IBM

Series/1

GA34-0152-0

File No. S1-01

IBM Series/1
Principles of Operation

IBM

Series/1

GA34-0152-0

File No. S1-01

IBM Series/1
Principles of Operation

First Edition (April 1981)

Use this publication only for the purpose stated in the Preface.

Changes are periodically made to the information herein; any such changes will be reported in subsequent revisions or Technical Newsletters.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address given below. Requests for copies of IBM publications should be made to your IBM representative or the IBM branch office serving your locality.

This publication could contain technical inaccuracies or typographical errors. A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to IBM Corporation, Information Development, Department 27T, P.O. Box 1328, Boca Raton, Florida 33432. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Preface

This publication describes the common functional characteristics of IBM Series/1 processors and their optional features.

The reader should understand data processing terminology and be familiar with binary and hexadecimal numbering systems.

This publication is intended primarily as a reference manual for experienced programmers who require machine code information to plan, correct, and modify programs written in the assembler language. It is also intended for the person who requires machine status information and interrupt-handling procedures.

This manual is to be used in conjunction with Series/1 processor and I/O description manuals.

Chapter 1. Introduction is an introduction to the Series/1. It contains a general description of the processors and features.

Chapter 2. Processor Unit Description contains a description of processor hardware, including registers and indicators.

Main storage data formats and addressing are presented in this chapter.

The "Program Execution" section covers:

- Basic instruction formats
- Effective-address generation
- Processor state control
- Initial program load (IPL)
- Jumping and branching
- Level switching and interrupts
- Stack operations

Chapter 3. Interrupts and Level Switching describes the priority interrupt levels and the interrupt processing for I/O devices and class interrupts. Related topics are:

- Program-controlled level switching
- Interrupt masking facilities
- Recovery from error conditions

Chapter 4. Input/Output Operations describes the I/O commands and control words that are used to operate the I/O devices. Condition codes and status information relative to the I/O operation are also explained. Specific command and status-word bit structures are contained in the I/O device description manuals.

Chapter 5. Storage Address Relocation Translator describes the relocation translator, including relocation addressing and address space management. The storage address relocation translator is not available on some processors.

Chapter 6. Clock/Comparator explains the functions of the clock/comparator. The clock/comparator is not available on some processors.

Chapter 7. Floating-Point Feature describes the optional floating-point feature. The floating-point feature is not available for some processors.

Chapter 8. Instructions describes the basic instruction set, including indicator settings and possible exception conditions. Individual instruction word formats that contain bit combinations for the operation codes and function fields are included. The instructions are arranged in alphabetical sequence based on assembler mnemonics.

Appendixes:

- Instruction formats
- Assembler syntax
- Number systems and conversion tables
- Character codes
- Carry and overflow indicators
- Reference information

Note: Refer to individual processor publications for a discussion of the optional programmer console.

Related Publications

Additional publications are listed in the *IBM Series/1 Graphic Bibliography*, GA34-0055.

Contents

Chapter 1. Introduction

Processor Characteristics

The IBM Series/1 processor is a compact, general-purpose computer that has the following characteristics:

- Four priority interrupt levels—independent registers and status indicators for each level. Automatic and program-controlled level switching.
- Instruction set that includes: stacking and linking facilities, multiply and divide, variable-field-length byte operations, and a variety of arithmetic and branching instructions.
- Supervisor and problem states.
- Designed for mounting in standard 483 mm (19-inch) rack; some models do not require rack-mounting.
- Basic console standard in processor unit; programmer console optional.
- An address translator (not installed on all processors).
- A clock/comparator (not installed on all processors).
- Channel capability:
  - Asynchronous, multidropped channel.
  - 256 input/output (I/O) devices can be addressed.
  - Direct program control and cycle-steal operations.

The processor unit contains power and space for additional features. The IBM 4959 Input/Output Expansion Unit and the IBM 4965 Diskette Drive and I/O Expansion Unit are available for additional features.

Processor Description

Figure 1-1 shows a block diagram of an IBM Series/1 processor and an IBM 4959 Input/Output Expansion Unit.

Four priority interrupt levels are implemented in the processor. Each level has an independent set of machine registers. Level switching can occur by program control or automatically upon acceptance of an I/O interrupt request. The interrupt mechanism provides 256 unique entry points for I/O devices.

The processor instruction set contains a variety of instruction types. These include: shift, register to register, register immediate, register to (or from) storage, bit manipulation, multiple register to storage, variable byte field, and storage to storage. Supervisor and problem states are implemented, with appropriate privileged instructions for the supervisor.

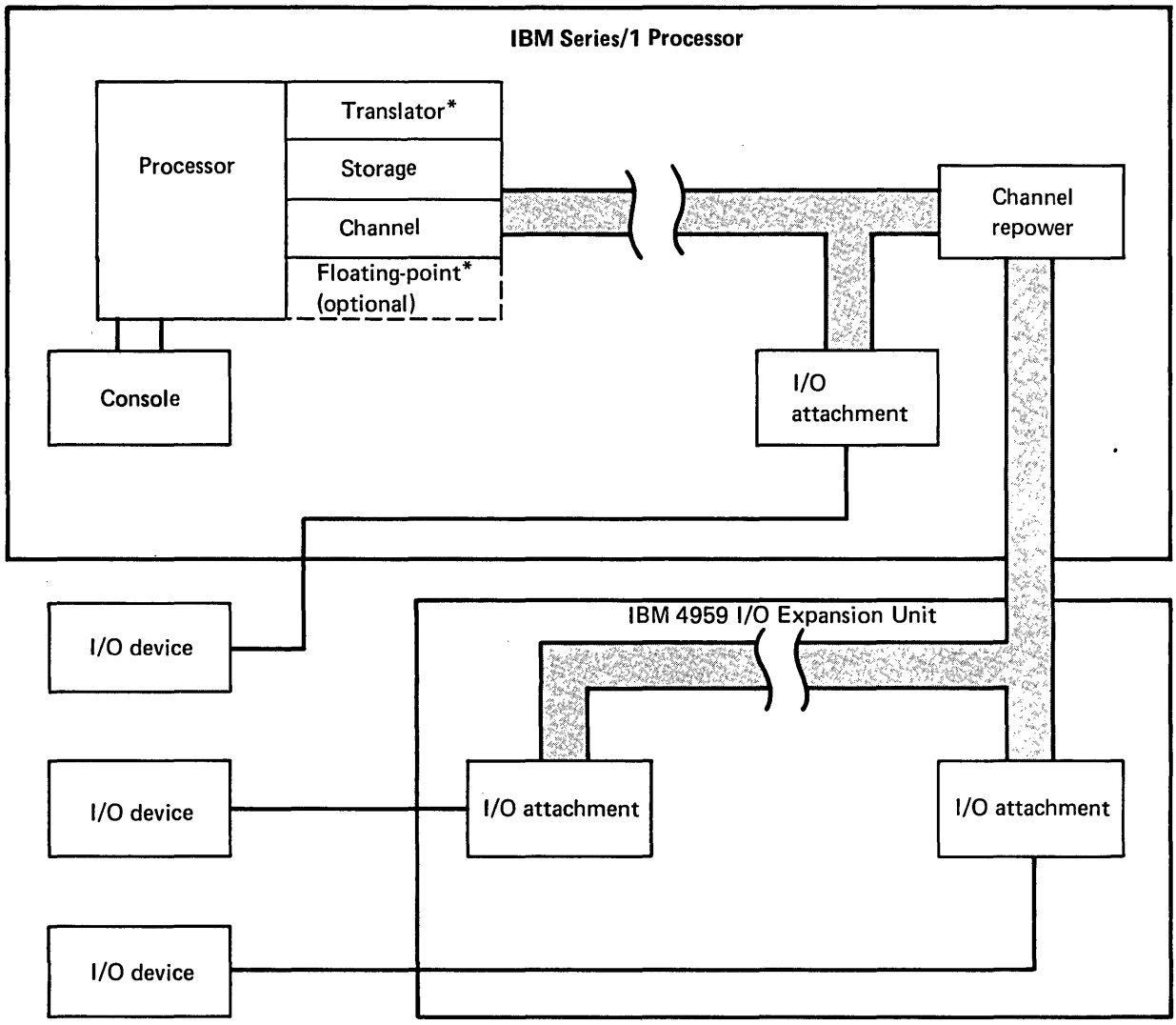


Figure 1-1. Block diagram of an IBM Series/1 processor and an IBM 4959 Input/Output Expansion Unit

The basic console is intended for dedicated systems that are used in a basically unattended environment. Only minimal controls are provided. A programmer console, which can be added as a feature, provides a variety of indicators and controls for operator-oriented systems.

I/O devices are attached to the processor through the processor I/O channel. The channel directs the flow of information between the I/O devices, the processor, and main storage. This channel accommodates a maximum of 256 directly addressable devices.

The channel supports:

- *Direct-program control operations*. Each Operate I/O instruction transfers a byte or word of data between main storage and the device. The operation may or may not terminate in an interrupt.
- *Cycle-steal operations*. Each Operate I/O instruction initiates multiple data transfers between main storage and the device (65,535 bytes maximum). Cycle-steal operations are overlapped with processing operations and always terminate in an interrupt.
- *Interrupt servicing*. Interrupt requests from the devices, along with cycle-steal requests, are presented and polled concurrently with data transfers.

Input/Output Units, I/O Features, and Processor Options

A variety of I/O units and features, plus several processor options, are available for use with the Series/1 processor. For a list and description of system units and features, refer to the *IBM Series/1 System Selection Guide*, GA34-0143, and the *IBM Series/1 System Summary*, GA34-0035. Detailed information about I/O units and features can be found in separate publications. The order numbers for these publications are contained in the *IBM Series/1 Graphic Bibliography*, GA34-0055.

Chapter 2. Processor Unit Description

Main Storage

Main storage holds data and instructions for applications to be processed on the system. The data and instructions are stored in units of information called bytes. Each byte consists of eight binary data bits plus a parity bit. Odd parity by byte is maintained throughout storage; even parity causes a machine-check error. Formats shown in this manual exclude the parity bits because they are not a part of the data flow manipulated by the instructions.

The bits within a byte are numbered consecutively, left to right, 0 through 7. When a format consists of multiple bytes, the numbering scheme is continued (for example, the bits in the second byte would be numbered 8 through 15). Leftmost bits are sometimes referred to as high-order or most-significant bits; rightmost bits are referred to as as low-order or least-significant bits.

Bytes can be handled separately or grouped together. A word is a group of two consecutive bytes, beginning on an even-address boundary, and is the basic building block of instructions. A doubleword is a group of four consecutive bytes, beginning on an even address boundary.

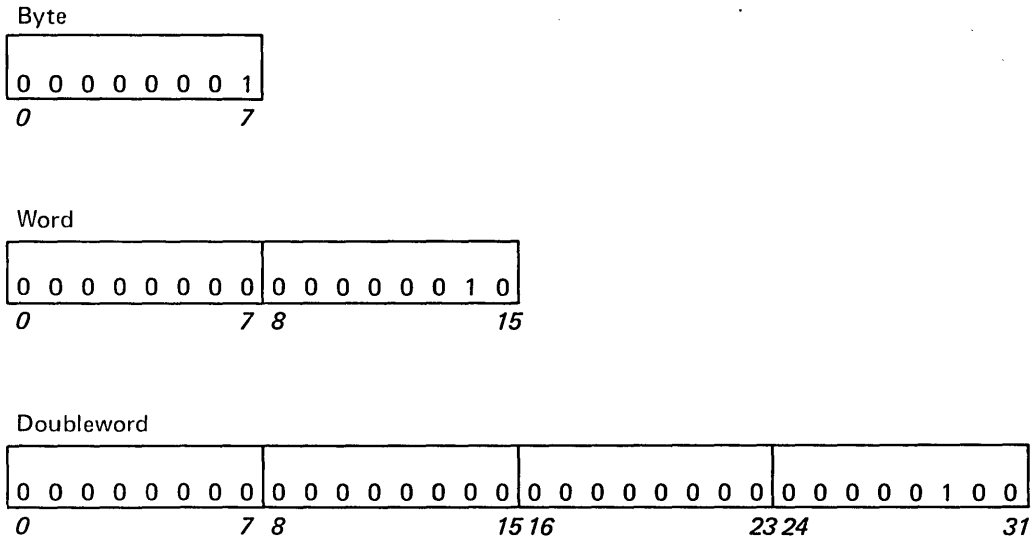
Addressing Main Storage

Each byte location in main storage is directly addressable. Byte locations in storage are numbered consecutively, starting with location 0; each number is considered to be the address of the corresponding byte. Storage addresses are 16-bit unsigned binary numbers. This permits a direct addressing range of 65,536 bytes.

When the storage address relocation translator is enabled, the logical address translates into a physical address that allows addressing beyond 65,536 bytes. Refer to individual processor publications for information regarding maximum fitted storage size.

Instruction and Operand Address Boundaries

As previously stated, all storage addressing is defined by byte location. Instructions can refer to bits, bytes, byte strings, words, or doublewords as data operands. All word and doubleword operand addresses must be on even-byte boundaries. All word and doubleword operand addresses point to the most-significant (leftmost) byte in the operand. Bit addresses are specified by a byte address and a bit displacement.



To provide maximum addressing range, some instructions refer to a byte, word, or doubleword displacement that is added to the contents of a register. In these cases, the operand is a word and the register must contain an even-byte address for valid results.

- All instructions must be on an even-byte boundary.
- The effective address for all branch type instructions must be on an even-byte boundary to be valid.

If the rules of even-byte addressing are violated, a program-check interrupt occurs with specification check set in the processor status word (PSW). The instruction is suppressed unless otherwise noted in the individual instruction description in Chapter 8.

Arithmetic and Logic Unit (ALU)

The arithmetic and logic unit (ALU) contains the hardware circuits that perform addition, subtraction, and logical operations; such as, AND, OR, and Exclusive OR. The ALU performs address arithmetic as well as the operations required to process the instruction operands. Operands may be regarded as signed or unsigned by the programmer. However, the ALU does not distinguish between them. Refer to "Numbering Representation" in this chapter for a detailed discussion of signed or unsigned operands. For many instructions, indicators are set to reflect the result of the ALU operation. Refer to "Indicator Bits" in this chapter for a detailed discussion of indicator settings.

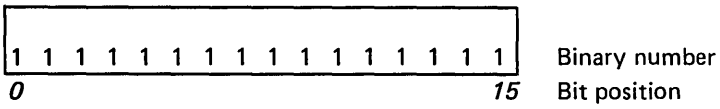
Numbering Representation

Operands may be signed or unsigned depending on how they are used by the programmer. An unsigned number is a binary integer in which all bits contribute to the magnitude. A storage address is an example of an unsigned number. A signed number is one where the high-order bit is used to indicate the sign, and the remaining bits define the magnitude. Signed positive numbers are represented in true binary notation with the sign bit (high-order bit) set to 0. Signed negative numbers are represented in two's complement notation with the sign bit (high-order bit) set to 1. The two's complement of a number is obtained by inverting each bit of the number and adding a 1 to the low-order bit position. Two's complement notation does not include a negative 0. The maximum positive number consists of an all-1's integer field with a sign bit of 0; the maximum negative number (the negative number with the greatest absolute value) consists of an all-0's integer field with a 1-bit for the sign.

The following examples show:

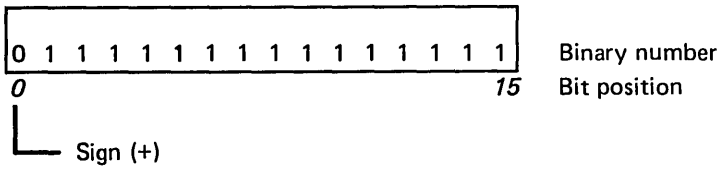
- An unsigned 16-bit number
- A signed 16-bit positive number
- A signed 16-bit negative number

Example of an unsigned 16-bit number:



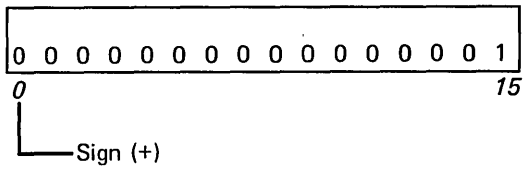
Decimal value	65535	(The largest unsigned number
Hexadecimal value	FFFF	representable in 16 bits.)

Example of a signed 16-bit positive number:



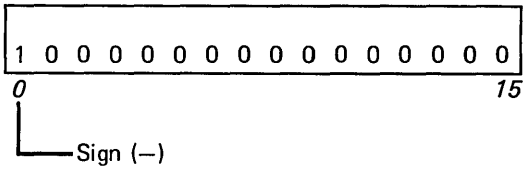
Decimal value	+32767	(The largest positive signed
Hexadecimal value	7FFF	number representable in 16 bits.)

When the number is positive, all bits to the left of the most-significant bit of the number, including the sign bit, are 0's.



Decimal value	+1
Hexadecimal value	0001

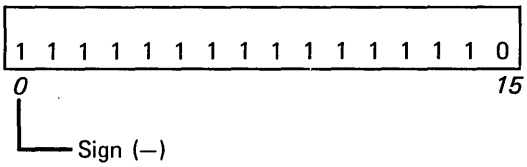
Example of a signed 16-bit negative number:



Decimal value	–32768	(The largest negative signed
Hexadecimal value	8000	number representable in 16 bits.)

Note: This form of representation yields a negative range of one more than the positive range.

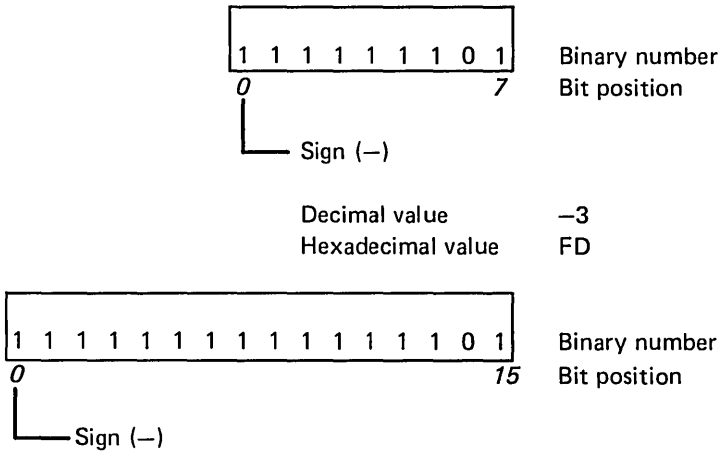
When the number is negative, all bits to the left of the most-significant bit of the number, including the sign bit, are set to 1's.



Decimal value	–2
Hexadecimal value	FFFE

When a signed-number operand must be extended with high-order bits, the extension is achieved by prefixing a field with each bit set equal to the high-order bit of the operand.

Example of an 8-bit field extended to a 16-bit field:



When performing the add and subtract operations, the processor does not regard the number as either signed or unsigned, but performs the designated operation on the values presented. Whether a given add or subtract operation is to be regarded as a signed operation or an unsigned operation is determined by the programmer's view of the values being presented as operands. The carry indicator and the overflow indicator of the level status register (LSR) are changed on various operations to reflect the result of that operation. This allows the programmer to make result tests for the number representation involved. The carry and overflow indicator settings are explained in "Indicator Bits" in this chapter.

Registers

There are two general types of registers: system and level registers. The system registers are one-of-a-kind registers that retain information common to all priority-interrupt levels. The level registers, which are duplicated for each priority-interrupt level, retain information that must be saved when a level is preempted.

Information that pertains only to the current process is kept in registers common to all levels. The registers in each category are listed in this section.

Registers supplied on a system basis:

- Processor status word (PSW) register
- Mask register (interrupt level)
- Clock register(not installed on all processors)
- Comparator register(not installed on all processors)
- Segmentation registers (not installed on all processors)

Registers supplied on a system basis, using the programmer console:

- Console data buffer register
- Current-instruction address register (CIAR)
- Storage address register (SAR)
- Console address key register
- Console stop-on-address register

Registers supplied on a level basis:

- Address key register (AKR)
- General registers (eight per level)
- Instruction address register (IAR)
- Level status register (LSR)
- Floating-point registers (optional; not available for some processors)

Note: For a specific level, the contents of the IAR, AKR, LSR, and the general registers are known as a level status block (LSB). The LSB is a 22-byte entity used by hardware and software for task control and task switching.

System Registers

Processor Status Word (PSW) Register

The processor status word (PSW) is a 16-bit register used to record error or exception conditions that may prevent further processing, and to hold certain flags that aid in error recovery. Error or exception conditions recorded in the PSW result in a class interrupt. Each bit in the PSW is described in detail in Chapter 3. The PSW can be accessed by using the Copy Processor Status and Reset (CPPSR) instruction. Refer to Chapter 8 for a detailed description of this instruction.

Mask Register

The mask register is used for control of interrupts. Bit 0 controls level 0, bit 1 controls level 1, and so on.

A 1-bit enables interrupts on a level; a 0-bit disables interrupts. For example, if bit 3 is set to a 1, interrupts are enabled on level 3.

Clock Register

The clock register is a 32-bit register that is incremented at 1-millisecond intervals. Refer to Chapter 6 for further information concerning the clock register.

Comparator Register

The comparator register is a 32-bit register that is used in conjunction with the clock register to generate the clock class interrupt. Refer to Chapter 6 for further information concerning the comparator register.

Segmentation Registers

A segmentation register is a register that changes a logical address to a physical address. Refer to Chapter 5 for further information concerning the segmentation registers.

Console Data Buffer Register

The console data buffer is a 16-bit register associated with the programmer console. The contents of the console data buffer can be loaded into a specified general register by using the Copy Console Data Buffer (CPCON) instruction. Refer to Chapter 8 for a detailed description of this instruction. Refer to individual processor publication for further information concerning the programmer console.

Current-Instruction Address Register (CIAR)

The current-instruction address register (CIAR) is not addressable by software. It may be displayed from the programmer console. When the processor enters the stop state, the CIAR contains the address of the last instruction that was executed. Refer to "Stop State" under "Processor State Control" in this chapter for methods of entering stop state.

Storage Address Register (SAR)

The storage address register (SAR) is not addressable by software. It is used for certain programmer console operations. SAR is a 16-bit register that contains the main-storage address for the last attempted processor storage cycle. Refer to individual processor publications for information concerning the programmer console.

Console Address Key Register

The console address key register is not addressable by software. When the programmer console is installed, this register is used for certain console operations. Refer to individual processor publications for information concerning the programmer console.

Console Stop-On-Address Register

The console stop-on-address register is not addressable by software. When the programmer console is installed, this register is used for certain console operations. Refer to individual processor publications for information concerning the programmer console.

Level Registers

Address Key Register (AKR)

The address key register (AKR) is a 16-bit register that contains three address keys and an address-key control bit. Separate three-bit fields contain an address key for instruction address space, operand-1 address space, and operand-2 address space.

General Registers

Subsequently referred to simply as registers, the general registers are 16-bit registers available to the program for general purposes. Eight registers are provided for each level. The R- and RB fields in the instructions control the selection of these registers.

Instruction Address Register (IAR)

The instruction address register (IAR) is a 16-bit register that holds the main storage address used to fetch an instruction. After an instruction has been fetched, the IAR is updated to point to the next instruction to be fetched.

Note: These registers are sometimes referred to as IAR0, IAR1, IAR2, and IAR3. The numbers represent the priority level IAR.

Level Status Register (LSR)

The level status register (LSR) is a 16-bit register that holds:

- Indicator bits, which are set as a result of arithmetic, logical, or I/O operations
- A supervisor state bit
- An in-process bit
- A trace bit
- A summary mask bit

These bits are discussed further in the following paragraphs. Seven other bits in the LSR are not used and are always set to 0's.

Floating-Point Registers

A floating-point register is a 64-bit register. The floating-point feature includes four 64-bit floating-point registers for each of the four priority interrupts levels in the processor. Refer to Chapter 7 for a detailed discussion of the floating-point feature.

Indicator Bits

The indicators are located in bits 0–4 of the level status register (LSR). Figure 2-1 shows the indicators and how they are set for arithmetic operations. The indicator bits are changed or not changed depending on the instruction being executed. Some instructions do not affect the indicators, other instructions change all of the indicators, and still other instructions change only specific indicators. Refer to the individual instruction descriptions in Chapter 8 for the indicators that are changed by each instruction.

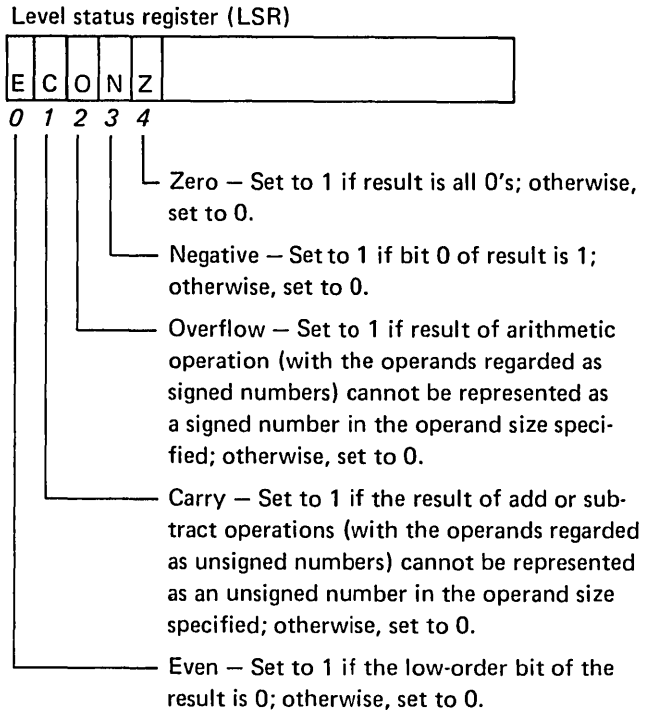


Figure 2-1. How indicators are set for signed and unsigned (logical) operations

The indicators are changed in a specialized manner for certain operations. These operations are described briefly. Additional information is provided in subsequent paragraphs for those operations where more detail is required.

- *Add, subtract, or logical operations.* The even, negative, and zero indicators are result indicators. For add and subtract operations, the carry and overflow indicators are changed to provide information for both signed and unsigned number representations.
- *Multiply and divide operations.* Signed number operands are always assumed for these operations. The carry indicator is used to provide a divide by 0 indication for the divide instruction. The overflow indicator defines an unrepresentable product for multiply operations. Refer to the individual instruction descriptions in Chapter 8.
- *Priority interrupts and input/output operations.* The even, carry, and overflow indicators are used to form a three-bit condition code that is set as a binary value.

- *Compare operations.* The indicators are set in the same manner as in a subtract operation.
- *Shift operations.* The carry and overflow indicators have a special meaning for shift left logical operations.
- *Complement operations.* The overflow indicator is set if an attempt is made to complement the maximum negative number. This number is not representable.
- *Set Indicators (SEIND) and Set Level Block (SELB) instructions.* All indicators are changed by the data associated with these instructions.

Even, Negative, and Zero Result Indicators

The even, negative, and zero indicators are called the result indicators. A positive result is indicated when the zero and negative indicators are both off (set to 0's). These indicators are set to reflect the result of the last arithmetic or logical operation performed. A logical operation in this sense includes data movement instructions. Refer to the individual instruction descriptions in Chapter 8 for the indicators changed for specific instructions.

Even, Carry, and Overflow Indicators—Condition Code for Input/Output Operations

The even, carry, and overflow indicators contain the I/O condition code following the execution of an Operate I/O instruction and following an I/O interrupt.

These indicators are used to form a three-bit binary number that results in a condition code value. For additional information about condition codes, refer to Branch on Condition Code (BCC) and Branch on Not Condition Code (BNCC) instructions in Chapter 8 and "I/O Condition Codes and Status Information" in Chapter 4.

Carry and Overflow Indicators—Add and Subtract Operations

A common set of add and subtract integer operations performs both signed and unsigned arithmetic. Whether a given add or subtract operation is to be regarded as a signed operation or an unsigned operation is determined by the programmer's view of the values being presented as operands. The carry and overflow indicators are set to reflect the results for both cases.

Carry Indicator Setting

The carry indicator is used to signal overflow of the result when operands are presented as unsigned numbers.

Overflow Indicator Setting

The overflow indicator is used to signal overflow of the result when the operands are presented as signed numbers.

Note: Appendix E explains the meaning of these indicators for signed and unsigned numbers. The appendix also provides examples for setting the carry and overflow indicators.

Carry and Overflow Indicators—Shift Operations

The carry and overflow indicators are changed for shift left logical operations and shift left and test operations. These operations affect the indicators as follows:

- The carry indicator is set to reflect the value of the last bit shifted out of the target register (register where bits are being shifted).
- The overflow indicator is set to 1 if bit 0 of the target register was changed during the shift; otherwise, it is set to 0.

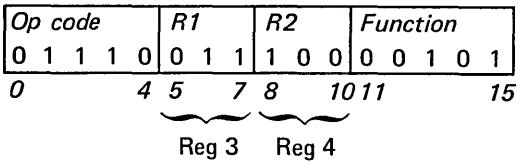
Indicators—Compare Operations

A compare operation sets the indicators in the same manner as a subtract operation. The even, negative, and zero indicators reflect the result. The carry and overflow indicators are set as described previously.

Compare instructions provide a test between two operands (without altering either operand) so that conditional branch and jump instructions may be used to control the programming logic flow. The conditions specified in branch and jump instructions are named such that, when the condition of the "subtracted from" operand relative to the other operand is true, the jump or branch occurs; otherwise, the next sequential instruction is executed. This is illustrated in the following example.

Example of compare operation:

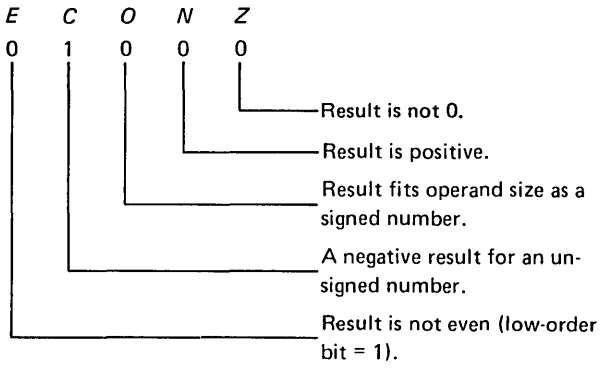
Instruction name	Assembler mnemonic	Operands
Compare word	CW	Reg 3, Reg 4



In this example, the contents of register 3 are subtracted from register 4:

		Decimal Unsigned	Signed
Reg 4 contents	0000 0000 0000 0010	2	+2
Reg 3 contents	1111 1111 1111 1011	65531	−5
Subtract result		−65529	+7
Machine operation:			
Minuend	0000 0000 0000 0010		
Subtrahend	0000 0000 0000 0100	one's complement	
Constant	1	for two's complement	
Result	0000 0000 0000 0111		

Indicator settings:



If the programmer wants to compare unsigned numbers, such as storage addresses, the logical conditional tests should be used (refer to Figure 2-2). In this example, assuming unsigned number representation, register 4 is logically less than register 3 and unequal to register 3. Therefore, the following branch instructions cause a transfer to symbolic location A (assuming register values shown in the example):

```
    CW    Reg 3,Reg 4
    BLLT  A
or
    CW    Reg 3,Reg 4
    BNE   A
```

The complementary tests (BLGT and BE) do not cause a transfer in this case.

If the programmer wants to compare signed numbers, the arithmetic conditional tests should be used (refer to Figure 2-2). In the previous compare word example, assuming signed number representation, register 4 is greater than register 3 and unequal to register 3. The following branch instructions would cause a transfer to symbolic location A.

```
    CW    Reg 3,Reg 4
    BGT   A
or
    CW    Reg 3,Reg 4
    BNE   A
```

The complementary tests (BLT and BE) do not cause a transfer.

Note: Jump instructions are also available for the logical and arithmetic conditional tests.

It must be emphasized again that the processor does not regard the numbers as either signed or unsigned. The compare word instruction results in a subtract operation being performed on the values presented. The programmer must then choose the correct conditional test (logical or arithmetic) for the number representation involved.

Condition tested by conditional branch or jump instruction	Assembler extended mnemonics	Indicators tested				
		0	1	2	3	4
		E	C	O	N	Z
Zero or equal	BE, BZ, JE, JZ					1
Not zero or unequal	BNE, BNZ, JNE, JNZ					0
Positive and not zero	BP, JP				0	0
Not positive	BNP, JNP				1	1
Negative	BN, JN				1	
Not negative	BNN, JNN				0	
Even	BEV, JEV	1				
Not even	BNEV, JNEV	0				
Arithmetically less than	BLT, JLT			0 1	1 0	
Arithmetically less than or equal	BLE, JLE			0 1	1 0	1
Arithmetically greater than or equal	BGE, JGE			1 0	1 0	
Arithmetically greater than	BGT, JGT			1 0	1 0	0 0
Logically less than or equal	BLLE, JLLE		1			1
Logically less than (carry)	BLLT, JLLT		1			
Logically greater than	BLGT, JLGT		0			0
Logically greater than or equal (no carry)	BLGE, JLGE		0			

Legend:

LSR bit	Indicator
0	E – Even
1	C – Carry
2	O – Overflow
3	N – Negative
4	Z – Zero

Figure 2-2. Indicators tested by conditional branch and jump instructions

Indicators—Multiple Word Operands

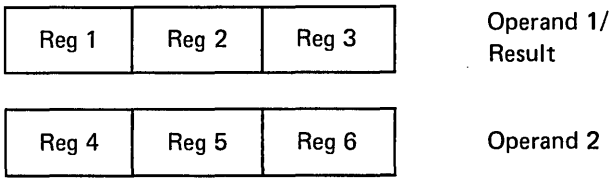
A programmer may desire to work with numbers that cannot be represented in one word or in a doubleword. It may take three or more words to represent the number.

The following register-to-register instructions allow the programmer to add or subtract these multi-word operands and then have the indicators reflect the multi-word result:

- Add Carry Register (ACY)
- Add Word With Carry (AWCY)
- Subtract Carry Register (SCY)
- Subtract Word With Carry (SWCY)

The following two examples show how the add instructions are used. A subtract operation is similar. Refer to Chapter 8 for details of the individual instructions.

Example 1. (Equal-length operands)



Program steps:

```
AW      Reg 6,Reg 3
AWCY    Reg 5,Reg 2
AWCY    Reg 4,Reg 1
```

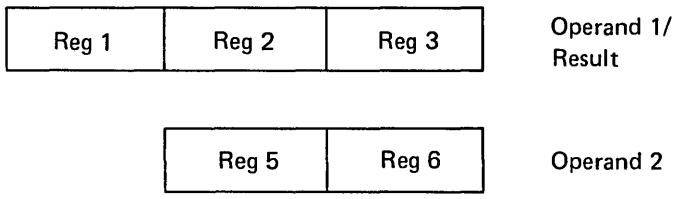
Explanation:

Step 1: The contents of register 6 are added to the contents of register 3.

Step 2: The contents of register 5 are added to the contents of register 2 plus any carry from the previous operation.

Step 3: The contents of register 4 are added to the contents of register 1 plus any carry from the previous operation.

Example 2. (Unequal-length operands)



Note: In this example, operand 2 must be an unsigned number or must be positive.

Program steps:

```
AW      Reg 6,Reg 3
AWCY    Reg 5,Reg 2
ACY     Reg 1
```

Explanation:

Step 1: The contents of register 6 are added to the contents of register 3.

Step 2: The contents of register 5 are added to the contents of register 2, plus any carry from the previous operation.

Step 3: Any carry from the previous operation is added to the contents of register 1.

Note: In both examples, the final indicator settings reflect the status of the three-word result.

Even	Set to 1 if the result low-order bit of register 3 is 0.
Carry	Set to 1 if the result cannot be represented as an unsigned three-word number.
Overflow	Set to 1 if the result cannot be represented as a signed three-word number.
Negative	Set to 1 if the result high-order bit of register 1 is 1.
Zero	Set to 1 if all three result registers contain 0's.

Testing Indicators with Conditional Branch and Jump Instructions

The indicators are tested according to a selected condition when a conditional branch or a conditional jump instruction is executed, as shown in Figure 2-2.

The conditional instructions are:

- Branch on Condition (BC)
- Branch on Not Condition (BNC)
- Jump on Condition (JC)
- Jump on Not Condition (JNC)

The assembler also provides extended mnemonics for the conditions shown in Figure 2-2. Refer to the individual instructions in Chapter 8.

Supervisor State Bit

Level status register (LSR) bit 8, when set to 1, indicates that the processor is in the supervisor state, which allows privileged instructions to be executed. This bit is set by any of the following:

- Class interrupt
  - Machine-check condition
  - Program-check condition
  - Power/thermal warning
  - Supervisor Call (SVC) instruction
  - Soft-exception trap condition
  - Trace
  - Console interrupt
  - Clock/comparator

- I/O interrupt
- Initial program load (IPL)
- System reset
- Power-on reset

When LSR bit 8 is set to 0, the processor is in problem state. For a selected priority level, the supervisor can alter the supervisor state bit by using a Set Level Block (SELB) instruction. For additional information, refer to "Processor State Control" in this chapter.

Class interrupts and I/O interrupts are described in Chapter 3. IPL is discussed under "Initial Program Load (IPL)" in this chapter.

In-Process Bit

Level status register (LSR) bit 9, when set to 1, indicates that a priority level is currently active or was preempted by a higher priority level before completing its task. Bit 9 is set to 0 by a Level Exit (LEX) instruction. Bit 9 can also be turned on or off by a Set Level Block (SELB) instruction. The in-process bit also affects level switching under program control. Refer to Chapter 3, "Interrupts and Level Switching," for further information.

Trace Bit

Level status register (LSR) bit 10, when set to 1, causes a trace class interrupt at the beginning of each instruction. The bit can be turned on or off with the Set Level Block (SELB) instruction. The trace bit aids in debugging programs. Refer to "Class Interrupts" in Chapter 3 for further information.

Summary Mask Bit

Level status register (LSR) bit 11, when set to 0 (disabled), inhibits all priority interrupts on all levels. It also inhibits power/thermal, clock, and console class interrupts. When this bit is set to 1 (enabled), normal interrupt processing is allowed. Refer to "Summary Mask" in Chapter 3 for details relating to control of the summary mask.

Program Execution

Instruction Formats

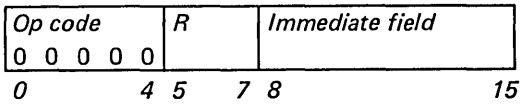
The processor instruction formats are designed for efficient use of bit combinations to specify the operation to be performed (operation code) and the operands that participate. Some formats also include an immediate data field or word, an address displacement or address word, and a function field that further modifies the operation code. Various combinations of these fields are used by the individual instructions. Some typical instruction formats are presented here. All formats are shown in Appendix A, "Instruction Formats."

One-Word Instructions

The basic instruction length is one word (16 bits). The operation code field (bits 0–4) is the only common field for all formats. This field, unless modified by a function field, specifies the operation to be performed. For a format without a function field, bits 5–15 specify the location of operands or data associated with an operand.

Example:

Instruction name	Assembler mnemonic	Syntax
Add Byte Immediate	ABI	byte,reg



Bits 0–4 Operation code (specifies ABI instruction).

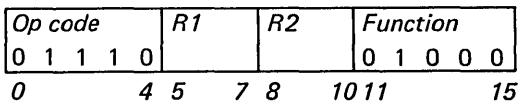
Bits 5–7 General register (0–7). This register contains data for the second operand.

Bits 8–15 Immediate data for the first operand.

In some cases, the operation code is the same for a group of instructions. The format for this group includes a function field. The bit combinations in the function field then determine the specific operation to be performed.

Example:

Instruction name	Assembler mnemonic	Syntax
Add Word	AW	reg,reg



Bits 0–4 Operation code for a group of instructions.

Bits 5–7 General register (0–7). This register contains data for the first operand.

Bits 8–10 General register (0–7). This register contains data for the second operand.

Bits 11–15 Function field. Modifies the operation code to specify the Add Word instruction.

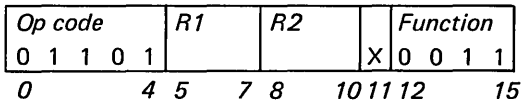
Note: For other instruction groups, the function field may vary as to location within the format and also to the number of bits used.

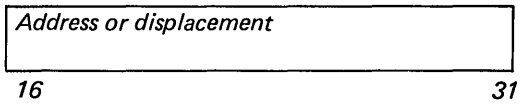
Two-Word Instructions

Bits 0–4 of the first word of this format are identical to the one-word instruction description. The second word (bits 16–31) contains either immediate data, an address, or a displacement. This word is used to provide data for an operand, or provide a main storage address or displacement for effective address generation. Refer to "Effective-Address Generation" in this chapter for further information.

Example:

Instruction name	**Assembler mnemonic**	**Syntax**
Branch and Link	BAL	longaddr,reg





Bits 0–4	Operation code.
Bits 5–7	General register (0–7) for the second operand.
Bits 8–10	General register (0–7) for the first operand.
Bit 11	Indirect address bit.
Bits 12–15	Function field.
Bits 16–31	A main storage address used for the first operand.

Note: Refer to "Branch and Link (BAL)" in Chapter 8 for further information.

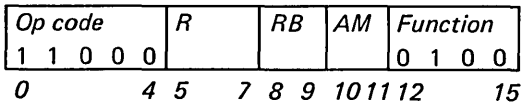
Variable-Length Instructions

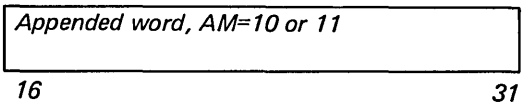
Some instructions use a selectable encoded technique for generating effective addresses. This method is referred to as an address argument technique in subsequent sections. These instruction formats contain a base register (RB) field and an address mode (AM) field. If both operands are using this technique, the format contains an RB and associated AM field for each. These fields are in the first instruction word. The AM field consists of two bits, and is referred to in binary notation (AM=00, 01, 10, or 11). If AM is equal to 10 or 11, an additional word is appended to the normal instruction word. For a format that contains two AM fields, two additional words may be appended. Refer to "Effective-Address Generation" in this chapter for a description of the appended words and how they are used.

For instructions with a single storage address argument, the RB field consists of two bits. An RB field of two bits, with its associated AM field of two bits, is referred to as a four-bit address argument or addr4 in assembler syntax.

Example:

Instruction name	Assembler mnemonic	Syntax
Compare byte	CB	addr4,reg





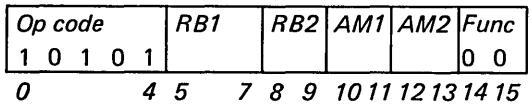
Bits 0–4 Operation code.
Bits 5–7 General register (0–7) for the second operand.
Bits 8–9 Base register (0–3).
Bits 10–11 Address mode.
Bits 12–15 Function field.
Bits 16–31 Appended word for the first operand.

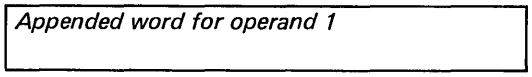
Note: The register specified by the RB field is a general register that is used as a base register for effective address generation.

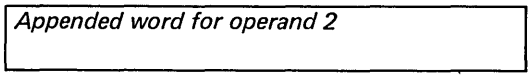
Some instruction formats have two storage address arguments. In this case, the first operand has a three-bit RB field, giving a five-bit address argument (addr5 in assembler syntax), and the second operand has a four-bit address argument.

Example:

Instruction name	Assembler mnemonic	Syntax
Add Word	AW	addr5,addr4







Bits 0–4	Operation code.
Bits 5–7	Base register (0–7) for the first operand.
Bits 8–9	Base register (0–3) for the second operand.
Bits 10–11	Address mode for the first operand.
Bits 12–13	Address mode for the second operand.
Bits 14–15	Function field.
Bits 16–31	Appended word for the first operand.
Bits 32–47	Appended word for the second operand.

Notes:

1. If there is no appended word for the first operand (AM1=00 or 01), the second operand word is appended to the instruction word in bits 16–31.
2. Registers specified by the RB fields are general registers.

Names of Instruction Formats

Names have been established for several categories of instructions. Each category has the same basic instruction format; therefore, the name is related to the format. In most cases, the name indicates the location of the operands or the type of instruction.

- Register/register instructions—General registers are used by both operands.
- Storage/storage instructions—Both operands reside in main storage.
- Register/storage instructions—One operand uses a general register; the other operand resides in main storage.
- Register immediate instructions—One operand uses a general register; the other operand uses an immediate data field. The immediate data field is the low-order byte of a one-word format or the second word of a two-word (long) format.

- Shift instructions with immediate count—This is a shift instruction with the count field contained within the instruction word.
- Storage immediate instructions—One operand is in main storage. The other operand uses an immediate data field. The immediate data field is the second word of a two-word format.
- Parametric instructions—For this instruction format, a parameter field (bits 8–15) is contained within the instruction word.

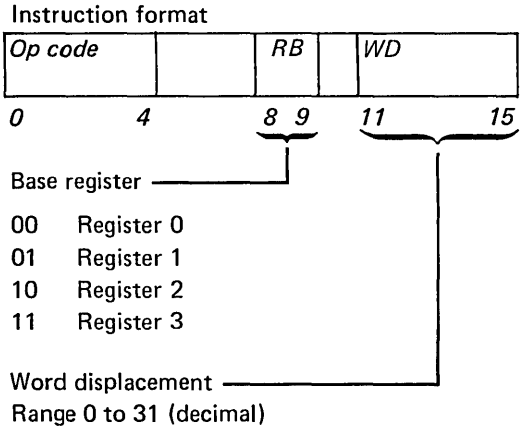
Effective-Address Generation

For purposes of storage efficiency, certain instructions formulate storage operand addresses in a specialized manner. These instructions have self-contained fields that are used when generating effective addresses. Standard methods for deriving effective addresses are included in this section. Other methods, such as bit displacements, are explained in the individual instruction descriptions in Chapter 8.

Programming Note: For the following instructions, the effective address points to a control block rather than to an operand:

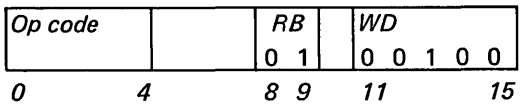
- Copy Level Block (CPLB)
- Load Multiple and Branch (LMB)
- Pop Byte (PB)
- Pop Doubleword (PD)
- Push Byte (PSB)
- Push Doubleword (PSD)
- Push Word (PSW)
- Pop Word (PW)
- Set Level Status Block (SELB)
- Store Multiple (STM)

Base Register Word Displacement Short



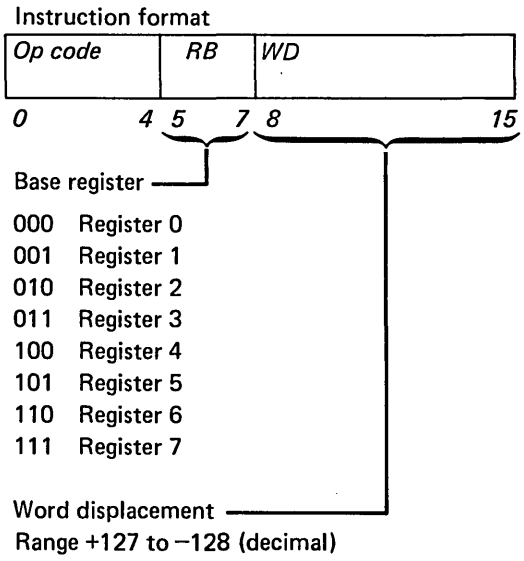
The five-bit unsigned integer (WD) is doubled in magnitude to form a byte displacement, and is then added to the contents of the specified base register to form the effective address. The contents of the base register must be even.

Example:



		Hex	Dec
Contents of register 1 (RB)	0000 0000 0110 0000	0060	0096
Word displacement (WD) doubled	+ 0 1000	8	8
Effective address	0000 0000 0110 1000	0068	0104

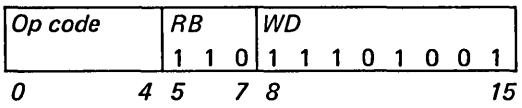
Base Register Word Displacement



The eight-bit signed integer (WD) is doubled in magnitude to form a byte displacement and is then added to the contents of the specified base register to form the effective address. The contents of the base register must be even.

The word displacement can be either positive or negative; bit 8 of the instruction word is the sign bit for the displacement value. If this high-order bit of the displacement field is a 0, the displacement is positive with a maximum value of +127 (decimal). If the high-order bit of the displacement field is a 1, the displacement is negative with a maximum value of −128. The negative number is represented in two's complement form.

Example:

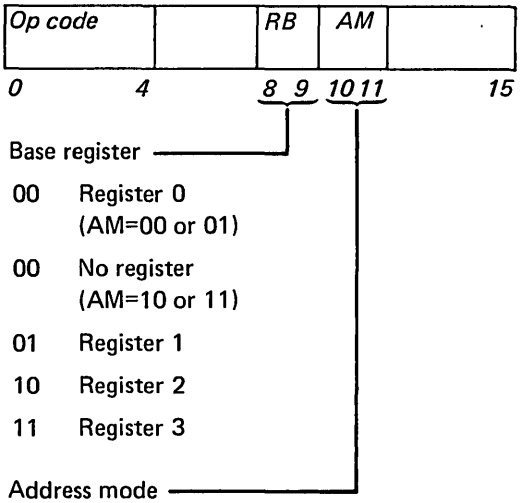


Note: This example uses a negative word displacement (–17 hex) shown in two's complement.

		Hex	Dec
Contents of register 6 (RB)	0000 0000 1000 0110	0086	0134
Word displacement (WD) doubled (sign bit is propagated left)	+1111 1111 1101 0010	– 2E	– 46
Effective address	0000 0000 0101 1000	0058	0088

Four-Bit Address Argument

Instruction format

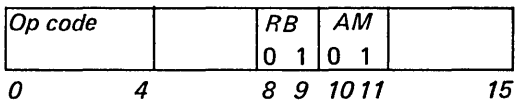


The address mode (AM) has the following significance:

AM=00. The contents of the selected base register form the effective address.

AM=01. The contents of the selected base register form the effective address. After use, the base register contents are incremented by the number of bytes in the operand. For some instructions, the effective address points to a control block rather than to an operand. When the effective address points to a control block, the base register contents are incremented by 2.

Example:

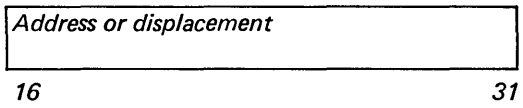


		Hex	Dec
Effective address (contents of register 1)	0000 0000 1000 0000	0080	0128
Contents of register 1 after instruction execution			
Byte operand	0000 0000 1000 0001	0081	0129
Word operand	0000 0000 1000 0010	0082	0130
Doubleword operand	0000 0000 1000 0100	0084	0132

Notes:

1. For register-to-storage instructions, if the specified register is the same for both operands, the register is incremented prior to using it as an operand.
2. Certain instructions (storage-to-storage) have two address arguments. Operand 1 has a three-bit RB field with its associated AM field. Operand 2 has a two-bit RB field with its associated AM field. If both RB fields specify the same register and both AM fields are equal to 01, the base register contents are incremented prior to fetching operand 2 and again after fetching operand 2. Assuming the same conditions, but with the operand 2 AM field not equal to 01, the base register contents are incremented prior to calculating the effective address for operand 2.
3. If the effective address points to a control block rather than to an operand; the base register contents are incremented by 2.

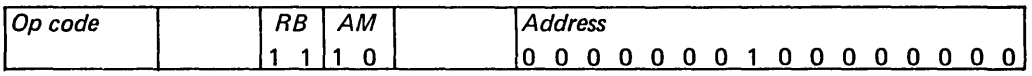
AM=10. An additional word is appended to the instruction. The word has the following format:



If RB is 0, the appended word contains the effective address.

If RB is not 0, the contents of the selected base register and the contents of the appended word (displacement) are added to form the effective address.

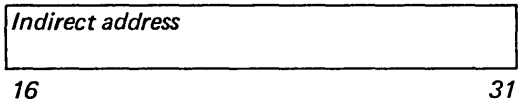
Example:



		Hex	Dec
Contents of register 3	0000 1000 0000 0000	0800	2048
Contents of appended word	+0000 0001 0000 0000	0100	0256
Effective address	0000 1001 0000 0000	0900	2304

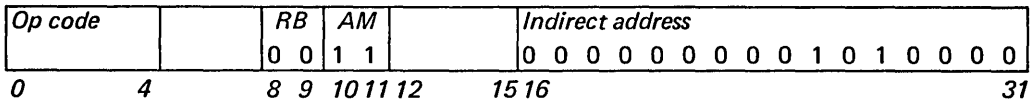
AM=11. An additional word is appended to the instruction.

If RB is 0, the appended word has the format:



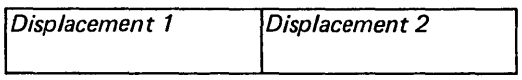
This address points to a main storage location, on an even-byte boundary, that contains the effective address.

Example:



		Hex	Dec
Contents of appended word	0000 0000 0101 0000	0050	0080
Effective address equals contents of storage at address 0080 (decimal)	0000 0100 0000 0000	0400	1024

If RB is not 0, the appended word has the format:



The two displacements are unsigned eight-bit integers. Displacement 2 is added to the contents of the selected base register to generate a main storage address. The contents of this storage location are added to displacement 1 and result in the effective address.

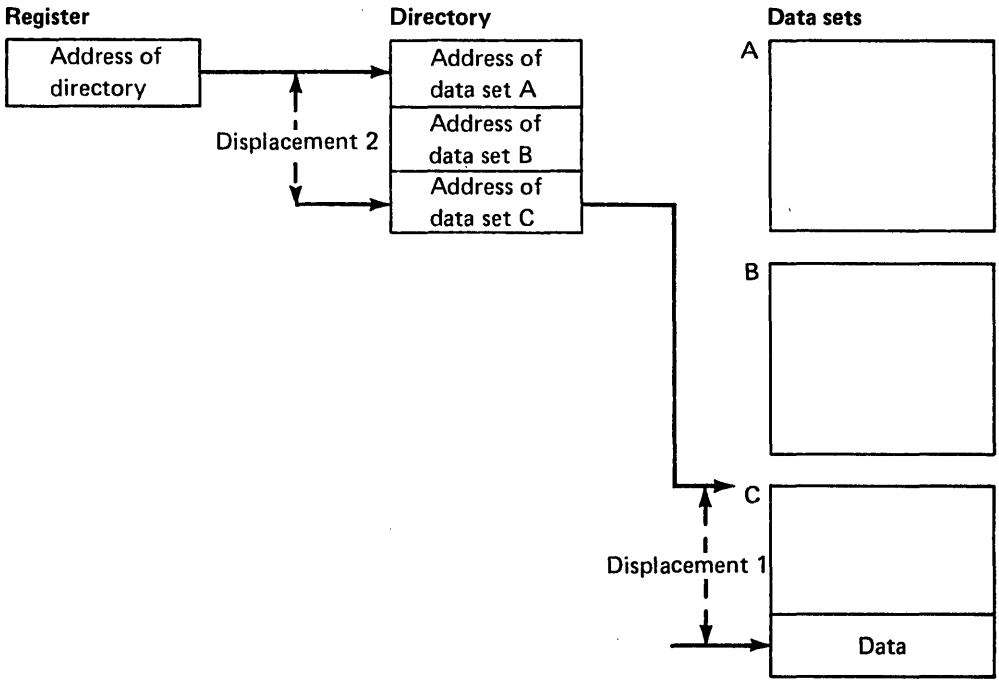
Example:

Op code		*RB*	*AM*		*Displacement 1*	*Displacement 2*
		1 0	1 1		0 0 1 0 0 1 0 1	0 1 0 0 0 0 1 0
0 4		*8 9*	*10 11*	*12*	*15 16*	*23 24* *31*

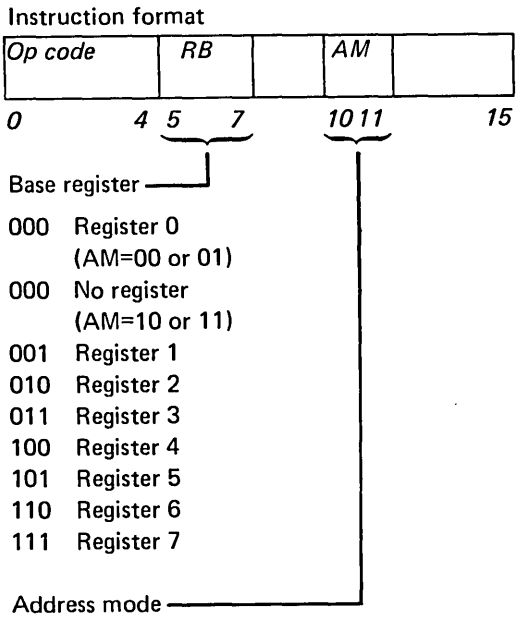
		Hex	Dec
Contents of register 2	0000 0101 0011 0101	0535	1333
Displacement 2	+ 0100 0010	42	66
Storage address	0000 0101 0111 0111	0577	1399
Contents of storage at address 1399 (decimal)	0000 0100 0001 0000	0410	1040
Displacement 1	+ 0010 0101	25	37
Effective address	0000 0100 0011 0101	0435	1077

Note: This example is invalid for other than a byte operand.

Programming Note: This addressing mode (AM=11, RB is not 0) is useful for the directorized data concept. For the addr4 or addr5 assembler syntax, the programmer codes the form displacement 1 (register, displacement 2)*. For addr4, the specified register is 1–3. For addr5, the specified register is 1–7. The asterisk denotes indirect addressing.

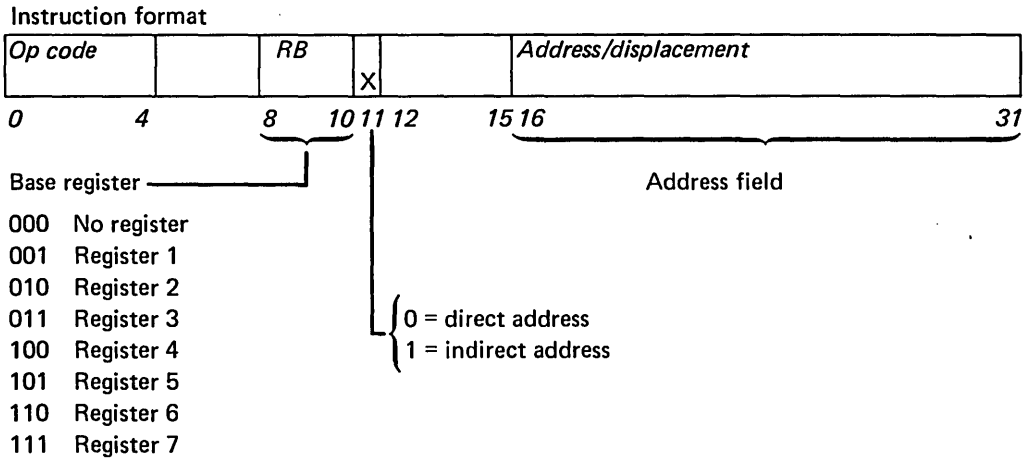


Five-Bit Address Argument



Operation of this mode is identical to the four-bit argument, but provides additional base registers.

Base Register Storage Address

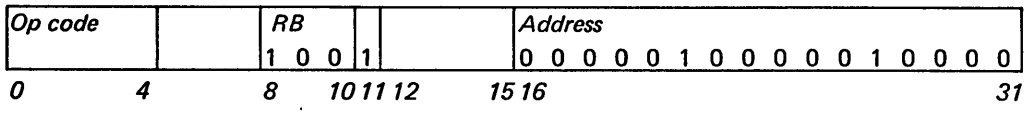


If RB is 0, the address field contains the effective address.

If RB is not 0, the contents of the selected base register and the contents of the address field are added together to form the effective address.

Note: Bit 11 specifies whether the effective addressing is direct or indirect addressing.

Example of indirect addressing:



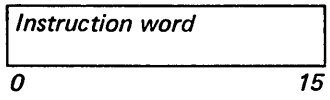
		Hex	Dec
Contents of register 4	0000 0001 0000 0000	0100	0256
Address field	+0000 0100 0001 0000	0410	1040
Storage address	0000 0101 0001 0000	0510	1296
Effective address Contents of storage at address 1296 (decimal)	0000 0110 0100 0000	0640	1600

Instruction Length Variations for Address Arguments

One-word instructions that contain a single AM field become two words in length if AM is equal to 10 or 11. The AM appended word follows the instruction word.

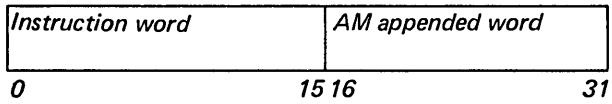
Example:

AM=00 or 01



No appended word

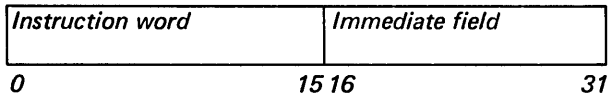
AM=10 or 11



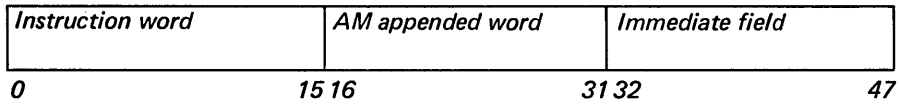
Two-word instructions that contain a single AM field become three words in length if AM is equal to 10 or 11. The AM word is appended to the first instruction word. The data or immediate field then becomes the third word of the instruction.

Example:

AM=00 or 01



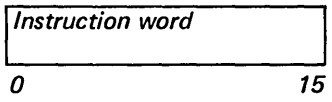
AM=10 or 11



One-word instructions that contain two AM fields (AM1 and AM2) may be one, two, or three words in length depending on the values of AM1 and AM2. The AM1 word is appended first; then the AM2 word is appended.

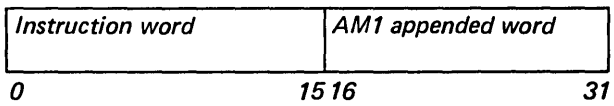
Example:

AM1=00 or 01
AM2=00 or 01

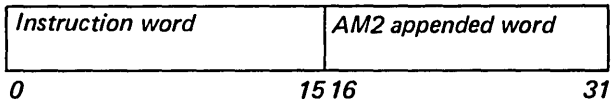


No appended word

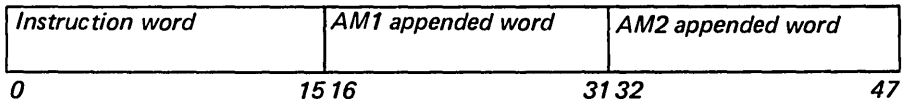
AM1=10 or 11
AM2=00 or 01



AM1=00 or 01
AM2=10 or 11



AM1=10 or 11
AM2=10 or 11



Processor State Control

If the processor is powered on, it is always in one of the following mutually exclusive states:

- Stop
- Wait
- Load
- Run—when in run state, programs can be executed in either:
  - Supervisor state or
  - Problem state

Stop State

The stop state is entered by any of the following methods:

- Pressing the Stop key on the programmer console.
- Execution of the Stop instruction when the Mode switch on the basic console is in the Diagnostic position and the optional programmer console is installed.
- An address match occurs (Stop On Address indicator on programmer console is lit).
- An instruction completes execution (Instruct Step indicator on programmer console is lit).
- An error occurs (Stop On Error indicator on programmer console is lit).
  - When the processor stops, the Check indicator is lit and the appropriate PSW bits are set to 1's.
  - Subsequently depressing any console key turns off the Check indicator but does not affect the PSW.
  - The next time the Start key is pressed (assuming no system reset) a class interrupt occurs (based on the PSW bit of the highest priority).
- Pressing the Reset key on the programmer console.
- Power-on reset occurs when the Mode switch is not in Auto IPL.

While the processor is in the stop state, the Stop light on the programmer console is on, the functions provided on the console can be activated, and no interrupt requests can be accepted by the processor.

Certain error or exception conditions cannot occur during stop state. These are specification check, privilege violate, invalid function, and stack exception. These conditions are explained under "Class Interrupts" in Chapter 3.

If an I/O check condition occurs during stop state, PSW bits 11 and 12 are set to 1's and the condition is preserved by hardware. The Check indicator is turned on. Pressing the Start key (assuming no system reset) allows a machine-check class interrupt to occur.

If a power/thermal warning condition occurs during stop state, PSW bit 15 is set to 1 and remains set for the duration of the condition. The Check indicator is not turned on. Subsequently depressing the Start key allows a power/thermal-warning class interrupt to occur, assuming that the condition is still active, the summary mask is enabled, and no system reset has occurred.

The processor exits the stop state by:

- Pressing the Load key on the basic console.
- Pressing the Start key on the programmer console. When the Start key is pressed, the processor returns to the state that was exited before entering stop state. If the run state is entered, one instruction is executed before interrupts are accepted by the processor. If the stop state is entered because of a reset (power-on reset or pressing the Reset key), pressing the Start key causes program execution to begin on level 0 with the instruction in location 0 of main storage. If the stop state is entered because of an error and the Stop On Error switch is set to on, a system reset or class interrupt can clear the error condition.

Notes:

1. Any manual entry into stop state is by the programmer console.
2. The Stop instruction performs no operation if the programmer console is not installed.

Wait State

The processor enters wait state when a Level Exit (LEX) instruction or a Set Level Block (SELB) instruction, which sets the current in-process bit off, is executed and no level is pending. While the processor is in the wait state, the Wait light on the basic console is on, and interrupts can be accepted under control of the system mask register and the summary mask, as defined by the LSR of the last active level.

The processor exits the wait state by:

- Pressing the Load key on the basic console.
- Pressing the Stop key on the programmer console.
- Pressing the Reset key on the programmer console.
- The processor accepting an I/O interrupt (the level must be enabled by the summary mask and the mask register).
- A class interrupt occuring.

Load State

The processor enters the load state when initial program load (IPL) begins. IPL occurs:

- When the Load key on the basic console is pressed.
- After a power-on reset, if the Mode switch is in the Auto IPL position.
- When an IPL signal is received from a host system.

While the processor is in load state, the Load light on the basic console is on. The processor exits the load state by:

- Successful completion of the IPL.
- Pressing the Stop key on the programmer console.
- Pressing the Reset key on the programmer console.

Refer to "Initial Program Load (IPL)" in this chapter for further information.

Run State

The processor enters the run state when it is not in the stop, wait, or load state. Run state is entered:

- From load state, upon successful completion of IPL
- From wait state, when an interrupt is accepted
- From stop state, when the Start key is pressed and the processor was in the run state prior to entering the stop state

The processor exits run state when entering stop, wait, or load state.

Supervisor State and Problem State

While in run state, instructions can be executed in either supervisor state or problem state. This is determined by level status register (LSR) bit 8:

- If LSR bit 8 is a 1, the processor is in supervisor state.
- If LSR bit 8 is a 0, the processor is in problem state.

Supervisor and problem states are discussed in the following paragraphs.

Supervisor State. The processor enters supervisor state when:

- A Supervisor Call (SVC) instruction is executed
- A class interrupt occurs
- An I/O interrupt is accepted
- After a successful initial program load (IPL)
- A reset occurs

Refer to Chapter 3, "Interrupts and Level Switching," for a detailed discussion of class interrupts and I/O interrupts.

When the processor is in supervisor state, the full instruction set may be executed. The following privileged instructions may be executed in supervisor state only:

Copy Address Key Register (CPAKR)
Copy Console Data Buffer (CPCON)
Copy Current Level (CPCL)
Copy Interrupt Mask Register (CPIMR)
Copy In-Process Flags (CPIPF)
Copy Instruction Space Key (CPISK)
Copy Floating Level Block (CPFLB)
Copy Level Block (CPLB)
Copy Operand 1 Key (CPOOK)
Copy Operand 2 Key (CPOTK)
Copy Processor Status and Reset (CPPSR)
Copy Segmentation Register (CPSR)
Copy Storage Key (CPSK)
Diagnose (DIAG)
Disable (DIS)
Enable (EN)
Interchange Operand Keys (IOPK)
Level Exit (LEX)
Operate I/O (IO)
Set Address Key Register (SEAKR)
Set Clock (SECLK)

Set Comparator (SECMP)
Set Console Data Lights (SECON)
Set Floating Level Block (SEFLB)
Set Instruction Space Key (SEISK)
Set Interrupt Mask Register (SEIMR)
Set Level Block (SELB)
Set Operand 1 Key (SEOOK)
Set Operand 2 Key (SEOTK)
Set Segmentation Register (SESR)
Set Storage Key (SESK)

Note: Refer to individual processor publications for further information concerning privileged instructions.

Problem State. The processor enters the problem state when the supervisor state bit (LSR bit 8) is set to 0. This is accomplished with a Set Level Status Block (SELB) instruction, which can change the contents of the registers for a selected priority interrupt level.

While the processor is in problem state, privileged instructions cannot be executed. If a privileged instruction execution is attempted, the instruction is suppressed and a program-check class interrupt occurs, with privilege violate (bit 2) set in the PSW.

Initial Program Load (IPL)

An initial program load function is provided to read an IPL record (set of instructions) from an external storage media, and automatically execute a start-up program. An IPL record is read into storage from a local I/O device or host system. The I/O attachments for the desired IPL sources are prewired at installation time. Two local sources, primary and alternate, can be wired and either one can be selected by using the IPL Source switch on the console.

IPL can be started by three methods:

- Manually, by pressing the Load key on the console.
- Automatically, after a power-on condition.
- Automatically, when a signal is received from a host system. The host system can be connected through a communications adapter.

The automatic power-on IPL is selected by the Mode switch on the console. When the Mode switch is in the Auto IPL position, IPL occurs whenever power turns on (either initially or after a power failure). Auto IPL is useful for unattended systems. A manual IPL can be initiated at any time by pressing the Load key on the console (even when in run state). The Mode switch has no effect on the manual IPL. For auto IPL and manual IPL, the local IPL source (primary or alternate) is selected. IPL from a host system can occur at any time and is initiated by the host system. The IPL record is transferred through the host system device (for example, the communications adapter). When an auto IPL occurs, bit 13 of the PSW is set to 1 to indicate the condition to the software. When a manual or host-system IPL occurs, this bit is set to 0.

The length of the IPL record depends on the media used by the IPL source.

Upon successful completion of an IPL, the processor enters supervisor state and begins execution on priority level 0. The summary mask is enabled and all priority interrupt levels in the mask register are enabled. The level 0 AKR is set to all 0's. The first instruction to be executed is at main storage location 0. The IPL source has a pending interrupt request on level 0. The system program must:

1. Perform housekeeping; for example, load vector table addresses in the reserved area of storage. Refer to "Automatic Interrupt Branching" in Chapter 3 for further information.
2. Issue a Level Exit (LEX) instruction. This allows the processor to accept the interrupt from the IPL source. When the interrupt is accepted, a forced branch is taken using the device-address vector table. The vector table entry is determined by the device address of the IPL source and results in a branch to the proper program routine for handling the interrupt. The device address of the IPL source is set into bits 8–15 of register 7 on level 0. Condition code 3, device end, is reported by the IPL source. For additional information, refer to "I/O Interrupts" in Chapter 3.

A system reset always occurs prior to an IPL. However, if any errors occur during the IPL, the results are unpredictable.

Sequential Instruction Execution

Normally, the operation of the processor is controlled by instructions taken in sequence. An instruction is fetched from the main storage location specified in the instruction address register (IAR). The instruction address in the IAR is then increased by the number of bytes in the instruction just fetched. The IAR now contains the address of the next sequential instruction. After the current instruction is executed, the same steps are repeated using the updated address in the IAR.

A change from sequential operation can be caused by branching, jumping, interrupts, level switching, or manual intervention.

Jumping and Branching

The normal sequential execution of instructions is changed when reference is made to a subroutine, when a two-way choice is encountered, or when a segment of coding, such as a loop, is to be repeated. All of these tasks can be accomplished with branching and jumping instructions. Provision is also made for subroutine linkage, permitting not only the introduction of a new instruction address, but also the preservation of the return address and associated information.

The conditional branch and jump instructions are used to test the indicators in the LSR. These indicators are set as the result of I/O operations and most arithmetic or logical operations. Single or multiple indicators are tested, as determined by the value in a three-bit field within the instruction. Refer to "Indicator Bits" and "Testing Indicators with Conditional Branch and Jump Instructions" in this chapter for further information.

Jumping

Jump instructions are used to specify a new instruction address relative to the address in the IAR. The new address must be within −256 to +254 of the byte following the jump instruction.

Note: The jump instruction contains a word displacement that is converted to a byte displacement when the instruction is executed. However, when the assembler is used, the programmer specifies a byte value, which is then converted to a word displacement by the assembler.

Branching

Branch instructions are used to specify a new full-width 16-bit address. A 16-bit value, range 0 to 65,535, is contained in the second word of the instruction or in a register. The value in the second word can be used as the effective branch address or added to the contents of a base register to form an effective address. Refer to "Base Register Storage Address" in this chapter for further information.

Level Switching and Interrupts

The processor can execute programs on four different interrupt priority levels. These levels, listed in priority sequence, are numbered 0, 1, 2, and 3, with level 0 having highest priority. The processor switches from one level to another in two ways:

- Automatically, when an interrupt request is accepted from an I/O device on a higher priority level than the current level.
- Under program control, by using the Set Level Block (SELB) instruction or the LEX instruction.

Both types of level switching are discussed in detail in Chapter 3.

Stack Operations

The processing unit provides two types of stacking facilities. The two types of stacking facilities are:

- *Data stacking.* This facility provides an efficient and simple way to handle last-in first-out queues of data items and/or parameters in main storage. The data items or parameters are called stack elements. For a given queue (or stack), each element is one, two, or four bytes wide. Instructions for each element size (byte, word, or doubleword) are provided to push an element into a stack (register-to-storage) or pop an element from a stack (storage-to-register).
- *Linkage stacking.* This facility provides an easy method for linking subroutines to a calling program. A word stack is used for saving and restoring the status of general registers and for allocating dynamic work areas. The Store Multiple (STM) instruction stores the contents of the registers into the stack and reserves a designated number of bytes in the stack as a work area. The Load Multiple and Branch (LMB) instruction reloads the registers, releases the stack elements, and causes a branch by register 7 back to the calling program.

  Note: The Store Multiple instruction pushes a block of information into a stack. This block is referred to as a register block. The Load Multiple and Branch instruction pops a register block from a stack.

Any contiguous area of main storage can be defined as a stack, and each stack is defined by a stack control block. Figure 2-3 shows a data stack and its associated stack control block. Stack control blocks must be aligned on a word boundary.

The words in the stack control block are used as follows:

High-Limit Address (HLA). This word contains the address of the first byte beyond the area being used for the stack. All data in the stack has a lower address than the contents of the HLA. Note that the HLA points to the first byte beyond the bottom of an empty stack.

Low-Limit Address (LLA). This word designates the lowest storage location that can be used for a stack element. Note that the LLA points to the top of a stack.

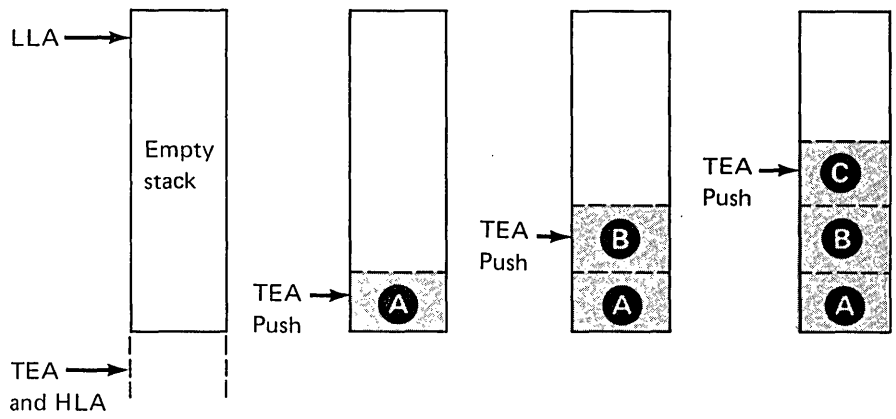
Top-Element Address (TEA). This word points to the stack element that is currently on top of the stack. For empty stacks, the TEA points to the same location as the high limit address (HLA).

Notes:

1. For word, doubleword, and register block operations, the HLA, LLA, and TEA must all contain even addresses to ensure data alignment on a word boundary.
2. The HLA and LLA define a contiguous range of addresses. These addresses must not cross the 64K-byte boundary that causes storage to wrap.

Push Operation. When a new element is pushed into a stack, the address value in the TEA is decremented by the length of the element (one, two, or four bytes) and compared against the LLA. If the TEA is less than the LLA, a stack overflow exists. A soft-exception trap interrupt occurs with stack exception set in the PSW. The TEA is unchanged. If the stack does not overflow, the TEA is updated and the new element is moved to the top location defined by the TEA.

The following diagram shows how elements are pushed into a stack. Note that each push operation always places an element at a lower address in the stack than the preceding element.



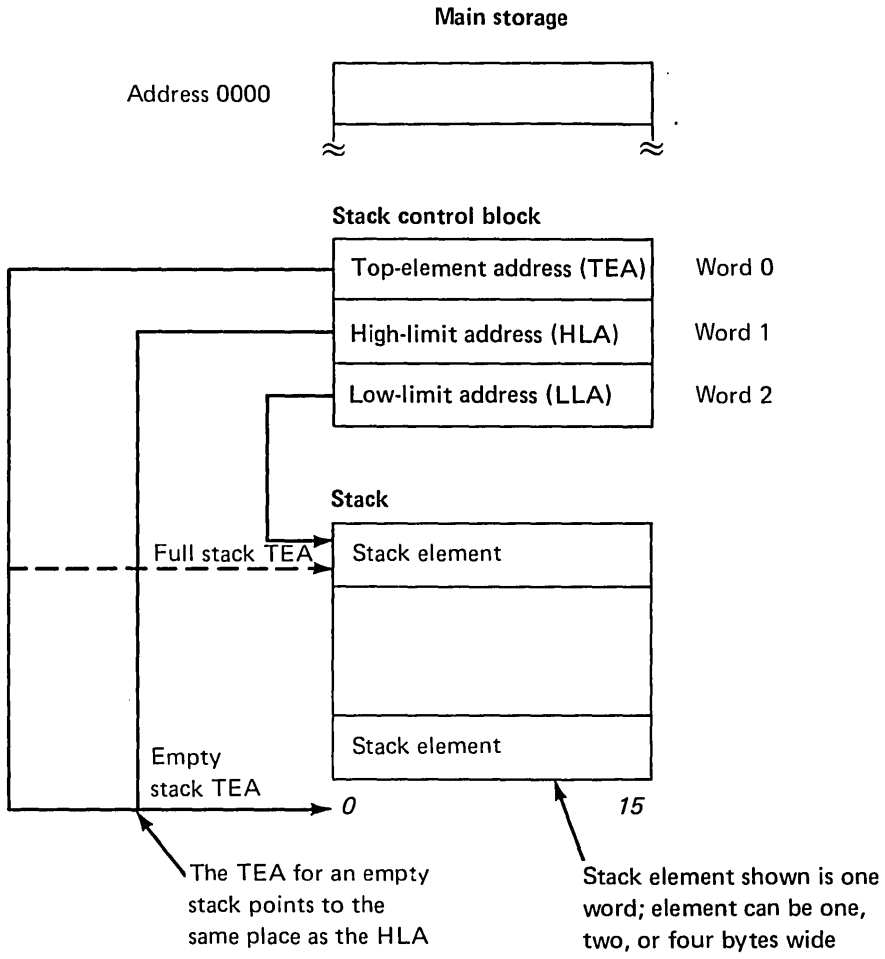


Figure 2-3. Relationship of stack control block to data stack

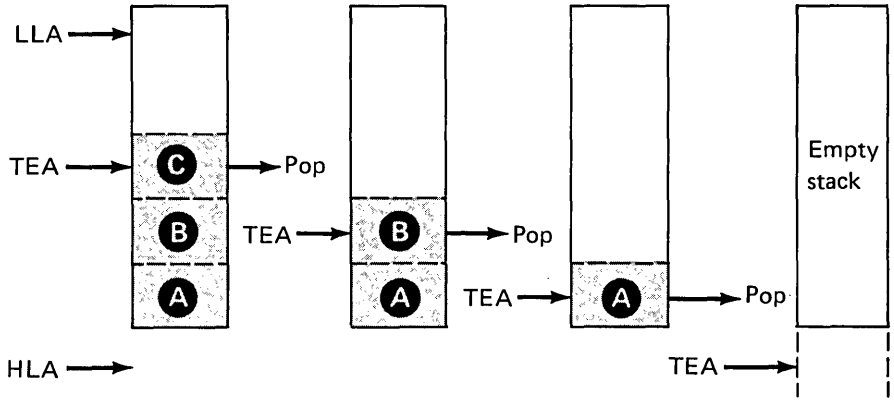
Refer to Chapter 8 for descriptions of the following instructions:

- Push Byte (PSB)
- Push Word (PSW)
- Push Doubleword (PSD)

Note: For a Push Doubleword operation, the TEA points to the high-order word of the doubleword operand.

Pop Operation. When an element is popped from a stack, the TEA is compared against the HLA. If it is equal to or greater than the HLA, an underflow condition exists. A soft exception trap interrupt occurs with stack exception set in the PSW. If the stack does not underflow, the stack element defined by the TEA is moved to the specified register and the TEA is incremented by the length of the element.

The following diagram shows how elements are popped from a stack:



Refer to Chapter 8 for descriptions of the following instructions:

- Pop Byte (PB)
- Pop Word (PW)
- Pop Doubleword (PD)

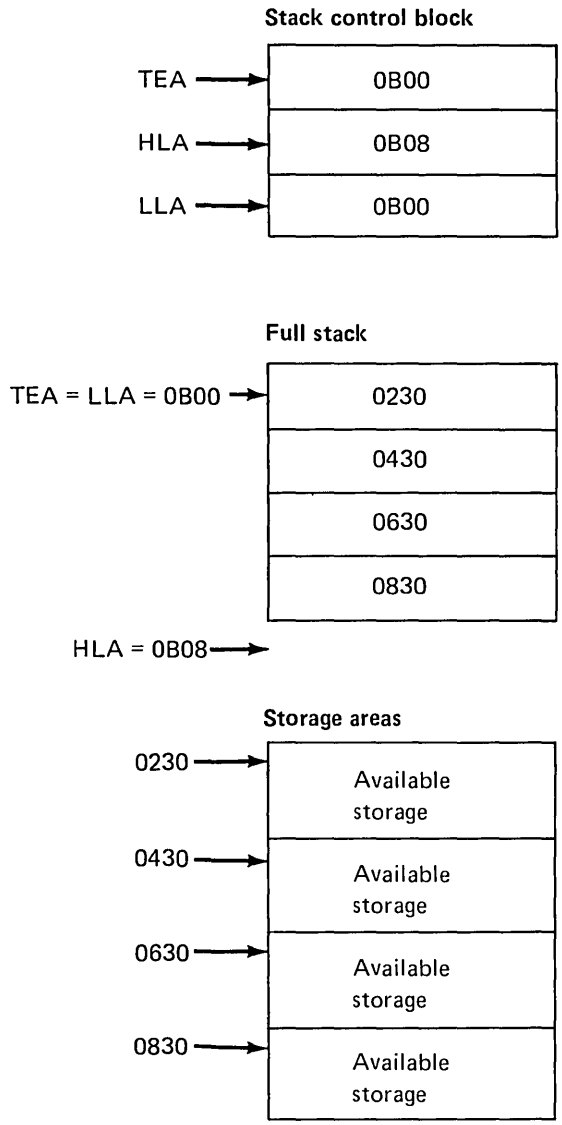
Note: It is possible to pop data from beyond a stack boundary if the TEA is less than the HLA, and the operand size is greater than HLA minus TEA.

Data Stacking Example—Allocating Fixed Storage Areas

Many programs require temporary main storage work areas. It is very useful to be able to dynamically assign such work area storage to a program only when that storage is needed. Conversely, when work-area storage is no longer needed by a program, it is desirable to free that resource so that it may be used by other programs. Use of the stacking mechanism can assist in the programming of the dynamic storage management function.

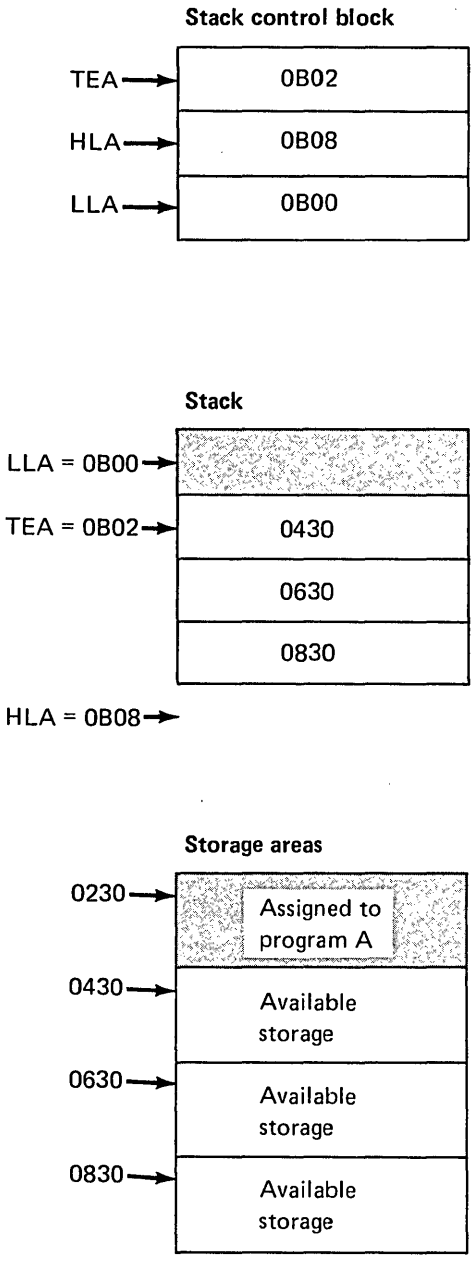
The following is an example of how storage areas could be allocated using the stacking mechanism.

A stack is initialized with addresses that point to fixed areas of storage. Each element in the stack represents the starting address of a block of storage consisting of 512 bytes (for example, addresses 0230 through 03FF). As storage is needed, the starting address for a block or storage is popped from the stack. When the block of storage is no longer needed, the starting address is pushed back into the stack. The stack control block, stack, and storage areas appear initially as follows:



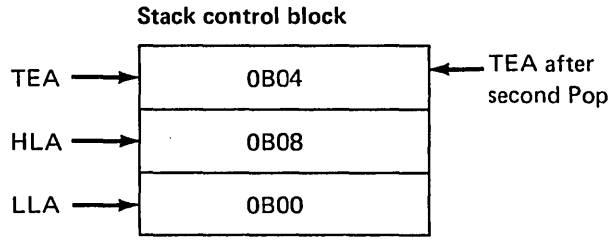
Notice that each stack element is one word long, that addresses of storage areas are the stack elements, and that the TEA points to the lowest location of the last element because the initialized stack is full. Contrast this with an empty stack, in which the TEA points to the same location as the HLA.

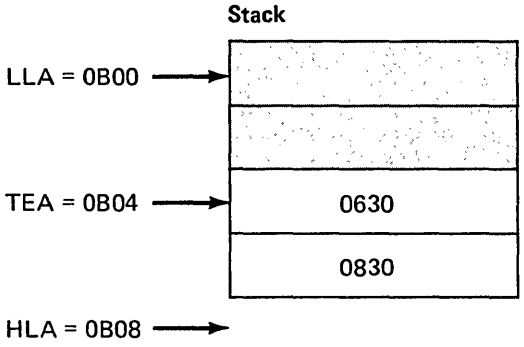
Now assume that program A requires a block of storage. Program A (or a storage management function at the request of program A) issues a Pop Word instruction against the stack control block. The TEA is updated as follows:

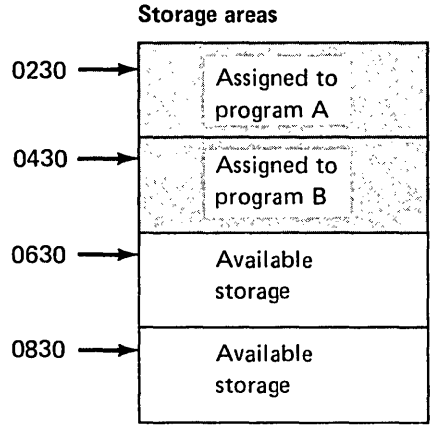


The word element popped is placed in the register specified by the Pop Word instruction executed by program A. This is the address of the 512-byte storage area beginning at address 0230.

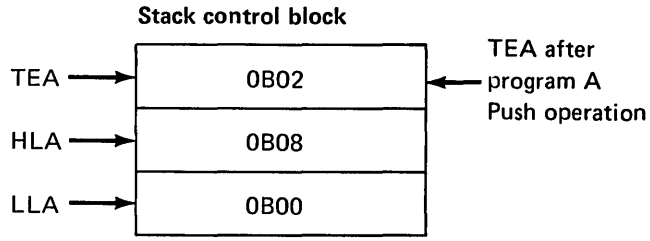
At this time, assume that program B (operating on a different hardware level than program A) also requires a storage area. Program B also executes a Pop Word instruction against the stack. The next element is moved to the register specified and points to the next available storage area; the TEA is updated:

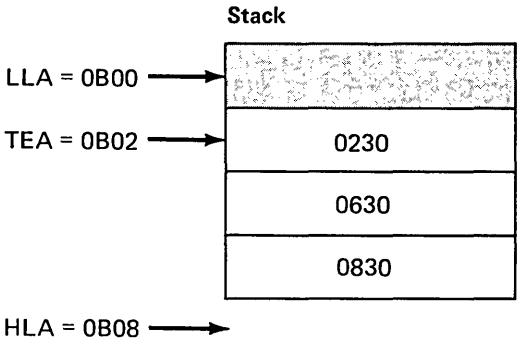


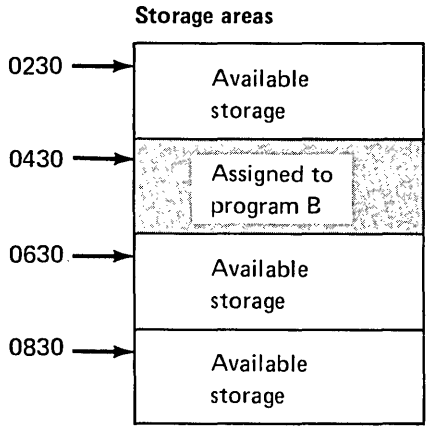




Before any further requests occur, program A terminates its need for a work area. Program A then issues a Push Word instruction against the stack and returns the address of the area it was using so that other programs may use it:







A similar operation is performed by Program B when it releases its storage to the stack, popping address 0400 into location 0B00. While the addresses are obviously shuffled in the stack (from the values initially established), this presents no problem because each program requires only an area of storage—the area location is not important.

Linkage Stacking Description

As previously described, a word-stack mechanism may be used for subroutine linkage. This mechanism saves and restores registers and allocates dynamic work areas.

The letters in the following description correspond to the letters of Figure 2-4.

The Store Multiple (STM) instruction specifies:

- Ⓐ Stack control block address
- Ⓑ Limit register (RL) number
- Ⓒ Number (N) of words to allocate for work areas

When the STM instruction is executed, the allocate value (N), plus the number of registers saved, and one control word is the requested block size in words. The block size (converted to bytes) is used to decrement the TEA before an overflow check is made. If no overflow occurs, the operation proceeds. The link register (register 7) and register 0 through the specified limit register (RL) are saved sequentially in the stack. If register 7 is specified as the limit register, only register 7 is stored in the stack. The dynamic work space is allocated and a pointer to the work area is returned in register RL. If no work area is specified, the returned pointer contains the location of register 7 in the stack. The values of RL and N are also saved as an entry in the stack. The TEA is updated to point to the new top of stack location.

When a Load Multiple and Branch (LMB) instruction is executed, the values of RL and N are retrieved from the stack and an underflow check is made. The value of RL controls the reloading of the registers; the values of RL and N are used to restore the stack pointer (TEA) to its former status. The contents of register 7 are then loaded into the instruction address register, and program control is returned to the calling routine.

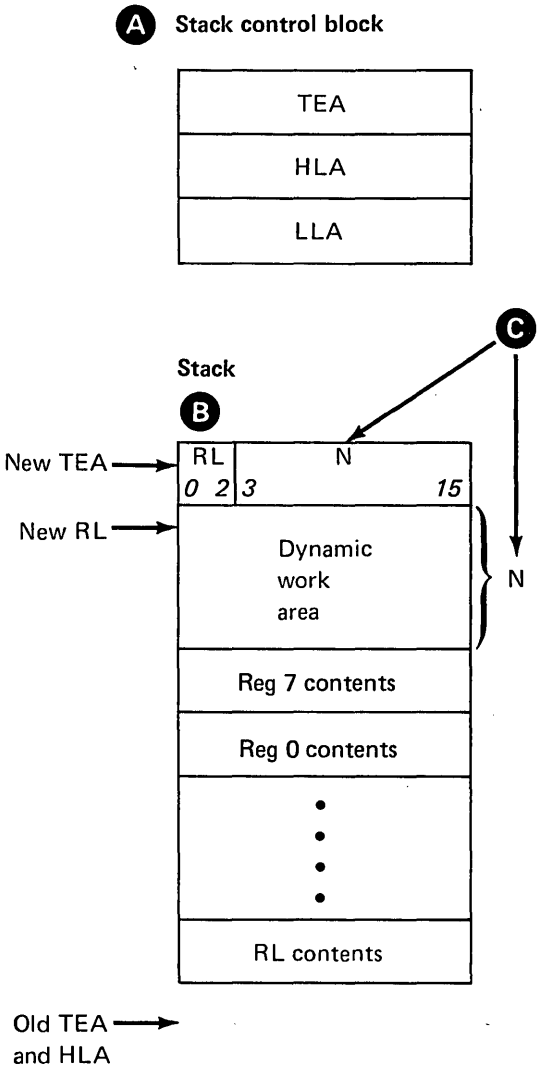


Figure 2-4. Word stack for subroutine linkage

Linkage Stacking Example—Reenterable Subroutine

A subroutine may be used by programs that operate on different interrupt levels. Rather than providing copies of the subroutine (one copy for each program that needs it) the subroutine can be made reenterable. Here, only one copy of the subroutine is provided; the single copy is used by all requesting programs. Two items must be considered in the reenterable subroutine code:

- Saving the register contents of each calling program. The subroutine is then free to use the same registers, restoring their contents to the calling-program's values just prior to returning to the calling program.
- Preserving the applicable variable data (generated by the subroutine) that is related to each call of the subroutine. That is, data associated with one call must not be disturbed when subroutine execution is restarted due to another call from a higher priority program.

The stacking mechanism, by means of the STM and LMB instructions, handles the two items just mentioned. For example, operation could proceed as follows:

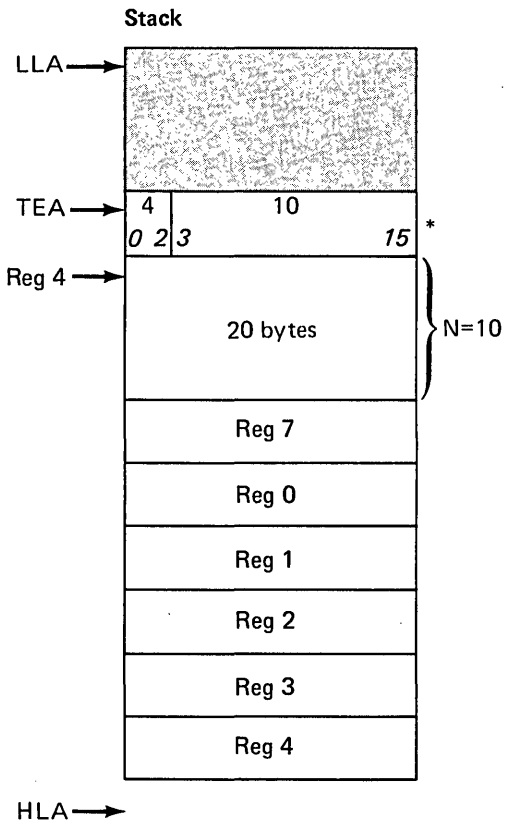
1. Program A calls the subroutine by means of a branch and link instruction (return address is in register 7).

   BAL SUBRT,7

2. The subroutine, in this example, uses register 3 and register 4 during its execution. The subroutine receives (from program A) a parameter list address in register 0 and the address of the stack control block in register 1. Also, the subroutine requires 20 bytes of work space. Thus, the subroutine executes, upon entry, the following store multiple instruction:

   SUBRT STM 4,(1),20

After execution of the STM, the stack contains the following:



*The last word contains a value that specifies the last register stored (Reg 4 in this example) and the size of the dynamic work area (in words).

Register 4 (the last register stored in the stack) is loaded automatically, during the STM operation, with the address of the work area to be used by the subroutine to hold its work data.

3. When subroutine processing for this call is completed, the subroutine executes a single Load Multiple instruction to reload the registers and return (by register 7) to the calling program:

   LMB (1)

   If a second call to the subroutine occurs prior to execution of the LMB, action similar to that just stated occurs again; however, another stack area is used. Then, when subroutine execution is complete for the second call and all higher priority interrupt level processing is complete, a return is made to the interrupted subroutine for completion of processing for the first call.

   Thus, multiple calls to a single subroutine are processed without interfering with the integrity of data associated with any other call to the subroutine.

Chapter 3. Interrupts and Level Switching

Efficient operation of the processor depends on prompt responses to input/output (I/O) service requests. This is accomplished by an interrupt scheme that stops the current processor operation, branches to a device service routine, handles device service, and then returns to continue the interrupted operation. One processor can control many I/O devices; therefore, an interrupt priority is established.

Certain error or exception conditions (such as machine check) also cause interrupts. These are called class interrupts and are processed in a manner similar to I/O interrupts. Both I/O and class interrupts are explained in the following paragraphs.

I/O interrupt priority is established by four priority levels of processing. These levels, listed in priority sequence, are numbered 0, 1, 2, and 3, with level 0 having the highest priority. Interrupt levels are assigned to I/O devices by program control. This provides flexibility for reassigning device priority as the application changes.

Each of the four priority levels has its own set of registers. These consist of an address key register (AKR), a level status register (LSR), eight general registers 0–7, and an instruction address register (IAR). Information pertaining to a level is automatically retained in these hardware registers when an interrupt occurs.

Processor priority level switching, under program control, can be accomplished by using the Set Level Block (SELB) or Level Exit (LEX) instructions. Details of this method are discussed under "Program-Controlled Level Switching" in this chapter.

Fixed locations in main storage are reserved for branch addresses or pointers that are referenced during interrupt processing. I/O and class interrupts cause automatic branching to a service routine. Refer to individual processor publications for storage allocation information.

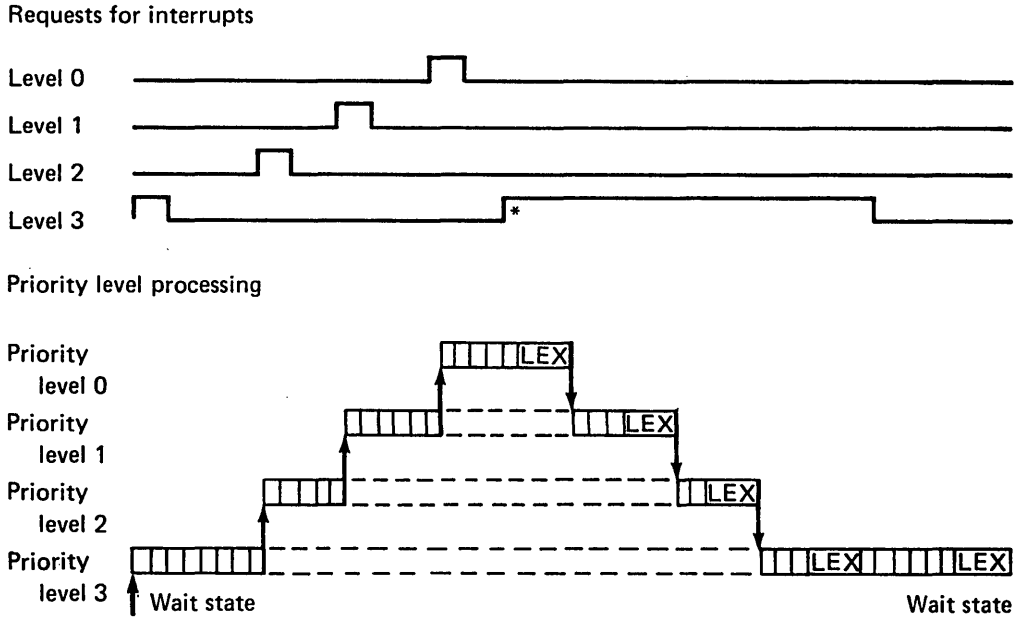
Interrupt masking facilities, to enable or disable interrupts, provide additional program control over the four priority levels. System and level masking are controlled by the summary mask and the interrupt level mask register. Device masking is controlled by a Prepare command in conjunction with an Operate I/O instruction. Manipulation of the mask bits can enable or disable interrupts on all levels, a specific level, or for a specific device. Masking is described under "Interrupt Masking Facilities" in this chapter.

Interrupt Scheme

Each I/O device is assigned to an interrupt level, depending on the application. When an interrupt on a given level is accepted, that level remains active until a Level Exit (LEX) instruction is executed, a Set Level Block (SELB) instruction causes a level switch, or a higher priority interrupt is accepted. In the first two cases, the active level at the time is cleared. In the last case, the processor switches to the higher level, completes execution (including a LEX or SELB instruction), and then automatically returns to the interrupted-from level. This automatic return can be delayed by other higher priority interrupts.

If an interrupt request is pending on the currently active level, it is not accepted until the level is cleared by a LEX or SELB instruction. If no other level of interrupt is pending when a program exits the current level, the processor enters the wait state. In the wait state, no processing is performed, but the processor can accept interrupts that are expected to occur (see Figure 3-1).

Class interrupts take precedence over I/O interrupts and do not change priority levels. Class interrupts are processed on the current active level. If the processor is in the wait state when a class interrupt occurs, priority level 0 is used to process the interrupt.



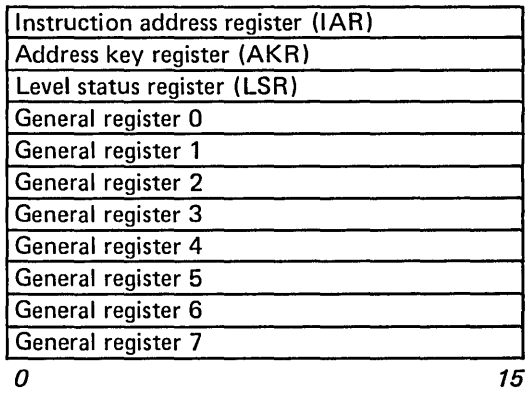
*This interrupt request cannot be honored until the first shown LEX, on priority level 3, has been executed.

Figure 3-1. Interrupt priority scheme

Level Status Block (LSB)

There are two types of LSBs: the hardware LSB and the main storage LSB, both of which are used for task control and switching.

The hardware LSB (one for each level of priority interrupt) consists of 22 bytes of data located in the local storage registers. Although the data bytes are not physically contiguous, they are mapped in a logically contiguous manner, in the following order: IAR, AKR, LSR, and general registers 0–7.



Level status block

The main storage LSB, which is made up of the mapped-hardware LSB at class interrupt time, is utilized for handling class interrupts and SELB instructions.

The main storage LSB, when produced by a class interrupt, contains the current-priority-level hardware LSB at the time of the class interrupt; this hardware LSB data is placed into the correct storage location.

A main storage LSB that is used by a SELB instruction contains data produced prior to the execution of the SELB instruction. When the SELB instruction is executed, the data in the main storage LSB replaces the chosen priority-level hardware LSB's data.

Refer to "Class Interrupts" and "Program-Controlled Level Switching" in this chapter for additional information.

Automatic Interrupt Branching

Automatic interrupt branching to a servicing routine, which provides a unique address in a reserved storage area of main storage, is accomplished by using: an I/O device address, a fixed-class interrupt vector (address), and a fixed-restart vector.

When an interrupt occurs, it is processed by an interrupt algorithm to locate the proper address in this reserved storage area. When that address is located, a branch operation to service the interrupt is begun.

The reserved storage area is located in the first 64K-byte block beginning at address 0000. One word (two bytes) is reserved for each interrupting source. The system interrupts utilize addresses 0000 through 002F (hex), and the device addresses begin at address 0030 (hex). The maximum number of devices that can be attached is 256; therefore, the last address

of this reserved area is 022F (hex) and the last addressable interrupt vector is 022E (hex).

Note: The first word of a device data block must be on an even-byte address.

The reserved storage locations and their assignments are shown in Figure 3-2. Refer to Chapter 5, "Storage Address Relocation Translator," for additional information.

Main storage address (hex)	Contents of word
022E	Device FF DDB pointer
≈	≈
0032	Device 01 DDB pointer
0030	Device 00 DDB pointer
002E	Reserved
002C	Reserved
002A	Reserved
0028	Reserved
0026	Clock SIA
0024	Clock LSB pointer
0022	Soft-exception trap SIA
0020	Soft-exception trap LSB pointer
001E	Console SIA
001C	Console interrupt LSB pointer
001A	Trace SIA
0018	Trace LSB pointer
0016	Power/thermal SIA
0014	Power/thermal LSB pointer
0012	SVC SIA
0010	SVC LSB pointer
000E	Program-check SIA
000C	Program-check LSB pointer
000A	Machine-check SIA
0008	Machine-check LSB pointer
0006	Reserved
0004	Reserved
0002	Restart instruction word 2
0000	Restart instruction word 1

Addresses used for I/O interrupts. The device data block (DDB) pointer is the address of the first word of a device data block. This word is used to obtain the start instruction address for the service routine. Refer to "I/O Interrupts" in this chapter for additional information.

Address used for class interrupts. The level status block (LSB) pointer is the first address of an area where a level status block will be stored. The start instruction address (SIA) points to the first instruction of a service routine.

Restart instruction. Following IPL, a forced branch is made to address 0000.

Note: Device addresses range from 00 through FF hex; the interrupt vector for device 00 hex is main storage address 0030 (hex).

Figure 3-2. Reserved storage locations

I/O Interrupts

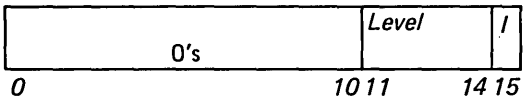
An I/O interrupt is caused by the termination of an I/O operation or by an external event at the I/O device.

Prepare I/O Device for Interrupt

I/O device interrupt parameters are established by program control. The Operate I/O (IO) instruction initiates the device operation and, in conjunction with the Prepare command, sets the device mask (I-bit) and assigns the priority level to use for interrupts. Refer to "Prepare" under "I/O Commands" in Chapter 4 for additional information on the Prepare command.

Refer to Chapter 8, "Instructions," and Chapter 4, "Input/Output Operations," for details of the Operate I/O instruction.

Execution of the Prepare command transfers a word to the addressed device that controls its interrupt parameters. This word has the format:



Bits	Contents
0–10	Set to 0's
11–14	*Level.* A four-bit encoded field that assigns an interrupt-priority level to the device (see Note). *Example:* 0000 – level 0, 0001 – level 1, 0010 – level 2, 0011 – level 3.
15	*Device mask or I-bit.* This bit sets the interrupt mask in the device. When set to 1, the device can interrupt. When set to 0, the device cannot request an interrupt.

Note: Refer to individual device publications for interrupt priority levels.

An interrupting device is always able to accept and execute a Prepare command, even if it is presently busy or has an interrupt request pending from a previous command. This allows the software to change the device mask and interrupt level at any time. Any pending interrupt request is then serviced on the new interrupt level.

Present and Accept I/O Interrupt

The I/O device presents an interrupt request on its assigned priority level. The interrupt request is applied to the interrupt algorithm for acceptance determination.

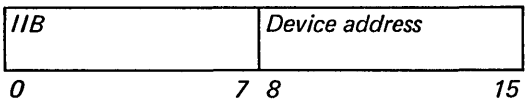
For an I/O interrupt to be serviced, the following conditions must exist:

- The summary mask must be set to 1 (enabled).
- The mask bit (interrupt level mask register) for the interrupting level must be set to 1 (enabled).
- The interrupt request must be the highest priority of the outstanding requests and higher than the current level of the processor.
- The processor must not be in the stop state.
- A class interrupt must not be pending.

Supervisor state is entered upon acceptance of all priority interrupts.

Following acceptance of an I/O interrupt, the device sends the device address and a condition code to the processor. The condition code is placed in the even, carry, and overflow indicators for the interrupted-to level. The device address and the interrupt information byte (IIB) form an interrupt identification (ID) word. The interrupt ID word is placed in register 7 of the interrupted-to level.

Interrupt ID word



Bits 0–7 *Interrupt information byte (IIB).* For interrupt condition codes 2 and 6, the IIB has a special format and is called an interrupt status byte (ISB). For interrupt condition codes reported by a device, the IIB contains:

CC0. The IIB is set to 0.
CC1 or CC5. The IIB contains a DCB identifier.
CC3 or CC7. Bit 0 may be set to 1 if suppress exception is in effect, and an exception has been suppressed. Other bits are device-dependent.
CC4. All bits are device-dependent.

Bits 8–15 *Device address.* This byte contains the address of the interrupting device.

Refer to Chapter 4 for condition codes and interrupt information byte (IIB) details.

For an example of I/O interrupt with automatic branching, refer to the following text and Figure 3-3.

The processor hardware switches from the registers and status of the interrupted-from level to the registers and status of the interrupted-to level Ⓐ. The interrupt ID word is placed in register 7 of the interrupted-to level Ⓑ. The device address is used by hardware to cause a forced branch to the reserved-storage location designated for this interrupting device Ⓒ. Refer to "Automatic Interrupt Branching" in this chapter for additional information.

The location branched to in the reserved storage area contains the device data block (DDB) address pointer (location of this DDB in main storage); this address pointer is placed in register 1 of the interrupted-to level Ⓓ. Hardware forces a branch to the address of the DDB Ⓔ. The first word of the DDB contains the address pointer to the start instruction address (SIA). The SIA pointer is loaded into the interrupted-to level IAR Ⓕ, and execution on the new priority-interrupt level begins Ⓖ.

When the LEX instruction is executed on this operating level and no other higher priority interrupts are pending, the execution of instructions at the interrupted-from level starts automatically Ⓗ.

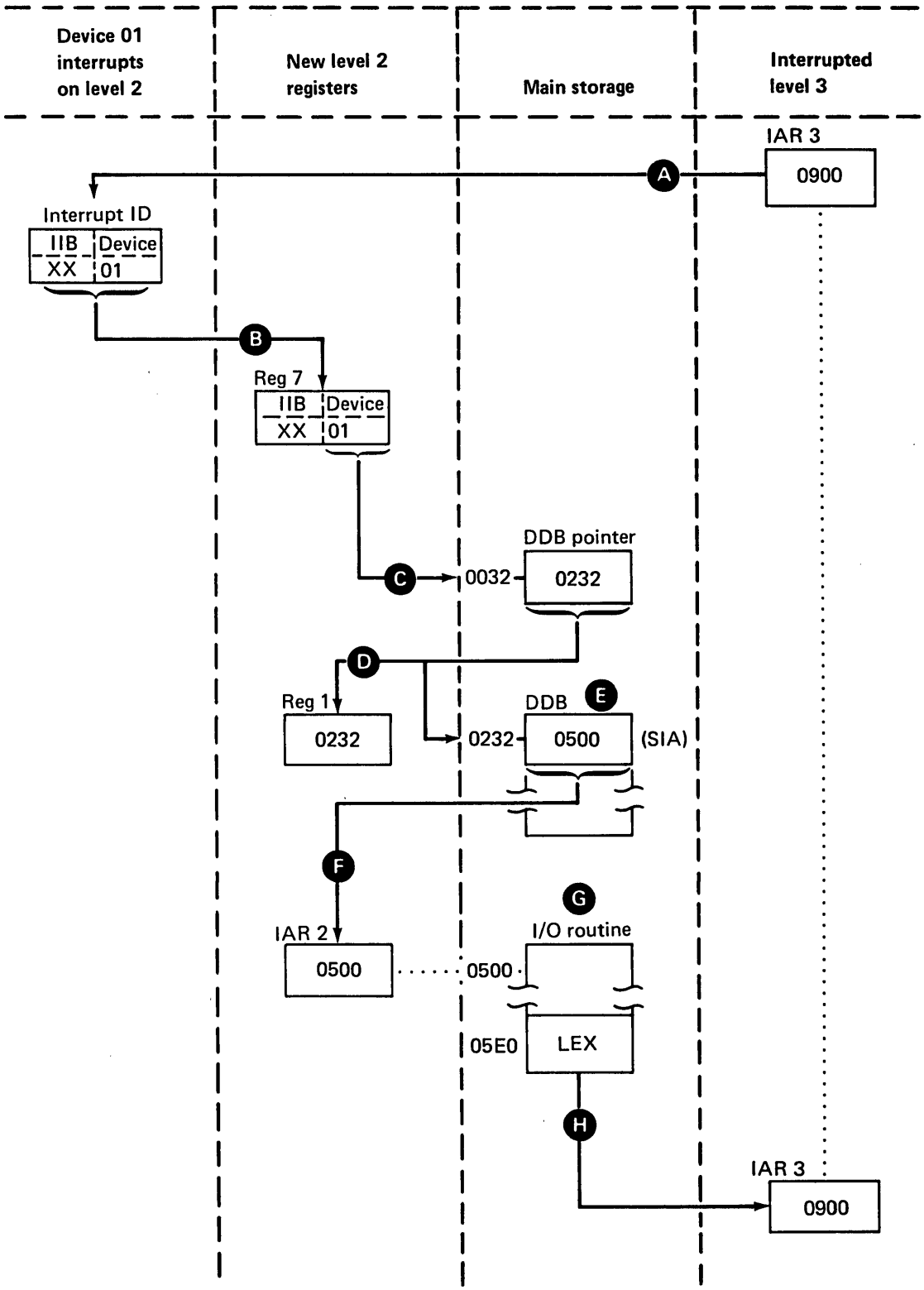


Figure 3-3. Example of I/O interrupt with automatic branching

Class Interrupts

A class interrupt alerts the system to an error or exception condition. Class interrupts utilize a level status block (LSB) scheme to present the identity of the error or exception to the software. Recovery can then occur in a manner that allows normal processing to continue with a minimum of disruption. Class interrupts are processed on the current active level in a priority-order-by-exception condition.

System error or exception conditions can cause eight types of class interrupts:

- Machine check, caused by a hardware error
- Program check, caused by a software error
- Power/thermal warning, caused by a power or temperature irregularity
- Supervisor call, caused by execution of a Supervisor Call (SVC) instruction
- Soft-exception trap, caused by a software error
- Trace, caused by instruction execution (trace enabled in the current LSR)
- Clock, caused by a program-controlled time interval
- Console, caused by pressing the Console Interrupt key when the programmer console is installed

Machine check, program check, soft-exception trap, and power/thermal warning are defined by bits in the processor status word (PSW). Software can refer to the processor status word for a specific condition and any related status information. Refer to "Processor Status Word (PSW)" in this chapter for additional information.

Class interrupts take precedence over I/O interrupts and do not cause a change in priority level. The interrupt is serviced on the level that is active when the condition occurs. If the processor is in the wait state, the interrupt is serviced on priority level 0. Independent routines are used to handle each type of class interrupt regardless of priority level.

All class interrupts cause the processor to enter supervisor state. Refer to "Present and Accept Class Interrupt" in this chapter for details of the hardware processing.

Programming Notes:

1. Three class interrupts (clock, power/thermal warning, and console) are disabled when the summary mask is disabled.
2. If the programmer console is installed and check restart mode is selected, machine-check, power/thermal-warning, and program-check interrupts do not occur. If stop-on-error mode is selected, a stop occurs before a machine-check, power/thermal-warning, or program-check interrupt is serviced.

Refer to individual processor publications for additional information regarding class interrupts.

Priority of Class Interrupts

Although class interrupts are serviced on the current priority level, they are serviced according to an error or exception condition priority.

The following table lists the error or exception conditions in priority sequence, with 0 being the highest priority. Two conditions of the same priority, such as protect check and specification check, may be reported to the PSW simultaneously. Refer to "Processor Status Word (PSW)" in this chapter for PSW-bit meanings. The table also shows the associated types of class interrupt exception conditions.

Priority	Error or exception condition	Type of class interrupt
0	CPU control check I/O check	Machine check
1	Invalid function (Note 1)	Program check
2	Privilege violate	
3	Invalid function	
4	Protect check Specification check	
5	Invalid storage address Specification check	
6	Storage parity	Machine check
7	Power warning Thermal warning	Power/thermal warning
8	Supervisor call	Supervisor call
9	Invalid function (Note 2)	Soft-exception trap
10	Floating-point exception	
11	Stack exception	
12	Trace	Trace
13	Clock	Clock
14	Console	Console

Notes:

1. Caused by an illegal operation or function combination.
2. A floating-point instruction is attempted and floating-point is not installed.

Present and Accept Class Interrupt

When a class interrupt occurs, it is serviced on the currently active level or, if the processor is in the wait state, priority level 0 is forced active. The interrupt causes the following to occur:

- Register contents are saved.
- Supervisor state is entered (LSR bit 8 is set to 1).
- Trace is reset (LSR bit 10 is set to 0).
- Summary mask is disabled (LSR bit 11 is set to 0).
- The address key register is set to a predetermined value, depending on the type of class interrupt.
- An automatic branch is taken to the reserved area of main storage.

Each type of class interrupt has an associated LSB pointer and SIA in the reserved area of main storage (refer to Figure 3-2). Reference is made to the reserved area to:

- Store current level IAR, AKR, general registers, and LSR into a level status block (LSB) in main storage.
- Branch automatically to a service routine by using the start instruction address (SIA).

Priority level 0 is forced active when a class interrupt occurs in the wait state. The level 0 hardware LSB is stored into main storage. The in-process bit (LSR bit 9) is set to 0 in the stored LSB.

The operand 1 key (OP1K) address key value is set in anticipation of the address spaces required by the interrupt service routine.

Contents of the main storage level status block are as follows:

Main storage address (LSB) pointer	
	Instruction address register (IAR)
	Address key register (AKR)
	Level status register (LSR)
	General register 0
	General register 1
	General register 2
	General register 3
	General register 4
	General register 5
	General register 6
+14 (hex)	General register 7
	0 *15*

The instruction address (contents of IAR) stored in the LSB depends on the type of class interrupt, as shown in the following chart:

Type of class interrupt	Contents of IAR (stored in LSB)
Program check Soft-exception trap	Address of the instruction that caused the interrupt
Supervisor call Trace Clock Console Power/thermal warning	Address of the next instruction
Machine check (with sequence indicator off)	Address of the instruction that caused the interrupt
Machine check (with sequence indicator on)	Address of the instruction that was being executed at the time of the error

Machine Check

A machine-check class interrupt is caused by a hardware malfunction and is considered a system-wide incident. There are three machine-check class interrupts.

- Storage parity check (PSW bit 8)
- CPU control check (PSW bit 10)
- I/O check (PSW bit 11)

A level status block is stored, starting at the location in main storage designated by the machine check LSB pointer. The contents of the storage address register (SAR) are loaded into register 7. The last active processor address key is placed into the operand 1 key (OP1K) address key of the AKR; then, operand 2 key (OP2K), equate operand spaces (EOS) bit, and instruction space key (ISK) are set to 0's. The machine check SIA is loaded into the IAR, and it becomes the address of the next instruction to be fetched.

Note: When the error condition occurs:

1. The IAR contains the true address of the first word of the instruction; it is not incremented if the error occurs in the second or third word of a long instruction.
2. For a storage parity check, the last active processor address key defines the address space corresponding to the storage address loaded into register 7. For a CPU control check or an I/O check, this address key and register 7 provide no useful information.

Program Check

A program-check class interrupt is caused by a software error. If a program-check class interrupt occurs, PSW bit 0, 1, 2, 3, or 4 is set to 1. There are five program-check class interrupts.

- *Specification check (PSW bit 0)*—A specification check occurs when the storage address violates the boundary requirements. The instruction is suppressed unless otherwise noted in the individual instruction description in Chapter 8.
- *Invalid storage address (PSW bit 1)*—An invalid storage address occurs when one or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is suppressed unless otherwise noted in the individual instruction description in Chapter 8.
- *Privilege violate (PSW bit 2)*—Privilege violate occurs when a privileged instruction is encountered while the processor is in the problem state. The instruction is suppressed.
- *Protect check (PSW bit 3)*—Protect check occurs when the processor is in the problem state and an instruction or data is accessed from a storage area not assigned to the current operation, or an attempt is made to change an operand in a storage area assigned as read-only. The instruction is suppressed unless otherwise noted in the individual instruction description in Chapter 8.
- *Invalid function (PSW bit 4)*—Invalid function occurs when an illegal operation code or function combination is encountered during instruction execution. The instruction is suppressed unless otherwise noted in the individual instruction description in Chapter 8.

A level status block is stored, starting at the location in main storage designated by the program check LSB pointer. The contents of the storage address register (SAR) are loaded into register 7. The last active processor address key is placed into the OP1K address key of the AKR; then, OP2K, EOS bit, and ISK are set to 0's. The program check SIA is loaded into the IAR, and it becomes the address of the next instruction to be fetched.

Notes:

1. A program-check class interrupt condition on one priority level does not affect software on other levels.
2. For a specification check, an invalid storage address, and a protect check, the last active processor address key defines the address space corresponding to the storage address loaded into register 7. For privilege violate and invalid function, this address key and register 7 provide no useful information.

Power/Thermal Warning

A power/thermal-warning class interrupt occurs when PSW bit 15 is set to 1. A power/thermal-warning class interrupt is initiated by:

- A power-warning signal that is generated when the power line decreases to about 85% of its rated value.
- A thermal-warning signal that is generated when the temperature limits inside the enclosure are exceeded.

In both cases, the instruction address that is stored in the LSB points to the next instruction to be executed.

A level status block is stored, starting at the location in main storage designated by the power/thermal LSB pointer. The EOS bit and all address keys in the AKR are set to 0's. The power/thermal SIA is loaded into the IAR, and it becomes the address of the next instruction to be fetched.

A power/thermal-warning class interrupt can occur when the system in the run or wait state, assuming that the summary mask is enabled and the programmer console is not in check restart mode. A power/thermal-warning interrupt is accepted by the processor only if both conditions are met.

If the optional battery backup unit is installed and a power warning occurs, PSW bit 15 remains on as long as power is supplied by the battery. If a thermal warning occurs, the processor powers down regardless of the battery backup unit. The minimum time before the processor powers down is 20 milliseconds. The IBM 4999 Battery Backup Unit is explained in a separate publication, *IBM Series/1 4999 Battery Backup Unit Description*, GA34-0032. Power/thermal-warning class interrupts are not accepted by the processor until the first instruction is executed following a power-on reset, an IPL, or exit from stop state.

Note: If the processor is in the wait state when the power/thermal condition occurs:

1. The interrupt is serviced on priority level 0. The level 0 LSB is stored into main storage. Additional power/thermal interrupts, along with priority interrupts, are disabled at this time because the summary mask is set to 0 by the class interrupt.
2. The instruction address stored in the LSB is unpredictable.

Supervisor Call

A supervisor-call class interrupt is initiated by executing an SVC instruction. The SVC instruction is described in Chapter 8.

A level status block is stored, starting at the main storage location designated by the supervisor call LSB pointer. The OP2K address key is placed into the OP1K address key in the AKR; then, OP2K, EOS bit, and ISK are set to 0's. The supervisor call SIA is loaded into the IAR, and it becomes the address of the next instruction to be fetched.

Soft-Exception Trap

A soft-exception-trap class interrupt is caused by a software error. A soft-exception-trap class interrupt occurs when bit 4, 5, or 6 of the PSW is set to 1. There are three soft-exception-trap class interrupts.

- *Invalid function (PSW bit 4)*—Invalid function occurs when a floating-point instruction attempts execution and floating-point is not installed. The register-to-register instructions are suppressed; the storage-to-register instructions are terminated.
- *Floating-point exception (PSW bit 5)*—When floating-point is installed, a floating-point exception occurs when an arithmetic error condition is detected. The instruction completes execution.
- *Stack exception (PSW bit 6)*—A stack exception occurs when an instruction attempts to pop an operand from an empty stack or push an operand into a full stack. The instruction is suppressed.

These exception conditions may be handled by software; therefore, they do not constitute an error condition.

A level status block is stored, starting at the location in main storage designated by the soft-exception-trap LSB pointer. The contents of the storage address register (SAR) are loaded into register 7. The OP2K address key is placed into the OP1K address key in the AKR; then, OP2K, EOS bit, and ISK are set to 0's. The soft-exception-trap SIA is loaded into the IAR, and it becomes the address of the next instruction to be fetched.

Trace

The trace class interrupt provides instruction tracking for software debugging. Instruction tracing can occur on any priority level, and is enabled by the trace bit (LSR bit 10). The tracing occurs when bit 10 of the current LSR is set to 1. When trace is enabled, a trace class interrupt occurs at the beginning of each instruction.

A level status block is stored, starting at the location in main storage designated by the trace LSB pointer. The ISK address key is placed into the OP1K address key in the AKR; then, OP2K, EOS bit, and ISK are set to 0's. The trace SIA is loaded into the IAR, and it becomes the address of the next instruction to be fetched.

Note: After the LSB is stored, and before the next instruction is fetched, supervisor state (LSR bit 8) is set to 1 (on), trace (LSR bit 10) is set to 0 (off), and the summary mask (LSR bit 11) is set to 0 (disabled).

Programming Note: When trace is enabled, a trace class interrupt occurs prior to executing each instruction. Hardware processing of the interrupt provides an automatic branch to the programmer's trace routine. To prevent retracing the same instruction, the program exits the trace routine by using the Set Level Block (SELB) instruction with the specified inhibit trace (IT) bit set to 1. The inhibit trace bit prevents a trace interrupt from occurring for the duration of one instruction. Refer to "Set Level Block (SELB)" in Chapter 8 for additional information. A double trace of an instruction can also occur when the instruction is interrupted and must be reexecuted. For example, a class interrupt occurs during execution of a variable-field-length instruction. Under this condition, exit from the class

interrupt routine should be by a SELB instruction with the inhibit trace bit set to 1.

The occurrence of any class interrupt or priority interrupt causes the trace bit (LSR bit 10) to be set to 0. This action permits tracing only problem state code. If it is desired to trace supervisor code, the programmer must make provisions within the service routine to enable the trace bit.

The following three conditions inhibit a trace class interrupt:

1. A SELB instruction sets the trace bit to 1 and the in-process bit to 1 in the LSR of a selected level lower than the current level; then, when the selected level becomes active, the first instruction executed is not preceded by a trace interrupt.
2. The programmer console is in diagnostic mode and a Stop instruction is encountered while tracing; then, when the Start key is pressed, a trace interrupt does not occur prior to executing the first instruction.
3. When a level is exited by either a LEX or a SELB instruction and processing is to continue on a pending level, one instruction is executed on the pending level prior to sampling for a trace class interrupt.

Clock

If the clock value is greater than or equal to the value in the comparator and the ability to interrupt has been enabled (by a Set Comparator instruction), a clock class interrupt occurs.

A clock class interrupt is recognized by the processor only when the processor is in run or wait state and when the summary mask is enabled.

If a clock class interrupt condition occurs when the summary mask is disabled, the interrupt is held pending until the summary mask is enabled.

The ability to interrupt is disabled by power-on reset or system reset. To restore the ability to interrupt, a Set Comparator instruction must be executed.

When a clock class interrupt occurs:

- Further clock interrupts are blocked. The processor stores the current LSB at the storage location defined by the storage address in main storage.
- The processor enters supervisor state.
- The in-process bit is set to 1.
- The trace bit is set to 0.
- The summary mask bit is set to 0.
- The AKR is set to 0.
- The PSW is unchanged.
- The processor resumes execution at the storage location defined by the storage address, which contains the clock start instruction address (SIA).

If the processor is in wait state when a clock interrupt condition occurs, it forces level 0 active. The values stored in the LSB are the residual values in the appropriate registers for level 0. The in-process bit (LSR bit 9) is set to 0 in the stored LSB.

Console

A console class interrupt function is provided when the programmer console is installed. To recognize the interrupt, the processor must have the summary mask enabled and be in the run state or wait state.

A level status block is stored, starting at the main storage location designated by the console interrupt LSB pointer. The EOS bit and all address keys are set to 0's. The console SIA is loaded into the IAR, and it becomes the address of the next instruction to be fetched.

Notes:

1. If the processor is in the wait state when a console class interrupt occurs, the interrupt is serviced on priority level 0.
2. If the summary mask is disabled, the console class interrupt is ignored because it is not buffered.

Recovery Procedures for Class Interrupts

Recovery procedures, initiated by software, depend on the application involved, the type of error or interrupt, and the number of recommended retries.

The class interrupt provides an automatic branch to a service routine. This routine can interrogate the PSW for specific information, and can then initiate the required action. If an error occurs during a priority interrupt sequence, the priority level switch is completed before the class interrupt is processed. This facilitates automatic register retention. A reset is generated by machine-check class interrupts caused by an I/O check or a CPU control check. A reset is not generated by program-check or power/thermal-warning class interrupts.

Machine Check

Storage Parity Check. A storage parity check initiates a machine-check class interrupt. The error may occur when accessing a storage location that has not been validated since power on. Any retry procedure should include refreshing data in the failing location. Two unsuccessful retries are considered a permanent failure, and the storage location should not be used.

CPU Control Check. A CPU control check, which occurs if hardware detects a malfunction of the processor controls, is a machine-wide error that initiates a machine-check class interrupt. A reset is generated to the channel, the I/O attachment features, and all attached I/O devices. The processor, sensor-based output points, and timer values are not reset. The generated reset should clear the error condition, but validity of any previous execution is not guaranteed. A retry is not recommended, and an IPL should be initiated.

I/O Check. An I/O-check condition occurs when a hardware error prevents further communication with I/O devices. A machine-check class interrupt is initiated and a reset is generated to the I/O attachment features, the channel, and all I/O devices. Error recovery from an I/O check depends on the sequence indicator setting (PSW bit 12).

If the sequence indicator is set to 0, the error occurred during an Operate I/O instruction. The address of the failing instruction (IAR contents) is available in the stored LSB. Retry should be attempted twice. After two unsuccessful retries, use of the device should be discontinued.

If the sequence indicator is set to 1, the error occurred during an interrupt or cycle-steal operation. The instruction address (IAR contents) stored in the LSB is not related to the error. The sequence of events leading to the I/O check is lost, along with all pending interrupt requests within the devices. Retry is not recommended.

Program Check

A program check is caused by a software error and initiates a program-check class interrupt. Error retry depends on the application. All necessary parameters are made available for locating and, if required, correcting the invalid condition. The priority level and operands are not changed during a program check class interrupt. The stored LSB reflects conditions at the time the interrupt occurred and contains:

- The address of the failing instruction (IAR contents).
- Status information (AKR and LSR contents).
- The contents of all general registers.

The contents of the storage address register (SAR) are loaded into register 7, but have meaning only for specification check, invalid storage address, and protect check. The programmer must reference the PSW to determine the type of program check.

Power/Thermal Warning

When a power/thermal-warning class interrupt occurs, the minimum time before the processor powers down is 20 milliseconds. If the optional battery backup unit is installed and a power warning occurs, PSW bit 15 remains set to 1 as long as power is supplied by the battery. If a thermal warning occurs, the processor powers down regardless of the battery backup unit.

Supervisor Call

The supervisor-call class interrupt is used to place the processor in supervisor state to allow execution of privileged instructions. This interrupt is not an error; therefore, there is no recovery procedure.

Soft-Exception Trap

A soft-exception-trap interrupt is the result of an exception condition that software may choose to handle dynamically. All necessary parameters are available to locate and correct the condition. The address of the instruction (IAR contents) causing the exception is retained in the level status block in main storage. The processor is not reset. The programmer must reference the PSW to determine the soft-exception type.

Trace

The trace class interrupt is a programming tool used to trace errors. This is a normal operation and there is no recovery procedure.

Clock

The clock class interrupt is a programming tool used for specific functions. It is a normal operation and there is no recovery procedure. After the interrupt, the processor resumes execution at the storage location defined by the clock SIA.

Console

The console class interrupt is a programming tool used for programmed applications and problem determination. When a console class interrupt is recognized by the processor, the console interrupt SIA is loaded into the IAR, and it becomes the address of the next instruction to be fetched.

Processor Status Word (PSW)

The processor status word (PSW) is used to record error or exception conditions, in the system, that may prevent further processing. It also contains certain status flags related to error recovery. Error or exception conditions recorded in the PSW cause one of four class interrupts to occur: machine check, program check, soft-exception trap, or power/thermal warning. Refer to "Class Interrupts" in this chapter for additional information.

The Copy Processor Status and Reset (CPPSR) instruction can be used to examine the PSW. This instruction stores the contents of the PSW at a specified location in main storage.

The PSW is contained in a 16-bit register and has the following bit representation:

Bit	Error or exception condition	Class interrupt	Remarks
0	Specification check	Program check	
1	Invalid storage address	Program check	
2	Privilege violate	Program check	
3	Protect check	Program check	
4	Invalid function	Program check or soft-exception trap	
5	Floating-point	Soft-exception trap	Note 1
6	Stack exception	Soft exception trap	
7	Not used		Always 0
8	Storage parity check	Machine check	
9	Not used		Always 0
10	CPU control check	Machine check	
11	I/O check	Machine check	
12	Sequence indicator	None	Status flag
13	Auto-IPL	None	Status flag
14	Translator enabled	None	Note 1
15	Power/thermal warning	Power/thermal	Note 2

Notes:

1. Refer to individual processor publications for further information.
2. The power/thermal-warning class interrupt is controlled by the summary mask.

Bit 0—Specification Check. This bit is set to 1 if the storage address violates the boundary requirements of the specified data type.

Bit 1—Invalid Storage Address. This bit is set to 1 when an attempt is made to access a storage address outside the storage size of the system. This can occur on an instruction fetch, an operand fetch, or an operand store.

Bit 2—Privilege Violate. This bit is set to 1 when a privileged instruction is attempted in the problem state. Supervisor state bit (LSR bit 8) is set to 0 (off).

Bit 3—Protect Check. In the problem state, an attempt is made to alter storage using a segmentation register with the read-only bit (bit 14) set to 1 and the address translator enabled.

A program-check class interrupt occurs with protect check (bit 3) set to 1 in the PSW.

Bit 4—Invalid Function. This bit is set to 1 by one of the following conditions:

1. Attempted execution of an invalid operation code or function combination. These are:

Op code	Function field bits
00101	All (when register 7 is specified in the R1 or R2 field of the instruction)
00111	All
01000	0001, 0010, 0011, 0101, 0110, 0111
01011	0101, 0111
01100	111
01110	11000, 11010, 11011, 11100, 11110, 11111
01111	1X11X, 01XXX, 1X011, 10001
11011	All
10110	All
11101	1100, 1101, 1110, 1111

   Note: The preceding invalid conditions cause a program-check class interrupt to occur.

2. The processor attempts to execute an instruction associated with a feature that is not installed. These are:

Op code	Function field bits
00100	All
01011	0011, 1011 (when in supervisor state)

   Note: The preceding conditions cause a soft-exception-trap class interrupt to occur.

Bit 5—Floating-Point. This bit is set to 1 when an arithmetic error condition is detected.

Bit 06—Stack Exception. This bit is set to 1 when an attempt has been made to pop an operand from an empty main storage stack or to push an operand into a full main storage stack. A stack exception also occurs when the stack cannot contain the number of words to be stored by a Store Multiple (STM) instruction.

Bit 7—Not Used. This bit is always 0.

Bit 8—Storage Parity. This bit is set to 1 when a parity error has been detected on data being read out of storage by the processor. This error can occur when accessing a storage location that has not been validated since power on.

Bit 9—Not Used. This bit is always 0.

Bit 10—CPU Control Check. This bit is set to 1 to indicate a malfunction of the CPU controls. This is a machine-wide error. (Refer to the Note under "Bit 11—I/O Check.")

Bit 11—I/O Check. This bit is set to 1 when a hardware error that may prevent further communication with any I/O device occurs on the I/O channel.

PSW bit 12 (sequence indicator) is used in conjunction with PSW bit 11 (I/O check) to further define the last I/O sequence before an I/O check condition.

Note: The machine-check class interrupt initiated by a CPU control check or an I/O check causes a reset. The I/O channel and all devices in the system are reset as if a Halt I/O (channel-directed command) had been executed. The processor, sensor-based output points, and timer values are not reset.

Bit 12— Sequence Indicator. This bit reflects the last I/O operation or sequence to occur.

PSW bit 12 (sequence indicator) is set to 0 if the error occurred during an Operate I/O instruction and is set to 1 if the error occurred during a cycle-steal operation or an interrupt-accept sequence. The sequence indicator bit is not an error in itself, but it reflects the last operation or sequence at any time. An I/O check cannot be caused by a software error.

Refer to "Bit 11—I/O Check" previously explained.

Bit 13—Auto IPL. This bit is set to 0 by:

- A power-on reset when auto-IPL mode is not selected.
- Pressing the Load key.
- An IPL initiated by a host system.

Refer to "Initial Program Load (IPL)" in Chapter 2 for additional information.

This bit is set to 1 when an automatic IPL occurs.

Bit 14—Translator Enabled. This bit is set to 0 when:

- A Disable (DIS) instruction is executed with bit 14 of the instruction word set to 1.
- An Enable (EN) instruction is executed with bit 12 of the instruction word set to 1.
- A processor reset (power-on reset, check restart, IPL, or programmer console system Reset key) occurs.

This bit is set to 1 when:

- An Enable (EN) instruction is executed with bit 12 of the instruction word set to 0 and bit 14 set to 1.

Bit 15—Power Warning and Thermal Warning. This bit is set to 1 when a power failure is imminent, or when a thermal condition causes the power to go off. Refer to "Power/Thermal Warning" under "Present and Accept Class Interrupts" in this chapter for additional information. The power/thermal-warning class interrupt is controlled by the summary mask.

Interrupt Masking Facilities

Three levels of priority interrupt masking are provided to the programmer for the control of interrupt processing:

- Summary mask (LSR bit 11)
- Interrupt level mask register
- Device mask (I-bit)

Each masking facility has specific control, as explained in the following paragraphs.

Summary Mask

The summary mask provides a masking facility for priority interrupts and certain class interrupts. The state of the summary mask (enabled or disabled) is controlled by bit 11 in the level status register (LSR) of the active priority level. When bit 11 is set to 0, the summary mask is disabled and prevents *all* priority interrupts regardless of priority level, and prevents power/thermal, clock, and console class interrupts. All other class interrupts are not masked. When bit 11 is set to 1, the mask is enabled and the interrupts are allowed.

The summary mask is disabled (set to 0) by:

- Execution of a Supervisor Call (SVC) instruction.
- Execution of a Disable (DIS) instruction, with bit 15 of the instruction set to 1.
- Occurrence of a class interrupt.
- Execution of a Set Level Block (SELB) instruction with bit 11 of the LSR set to 0.

The summary mask bit is enabled (set to 1) by:

- Execution of an Enable (EN) instruction, with bit 15 of the instruction set to 1.
- Execution of a Set Level Block (SELB) instruction with bit 11 of the LSR set to 1.
- Acceptance of a priority interrupt on the interrupted-to level.
- System reset, power-on reset, or IPL.

Note: If the processor is in the wait state, the summary mask is enabled or disabled as defined by bit 11 in the LSR of the last active priority level.

Interrupt Level Mask Register

The interrupt level mask register is a four-bit register used to control interrupts on specific priority levels. Each level is controlled by a separate bit of the mask register, as shown here:

Interrupt level mask register

With a bit position set to 1, the corresponding priority level is enabled and permits interrupts. With a bit position set to 0, the corresponding priority level is disabled. The Set Interrupt Mask Register (SEIMR) instruction is used to control bit settings in the interrupt level mask register. The Copy Interrupt Mask Register (CPIMR) instruction may be used to interrogate the register.

Note: All levels are enabled (set to 1) by a power-on reset, IPL, or programmer console system Reset key.

Device Mask (I-Bit)

Each interrupting device contains a one-bit mask called the device mask or interrupt bit (I-bit). Interrupts by the device are permitted when its device mask is enabled (set to 1). With the device mask bit disabled (set to 0), that device cannot cause an interrupt. The device mask is controlled by a Prepare command in conjunction with an Operate I/O instruction. Refer to Chapter 8, "Instructions," and Chapter 4, "Input/Output Operations," for additional information.

Program-Controlled Level Switching

Level switching under program control may be accomplished by using the Set Level Block (SELB) instruction. This instruction is described in detail in Chapter 8, "Instructions." In general, this instruction:

- Specifies the location of a level status block (LSB) at an effective address in main storage.
- Specifies a selected priority level associated with the main storage LSB.
- Loads the main storage LSB into the hardware LSB for the selected level.

The hardware LSB consists of the following hardware registers for the selected level:

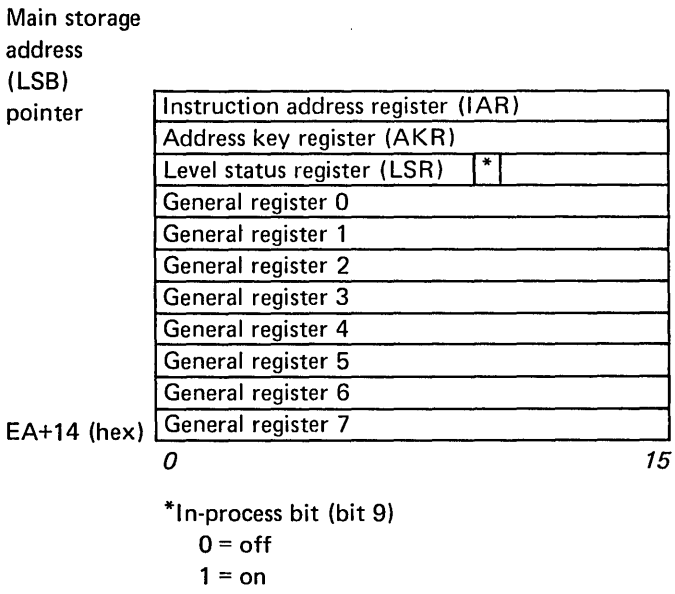
- Instruction address register
- Address key register
- Level status register
- Eight general registers (0–7)

System programmers should be familiar with the execution of the SELB instruction, in order to prevent adverse effects within the programming system.

- The current execution level
- The selected level specified in the SELB instruction
- The state of the in-process bit (LSR bit 9) contained in the main storage LSB

Note: Interrupt masking, provided by the summary mask and the interrupt level mask register, does not apply to program-controlled level switching.

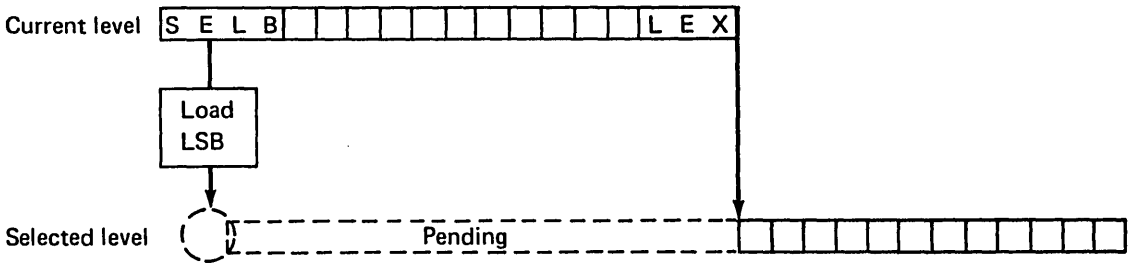
The main storage LSB and the location of the in-process bit are shown in the following diagram:



Execution of the SELB instruction may result in level switching or a change in the pending status of a level as described in the following paragraphs.

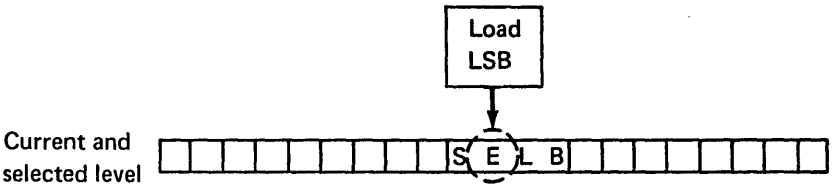
Selected Level Lower Than Current Level and In-Process Bit On

These conditions cause the selected level to become pending. The main storage LSB is loaded into the hardware LSB for the selected level. Execution of a LEX instruction on the current level causes the selected level to become active, provided that no higher priority interrupts are being requested.



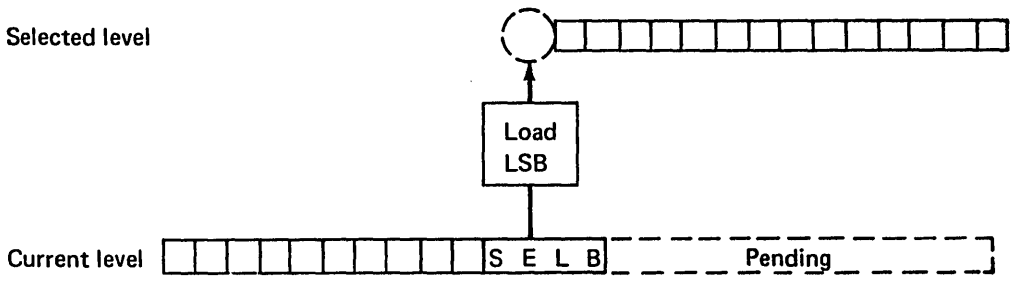
Selected Level Equal to Current Level and In-Process Bit On

These conditions cause the selected level to become the current level. The main storage LSB is loaded into the hardware LSB for the selected level. The effect is a task-switch on the current level, with no level change.



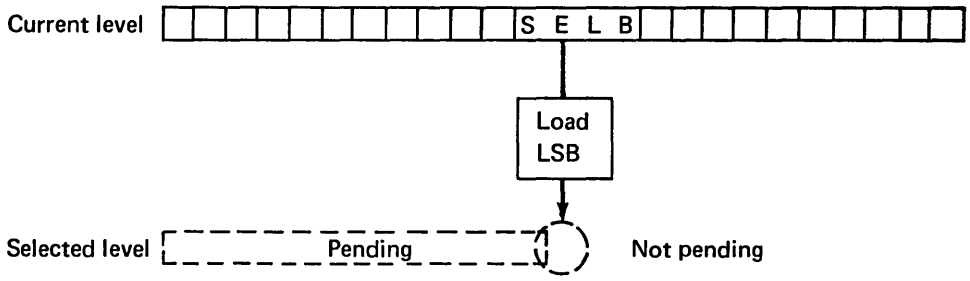
Selected Level Higher Than Current Level and In-Process Bit On

These conditions cause the selected level to become the current level. The main storage LSB is loaded into the hardware LSB for the selected level. This is a level switch to the higher (selected) level and causes the lower level to be pending.



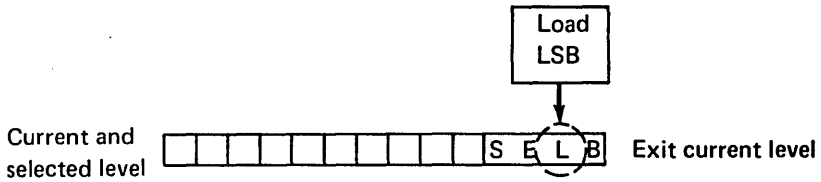
Selected Level Lower Than Current Level and In-Process Bit Off

These conditions cause the pending selected level to be reset. The main storage LSB is loaded into the hardware LSB for the selected level.



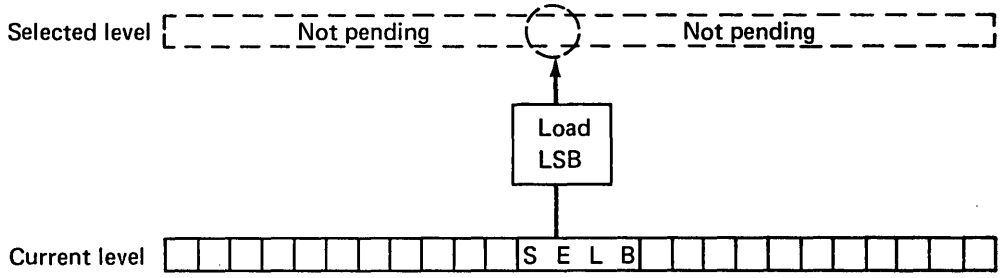
Selected Level Equal to Current Level and In-Process Bit Off

These conditions cause an exit from the selected (current) level. This exit is identical to executing a LEX instruction except that the main storage LSB is loaded into the hardware LSB for the selected level. Refer to "Level Exit (LEX)" in Chapter 8 for additional information.



Selected Level Higher Than Current Level and In-Process Bit Off

The main storage LSB is loaded into the hardware LSB for the higher (selected) level.



Chapter 4. Input/Output Operations

Input/output (I/O) operations involve the use of devices to enter data into the system, to receive data from the system, or both. These devices are attached to the processor and main storage by the I/O channel, with the channel directing the flow of information. The I/O channel can accommodate a maximum of 256 addressable devices. The general data flow is shown in Figure 4-1.

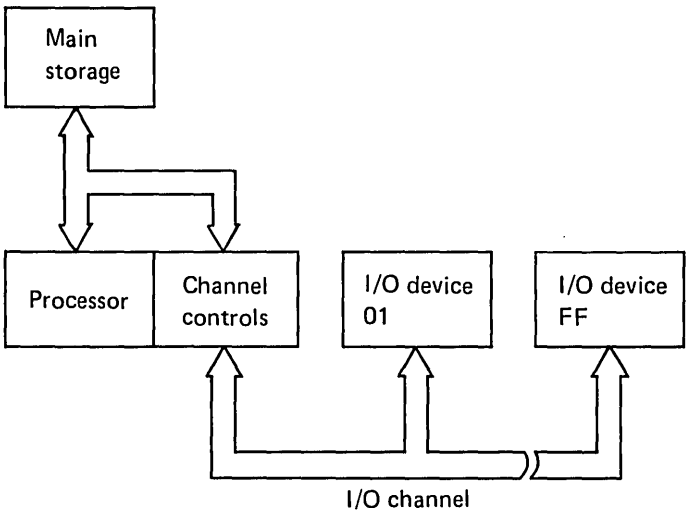


Figure 4-1. General data flow

The channel supports three basic types of operations:

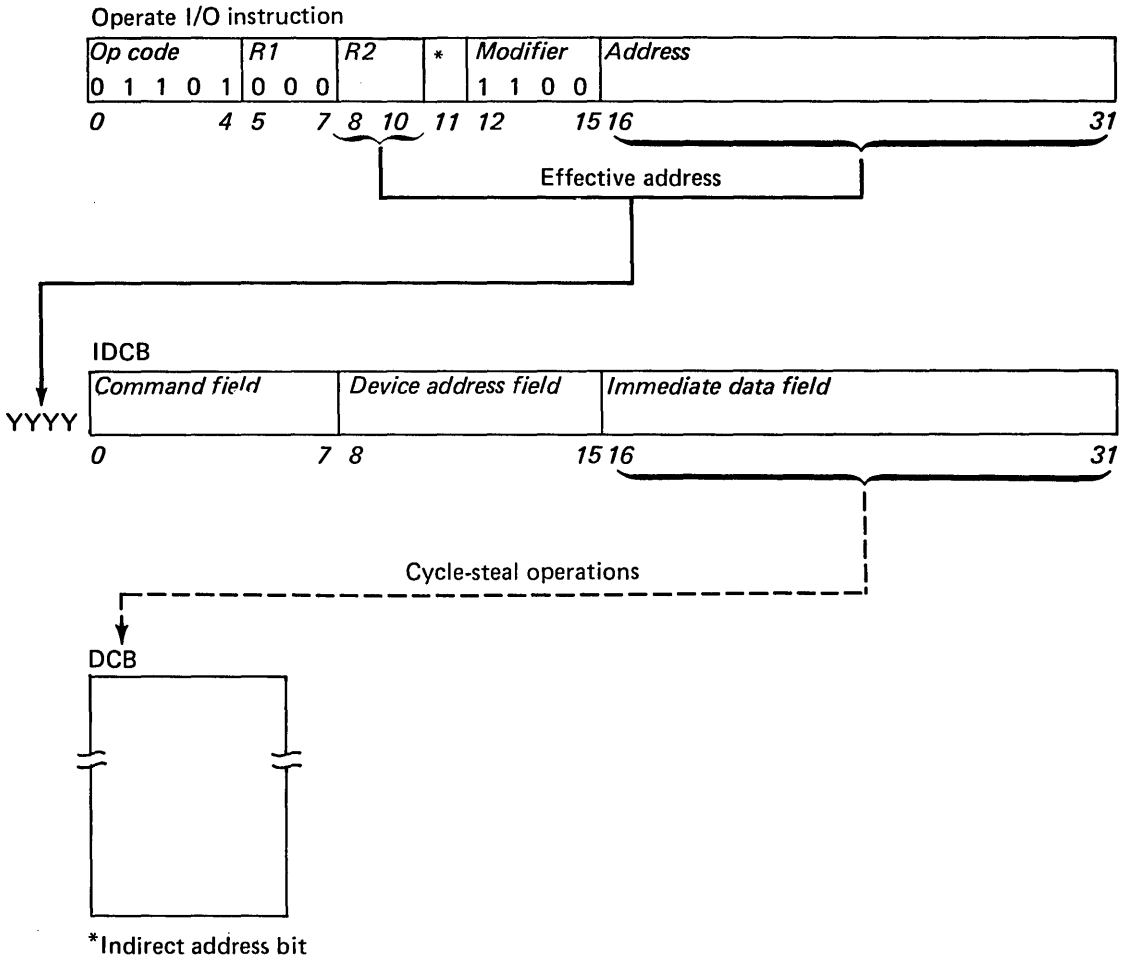
- *Direct Program Control (DPC) Operations*—An immediate data transfer is made between main storage and the device for each Operate I/O instruction. The data may consist of one byte or one word. The operation may or may not terminate with an interrupt.
- *Cycle-Steal Operations*—An Operate I/O instruction can initiate cycle-steal data transfers of up to 65,535 bytes between main storage and the device. Cycle-steal operations are interleaved with processing operations. Word or byte transfers, device control block (DCB) chaining, burst mode, and program-controlled interrupts (PCIs) can be supported. All cycle-steal operations terminate with an interrupt.
- *Interrupt Servicing*—Four priority interrupt levels are available to provide device transfers. The device interrupt level is assigned by the program. In addition, the device interrupt capability is controlled by the mask register, which is set by the Set Interrupt Mask Register (SEIMR) instruction. Interrupt requests, along with cycle-steal requests, are presented and polled concurrently with DPC and cycle-steal data transfers.

The channel provides comprehensive error checking, including time-outs, sequence checking, and parity checking. Error, exception, and status reporting are facilitated by recording condition codes in the processor during execution of Operate I/O instructions, and recording condition codes and an interrupt information byte (IIB) in the processor during interrupt acceptance. Additional status words may be used by the device, as necessary, to describe its status (refer to "I/O Condition Codes and Status Information" in this chapter).

Operate I/O Instruction

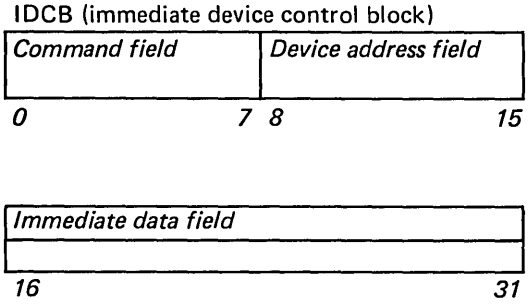
The Operate I/O instruction, which initiates all I/O operations from the processor, is a privileged instruction and is independent of specific I/O parameters. The generated effective address points to an immediate device control block (IDCB) in main storage. The IDCB consists of two words that contain an I/O command, a device address, and an immediate data field. For DPC operations, the immediate data field is used as a device data word. For cycle-steal operations, the immediate data field points to a DCB that provides additional information needed for the operation. For more details about the Operate I/O instruction, refer to Chapter 8.

Note: DPC operations are performed by all devices, but some devices do not operate in cycle-steal mode.



Immediate Device Control Block (IDCB)

The storage location specified by the Operate I/O instruction's effective address contains the first word of the IDCB. The IDCB contains an I/O command that describes the type of I/O operation. This command is used by the channel for execution of the operation. The IDCB must always be on an even-byte address boundary, and has the following format:



Command Field (Bits 0–7).

Bit 0 *Channel-directed.* If this bit is set to 1, the I/O command is directed to the channel rather than to a specific device. The Halt I/O command is the only valid channel-directed command. Any other command with bit 0 set to 1 causes a command reject exception condition.

Bit 1 *Read/write.* If this bit is set to 1 (write), the data contained in the immediate data field is transferred to the addressed I/O device. If this bit is set to 0 (read), the immediate data field contains the data received from the I/O device at the conclusion of the I/O instruction.

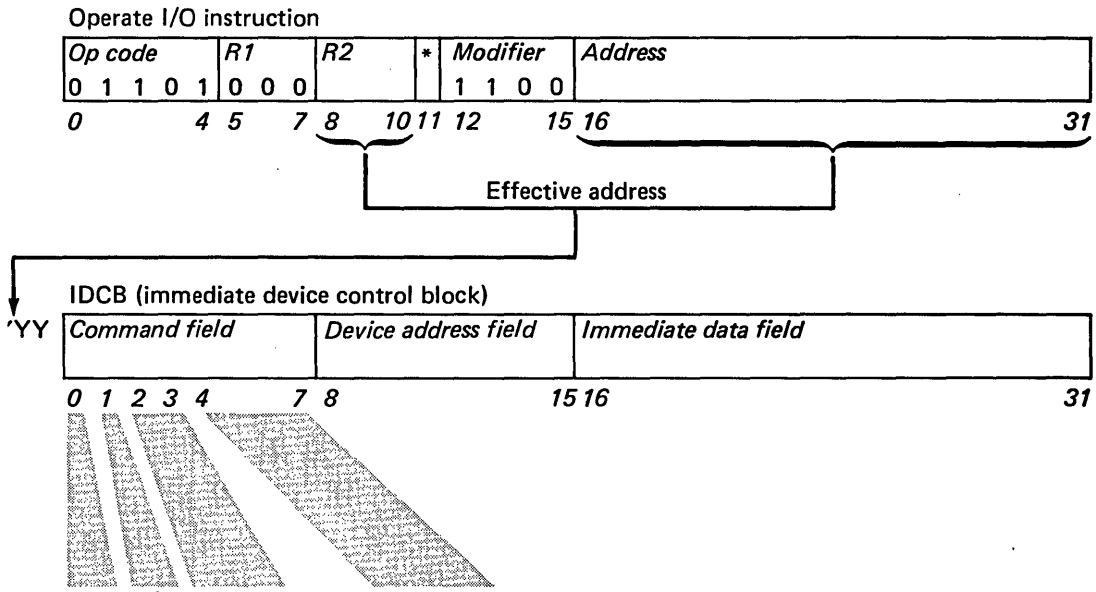
Bits 2–3 *Function.* This field specifies the general type of I/O operation to be performed (see Figure 4-2).

Bits 4–7 *Modifier.* This field further defines the functions.

Device Address Field (Bits 8–15). This field contains the I/O device address. A unique I/O device address is assigned to each I/O device. The address range is 00 through FF (hex).

Immediate Data Field (Bits 16–31). For DPC operations, the immediate data field contains a data word; for cycle-steal operations, this field points to a device control block.

Figure 4-2 shows the relationship between the IDCB and the Operate I/O instruction, with a chart for the various I/O commands. The Start and the Start Cycle Steal Status commands are used to initiate cycle-steal operations. The remaining commands are used for DPC operations only.



Chan	R/W	Function	Modifier**	Hex	Specific command	Type of operation
0	0	00 Read	XXXX	0X	Read	DPC
0	0	01 Read	XXXX	1X	Read	DPC
0	0	10 Read status	0000	20	Read ID	DPC
0	0	10 Read status	XXXX	2X	Read Status	DPC
0	0	11		3X	Unused***	Unused
0	1	00 Write	XXXX	4X	Write	DPC
0	1	01 Write	XXXX	5X	Write	DPC
0	1	10 Control	0000	60	Prepare	DPC
0	1	10 Control	XXXX	6X	Control	DPC
0	1	10 Control	1111	6F	Device Reset	DPC
0	1	11 Start	XXXX	7X	Start	Cycle-steal
0	1	11 Start	1111	7F	Start Cycle Steal Status	Cycle-steal
1	1	11 Channel	0000	F0	Halt I/O	Channel

*Indirect address bit.
**Modifier XXXX is device-dependent. Other modifiers are system-defined.
***To avoid future code obsolescence, this command format must not be used.

Figure 4-2. IDCB and I/O commands

Device Control Block (DCB)

This section describes the standard device control block that is used for a cycle-steal operation. The actual cycle-steal operation is explained under "Cycle-Steal" in this chapter. The DCB is an eight-word control block, residing in main storage, that contains the specific parameters for a cycle-steal operation. The device fetches the DCB using the cycle-steal mechanism.

All devices use the standard DCB format (see Figure 4-3). Some devices may also use additional formats, which are explained in the individual device publications.

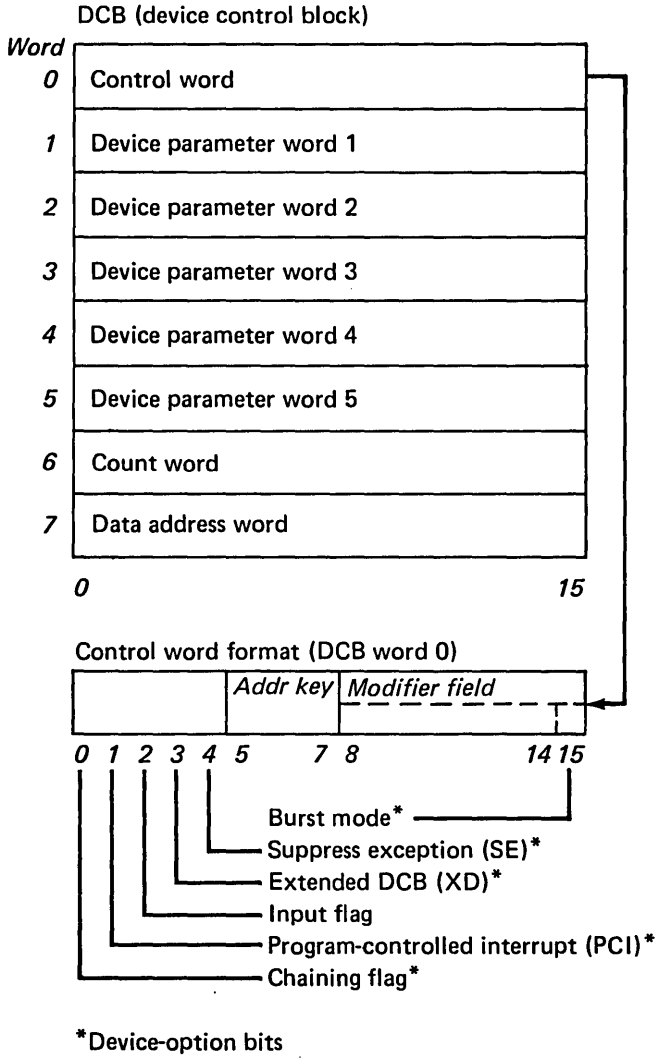


Figure 4-3. Device control block

The DCB words have the following meanings:

Control Word

Bit 0* *Chaining flag.* If this bit is set to 1, a DCB chaining operation is indicated.

Bit 1* *Program-controlled interrupt (PCI).* If this bit is set to 1, the device presents a PCI at the completion of the DCB fetch.

Bit 2 *Input flag.* This bit indicates to the device the direction of data transfer.
0 = output (main storage to device)
1 = input (device to main storage)
For bidirectional data transfers under a DCB operation, this bit must be set to 1. For control operations involving no data transfer, this bit must be set to 0.

Bit 3* *Extended DCB.* This bit, when set to 1, specifies that the DCB is a non-standard type.

Bit 4* *Suppress exception.* If this bit is set to 1, the device is allowed to suppress the reporting of certain exception conditions. The device can then take alternative action, depending on the condition.

Bits 5–7 *Cycle-steal address key.* These bits are presented by the device, during data transfers, to ascertain storage access authorization.

Bit 8–15 *Modifier.* These are device- dependent bits, with the following exceptions: (1) when extended DCB=1, bits 8–11 further identify the DCB type, and (2) when a device uses burst mode, it is specified in bit 15. Otherwise, these bits may be used for functions that are unique to a particular device.

*These bits are used with device options that are available on a device-feature basis. All bits not used by the device must be set to 0's. If the bits are set improperly, the devices may report a DCB specification check. Refer to the individual device publications for additional information.

Device Parameter Words 1–2

Device parameter words 1–2 are device-dependent control words and are used as required. Refer to the individual device publications for definitions of these words.

Device Parameter Word 3

The high-order byte (bits 0–7) of device parameter word 3 is used as a DCB identifier when PCI is specified. The device places the identifier in the interrupt information byte when the PCI is processed. The high-order byte (bits 0–7) is device-dependent when PCI is not specified. The low-order byte (bits 8–15) is always device-dependent.

Device Parameter Word 4

Device parameter word 4 specifies a 16-bit main storage address called the status address, if suppress exception is used by a device. The status address points to a residual status block that is stored by the device following completion of the DCB operation.

When suppress exception is not used by a device, a residual status block is not stored. In this case, parameter word 4 is device-dependent.

Device Parameter Word 5

Device parameter word 5 specifies a 16-bit main storage address of the next DCB in the chain, when the DCB chaining bit (bit 0) of the control word) is set to 1. When chaining is not indicated, parameter word 5 is device-dependent.

Count Word 6

The count word contains a 16-bit unsigned integer that represents the number of data bytes to be transferred for the current DCB. The count word specifies bytes with a range of 0 through 65,535. The count word must contain an even number for word-only devices.

Data Address Word 7

The data address word contains the starting main storage address for data transfers.

Programming Considerations When Using the DCB

1. Only those words required for the cycle-stealing operation are used by the device and they may be used in any order. Contents of the words must be specified correctly; if not, the device records a DCB specification check in the interrupt status byte (ISB) and terminates the cycle-steal operation with an exception interrupt.
2. The DCB address (in the IDCB), the chain address, and the status address must be on an even-byte address boundary. If the DCB address is odd, the device records a command reject condition code and terminates the operation. An odd chain address or an odd status address results in a DCB specification check.

Note: Condition codes and status recording are explained in detail under "I/O Condition Codes and Status Information" in this chapter.

I/O Commands

This section describes each I/O command and shows the related IDCB. The command field (bits 0–7) of the IDCB contains the hex value of the command. An X in this field means that the value is device-dependent.

Read

IDCB (immediate device control block)

Command field	*Device address field*
0 0 0 X X X X X	X X X X X X X X
0 ... *7*	*8* ... *15*
0X, 1X	00–FF

Immediate data field
Data word
16 ... *31*

The Read command transfers a word or a byte from the addressed device to the data word of the IDCB. If a single byte is transferred, it is placed in bits 24–31 of the data word, with bits 16–23 set to 0's. Correct parity is always maintained and checked for both bytes on the I/O channel. The individual devices may use either the 0X- or 1X-type of Read command.

Read ID

IDCB (immediate device control block)

Command field	*Device address field*
0 0 1 0 0 0 0 0	X X X X X X X X
0 ... *7*	*8* ... *15*
20	00–FF

Immediate data field
Data word
16 ... *31*

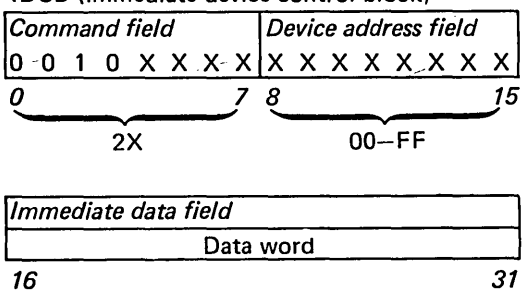
The Read ID command transfers a device identification (ID) word from the device to the data word of the IDCB. The device ID word contains physical information about the device and may be used to determine the devices that are attached to the system. This ID word is not related to the interrupt ID word associated with interrupt processing. The device ID word format is:

Class	0	Assigned code	CS	D
0 ... *3*	*4*	*5* ... *13*	*14*	*15*

Bits 0–3	Assigned class code
Bit 4	Reserved; always 0
Bits 5–13	Assigned code
Bit 14	0 – not a cycle-steal device
	1 – cycle-steal device
Bit 15	0 – IBM device
	1 – OEM device

Read Status

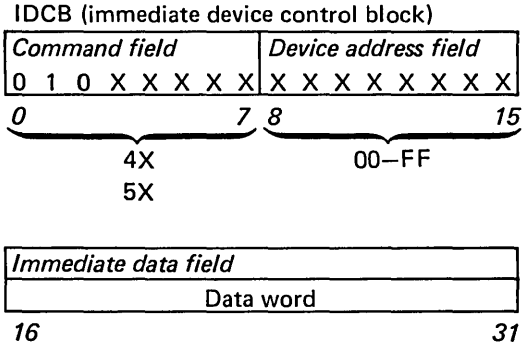
IDCB (immediate device control block)



The Read Status command transfers a device status word from the device to the data word of the IDCB. Contents of the device status word are device-dependent.

Write

IDCB (immediate device control block)

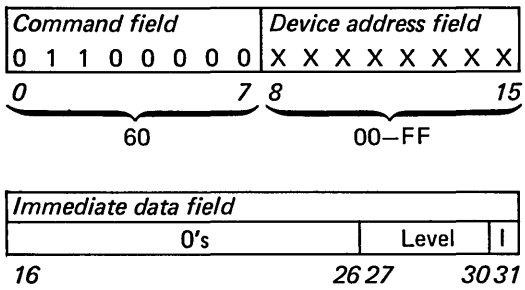


The Write command transfers a word or a byte from the data word of the IDCB to the addressed device. An individual device may use either the word format or the byte format. If a single byte is to be transferred, it must be placed in bits 24–31 of the data word and bits 16–23 must be set to 0's. A byte-oriented device may ignore bits 16–23 (including the parity bit) on the I/O channel, but these bits should be set to 0's to avoid future code obsolescence.

Note: Both bytes of the IDCB data word are fetched by the channel and placed on the I/O data bus (with good parity) even if both bytes are not required by the device.

Prepare

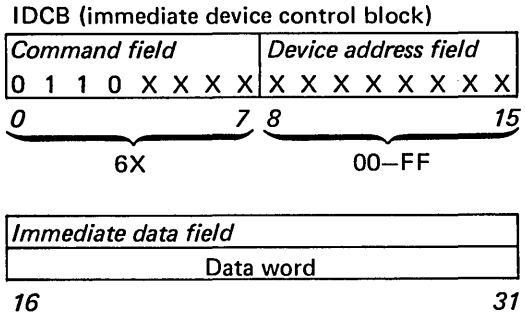
IDCB (immediate device control block)



The Prepare command transfers a device interrupt control word to the addressed device that controls the device interrupt parameters. The device interrupt control word is transferred from the immediate data field of the IDCB in the format shown. A priority interrupt level is assigned to the device by the level field. The I-bit (device mask) controls the device interrupt capability. If the I-bit is set to 1, the device is allowed to interrupt. If the I-bit is set to 0, the device cannot interrupt. Refer to "Prepare I/O Device for Interrupt" in Chapter 3.

Control

IDCB (immediate device control block)



The Control command initiates a control action in the addressed device. A word or byte transfer from the data word of the IDCB to the addressed device may or may not occur, depending on device requirements. If a single byte is to be transferred, it must be placed in bits 24–31 of the data word and bits 16–23 must be set to 0's.

Note: Both bytes of the IDCB data word are fetched by the channel and placed on the I/O data bus (with good parity) even if they are not required by the device.

Device Reset

IDCB (immediate device control block)

Command field	*Device address field*
0 1 1 0 1 1 1 1	X X X X X X X X
0 ... *7*	*8* ... *15*
6F	00–FF

Immediate data field
0's
16 ... *31*

The Device Reset command resets the addressed device. A pending interrupt from this device (or a busy condition) is cleared. The device mask (I-bit) is not changed. The assigned priority level for the device is not changed. The residual address (device status) and output sensor points are not reset. The IDCB data word is not checked for parity.

Start

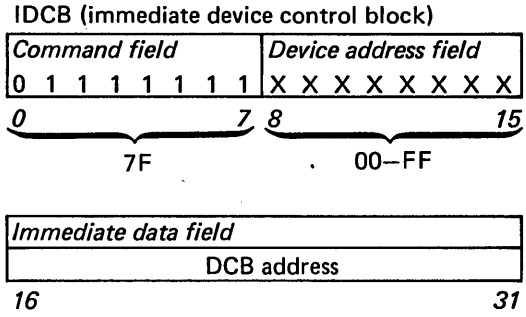
IDCB (immediate device control block)

Command field	*Device address field*
0 1 1 1 X X X X	X X X X X X X X
0 ... *7*	*8* ... *15*
70–7E	00–FF

Immediate data field
DCB address
16 ... *31*

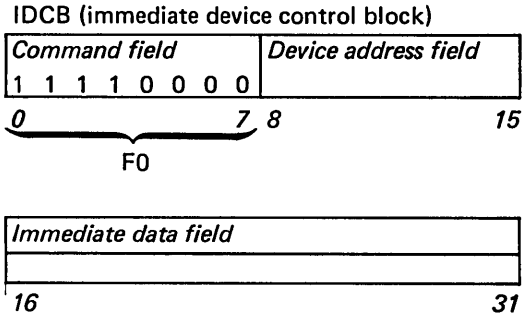
The Start command initiates a cycle-steal operation for the addressed device. The second word of the IDCB contains a 16-bit logical storage address of a DCB and is transferred to the device. Refer to "Start Command" in this chapter for additional information.

Start Cycle Steal Status



The Start Cycle Steal Status command initiates a cycle-steal operation for the addressed device. Status information is collected from the addressed device. The second word of the IDCB contains a 16-bit logical storage address of a DCB and is transferred to the device. Refer to "Start Cycle Steal Status Command Operation" in this chapter for additional information.

Halt I/O



Halt I/O is a channel-directed command that causes a halt of all I/O activity on the I/O channel and resets all devices. Data is not associated with this command. All device interrupts pending are cleared. Device priority-interrupt-level assignments and device masks (I-bits) are not changed. The residual address (device status) and output sensor points are not reset.

Note: The channel always accepts and executes the Halt I/O command, and it is the only valid channel-directed command.

DPC Operation

A DPC operation is an immediate transfer of data or control information to or from an I/O device under the control of an Operate I/O instruction. The Operate I/O instruction must be executed for each data transfer. Refer to Figure 4-4 for an explanation of the following steps:

1. The Operate I/O instruction's effective address points to an IDCB in main storage Ⓐ.
2. The I/O channel uses the IDCB to select the addressed device and to determine the operation to be performed Ⓑ.
3. The I/O channel transfers data to the device from main storage, or transfers data from the device to main storage Ⓒ.
4. The device transfers an I/O instruction condition code to the current level status register (LSR) in the processor Ⓓ.

Notes:

1. The DPC operation may end with a priority interrupt. Refer to "I/O Interrupts" in Chapter 3 for additional information.
2. There are two types of condition codes: the first is an I/O instruction condition code, and it is presented immediately after completion of an Operate I/O instruction; the second is an interrupt condition code, and it is presented upon acceptance of a priority interrupt. The condition code significance is different for the two cases. Refer to "I/O Condition Codes and Status Information" in this chapter for additional information.

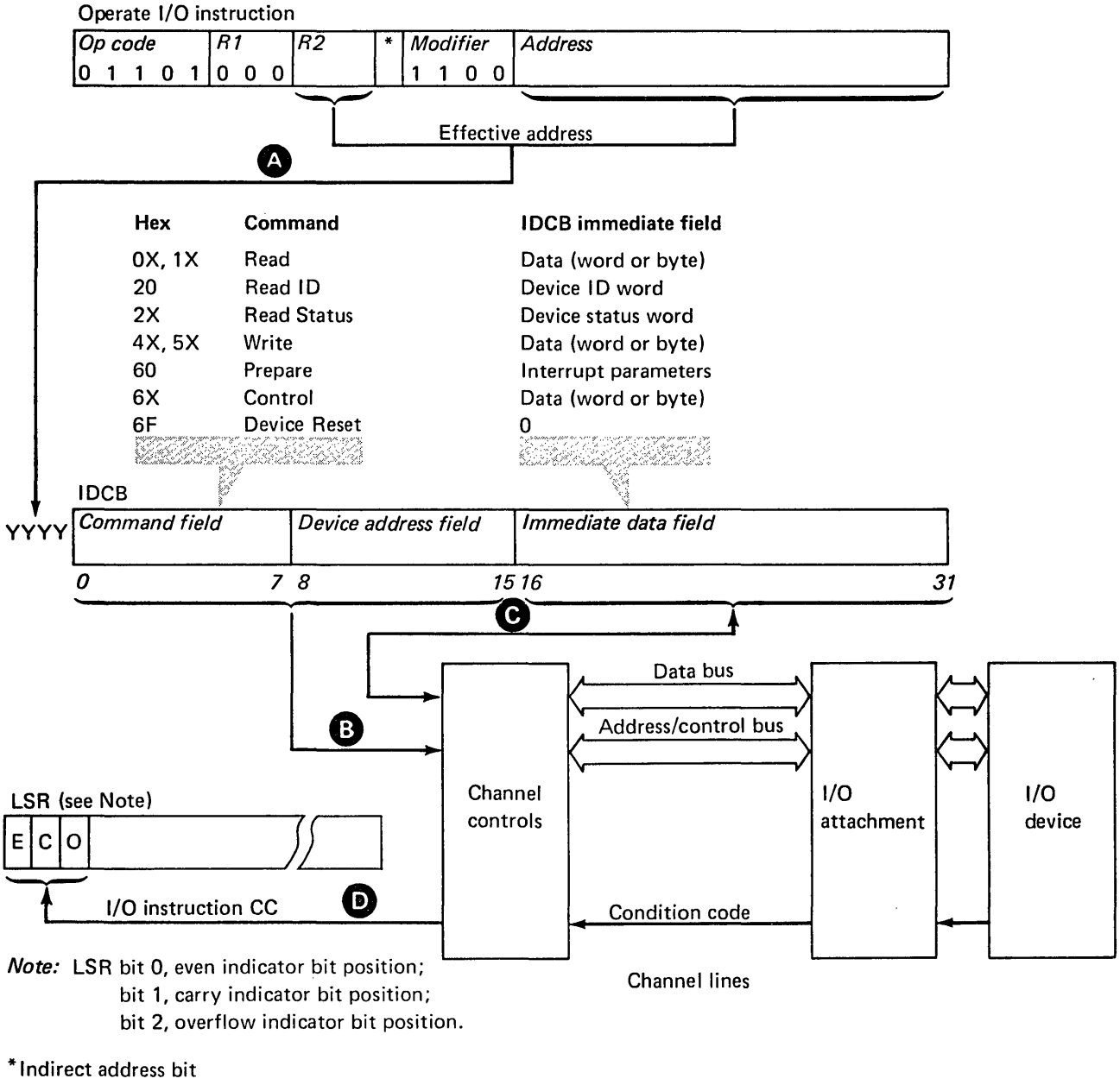


Figure 4-4. Direct program control I/O operation

Cycle-Steal

Cycle-stealing allows data transfer to or from an I/O device while the processor is processing instructions. This interleaved operation allows multiple data transfers to be started by one Operate I/O instruction. The processor executes the Operate I/O instruction, then continues processing instructions while the I/O device steals main storage data cycles when needed. The channel resolves contention among multiple devices requesting cycle-steal transfers. The operation always ends with a priority interrupt from the device.

The cycle-steal operation capabilities depend on the device options that are provided by a device feature basis.

- Burst mode
- DCB chaining
- Extended DCB
- Program-controlled interrupt (PCI)
- Suppress exception
- Storage addresses and data transfers by byte or word

Refer to "Cycle-Steal Device Options" in this chapter for additional information about each option.

All cycle-steal operations terminate with a priority interrupt, provided that the device has been prepared with a successful Prepare command, with the device mask (I-bit) set to 1. If the device mask is (I-bit) is set to 0, the interrupt presentation is blocked and the device remains busy until the condition is cleared by a reset, or the proper Prepare command is executed.

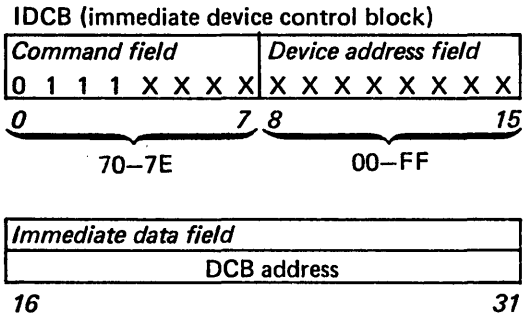
All cycle-steal operations are started by an Operate I/O instruction that points to an IDCB. The immediate data field of the IDCB contains the address of a DCB. The DCB is fetched by the device using a cycle-steal address key of 0. Refer to "Device Control Block (DCB)" in this chapter for a additional information about the DCB.

There are two types of cycle-steal commands:

- Start
- Start Cycle Steal Status

Start Command

The IDCB for the Start command, pointed to by an Operate I/O instruction, has the following format:



The Start command initiates a cycle-steal operation for the addressed device. The second word of the IDCB contains a 16-bit logical storage address of a DCB and is transferred to the device. The device uses this storage address for fetching the DCB.

A cycle-steal operation is described in the following chart. Use Figure 4-5 in conjunction with this chart. Condition codes used in the chart are explained under "I/O Condition Codes and Status Information" in this chapter.

Note: An I/O device must be properly prepared (using a Prepare command) before it is allowed to interrupt.

Cycle-steal major steps	**Remarks**
Start cycle-steal	1. Execute I/O instruction. 2. IDCB contains a Start command and points to a DCB. The DCB address is sent to the device.Ⓐ 3. Device presents condition code 7 (bits 0–2 in the LSR).Ⓑ
Device fetches DCB	1. Device uses cycle-steal mechanism to fetch DCB.Ⓒ 2. Cycle-steal address key of 0 is used.
Data transfer	1. Data is transferred to or from the device in word or byte format.Ⓓ 2. Transfers continue until count in DCB is exhausted. 3. DCB specifies cycle-steal address key for data area.
Termination (no error condition)	1. Device presents interrupt request. 2. Channel polls I/O attachment feature and accepts request. 3. Device sends interrupt ID word and interrupt condition code 3 (device end).
Termination (exception condition)	1. Device presents interrupt request. 2. Channel polls I/O attachment feature and accepts request. 3. Device sends interrupt ID word and interrupt condition code 2 (exception).

Other events that might occur during the cycle-steal operation are:

Chaining	1. Device completes the current DCB operation but does not present an interrupt request. 2. Device fetches next DCB in the chain.Ⓔ
Program-controlled interrupt	1. Device fetches DCB (PCI bit=1). 2. Device initiates an interrupt and sends an interrupt ID word and interrupt condition code 1 (PCI).
Suppress exception	1. Device completes current operation. 2. Device stores status at the main storage location defined by DCB parameter word 4, using a cycle-steal address key of 0.

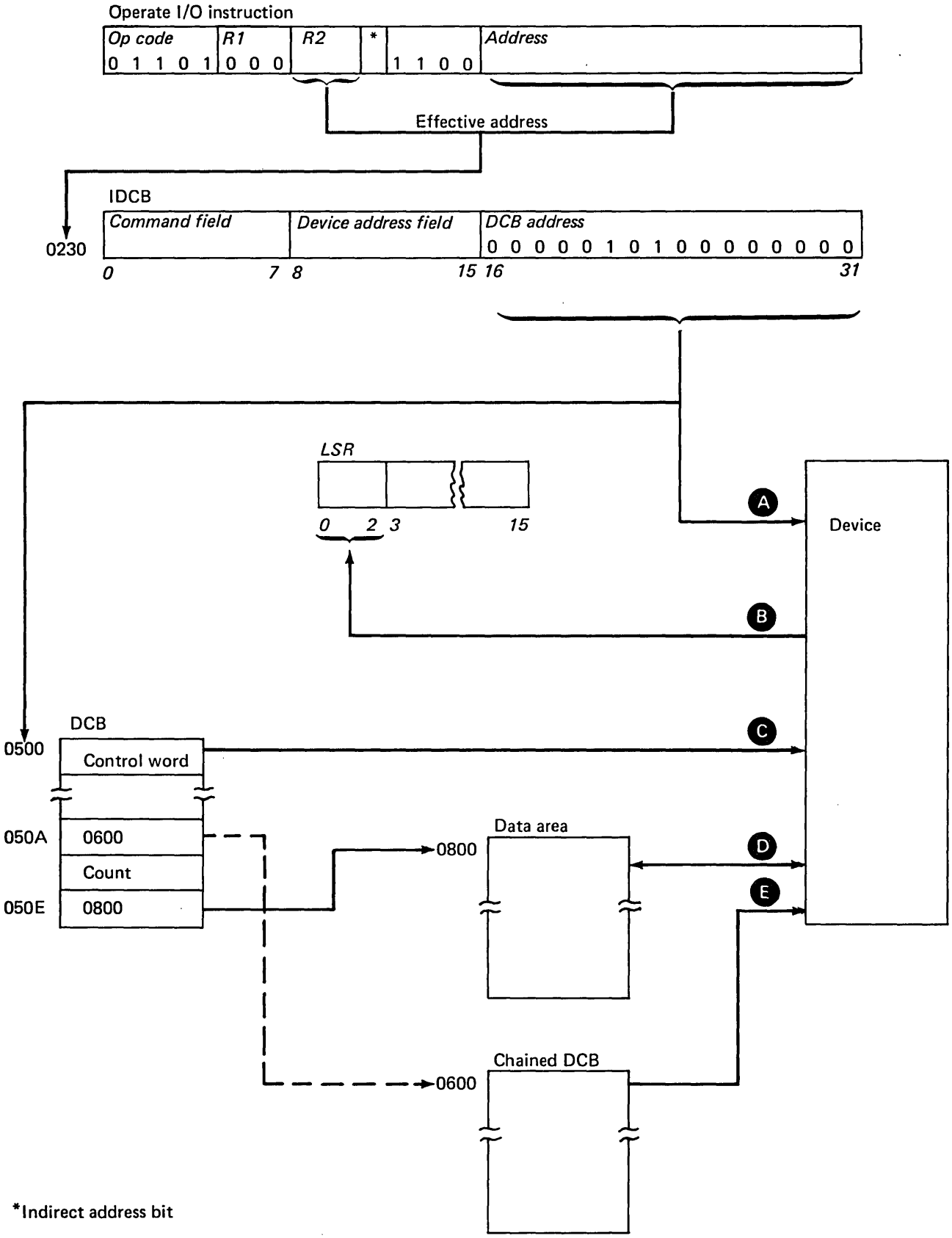
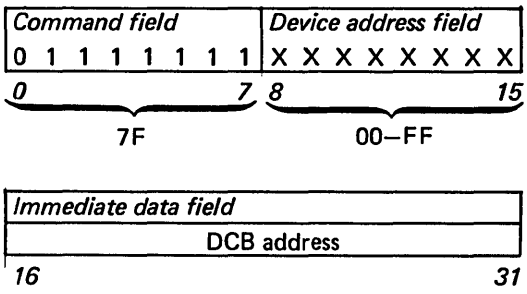


Figure 4-5. Example of cycle-steal control information

Start Cycle-Steal Status Command

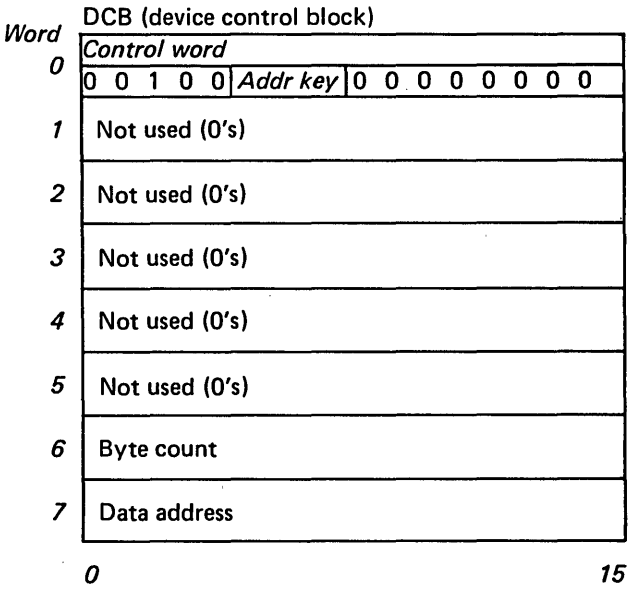
The Start Cycle-Steal Status command initiates a cycle-steal operation for the addressed device. Status information is collected from the addressed device if the previous operation terminated due to an error or exception condition. The second word of the IDCB contains a 16-bit logical address of a DCB and is transferred to the device. The IDCB format is:

IDCB (immediate device control block)



The Start Cycle-Steal Status command DCB format is:

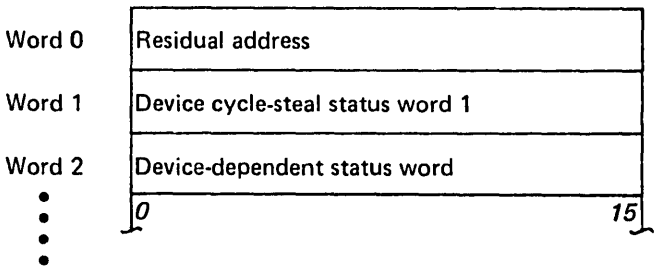
DCB (device control block)



Programming Note: For the start cycle-steal status operation, the DCB has the following parameters:

- Bits designated as 0's are not checked by hardware (see Figure 4-6).
- The count is specified in bytes.
- The maximum count is device-dependent.
- The validity of a count value less than the maximum value is device-dependent.
- If the maximum count is exceeded, or a count value is specified that indicates the partial storing of a word-length parameter, the device records a DCB specification check in the interrupt status byte and terminates the operation.
- An odd data address also results in a DCB specification check.

Cycle steal status data is transferred to main storage starting at the data address specified in the DCB. This data consists of residual parameters and device-dependent status information, and has the following format:



Residual Address. This word contains the main storage address of the last cycle-steal transfer attempted with a Start command. The address may be a data address, a DCB address, or a residual-status-block address. For word transfers, the residual address points to the higher address (low-order) byte of the word. If an error occurs during a start cycle-steal status operation, the residual address (as contained within the device) is not altered. Device Reset, Halt I/O, machine check, and system reset have no effect on the residual address in the device. This residual address is cleared by a power-on reset. Following a power-on reset, the residual address is:

- 0000 (hex) for a byte-oriented device.
- 0001 (hex) for a word-oriented device.

Device Cycle-Steal Status Word 1. This word contains the residual byte count of the previous cycle-steal operation initiated by a Start command. The byte count is initialized by the count field of a DCB during a Start command. The residual byte count is updated as each byte of data is successfully transferred by a cycle-steal operation. The residual byte count is not updated by cycle-steal transfers into the residual status block and is not altered if an error occurs during a start cycle-steal status operation. The residual byte count is reset by a power-on reset, system reset, device reset, Halt I/O, or machine-check condition.

Note: The contents of device cycle-steal-status word 1 are device-dependent. Some devices do not implement suppress exception or store a residual byte count as part of its cycle-steal status.

Device-Dependent Status Words. The number of words and their content are specified by the individual device. Three conditions can cause bits to be set in the device-dependent status words (refer to individual device publications):

- Execution of an I/O command that causes an exception interrupt.
- Asynchronous conditions in the device that indicate an error, an exception, or a state condition.
- Conditions as defined by the individual device.

The bits are reset as follows:

- Exception interrupt bits are reset by the acceptance of the next I/O command (except Start Cycle Steal Status). These exception interrupt bits are also reset by a power-on reset, system reset, or execution of a Halt I/O command.
- Asynchronous condition bits are reset as defined by the individual device.
- Individual device condition bits are reset as defined by the individual device.

Cycle-Steal Device Options

Bits in the DCB control word are used to activate cycle-steal device options. Refer to the individual device publications for device options.

Burst Mode

Burst mode is specified by bit 15 of the DCB control word. If bit 15 is set to 1, the transfer of data takes place in burst mode. This mode dedicates the I/O channel to the device until the last data transfer is completed (DCB count is 0). Cycle-steal interleaving by other devices is prevented. Burst mode also prevents all priority interrupt requests from being accepted by the processor.

Chaining

Chaining allows the programmer to sequence an I/O device through a set of operations by using a chain of DCBs. Bit 0 of the DCB control word (when set to 1) indicates a chaining operation. Each chained DCB, fetched by the device, is interpreted as a new operation (or function) to be performed.

When the current DCB indicates a chaining operation, device parameter word 5 of the DCB must contain a main storage address that points to the next DCB in the chain. The device completes the current operation but does not present an interrupt request (excluding PCI) to the processor. Instead, the device fetches the next DCB in the chain and executes its operation.

Note: The chaining operation does not affect PCIs. These interrupts, when specified in the DCB, still occur at the completion of the DCB fetch operations.

Extended DCB

This option allows a device to use additional DCB types. Each DCB type is designed to support a specific operation, such as data chaining, and is assigned a unique name in order to distinguish it from a standard DCB. Bit settings in the control word of the DCB determine the type. The extended DCBs, if used by a device, are explained in the individual device publication.

Program-Controlled Interrupt (PCI)

Bit 1 of the DCB control word (when set to 1) instructs the device to present a PCI to the processor at the completion of the DCB fetch and prior to data transfer.

When the PCI is serviced, a DCB identifier byte is returned to the processor in the interrupt information byte. Refer to "Device Parameter Word 3" under "Device Control Block (DCB)" in this chapter.

Chaining and data transfers associated with the DCB may commence even if the PCI is pending.

If the PCI is pending when the device encounters the next interrupt-causing condition, the PCI condition is discarded by the device and replaced with the new interrupt condition.

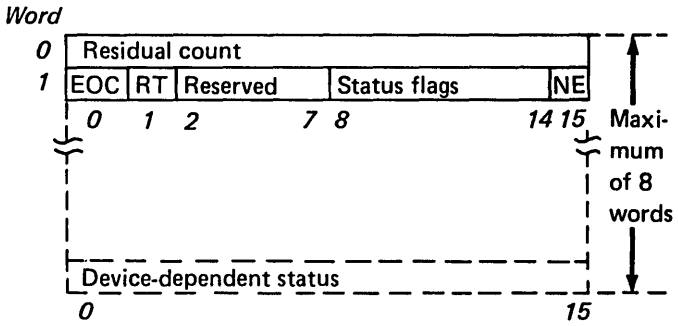
Suppress Exception

A device using suppress exception is allowed to suppress the reporting of certain exception conditions that would normally cause an exception interrupt. The device is then allowed to take alternative action, depending on the condition. The suppressed exception conditions are reported to the programmer as status information upon completion of the operation. Refer to "Suppression of Exceptions" in this section for details about various actions that a device might take.

The suppress exception option also provides automatic logging of status information (including suppressed exceptions) in main storage. When the suppress exception bit of a DCB is set to 1, the device always stores a residual status block in main storage after successful completion of the data transfer. Device parameter word 4 of the DCB must be used to specify the residual status block starting address in main storage. A residual status block is stored even when there are no exception conditions to be suppressed.

Residual Status Block

The residual status block is stored in main storage at the location pointed to by the status address (DCB word 4). The device uses an address key that corresponds to the DCB address space, for this operation. The size of a residual status block is fixed for each device with a limit of eight words. For a standard DCB, the format is:



Word 0 Contains the residual byte count associated with the DCB.

Word 1 EOC is the end-of-chain bit, and is set to 1 for all conditions that would terminate a chaining operation. RT is the retry bit, and is set to 1 when the device has attempted a retry operation. NE is the no exception bit, and is set to 1 when the operation is completed and no exceptions are reported. The status flags are device-dependent flags that indicate suppressed-exception conditions.

Any additional words are device-dependent as to number and content. Refer to the individual device publication for the additional status information and for the bit significance of the status flags.

Note: The words in a residual status block for a non-standard DCB may have different meanings. Refer to the individual device publications.

Suppression of Exceptions

An exception condition can be suppressed by a device only when it occurs during a data transfer operation. It cannot be suppressed if it occurs during:

- DCB fetch
- Storing of a residual status block
- Cycle-steal status operation

A second requirement of a suppressible exception is that the device be capable of continuing operation in a normal and predictable manner after occurrence of the exception. If these conditions are not met, the exception condition causes an exception interrupt. The number of action types used by a device and the suppressible exceptions for each type are a device specification. When a suppressible exception is encountered, the device initiates one of four possible types of actions, depending on the device and the exception condition. Refer to the individual device publications for additional information. The four action types are:

1. Suppress exception and continue. The exception condition occurs but data transfer is allowed to proceed. At the completion of the data transfer (defined by the DCB), a residual status block is stored. The device may then continue with the next DCB, if chaining is specified.
2. Suppress exception and retry. Upon detecting the exception condition, the device restarts the data transfer defined by the DCB. The number of retries to be attempted is a device specification. A residual status block is stored after a successful retry or after all retries have failed.
3. Suppress exception and terminate data transfer. Upon detecting the exception condition, the device terminates the data transfer for this DCB. The device stores a residual status block and continues with the next DCB, if chaining is specified.
4. Suppress exception and terminate chain. Upon detecting this exception condition, the device terminates the data transfer for this DCB, and ignores any commands specifying further chaining.

   The device stores a residual status block and then presents a permissive device-end interrupt. Refer to "Interrupt Condition Codes" in this chapter for additional information.

Priority of Suppress-Exception Actions. Multiple exceptions that are suppressible can occur during an operation. They are noted in the residual status block by setting multiple status flags. The type of action taken by a device depends on the exception/action combination with the highest priority. The priority sequence is type 4, type 3, type 2, and type 1, with type 4 having the highest priority.

I/O Condition Codes and Status Information

Each time an Operate I/O instruction is issued, the device, controller, or channel immediately reports to the processor one of eight condition codes pertaining to the execution of the I/O command. These codes are called I/O instruction condition codes. Three bits are used to encode a condition-code value (range 0 through 7). The bits are recorded in the even, carry, and overflow positions of the LSR and may be interrogated by specific instructions such as Branch on Condition Code and Branch on Not Condition Code. Refer to Chapter 8 for details of these instructions.

For interrupting devices, condition codes are also reported during a priority interrupt. These codes are called interrupt condition codes, and pertain to operations that continue beyond execution of the Operate I/O instruction (such as cycle-stealing of data). The interrupt condition codes are recorded in the current LSR and interrogated in the same manner as the I/O instruction codes. Along with the interrupt condition code, the device also transfers an interrupt ID word to the processor. Bits 0–7 of the interrupt ID word contain status information related to the interrupt processing and are called the interrupt information byte. Refer to "Interrupt ID Word" under "I/O Status Information" in this chapter for additional information.

I/O Instruction Condition Codes

I/O instruction condition codes are reported during execution of an Operate I/O instruction.

Condition code (CC) value	LSR position			Reported by	Meaning
	Even	Carry	Overflow		
0	0	0	0	Channel	Device not attached
1	0	0	1	Device	Busy
2	0	1	0	Device	Busy after reset
3	0	1	1	Chan/dev	Command reject
4	1	0	0	Device	Intervention required
5	1	0	1	Chan/dev	Interface data check
6	1	1	0	Controller	Controller busy
7	1	1	1	Chan/dev	Satisfactory

CC0 *Device not attached.* Reported by the channel when the addressed device is not attached to the system.

CC1 *Busy.* Reported by the device when it is unable to execute a command because it is in the busy state. The device enters the busy state upon acceptance of a command that requires an interrupt for termination. The device exits the busy state when the processor accepts the interrupt. Certain devices also enter the busy state when an external event causes an interrupt. When this condition code is reported, a subsequent priority interrupt from the addressed device always occurs.

CC2 *Busy after reset.* Reported by the device when it is unable to execute a command because of a reset, and the device has not had sufficient time to return to the quiescent state. An interrupt does not occur to indicate termination of this condition.

CC3 *Command reject.* Reported by the device or the channel when:

- A command (in the IDCB) that is outside the device command set is issued.
- The device is in an improper state to execute the command.
- The IDCB contains an incorrect parameter (for example, an odd-byte DCB address or an incorrect function/modifier combination).

When a cycle-steal device reports command reject, it does not fetch the DCB.

CC4 *Intervention required.* Reported by the device when it is unable to execute a command due to a condition that requires manual intervention.

CC5 *Interface data check.* Reported by the device or the channel (whichever is receiving the data) when a parity error is detected on the I/O data bus during a data transfer.

CC6 *Controller busy.* This condition is reported by a device controller, and not the addressed device, when the controller is busy. Controller busy is reported only by controllers that have two or more devices attached (where each device has a unique address).

CC7 *Satisfactory.* Reported by the device or the channel when it accepts the command.

Interrupt Condition Codes

Interrupt condition codes are reported by the device or controller during priority interrupt acceptance:

Condition code (CC) value	LSR position Even	Carry	Overflow	Reported by	Meaning
0	0	0	0	Controller	Controller end
1	0	0	1	Device	Program-controlled interrupt (PCI)
2	0	1	0	Device	Exception
3	0	1	1	Device	Device end
4	1	0	0	Device	Attention
5	1	0	1	Device	Attention and PCI
6	1	1	0	Device	Attention and exception
7	1	1	1	Device	Attention and device end

CC0 *Controller end.* May be reported by a controller when controller busy (I/O instruction condition code) has been previously reported one or more times. Controller end signifies that the controller is now free to accept I/O commands for devices under its control. The device address reported with controller end is always the lowest address (numerical value) of the group of devices serviced by the controller. The interrupt information byte in the interrupt ID word is set to 0.

CC1 *Program-controlled interrupt.* Reported when the interrupt indicates that a DCB with the PCI bit set to 1 has been transferred by cycle-steal to the device and no error or exception condition has occurred. The device places the DCB identifier into the interrupt information byte.

CC2 *Exception.* Reported when an error or exception condition is associated with the interrupt. The condition is described in the interrupt status byte or in device-dependent status words.

CC3 *Device end.* Reported when no error, exception, or attention condition has occurred during the I/O operation, and the interrupt is not the result of a PCI (for example, an operation terminats normally).

Note: If the device comes to a normal end while using suppress exception (suppress exception bit set to 1) and an exception was suppressed since the last Start command, then bit 0 of the interrupt status byte is set to 1. This condition is called permissive device end, and it indicates that errors or exceptions have been suppressed. Related status information is contained in the residual status block.

CC4 *Attention.* Reported when the interrupt is caused by an external event rather than caused by the execution of an Operate I/O instruction. Additional status information is not provided unless the event requires further definition (for example, code bits for a keyboard function).

CC5 *Attention and PCI.* Reported when attention and PCI are both present. In this case, the interrupt information byte contains the DCB identifier.

CC6 *Attention and exception.* Reported when attention and exception are both present.

CC7 *Attention and device end.* Reported when attention and device end are both present. For this condition code, device end can also mean permissive device end. Refer to interrupt condition code 3.

The interrupt condition codes are mutually exclusive with each other, and they have no priority sequence.

I/O Status Information

Status information is transferred from the device to the processor as a result of:

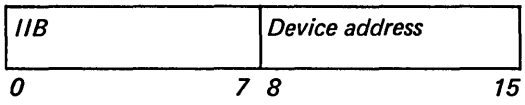
- A read status operation (refer to "Read Status" under "I/O Commands" in this chapter).
- A start cycle-steal status operation (refer to "Start Cycle Steal Status Command" in this chapter).
- Storing a residual status block (refer to "Cycle-Steal Device Options" in this chapter).
- A priority interrupt.

The interrupt status information is detailed in "Interrupt ID Word" and "Interrupt Status Byte (ISB)" in this section.

Interrupt ID Word

Acceptance of an I/O interrupt causes the device to present an interrupt ID word to the processor. Presentation of the interrupt ID word is explained in "I/O Interrupts" in Chapter 3. The interrupt ID word has the following format:

Interrupt ID word



Bits 0–7 *Interrupt information byte (IIB).* For interrupt condition codes 2 and 6, the IIB has a special format and is called an interrupt status byte (ISB). Refer to "Interrupt Status Byte" in this section. For other interrupt condition codes reported by a device, the IIB contains:

1. CC0. The IIB is set to 0.
2. CC1 or CC5. The IIB contains a DCB identifier.
3. CC3 or CC7. Bit 0 may be set to 1 if suppress exception is in effect. Other bits are device-dependent.
4. CC4. All bits are device-dependent.

Bits 8–15 *Device address.* This byte contains the address of the interrupting device.

Interrupt Status Byte (ISB)

The ISB is a special format of the interrupt information byte that contains detailed information about the nature of the interrupt. The ISB is reported only for error or exception conditions (interrupt condition code 2 or 6). The ISB bits are normally set as a result of:

- Status errors that occur during a DPC operation and that cannot be indicated by a condition code.
- Status errors that occur during a cycle-steal operation.

The ISB is never reported as 0 unless the condition code presentation of 2 or 6 is singular in meaning for devices that do not cycle-steal. After the processor has accepted the interrupt request, the device resets the ISB.

Bits 0–7 of the two special formats have the following meanings:

ISB (Devices That Do Not Cycle-Steal).

Bit 0 *Device-dependent status available.* This bit, when set to 1, signifies that additional status information is available from the device. The information content and method of reading is described in the individual device publication.

Bit 1 *Delayed command reject.* This bit is set to 1 if the device cannot execute the command (specified in the IDCB) due to an incorrect parameter in the IDCB, or it cannot execute the command due to its present state. For example, the IDCB specifies an incorrect function/modifier combination, or the device is temporarily not ready. The operation in progress is terminated. Delayed command reject is set in the ISB only if the device cannot report I/O instruction condition codes for the condition.

Bits 2–7 *Device-dependent.* These bits, if used, are described in the individual device publication.

ISB (Cycle-Stealing Device).

Bit 0 *Device-dependent status available.* This bit, when set to 1, signifies that additional status information is available from the device, or the device is in an improper state to execute a function specified by a DCB.

The operation is terminated. The content and method of reading the additional status information is described in the individual device publication.

Note: When bit 0 of the ISB is set to 1 and bits 2–7 are set to 0's, the contents of the residual-address word (cycle-steal status) are defined by the device.

Bit 1 *Delayed command reject.* This bit is set to 1 if the device cannot execute the command due to one of the following conditions:

- The IDCB contains an incorrect parameter (for example, an odd-byte DCB address or an incorrect function/modifier combination).
- The present state of the device, such as a not-ready condition, prevents execution of an I/O command specified in the IDCB.

Delayed command reject is set in the ISB only if the device cannot report I/O instruction condition codes for the condition. The operation is terminated. The DCB is not fetched.

Bit 2 *Incorrect-length record.* This bit is set to 1 when the device encounters a mismatch between byte count and actual record length after beginning execution of the DCB. For example, the byte count is reduced to 0 (with chaining flag off) and no end-of-record is encountered. Incorrect-length record is not reported when the suppress exception bit in the control word is set to 1. Reporting of incorrect-length record is a device-dependent feature and may be implemented regardless of the suppress-exception feature. The operation is terminated.

Bit 3 *DCB specification check.* This bit is set to 1 when the device cannot execute a command due to an incorrect parameter specification in the DCB (for example, an odd-byte DCB chaining or status address, an odd-byte count for a word-only device, an odd-byte data address for a word-only device, an invalid command or invalid bit settings in the control word, or an incorrect count). The operation is terminated.

Bit 4 *Storage data check.* This error condition applies to cycle-steal output operations only. If the bit is set to 1, it indicates that the main storage location accessed during the current output cycle contained bad parity. Parity in main storage is not corrected. The device terminates the operation. The bad parity data is not transferred to the I/O data bus. A machine-check condition does not occur.

Bit 5 *Invalid storage address.* This bit, when set to 1, indicates one of the following conditions:

- During a cycle-steal operation, the device presented a main storage address that is outside the storage size of the system.
- A cycle-stealing device attempted to access storage through a segmentation register and the valid bit in the segmentation register is set to 0. The relocation translator must be enabled before this condition can occur.

Invalid storage address can occur on a data transfer or on a DCB fetch operation. In either case, the cycle-steal operation is terminated.

Bit 6 *Protect check.* When set to 1, this bit indicates that the I/O device attempted to access a main storage location and presented an incorrect address key. Refer to individual device publications for additional information.

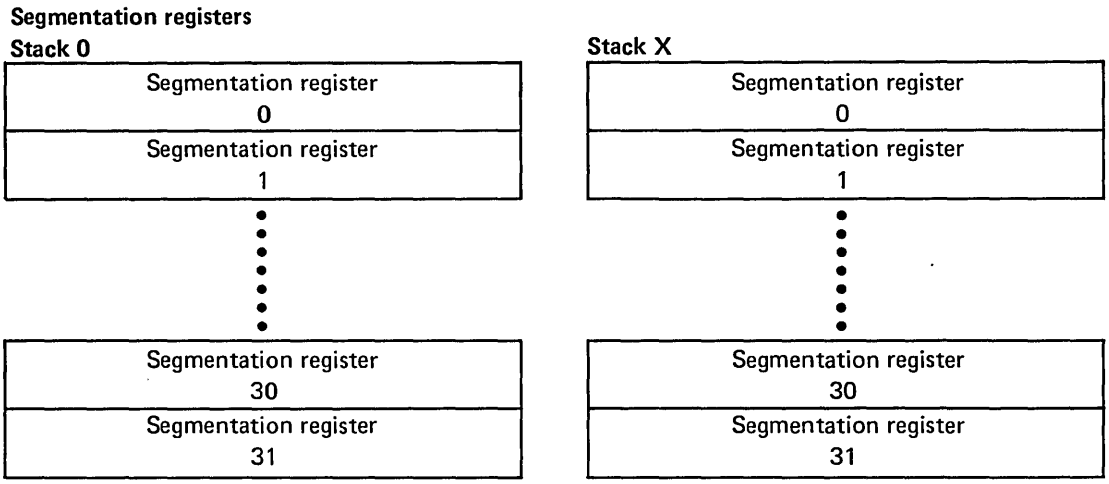
Bit 7 *Interface data check.* This bit, when set to 1, indicates that a parity error has been detected on the I/O data bus during a cycle-steal data transfer. This condition may be detected by the channel or the I/O device. In either case, the operation is terminated.

Chapter 5. Storage Address Relocation Translator

The relocation translator and segmentation registers permit addressing of main storage locations beyond 64K bytes and provide a read-only type of storage protection. The first 64K bytes can be addressed directly when the translator is disabled; therefore, the translator must be enabled when main storage is larger than 64K bytes.

Translator Description

The translator provides stacks of segmentation registers. The stacks are numbered consecutively from 0 to X to correspond to the possible values of the address keys. Each stack consists of 32 registers (0–31):



The stacks of segmentation registers are under supervisory program control. Four privileged instructions are used with the relocation translator and segmentation registers.

- *Set Segmentation Register (SESR).* This instruction loads one segmentation register.
- *Copy Segmentation Register (CPSR).* This instruction allows the supervisor to inspect the contents of a segmentation register.
- *Enable (EN).* This instruction enables the relocation translator. Until the translator is enabled, 16-bit addressing is in effect for the low-order 64K bytes of storage. Any storage above 64K bytes is not accessible to the program until the translator is enabled.
- *Disable (DIS).* This instruction disables the relocation translator.

Refer to Chapter 8 for detailed information of the preceding instructions. Refer to individual processor publications for further information concerning segmentation registers.

Storage Mapping

Mapping of main storage is achieved through the segmentation registers. Each segmentation register controls 2K-byte segments of storage. The SESR instruction is used to load each segmentation register with a unique physical segment address. This segment address is the physical address of a 2K-byte segment of storage.

Note, however, that more than one segmentation register can be loaded with the same segment address. For example, stack 0, register 15 (associated with the supervisor address key of 0), can be loaded with the same number as stack 1, register 6. This arrangement allows the supervisor to address control blocks within a problem program even though the address key for the supervisor is different than the key for the problem program. Once loaded, each stack of segmentation registers contains a complete map of 64K bytes scattered in 2K-byte physical segments.

Relocation Addressing

The relocation translator generates a physical address that allows any byte in storage to be addressed. Figure 5-1 shows an example of address translation. The letters in the following steps correspond to the letters on the figure:

- Ⓐ The active address key from the address key register selects a segmentation register stack. The address key pertains to the instruction being executed on the current priority level.
- Ⓑ The five high-order bits (0–4) of the 16-bit address (generated for the instruction being executed) select a segmentation register within the stack selected in step Ⓐ. These bits define the logical segment.
- Ⓒ The physical address is generated. The 13 high-order bits (0–12) are from the segmentation register; these bits specify the physical address of a 2K-byte segment of storage.

Bit 13 (Valid Bit). When set to 1, this bit specifies that the contents of the segmentation register are valid; the segmentation register can be used to perform the translation. When bit 13 is a 0, the segmentation register cannot be used for translation (no access). If translation is attempted, a program-check interrupt occurs with invalid storage address set in the processor status word (PSW).

Bit 14 (Read-Only Bit). When set to 1, this bit specifies that the block is read-only. When in the problem state, if an attempt is made to write into storage using a segmentation register with the read-only bit set to 1, a program-check interrupt occurs with protect check set in the PSW. Storage is not changed. Bit 14 is ignored by a cycle-steal access, or when in supervisor state.

- Ⓓ The 11 low-order bits (13–23) of the physical address are the 11 low-order bits (5–15) of the 16-bit logical address (generated for the instruction being executed); these bits specify the byte address within the 2K-byte segment.

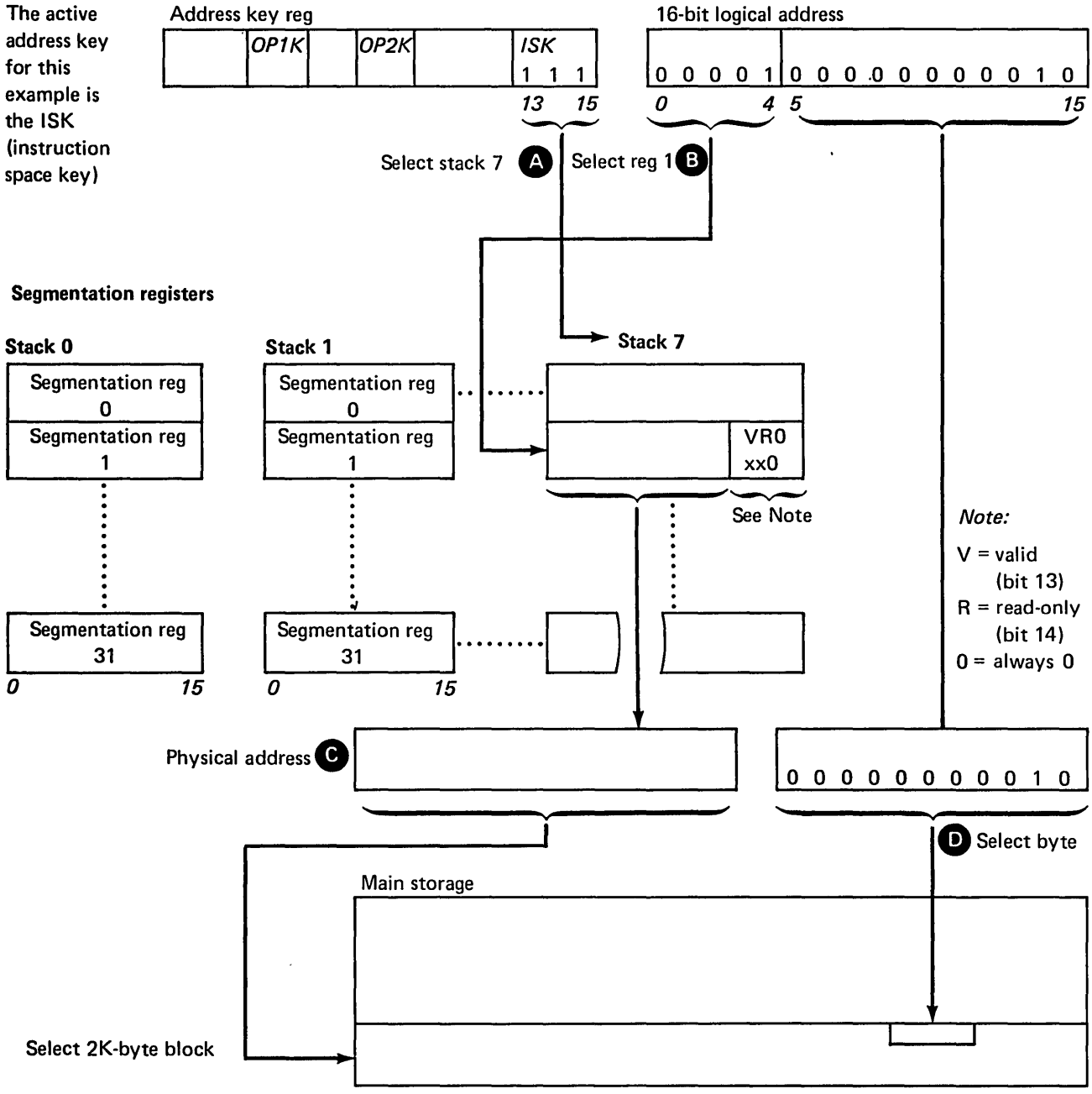


Figure 5-1. Address translation example

Storage Protection

The storage protection mechanism is enabled and disabled by the Enable (EN) and Disable (DIS) instruction described in Chapter 8. When storage protection is enabled, it protects against:

- Reading and writing to defined blocks of storage
- Writing in an undesired location within a defined block

Each processor handles storage protection in its own way, and is not effective when the relocation translator is active. Because each stack of segmentation registers has access to storage only within its assigned region, protection is provided against writing into storage or fetching instructions from another region. The translator also provides no-access and read-only protection within the regions controlled by each stack of segmentation registers. This allows storage protection of shared segments of storage. Bits 13 and 14 of the segmentation registers are used for this purpose.

Refer to individual processor publications for further information concerning storage protection.

I/O Storage Access Using the Relocation Translator

All storage access requests from I/O devices are translated by the hardware that handles storage requests from the processor. The device control block (DCB) must reside in the supervisor's address space; therefore, all I/O devices must use address key 0 to gain access to the DCB and to store the residual status block. The address key of the process requiring a cycle-steal operation resides in the DCB. The I/O device presents this address key along with a 16-bit logical address to the translator. This allows the I/O device to directly address the storage space for a particular process. The address key allows I/O storage protection to be established between address spaces, assuming that the supervisor ensures the integrity of the DCBs.

Status of Translator After Power Transitions and Resets

The translator is enabled only by the Enable (EN) instruction. The translator is disabled by the following:

- Disable (DIS) instruction
- Power-on reset
- Check Restart key on programmer console
- Initial program load (IPL)
- System Reset key on programmer console

All translator controls are reset when the translator is disabled.

Notes:

1. A machine-check interrupt does not disable the translator.
2. The segmentation registers are not reset when the translator is disabled.

Error-Recovery Considerations

Invalid Storage Address

The invalid storage address bit (bit 1 of the PSW) is set to 1 by any one of the following:

- Storage access was attempted using an address greater than the physical storage size.
- Storage access was attempted with bit 13 (valid bit) of the segmentation register set to 0. This signifies that the contents of the segmentation register are invalid.
- Storage access was attempted with an invalid address key.

The specific nature of the invalid storage address can be resolved as follows:

- Store the segmentation register following the program-check interrupt.
- Test the segmentation register for the presence of bit 13.
- If bit 13 is a 1, the supervisor's concept of the actual storage installed on the machine is incorrect.

Protect Check

A program-check class interrupt is initiated when the protect-check bit (bit 3 of the PSW) is set to 1. In the problem state, the protect-check bit is set to 1 when the selected segmentation register has bit 14 (read-only) set to 1 and the instruction being executed is a write operation.

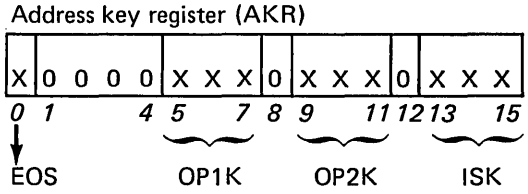
To resolve the cause of the protect-check error, the supervisor must determine if the translator is enabled.

Address Space Management

Active Address Key

The coding of the address key to be made active depends on the type of operation being performed.

Each level of priority interrupt has an associated address key register (AKR), each of which contains three address keys and an equate-operand-spaces (EOS) bit.



EOS — *Equate operand spaces.* This bit, when set to 1, causes all data operands to use the OP2K address key. Refer to "Equate Operand Spaces (EOS)" under "Address Space Management" in this chapter.

OP1K — *Operand 1 key.* These bits contain the binary-coded operand 1 address key.

OP2K — *Operand 2 key.* These bits contain the binary-coded operand 2 address key.

ISK — *Instruction space key.* These bits contain the binary-coded instruction address key.

Cycle-steal devices have a cycle-steal address key specified in their device control block.

When a programmer console is attached, the console address key may be used.

Any one of the five address keys mentioned (ISK, OP1K, OP2K, console address key, or the cycle-steal address key) may be used during a storage access as the active address key.

Equate Operand Spaces (EOS)

The equate operand spaces bit (bit 0) in the address key register controls the modification of the active address key.

When the EOS bit is set to 1 (enabled), all processor data fetches use a single address space defined by the OP2K address key. The OP1K is ignored, but not changed, and all normal OP1K operations use OP2K as an active key. When the EOS bit is equal to 0 (disabled), the OP1K address key functions in a normal manner.

Equate operand spaces (EOS) may be enabled by an Enable (EN) instruction, a Set Level Block (SELB) instruction, or a Set Address Key Register (SESKR) instruction. EOS may be disabled by a Disable (DIS) instruction, a Set Level Block (SELB) instruction, or a Set Address Key Register (SESKR) instruction. The EOS is also disabled by a priority interrupt or a class interrupt. These instructions are described in Chapter 8.

Address Space

An address key defines a specific address space, where:

- The address space is a range of logically contiguous storage.
- The address space is accessible by the effective address without intervention by a resource management function (the address space is not greater than 64K bytes).

All instruction fetches and effective address generation for the branch and jump instructions, use the address space defined by the instruction space key (ISK). For storage-to-storage instructions, the operand 1 fetches use the address space defined by the OP1K, assuming that the EOS bit is set to 0, and the operand 2 fetches use the address space defined by the OP2K. All other storage data accesses use the address space defined by the OP2K.

When the relocation translator is enabled, the address keys are used to help select a 2K-byte block of storage.

Examples:

ISK=OP1K=OP2K. For instruction processing, all storage accesses occur within the same address space.

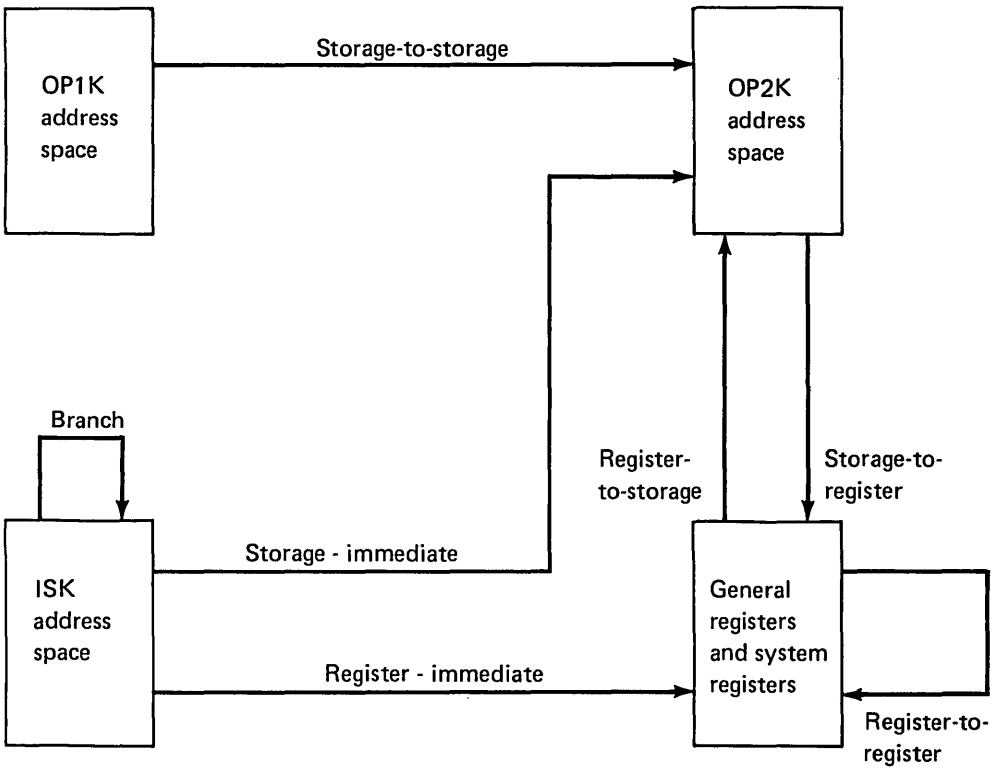
ISK≠OP1K, OP1K=OP2K. Instruction fetches occur in the ISK address space. Data access occurs in the OP2K address space.

ISK=OP1K, OP1K≠OP2K. Refer to Figure 5-2 for this example.

I/O operations that access main storage also use an address key. Cycle-steal operations (read or write) use the cycle-steal address key specified within the device control block. An address key of 0 is used when the device fetches the device control block. Direct program control (DPC) operations that write data to storage use the OP2K address key. The cycle-steal and DPC operations are explained in Chapter 4.

Other defined usage of the address key register are as follows:

- All indirect access for branching uses the ISK.
- Effective-address generation occurs in the address space of the particular data operand. The appended words in the instruction are accessed by the ISK.
- Storage access from the console is defined by the console address key register. Stop on address is based on the Stop On Address key when the translator is enabled.
- System reset and IPL set all address keys and the EOS bit to 0's.



Assembler syntax for address spaces

ISK	OP1K	OP2K	Example instructions	
	addr5	addr4	AW	addr5,addr4
	(reg)	(reg)	MVFD	(reg),(reg)
Bits 13-15 of AKR			MVBI	byte,reg
Bits 13-15 of AKR			B	longaddr*

*Indirect addressing.

Notes:

1. OP1K is used for the source operand in storage-to-storage operations.
2. OP2K is used for storage data access in all other operations (excluding branch/jump).
3. ISK (bits 13-15 of the AKR) is used for instruction fetch and branch/jump operations.

Figure 5-2. Data movement in address spaces when ISK=OP1K, OP1K≠OP2K

Address Key Values After Interrupts

When priority or class interrupts occur, certain values are set in the address keys of the affected AKR. These values anticipate the address spaces that the programmer might need for interrupt processing. The following chart shows the resulting AKR for each type of interrupt.

	Resulting AKR values			
Interrupt	**EOS**	**OP1K**	**OP2K**	**ISK**
Priority	0	0	0	0
Supervisor call	0	Note 1	0	0
Machine check	0	Note 2	0	0
Program check	0	Note 2	0	0
Soft-exception trap	0	Note 1	0	0
Trace	0	Note 3	0	0
Console	0	0	0	0
Power/thermal warning	0	0	0	0

Notes:

1. OP1K is set to the preceding key contained in OP2K.
2. OP1K is set to the last active processor address key.
3. OP1K is set to the preceding key contained in the ISK.

All interrupt service routines reside in address space 0; therefore, the ISK and OP2K are set to 0's when an interrupt occurs. Necessary information for processing a specific interrupt may reside in an address space other than 0. The address key related to the particular interrupt is placed in OP1K. The OP1K is set in anticipation of a storage-to-storage move of information from the interrupting address space to address space 0.

Note: Class interrupts cause a hardware-controlled storing of a level status block. This operation uses address key 0.

Chapter 6. Clock/Comparator

A clock/comparator is incorporated into the basic instruction set of the processor. The clock is a single 32-bit register, which is incremented at 1-millisecond intervals. This allows time to be represented up to 49 days, 17 hours, 2 minutes, and 47.295 seconds before the register wraps. The comparator is a 32-bit register, which can be set to a predetermined value by the Set Comparator instruction. The clock value and the comparator value are compared to determine when a class interrupt should occur. If the clock value is greater than or equal to the comparator value, a clock class interrupt is generated. (Refer to Chapter 3 for a detailed discussion of clock class interrupt.) The clock/comparator combination can be used for a predetermined time control operation. Four instructions are provided to set or copy the clock and comparator:

- Set Clock (SECLK)
- Set Comparator (SECMP)
- Copy Clock (CPCLK)
- Copy Comparator (CPCMP)

The setting of the clock and comparator is allowed only when the processor is in supervisor state; the copying of the clock and comparator is allowed in problem state. Detailed descriptions of the instructions are contained in Chapter 8.

Clock/Comparator Features

Clock

The features of the clock are:

- 32-bit register (counter)
- 1-millisecond resolution
- Set/Copy Clock instructions
- No timer external sync or 60-Hz sync

  Note: 60-Hz synchronization is provided on some 60-Hz processors. Refer to individual processor publications for further information.
- No alter/display from console
- No interrupt on clock overflow
- Set to 0 by power-on reset
- Continuously running (no ability to start/stop)

Comparator

The features of the comparator are:

- 32-bit register
- Set/Copy Comparator instructions
- No alter/display from console
- Set to 0 by power-on reset
- Clock class interrupt is disabled by power-on reset or system reset
- Clock class interrupt enabled by a Set Comparator instruction
- A clock class interrupt is held pending when the summary mask (bit 11 of the level status register (LSR)) is disabled.

Note: System reset does not affect the clock's operation or the comparator's value. Switching back and forth between power-good and battery backup or multiple IPL sequences does not affect the clock's operation.

Chapter 7. Floating-Point Feature

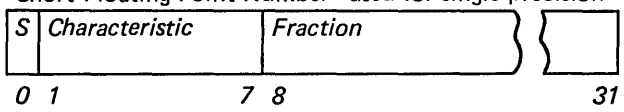
The floating-point feature includes the resources to execute all floating-point instructions and four 64-bit floating-point registers for each of the four priority interrupt levels in the processor. The floating-point instruction set performs calculations on operands with a wide range of magnitude. Results of these calculations are scaled to preserve precision. The floating-point registers are provided to avoid unnecessary storing and loading operations for results and operands.

A floating-point number consists of a signed exponent and a signed fraction. The quantity expressed by this number is the product of the fraction and the number 16 raised to the power of the exponent. The exponent is expressed in excess 64 binary notation; the fraction is expressed as a hexadecimal number having a radix point to the left of the high-order hexadecimal digit.

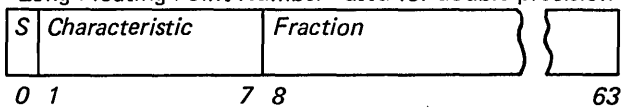
Data Format

Two fixed-length formats (short and long) may be used for floating-point data:

Short Floating-Point Number—used for single precision



Long Floating-Point Number—used for double precision



Both formats may be used in main storage and in the floating-point registers. The first bit in either format is the sign bit (S). The subsequent seven bit positions are occupied by the characteristic. The fraction field may have either six or 14 hexadecimal digits.

The entire set of floating-point instructions is available for both short and long operands. When single precision (short format) is specified, all operands and results are 32-bit floating-point words. With two exceptions, the rightmost 32-bits of the floating-point registers do not participate in single precision operations and are not changed by the operations. The two exceptions are:

- The product in multiply operations (it is a 64-bit floating-point word and occupies a full register)
- A storage to register move (the low-order 32-bits are set to 0's.

When double precision (long format) is specified, all operands and results are 64-bit floating-point words.

Although final results in short precision have six fraction digits, intermediate results in add and subtract operations may extend to seven fraction digits. The low-order digit of a seven-digit fraction is called the guard digit and serves to increase the precision of the final result. Intermediate results in long precision may extend to 15 fraction digits, with the 15th digit being the guard digit.

Number Representation

Floating-Point Numbers

The fraction of a floating-point number is expressed in hexadecimal digits. The radix point of the fraction is assumed to be immediately to the left of the high-order fraction digit. To provide the proper magnitude for the floating-point number, the fraction is considered to be multiplied by a power of 16. The characteristic portion, bits 1–7 of both floating-point formats, indicates this power. The bits within the characteristic field can represent numbers from 0 through 127. To accommodate large and small magnitudes, the characteristic is formed by adding 64 to the actual exponent. The range of the exponent is thus −64 through +63. This technique produces a characteristic in excess 64 notation.

Both positive and negative quantities have a true fraction, the difference in sign being indicated by the sign bit. The number is positive or negative accordingly as the sign bit is 0 or 1.

A floating-point number with zero characteristic, zero fraction, and plus sign is called a true zero. A true zero may arise as the result of an arithmetic operation because of the particular magnitude of the operands. A result is forced to be true zero when an exponent underflow occurs or when a result fraction is 0.

Conversion Example

Convert the decimal number 149.25 to a short-precision floating-point operand.

1. The number is converted to a decimal integer and a decimal fraction.

   149.25=149 plus 0.25
2. The decimal integer is converted to its hexadecimal representation.

   $149_{10}=95_{16}$
3. The decimal fraction is converted to its hexadecimal representation.

   $0.25_{10}=0.4_{16}$
4. Combine the integral and fractional parts and express as a fraction times a power of 16 (exponent).

   $95.4_{16}=(0.954 \times 10^{2})_{16}$
5. The characteristic is developed from the exponent and converted to binary.

   base + exponent = characteristic
   64+2=66 (1000010)
6. The fraction is converted to binary and grouped hexadecimally.

   0.954_{16}=.1001 0101 0100

7. The characteristic and the fraction are stored in short precision format. The sign position contains the sign of the fraction.

S	Characteristic	Fraction
0	1000010	1001 0101 0100 0000 0000 0000

Binary Integers in Main Storage

Signed binary integers occupy storage in one of two fixed-length formats:

- One-word format (16 bits)
- Doubleword format (32 bits)

Both formats may be used in main storage and are automatically converted to single- or double-precision floating-point numbers during floating move and convert operations that move data from storage to a floating-point register. Negative signed binary integers are in main storage in two's complement form. They are converted to contain a true fraction. An integer may be moved from main storage to a floating-point register, without conversion, by using the floating-move instruction. In this case, the integer is assumed to be a floating-point number.

Floating move and convert operations that move data from a floating-point register to storage accomplish the reverse process; the floating-point number in the register is automatically converted to an integer. This integer result is then placed in main storage. The floating move and floating move and convert operations are fully explained in Chapter 8, "Instructions."

Normalization

A quantity can be represented with the greatest precision by a floating-point number of given fraction length when that number is normalized. A normalized floating-point number has a nonzero high-order hexadecimal fraction digit. If one or more high-order fraction digit is 0, the number is said to be unnormalized. The process of normalization consists of shifting the fraction left until the high-order hexadecimal digit is nonzero and reducing the characteristic by the number of hexadecimal digits shifted.

Normalization takes place after the multiply operations, and after the add or subtract operations if an actual subtraction has taken place. For example, +A+(−B), +A−(+B), or −A−(−B). Normalization does not take place following a true addition or division; therefore, unnormalized operands can produce an unnormalized result. Floating-point numbers in main storage are assumed to be normalized.

Programming Considerations

Floating-Point Feature Not Installed

An attempt to execute a floating-point instruction when the feature is not installed results in a soft-exception-trap interrupt with invalid function set in the PSW. There are two exceptions to this rule:

- When attempting to execute a floating-point privileged instruction while in problem state, a program-check interrupt occurs with privilege violate set in the PSW.

- If the effective address is odd when attempting to execute a floating-point instruction, a program-check interrupt occurs with specification check set in the PSW.

Floating-Point Registers

Four floating-point registers are provided for each of the four priority interrupt levels associated with the processor. Floating-point register selection is determined by the R-field of the instruction. The R-field in the instruction format consists of two bits and may be labeled R, R1, and R2, as required by the individual instruction.

R-field value	Floating-point register selected
00	Register 0
01	Register 1
10	Register 2
11	Register 3

Note: The floating-point registers are not affected by a reset and must be initialized by the programmer.

Arithmetic Indicators

The processor indicators (carry, overflow, zero, negative, and even) are set or reset at the end of each floating-point instruction. Details of indicator settings are contained in the individual instruction description in Chapter 8.

Floating-Point Exceptions

Floating-point underflow, overflow, and divide check are considered exception conditions. When these conditions are recognized, a soft-exception-trap class interrupt occurs with floating-point exception (bit 5) set in the PSW. Note that the soft-exception-trap interrupt does not occur during floating-point compare operations. The overflow, carry, and even indicators are set as follows:

- The overflow indicator is set to 1 by an overflow, underflow, or divide check.
- The carry indicator is set to 1 by a divide check.
- The even indicator is set to 1 by an underflow.

Floating-Point Overflow

Add Operations. An exponent overflow occurs when a carry from the high-order position of the intermediate-sum fraction causes the characteristic of the sum to exceed 127. The operation is completed by forcing the characteristic to 127 and the result fraction bits to all 1's.

Subtract and Compare Operations. An exponent overflow occurs when a borrow from the high-order position of the intermediate-sum fraction causes the characteristic of the sum to exceed 127. The operation is completed by forcing the characteristic to 127 and the result fraction bits to all 1's.

Divide Operations. An exponent overflow occurs when the final-quotient characteristic exceeds 127. The operation is completed by forcing the characteristic to 127 and the result fraction bits to all 1's.

Multiply Operations. An exponent overflow occurs when the characteristic of the normalized product exceeds 127 and the fraction is not 0. The operation is completed by forcing the characterisitc to 127 and the result fraction bits to all 1's.

Floating-Point Underflow

Add Operations. An exponent underflow occurs when the characteristic of the normalized sum is less than 0 and the fraction is not 0. The result sign, characteristic, and fraction are forced to 0's.

Subtract and Compare Operations. An exponent underflow occurs when the characteristic of the normalized sum is less than 0 and the fraction is not 0. The result sign, characteristic, and fraction are forced to 0's.

Divide Operations. An exponent underflow occurs when the characteristic of the normalized quotient is less than 0 and the fraction is not 0. The result sign, characteristic, and fraction are forced to 0's.

Multiply Operations. An exponent underflow occurs when the characteristic of the normalized product is less than 0 and the fraction is not 0. The result sign, characteristic, and fraction are forced to 0's.

Divide Check

Divide Operations. A divide check occurs when division by 0 is attempted. The dividend is not changed.

Floating-Point Instructions

The floating-point instruction set provides a variety of instructions that deal with single- or double-precision floating-point data. The main categories are:

- Arithmetic instructions (add, subtract, multiply, divide, and compare)
- Data movement instructions (with or without conversion of binary integers)

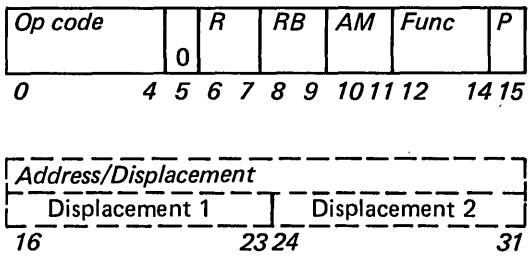
Two privileged instructions are also provided for interrogation of the floating-point registers. They are Copy Floating Level Block (CPFLB) and Set Floating Level Block (SEFLB).

All floating-point instructions use the floating-point registers. One group of instructions (storage/floating-point register) specifies a register for one operand, and an effective main storage address for the other operand. Another group (floating-point register to floating-point register) specifies registers for both operands.

Instruction Formats

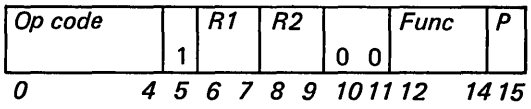
Arithmetic and data movement instructions use the following two formats:

Storage/Floating-Point Register



- The op-code field specifies the floating-point operation.
- The R-field specifies a floating-point register.
- The function field designates the function to be performed (add, subtract, multiply, divide, move, or move and convert).
- The RB and AM fields designate the effective address argument. Refer to "Effective-Address Generation" in Chapter 2 for additional information.
- The P-field designates precision of floating-point data. A 0 denotes single precision; a 1 denotes double precision.
- The second word (bits 16–31) is the address mode appended word for an AM field equal to 10 or 11.

Floating-Point Register/Floating-Point Register



- The op-code field specifies the floating-point operation.
- The R1 and R2 fields specify floating-point registers.
- Bits 10–11 designate the function modifier. These bits are not used and must be set to 0's to avoid future code obsolescence.
- The function field designates the function to be performed (add, subtract, multiply, divide, move, or compare).

  Note: To avoid future code obsolescence, function field bit combinations equal to 110 and 111 must not be used.
- The P-field designates the precision of floating-point data. A 0 denotes single precision; a 1 denotes double precision.

Another instruction format is used for the two privileged instruction (CPFLB and SEFLB). The three-bit R-field associated with this format specifies a processor general register (0–7). Refer to the individual instructions in Chapter 8 for the complete format.

Note: The instruction formats are also shown in Appendix A of this manual.

Exception Conditions

Exception conditions that might occur during instruction execution are shown with each instruction description.

Program-Check Conditions

Specification Check

A program-check class interrupt occurs with specification check (bit 0) set in the PSW.

Invalid Storage Address

A program-check class interrupt occurs with invalid storage address (bit 1) set in the PSW.

Note: If the instruction uses an AM field equal to 01, the instruction is terminated if the RB register is incremented. Refer to "Additional Error Information" in this chapter for details.

Privilege Violate

A program-check class interrupt occurs with privileged violate (bit 2) set in the PSW.

Protect Check

A program-check class interrupt occurs with protect check (bit 3) set in the PSW.

Note: If the instruction uses an AM field equal to 01, the instruction is terminated if the RB register is incremented. Refer to "Additional Error Information" in this chapter for details.

Soft-Exception Trap Conditions

Invalid Function

A soft-exception-trap class interrupt occurs with invalid function (bit 4) set in the PSW. For storage-to-storage instructions, the main storage address loaded into register 7 is the calculated effective address of data operand 2. For register-to-register instructions, the address of the attempted instruction is loaded into register 7.

Floating-Point Exception

A soft-exception-trap class interrupt occurs with floating-point exception (bit 5) set in the PSW.

Note: The resulting class interrupt causes the contents of the storage address register (SAR) to be loaded into general register 7. SAR contains either the calculated effective address of data operand 2, or the address of the attempted instruction for register-to-register operations.

Additional Error Information

The storage to register instructions use an AM field and an RB field for effective address generation. During normal operation, if no errors occur, the RB register is incremented by the number of bytes in the storage operand if the AM field is equal to 01. If an invalid storage address , a protect check, or a specification check occurs when the AM field is equal to 01, the RB register is incremented by 2 for CPFLB and SEFLB, or incremented by 1 for all other storage to register instructions.

Single Precision

Addition

Addition of two floating-point numbers is based on characteristic comparison and fraction addition. The characteristics of the two operands are compared, and the fraction accompanying the smaller characteristic is shifted right, with its characteristic increased by 1 for each hexadecimal digit shifted, until the two characteristics are equal.

When an operand is shifted right during alignment, the leftmost hexadecimal digit of the field shifted out is retained as a guard digit. The operand that is not shifted is considered to be extended with a low-order 0. Both operands are considered to be extended with low-order 0's when no alignment shift occurs. The 28-bit fractions are then added algebraically to form an intermediate sum.

The intermediate-sum fraction consists of seven hexadecimal digits and a possible carry. If a carry is present, the sum is shifted right one digit position, to make room for the carry, and the characteristic is increased by 1.

If the operand signs are unlike (resulting in a subtraction) and the fraction is not 0, normalization takes place. The intermediate sum is shifted left as necessary to form a normalized number. Vacated low-order digit positions are filled with 0's, and the characteristic is reduced by the number of hexadecimal digits shifted. The intermediate-sum fraction is subsequently truncated to the proper result fraction length of six hexadecimal digits.

Subtraction

Subtraction of two floating-point numbers is based on characteristic comparison and fraction subtraction. The characteristics of the two operands compared, and the fraction accompanying the smaller characteristic is shifted right, with its characteristic increased by 1 for each hexadecimal digit shifted, until the two characteristics are equal.

When an operand is shifted right during alignment, the leftmost hexadecimal digit of the field shifted out is retained as a guard digit. The operand that is not shifted is considered to be extended with low-order 0's when no alignment shift occurs. The 28-bit fractions are then subtracted algebraically to form an intermediate sum.

The intermediate-sum fraction consists of seven hexadecimal digits and a possible borrow. If a borrow is present, the sum is shifted right one digit position, and the characteristic is increased by 1.

If a true subtraction is performed and the fraction is not 0, normalization takes place. The intermediate sum is shifted left as necessary to form a normalized number. Vacated low-order digit positions are filled with 0's

and the characteristic is reduced by the number of hexadecimal digits shifted. The intermediate-sum fraction is subsequently truncated to the proper result-fraction length of six hexadecimal digits.

Multiplication

Multiplication of two floating-point numbers is based on exponent addition and fraction multiplication. The operands are assumed to be normalized. The sum of the characteristics of the operands less 64 is used as the characteristic of the intermedite product. When the result is normalized without requiring any post-normalization, the intermediate-product fraction is the result fraction, and the intermediate-product characteristic becomes the final-product characteristic. When the intermediate-product fraction has one leading 0-digit, it is shifted left one digit position and the intermediate-product characteristic is reduced by 1.

The multiplier and multiplicand have six-digit fractions. The product fraction has 14 digits. The two low-order fraction digits are always 0's, unless overflow occurs.

Division

Division of two floating-point numbers is based on characteristic subtraction and fraction division. The operands are assumed to be normalized. The difference between the dividend and divisor characteristics plus 64 is used as the characteristic of the intermediate quotient.

The sign of the quotient is determined by the rules of algebra unless the quotient is made a true zero; in this case, the sign is made plus.

All dividend and divisor fraction digits participate in forming the fraction of the quotient. The quotient fraction will be a 24-bit normalized result if the dividend and the divisor are normalized.

Double Precision

Addition

Addition of two floating-point numbers is based on characteristic comparison and fraction addition. The characteristics of the two operands are compared and the fraction accompanying the smaller characteristic is shifted right, with its characteristic increased by 1 for each hexadecimal digit shifted, until the two characteristics are equal. The fractions are then added algebraically to form an intermediate sum.

When an operand is shifted right during alignment, the last hexadecimal digit shifted out of the 64-bit register is preserved as a guard digit, with 15 digits participating in the arithmetic.

The long intermediate-sum fraction consists of 15 hexadecimal digits and a possible carry. If a carry is present, the sum is shifted right by one position, and the characteristic is increased by 1.

If the operand signs are unlike (resulting in a subtraction) and the fraction is not 0, normalization takes place. The intermediate sum, including the guard digit, is shifted left as necessary to form a normalized number. Vacated low-order digit positions are filled with 0's, and the characteristic is reduced by the number of hexadecimal digits shifted.

Subtraction

Subtraction of two floating-point numbers is based on characteristic comparison and fraction subtraction. The characteristics of the two operands are compared and the fraction accompanying the smaller characteristic is shifted right, with its characteristic increased by 1 for each hexadecimal digit shifted, until the two characteristics are equal.

When an operand is shifted right during alignment, the last hexadecimal digit shifted out fo the 64-bit register is preserved as a guard digit, with 15 digits participating in the arithmetic. The fractions are then subtracted algebraically to form an intermediate sum.

The long intermediate-sum fraction consists of 15 hexadecimal digits and a possible borrow. If a borrow is present, the sum is shifted right one digit position, and the characteristic is increased by 1.

If a true subtraction is performed and the fraction is not 0, normalization takes place. The intermediate sum, including the guard digit, is shifted left as necessary to form a normalized number. Vacated low-order digit positions are filled with 0's, and the characteristic is reduced by the number of hexadecimal digits shifted.

Multiplication

Multiplication of two floating-point numbers is based on exponent addition and fraction multiplication. The operands are assumed to be normalized. The sum of the characteristics of the operands less 64 is used as the characteristic of the intermediate product. When the result is normalized without requiring any post-normalization, the intermediate-product fraction is the result fraction, and the intermediate-product characteristic becomes the final-product characteristic. When the intermediate-product fraction has one leading 0-digit, it is shifted left one digit position and the intermediate-product characteristic is reduced by 1. The multiplier and multiplicand fractions have 14 digits and the result-product fraction is truncated to 14 digits.

Division

The division of two floating-point numbers is based on characteristic subtraction and fraction division. The operands are assumed to be normalized. The difference between the dividend and divisor characteristics plus 64 is used as the characteristic of the intermediate quotient.

All divident and divisor fraction digits participate in forming the fraction of the quotient. The quotient fraction will be a 56-bit normalized result if the dividend and divisor are normalized.

The sign of the quotient is determined by the rules of algebra unless the quotient is made a true zero; in this case, the sign is made plus.

Chapter 8. Instructions

This chapter contains instruction descriptions. The instructions are listed in alphabetical sequence based on assembler mnemonics.

Each instruction description contains:

- Assembler syntax
- Instruction format
- Explanation
- Indicator settings
- Exception conditions (which occur within each instruction).

In the instruction illustration, the field names R1 and R2 do not correspond to general register names or operand placement within assembler syntax. Refer to "Program Execution" in Chapter 2 for an explanation of the relationship between assembler syntax and machine-language instruction formats. For a detailed discussion of assembler syntax and operand usage, refer to the publication that describes the assembler program installed on each individual system. A listing of these publications can be found in the *IBM Series/1 Graphic Bibliography*, GA34-0055.

Under program-check conditions, "instruction is suppressed" means that the error condition was detected prior to the modification of any software accessible register or storage locations; "instruction is terminated" means that the error condition was detected after the modification of certain software accessible registers or storage locations.

For additional information, refer to:

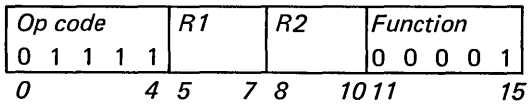
- "Effective-Address Generation" in Chapter 2 for a detailed explanation of the standard methods of deriving effective addresses.
- "Indicator Bits" in Chapter 2 for a detailed explanation of indicator settings.
- "Class Interrupts" in Chapter 3 for a detailed description of exception conditions.
- Appendix A for instruction formats grouped by operation codes.
- Appendix B for definitions of the assembler syntax.

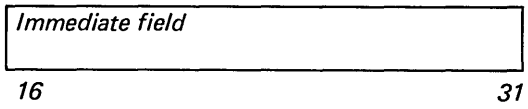
AA

Add Address (AA)

Register Immediate Long Format

AA raddr,reg[,reg]





The immediate field (an address value) is added to the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed if the R1 and R2 fields do not specify the same register.

Indicators

Carry. If a carry is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no carry is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the sum cannot be represented in one word; that is, if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

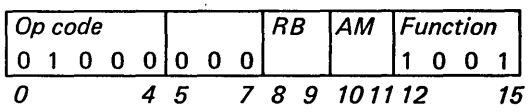
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

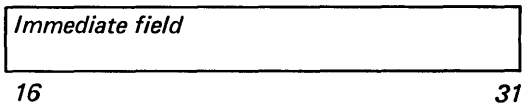
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Storage Immediate Format

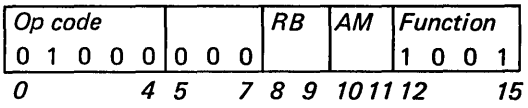
AA raddr,addr4

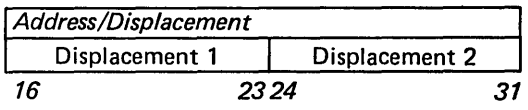
Format without appended word for effective addressing (AM = 00 or 01)

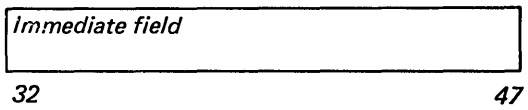




Format with appended word for effective addressing (AM = 10 or 11)







The immediate field (an address value) is added to the contents of the location specified by the effective address. The result replaces the contents of the storage location specified by the effective address. The immediate operand is not changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Carry. If a carry is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no carry is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the sum cannot be represented in one word; that is, if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

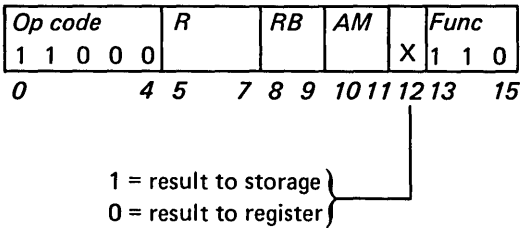
The instruction is terminated.

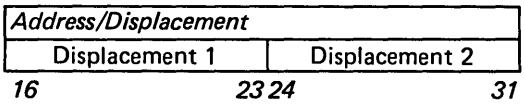
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

AB

Add Byte (AB)

AB reg,addr4
addr4,reg





An add operation is performed between the least-significant byte of the register specified by the R-field and the location specified by the effective address in main storage. The source operand and high-order byte of the register are not changed.

Bit 12 of the instruction specifies the destination of the result.

Indicators

Carry. If a carry is detected out of the high-order bit position of the byte, the carry indicator is set to 1. If no carry is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the sum cannot be represented in one byte; that is, if the sum is less than -2^7 or greater than $+2^7-1$.

If an overflow occurs, the result contains the correct low-order eight bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

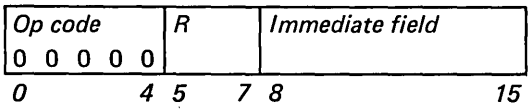
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Add Byte Immediate (ABI)

ABI byte,reg



The immediate field is expanded to 16 bits by sign propagation to the eight high-order bits. The field is then added to the contents of the register specified by the R-field. The result is placed in the register specified by the R-field.

Indicators

Carry. If a carry is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no carry is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the sum cannot be represented in one word; that is, if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

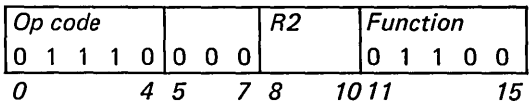
Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data accessed from a storage area not assigned to the current operation.

Add Carry Register (ACY)

ACY reg



The value of the carry indicator is added to the contents of the register specified by the R2 field, and the result is placed in the register specified by the R2 field.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Programming Note: This instruction can be used when adding multiple word operands. See "Indicators—Multiple Word Operands" in Chapter 2.

Indicators

Carry. If a carry is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no carry is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the sum cannot be represented in one word; that is, if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even. The even indicator is not changed.

Negative. The negative indicator is changed to reflect the result.

Zero. If on at entry, the zero indicator is changed to reflect the result. If off at entry, it remains off.

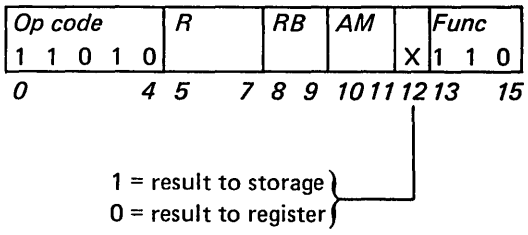
Program-Check Conditions

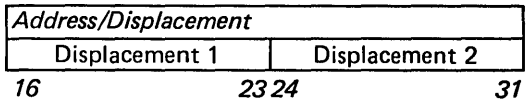
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Add Doubleword (AD)

Register/Storage Format

AD reg,addr4
addr4,reg





An add operation is performed between the register pair specified by the R-field and R+1 field and the doubleword in main storage specified by the effective address. The source operand is not changed.

If the R-field value is 7, register 7 and register 0 are used.

Bit 12 of the instruction specifies the destination of the result.

Indicators

Carry. If a carry is detected out of the high-order bit position of the doubleword, the carry indicator is set to 1. If no carry is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the sum cannot be represented in the doubleword; that is, if the sum is less than -2^{31} or greater than $+2^{31}-1$.

If an overflow occurs, the result contains the correct low-order 32 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

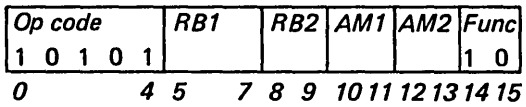
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

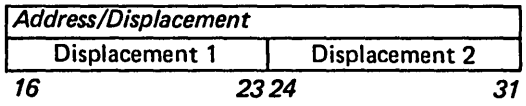
The instruction is terminated.

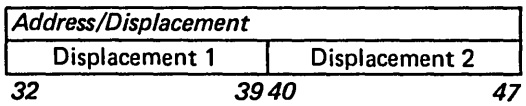
Specification Check. The effective address or indirect address results in an even-byte boundary violation.

Storage/Storage Format

AD addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. Doubleword operand 1 is added to doubleword operand 2. The result replaces operand 2. Operand 1 is not changed.

Indicators

Carry. If a carry is detected out of the high-order bit position of the doubleword, the carry indicator is set to 1. If no carry is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the sum cannot be represented in the doubleword; that is, if the sum is less than -2^{31} or greater than $+2^{31}-1$.

If an overflow occurs, the result contains the correct low-order 32 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

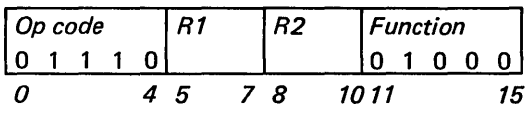
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Add Word (AW)

Register/Register Format

AW reg,reg



The contents of the register specified by the R1 field are added to the contents of the register specified by the R2 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed if the R1 and R2 fields do not specify the same register.

Indicators

Carry. If a carry is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no carry is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the sum cannot be represented in one word; that is, if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

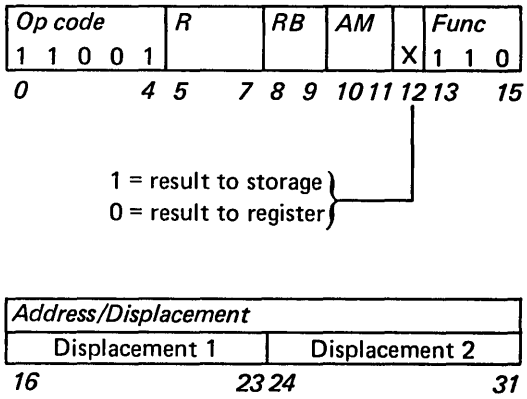
Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Register/Storage Format

AW reg,addr4
addr4,reg



An add operation is performed between the register specified by the R-field and the location specified by the effective address in main storage. The source operand is not changed.

Bit 12 of the instruction specifies the destination of the result.

Indicators

Carry. If a carry is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no carry is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the sum cannot be represented in one word; that is, if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

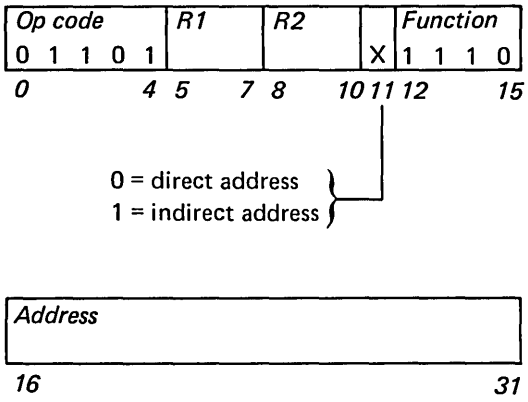
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Storage/Register Long Format

AW longaddr,reg



The contents of the main storage location specified by the effective address are added to the contents of the register specified by the R1 field. The result is placed in the register specified by the R1 field.

The effective main storage address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals 0, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0 (direct address). The result from step 1 is the effective address.

   Bit 11=1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Indicators

Carry. If a carry is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no carry is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the sum cannot be represented in one word; that is, if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

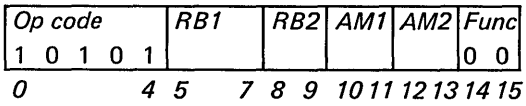
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation

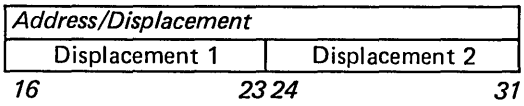
Specification Check. The effective address or indirect address results in an even-byte boundary violation.

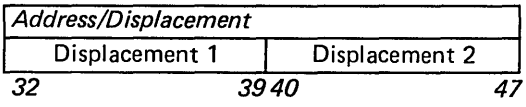
AW

Storage/Storage Format

AW addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. Word operand 1 is added to word operand 2. The result replaces operand 2. Operand 1 is not changed.

Indicators

Carry. If a carry is detected out of the high-order bit position of the word, The carry indicator is set to 1. If no carry is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the sum cannot be represented in one word; that is, if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state the instruction:

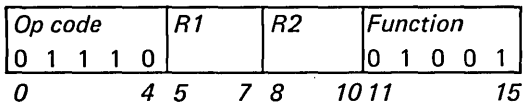
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Add Word With Carry (AWCY)

AWCY reg,reg



This instruction adds the contents of the register specified by the R1 field, the contents of the register specified by the R2 field, and the value of the carry indicator at entry.

The contents of the register specified by the R1 field are not changed if the R1 and R2 fields do not specify the same register. The final result replaces the contents of the register specified by the R2 field.

Programming Note: This instruction can be used when adding multiple word operands. Refer to "Indicators—Multiple Word Operands" in Chapter 2.

Indicators

Carry. If a carry is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no carry is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the sum cannot be represented in one word; that is, if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even. The even indicator is not changed.

Negative. The negative indicator is changed to reflect the result.

Zero. If on at entry, the zero indicator is changed to reflect the result. If off at entry, it remains off.

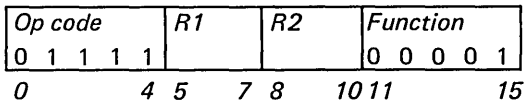
Program-Check Conditions

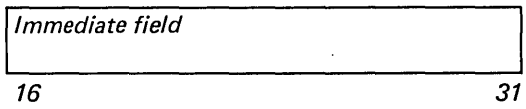
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Add Word Immediate (AWI)

Register Immediate Long Format

AWI word,reg[,reg]





The immediate field is added to the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed if the R1 and R2 fields do not specify the same register.

Indicators

Carry. If a carry is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no carry is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the sum cannot be represented in one word; that is, if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

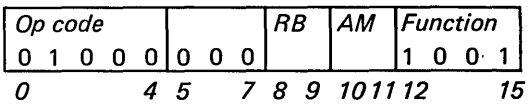
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

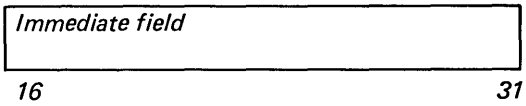
AWI

Storage Immediate Format

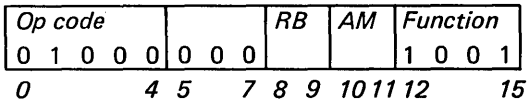
AWI word,addr4

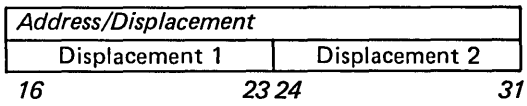
Format without appended word for effective addressing (AM = 00 or 01)





Format with appended word for effective addressing (AM = 10 or 11)





The immediate field is added to the contents of the location specified by the effective address. The result replaces the contents of the storage location specified by the effective address. The immediate operand is not changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Carry. If a carry is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no carry is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the sum cannot be represented in one word; that is, if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction;

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

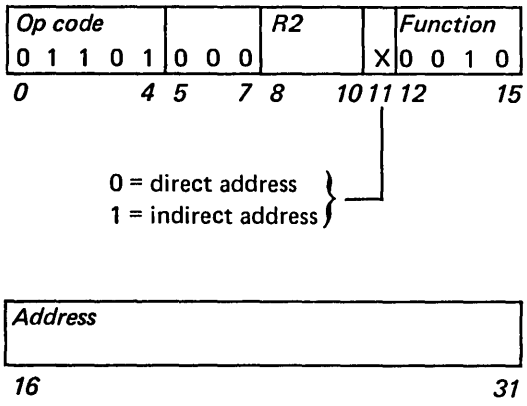
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Branch Unconditional (B)

B longaddr

Extended Assembler Mnemonic

BX vcon Branch External



An effective branch address is generated and loaded into the instruction address register. This becomes the next instruction to be fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals 0, no register contributes to the address generation. The contents of the register specified by the R2 field are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Specification Check. The effective address or indirect address results in an even-byte boundary violation.

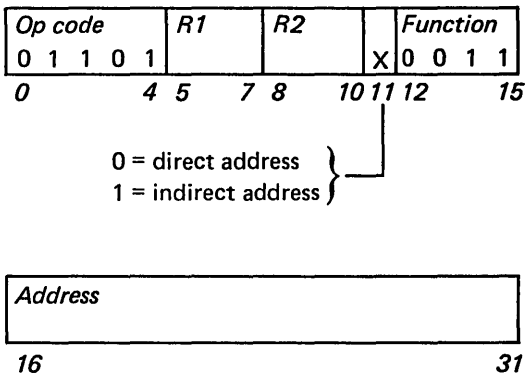
BAL

Branch and Link (BAL)

BAL longaddr,reg

Extended Assembler Mnemonic

BALX vcon,reg Branch and Link External



The updated contents of the instruction address register (the address of the next sequential instruction) are stored into the register specified by the R1 field. An effective branch address is then generated and loaded into the instruction address register. This becomes the next instruction to be fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals 0, no register contributes to the address generation. The contents of the register specified by the R2 field are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

Programming Note: If the R1 and R2 fields specify the same register, the initial contents are used in effective address computation and subsequently overwritten by the return data.

Indicators

The indicators are not changed.

Program-Check Conditions

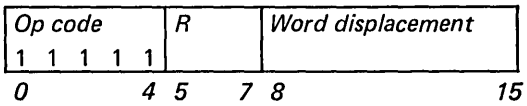
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. No branch is taken, but the contents of the register specified by the R1 field are changed. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. No branch is taken, but the contents of the register specified by the R1 field register are changed. The instruction is terminated.

Branch and Link Short (BALS)

BALS (reg,jdisp)*
(reg)*
addr*



The updated contents of the instruction address register (the location of the next sequential instruction) are stored in register 7.

Bit 8 (the leftmost bit of the word displacement field) is propagated left by seven bit positions and a 0 is appended at the low-order end; this results in a 16-bit word. (Word displacement is converted to a byte displacement.) This value is added to the contents of the register specified by the R-field to form an effective address. The contents of the storage location specified by the effective address are stored into the instruction address register, and become the address of the next instruction to be fetched.

Programming Note: If the implied register (register 7) is used as a base register, the initial contents of register 7 are used in effective-address computation and subsequently overwritten by the return data.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. Branching does not occur, but the updated instruction address is stored into register 7. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operations.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. Branching does not occur, but the updated instruction address is stored into register 7. The instruction is terminated.

BC

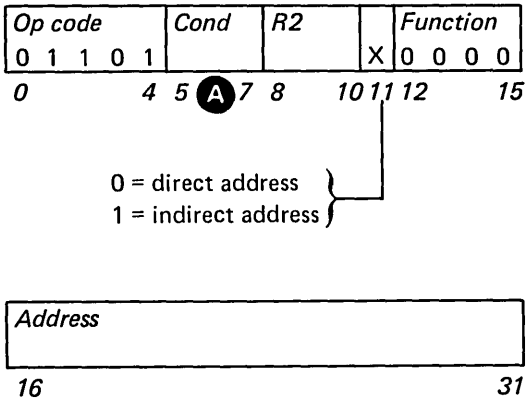
Branch on Condition (BC)

Mnemonic	Operand syntax	Instruction name	Condition field bits (see A)
BC	cond,longaddr	Branch on Condition	Any value listed below

Extended mnemonic	Operand syntax	Instruction name	Condition field bits (see A)
BE	longaddr	Branch on Equal	000
BOFF	longaddr	Branch if Off	000
BZ	longaddr	Branch on Zero	000
BP	longaddr	Branch on Positive	001
BMIX	longaddr	Branch if Mixed	001
BN	longaddr	Branch if Negative	010
BON	longaddr	Branch if On	010
BEV	longaddr	Branch on Even	011
BLT	longaddr	Branch on Arithmetically Less Than	100
BLE	longaddr	Branch on Arithmetically Less Than or Equal	101
BLLE	longaddr	Branch on Logically Less Than or Equal	110
BCY	longaddr	Branch on Carry	111
BLLT	longaddr	Branch on Logically Less Than	111

Cond field bits	Extended mnemonics: Branch	Indicators tested: 0 1 2 3 4 / E C O N Z
000	BE, BOFF, BZ	X X X X 1
	BNE, BNOFF, BNZ	X X X X 0
001	BMIX, BP	X X X 0 0
	BNMIX, BNP	X X X X 1 X X X 1 X
010	BN, BON	X X X 1 X
	BNN, BNON	X X X 0 X
011	BEV	1 X X X X
	BNEV	0 X X X X

Cond field bits	Extended mnemonics: Branch	Indicators tested: 0 1 2 3 4 / E C O N Z
100	BLT	X X 0 1 X X X 1 0 X
	BGE	X X 1 1 X X X 0 0 X
101	BLE	X X 0 1 X X X 1 0 X X X X X 1
	BGT	X X 1 1 0 X X 0 0 0
110	BLLE	X 1 X X X X X X X 1
	BLGT	X 0 X X 0
111	BCY, BLLT	X 1 X X X
	BLGE, BNCY	X 0 X X X



This instruction tests the condition of the various indicators (LSR bits 0–4). If the condition tested is met, the effective branch address is loaded into the instruction address register and becomes the next address to be fetched.

If the condition tested is not met, the next sequential instruction is fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals 0, no register contributes to the address generation. The contents of the register specified by the R2 field are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

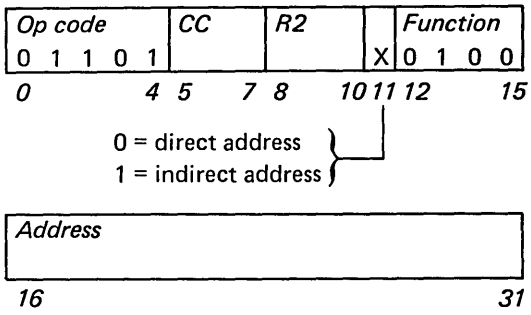
Specification Check. The effective address or indirect address results in an even-byte boundary violation.

Branch on Condition Code (BCC)

BCC cond,longaddr

Extended mnemonic

BNER longaddr Branch on Not Error (CC field = 111)



The value of the CC field is compared to the even, carry, and overflow indicators. These indicators hold the I/O condition code following an I/O instruction or an I/O interrupt.

CC bit	Indicator
5	Even
6	Carry
7	Overflow

If the conditions match, an effective branch address is generated and loaded into the instruction address register. This becomes the next instruction to be fetched.

If the conditions do not match, the next sequential instruction is fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals 0, no register contributes to the address generation. The contents of the register specified by the R2 field are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Specification Check. The effective address or indirect address results in an even-byte boundary violation.

I/O Condition Codes

The I/O condition codes are summarized in the following tables. Refer to Chapter 4 for a detailed description of each condition-code value. Some devices do not report all condition codes, refer to the specific I/O device descriptions.

Condition Codes Reported After I/O Instruction.

Condition code	Indicators: Even	Carry	Overflow	Meaning
0	0	0	0	Device not attached
1	0	0	1	Busy
2	0	1	0	Busy after reset
3	0	1	1	Command reject
4	1	0	0	Intervention required
5	1	0	1	Interface data check
6	1	1	0	Controller busy
7	1	1	1	Satisfactory

Condition Codes Reported During an I/O Interrupt.

Condition code	Indicators: Even	Carry	Overflow	Meaning
0	0	0	0	Controller end
1	0	0	1	PCI (program-controlled interrupt)
2	0	1	0	Exception
3	0	1	1	Device end
4	1	0	0	Attention
5	1	0	1	Attention and PCI
6	1	1	0	Attention and exception
7	1	1	1	Attention and device end

Branch on Not Condition (BNC)

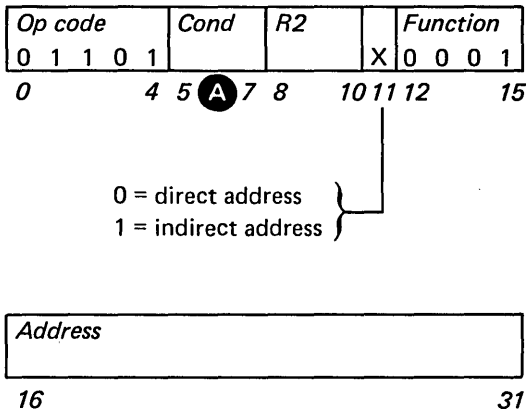
Mnemonic	Operand syntax	Instruction name	Condition field bits (see A)
BNC	cond,longaddr	Branch on Not Condition	Any value listed below

Extended mnemonic	Operand syntax	Instruction name	Condition field bits (see A)
BNE	longaddr	Branch on Not Equal	000
BNZ	longaddr	Branch on Not Zero	000
BNOFF	longaddr	Branch if Not Off	000
BNP	longaddr	Branch on Not Positive	001
BNMIX	longaddr	Branch on Not Mixed	001
BNN	longaddr	Branch on Not Negative	010
BNON	longaddr	Branch if Not On	010
BNEV	longaddr	Branch on Not Even	011
BGE	longaddr	Branch on Arithmetically Greater Than or Equal	100
BGT	longaddr	Branch on Arithmetically Greater Than	101
BLGT	longaddr	Branch on Logically Greater Than	110
BLGE	longaddr	Branch on Logically Greater Than or Equal	111
BNCY	longaddr	Branch on No carry	111

Cond field bits	Extended mnemonics: Branch	Indicators tested: 0 1 2 3 4 / E C O N Z
000	BE, BOFF, BZ	X X X X 1
	BNE, BNOFF, BNZ	X X X X 0
001	BMIX, BP	X X X 0 0
	BNMIX, BNP	X X X X 1 X X X 1 X
010	BN, BON	X X X 1 X
	BNN, BNON	X X X 0 X
011	BEV	1 X X X X
	BNEV	0 X X X X

Cond field bits	Extended mnemonics: Branch	Indicators tested: 0 1 2 3 4 / E C O N Z
100	BLT	X X 0 1 X X X 1 0 X
	BGE	X X 1 1 X X X 0 0 X
101	BLE	X X 0 1 X X X 1 0 X X X X X 1
	BGT	X X 1 1 0 X X 0 0 0
110	BLLE	X 1 X X X X X X X 1
	BLGT	X 0 X X 0
111	BCY, BLLT	X 1 X X X
	BLGE, BNCY	X 0 X X X

BNC



This instruction tests the condition of the various indicators (LSR bits 0–4). If the condition tested is met, the effective branch address is loaded into the instruction address register and becomes the next address to be fetched.

If the condition tested is not met, the next sequential instruction is fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals 0, no register contributes to the address generation. The contents of the register specified by the R2 field are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

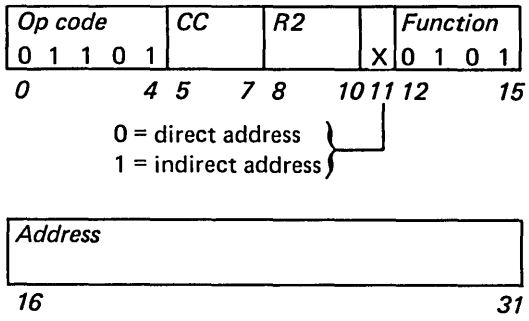
Specification Check. The effective address or indirect address results in an even-byte boundary violation.

Branch on Not Condition Code (BNCC)

BNCC cond,longaddr

Extended mnemonic

BER longaddr Branch on Error (CC field≠111)



The value of the CC field is compared to the even, carry, and overflow indicators. These indicators hold the I/O condition code following an I/O instruction or an I/O interrupt.

CC bit	Indicator
5	Even
6	Carry
7	Overflow

If the conditions do not match, an effective branch address is generated and loaded into the instruction address register. This becomes the next instruction to be fetched.

If the conditions match, the next sequential instruction is fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form main storage address. If the R2 field equals 0, no register contributes to the address generation. The contents of the register specified by the R2 field are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Specification Check. The effective address or indirect address results in an even-byte boundary violation.

I/O Condition Codes

The I/O condition codes are summarized in the following tables. Refer to Chapter 4 for a detailed description of each condition-code value. Some devices do not report all condition codes; refer to the specific I/O device descriptions.

Condition Codes Reported After I/O Instruction.

Condition code	Indicators: Even	Carry	Overflow	Meaning
0	0	0	0	Device not attached
1	0	0	1	Busy
2	0	1	0	Busy after reset
3	0	1	1	Command reject
4	1	0	0	Intervention required
5	1	0	1	Interface data check
6	1	1	0	Controller busy
7	1	1	1	Satisfactory

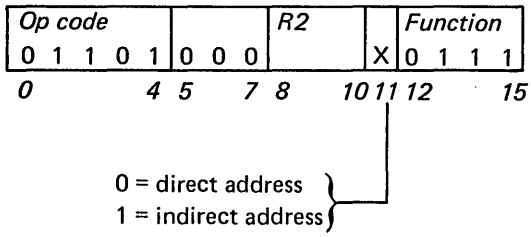
Condition Codes Reported During an I/O Interrupt.

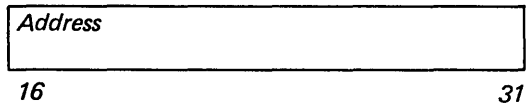
Condition code	Indicators: Even	Carry	Overflow	Meaning
0	0	0	0	Controller end
1	0	0	1	PCI (program-controlled interrupt)
2	0	1	0	Exception
3	0	1	1	Device end
4	1	0	0	Attention
5	1	0	1	Attention and PCI
6	1	1	0	Attention and exception
7	1	1	1	Attention and device end

BNOV

Branch on Not Overflow (BNOV)

BNOV longaddr





The overflow indicator is tested. If the indicator is off, the effective branch address is loaded into the instruction address register and becomes the next address to be fetched.

If the overflow indicator is on, the next sequential instruction is fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals 0, no register contributes to the address generation. The contents of the register specified by the R2 field are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

The indicators are not changed.

Program-Check Conditions

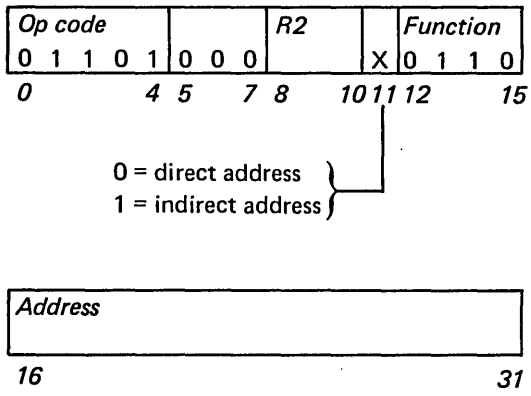
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Specification Check. The effective address or indirect address results in an even-byte boundary violation.

Branch on Overflow (BOV)

BOV longaddr



The overflow indicator is tested. If the indicator is on, the effective branch address is loaded into the instruction address register and becomes the next address to be fetched.

If the overflow indicator is off, the next sequential instruction is fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals 0, no register contributes to the address generation. The contents of the register specified by the R2 field are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

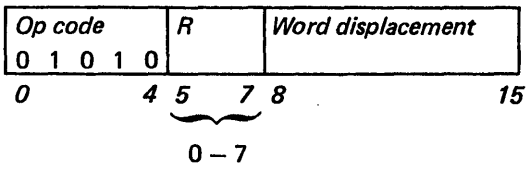
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Specification Check. The effective address or indirect address results in an even-byte boundary violation.

BXS

Branch Indexed Short (BXS)

BXS (reg$^{1-7}$,jdisp)
(reg$^{1-7}$)
addr



Bit 8 (the leftmost bit of the word displacement field) is propagated left seven bit positions and a 0 is appended at the low-order end; this results in a 16-bit word. (Word displacement is converted to a byte displacement.) This value is added to the contents of the register specified by the R-field. The result is stored into the instruction address register, and becomes the address of the next instruction to be fetched.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

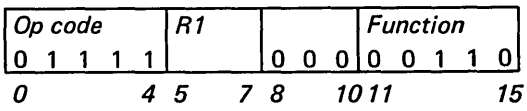
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

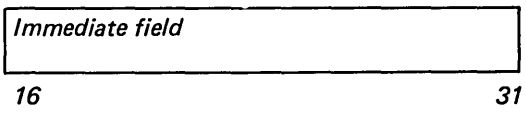
Specification Check. The effective address or indirect address results in an even-byte boundary violation.

Compare Address (CA)

Register Immediate Long Format

CA raddr,reg





The immediate field (an address value) is subtracted from the contents of the register specified by the R1 field. The contents of the register specified by the R1 field are not changed.

Bits 8–10 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

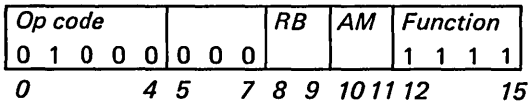
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

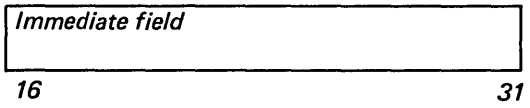
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Storage Immediate Format

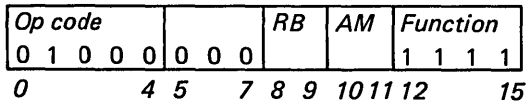
CA raddr,addr4

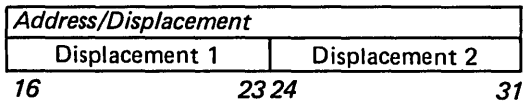
Format without appended word for effective addressing (AM = 00 or 01)

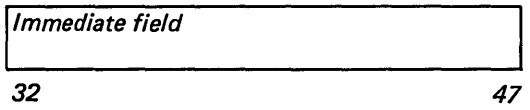




Format with appended word for effective addressing (AM = 10 or 11)







The immediate word (an address value) is subtracted from the contents of the location specified by the effective address. Neither operand is changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

CA

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the systems.

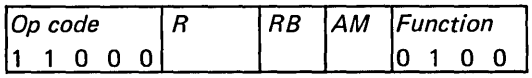
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

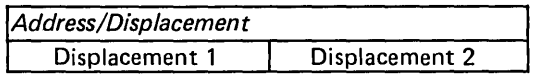
Specification Check. The effective address or indirect address results in an even-byte boundary violation.

Compare Byte (CB)

Register/Storage Format

CB addr4,reg





The contents of the location specified by the effective address in main storage are subtracted from the least-significant byte of the register specified by the R-field. Neither operand is changed.

Bit 12 of the instruction is not used and must be set to 0 to avoid future code obsolescence.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the byte, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one byte; that is, if the difference is less than -2^7 or greater than $+2^7-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

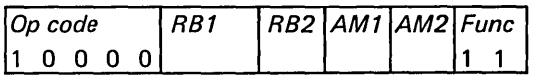
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

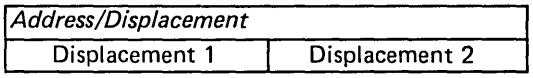
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

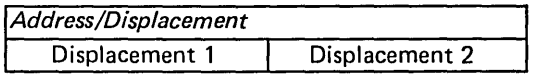
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Storage/Storage Format

CB addr5,addr4







The address arguments generate the effective addresses of the two operands in main storage. Byte operand 1 is subtracted from byte operand 2. Neither operand is changed.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the byte, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one byte; that is, if the difference is less than -2^7 or greater than $+2^7-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

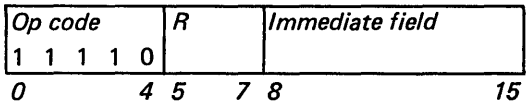
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Compare Byte Immediate (CBI)

CBI byte,reg



The immediate field is extended to 16 bits by sign propagation to the eight high-order bit positions. The result is subtracted from the contents of the register specified by the R-field. Neither operand is changed.

Note: If a byte of data from storage is to be compared with a CBI instruction, an MVB (storage to register) instruction must be performed first.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result.

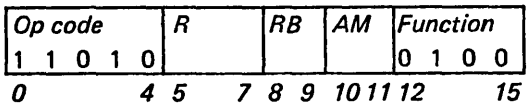
Program-Check Conditions

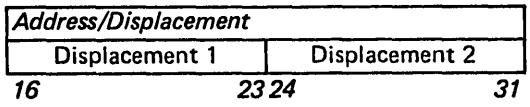
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Compare Doubleword (CD)

Register/Storage Format

CD addr4,reg





The contents of the doubleword in main storage specified by the effective address are subtracted from the contents of the register pair specified by the R-field and R+1 field. Neither operand is changed.

If the R-field value is 7, registers 7 and 0 are used.

Bit 12 of the instruction is not used and must be set to 0 to avoid future code obsolescence.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the doubleword, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in the doubleword; that is, if the difference is less than -2^{31} or greater than $+2^{31}-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

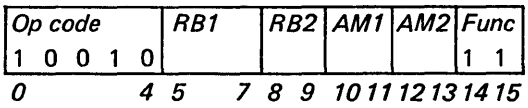
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

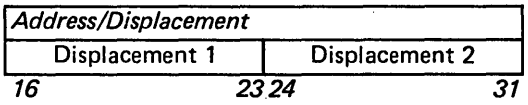
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

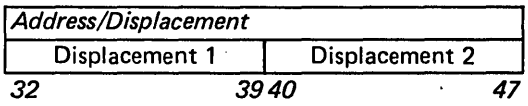
CD

Storage/Storage Format

CD addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. Doubleword operand 1 is subtracted from doubleword operand 2. Neither operand is changed.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the operand, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one doubleword; that is, if the difference is less than -2^{31} or greater than $+2^{31}-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

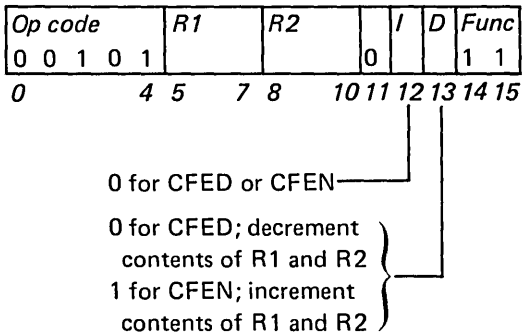
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Compare Byte Field Equal and Decrement (CFED)

Compare Byte Field Equal and Increment (CFEN)

CFED (reg),(reg)
CFEN (reg),(reg)



This instruction compares two fields in main storage on a byte-for-byte basis. Register 7 contains the number of bytes to be compared. This number is decremented after each byte is compared.

The register specified by the R1 field contains the address of operand 1. The register specified by the R2 field contains the address of operand 2. Operand 1 is subtracted from operand 2, but neither operand is changed. After each byte is compared, the addresses in the registers specified by the R1 and R2 fields are incremented or decremented (determined by bit 13 of the instruction). The operation terminates when either:

1. An equal condition is detected, or
2. The number of bytes specified in register 7 has been compared.

When an equal condition occurs, the addresses in the registers point to the next operands to be compared, but the count in register 7 is not updated.

Bit 11 of the instruction is not used and must be set to 0 to avoid future code obsolescence.

See "Scan Byte Field Equal and Decrement (SFED)" and "Scan Byte Field Equal and Increment (SFEN)" for other versions of this machine instruction.

Notes:

1. If the specified count in register 7 is 0, the instruction performs no operation (no-op).
2. Variable-field-length instructions can be interrupted. When this occurs and the interrupted level resumes operation, the processor treats the incomplete instruction as a new instruction, with the remaining byte count specified in register 7.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the byte, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one byte; that is, if the difference is less than -2^7 or greater than $+2^7-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Function. Register 7 is specified in the R1 or R2 field of the instruction. The instruction is terminated.

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

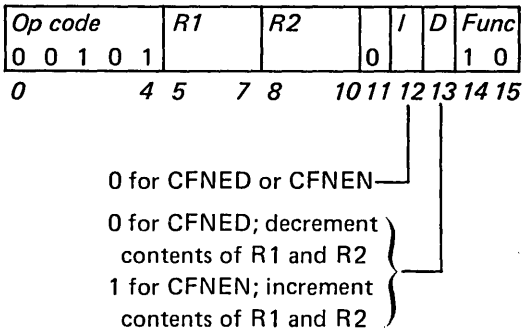
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

CFNED—CFNEN

Compare Byte Field Not Equal and Decrement (CFNED)

Compare Byte Field Not Equal and Increment (CFNEN)

CFNED (reg),(reg)
CFNEN (reg),(reg)



This instruction compares two fields in main storage on a byte-for-byte basis. Register 7 contains the number of bytes to be compared. This number is decremented after each byte is compared. The register specified by the R1 field contains the address of operand 1. The register specified by the R2 field contains the address of operand 2. Operand 1 is subtracted from operand 2, but neither operand is changed. After each byte is compared, the addresses in the register specified by the R1 and R2 fields are incremented or decremented (determined by bit 13 of the instruction). The operation terminates when either:

1. An unequal condition is detected, or
2. The number of bytes specified in register 7 has been compared.

When an unequal condition occurs, the addresses in the registers point to the next operands to be compared, but the count in register 7 is not updated.

Bit 11 of the instruction is not used and must be set to 0 to avoid future code obsolescence.

See "Scan Byte Field Not Equal and Decrement (SFNED)" and "Scan Byte Field Not Equal and Increment (SFNEN)" for other versions of this machine instruction.

Notes:

1. If the specified count in register 7 is 0, the instruction performs no operation (no-op).
2. Variable-field-length instructions can be interrupted. When this occurs and the interrupted level resumes operation, the processor treats the incomplete instruction as a new instruction, with the remaining byte count specified in register 7.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the byte, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one byte; that is, if the difference is less than -2^7 or greater than $+2^7-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

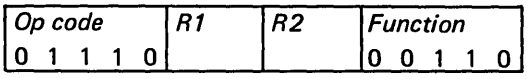
Invalid Function. Register 7 is specified in the R1 or R2 field of the instruction. The instruction is terminated.

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Complement Register (CMR)

CMR reg[,reg]



The contents of the register specified by the R1 field are converted to the two's complement. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed if the R1 and R2 fields do not specify the same register.

Indicators

Carry. The carry indicator is reset, and then set to 1 if the number to be complemented is 0.

Overflow. The overflow indicator is reset, and then set to 1 if the number to be complemented is the maximum negative number representable.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

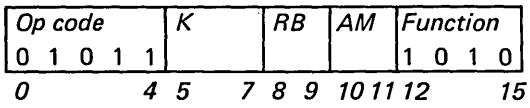
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

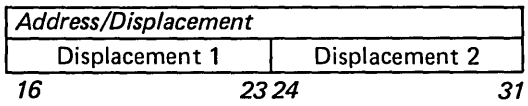
Copy Address Key Register (CPAKR)

System Register/Storage Format

Mnemonic	Syntax	Instruction name	K-field
CPAKR	addr4	Copy Address Key Register	011

Extended mnemonic	Syntax	Instruction name	K-field
CPISK	addr4	Copy Instruction Space Key	000
CPOOK	addr4	Copy Operand 1 Key	010
CPOTK	addr4	Copy Operand 2 Key	001





The contents of the address key register (AKR) field, specified by the K-field, are stored into the word location specified by the effective address. The contents of the AKR are not changed.

The K-field can specify a field within the AKR or the entire AKR.

K-field	Address key register field name	Bits
000	Instruction space key	13–15
001	Operand 2 key	9–11
010	Operand 1 key	5–7
011	Address key register	0–15
100	See Note	
101	See Note	
110	See Note	
111	See Note	

Note: To avoid future program obsolescence, these K-field values should not be used.

If the K-field specifies a specific field within the AKR, the specified field is stored in bits 13–15 of the word location in main storage. Bits 0–12 of the word in main storage are set to 0's. If the K-field specifies the entire AKR, the AKR is stored in the word location in main storage.

CPAKR

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

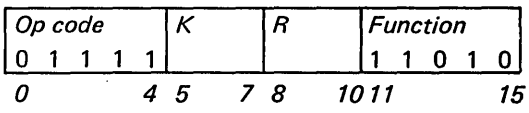
Privilege Violate. In the problem state, a privileged instruction is encountered.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

System Register/Register Format

Mnemonic	Syntax	Instruction name	K-field
CPAKR	reg	Copy Address Key Register	011

Extended mnemonic	Syntax	Instruction name	K-field
CPISK	reg	Copy Instruction Space Key	000
CPOOK	reg	Copy Operand 1 Key	010
CPOTK	reg	Copy Operand 2 Key	001



The contents of the address key register (AKR) field, specified by the K-field, are loaded into the register specified by the R-field. The contents of the AKR are not changed.

The K-field can specify a field within the AKR or the entire AKR.

K-field	Address key register field name	Bits
000	Instruction space key	13–15
001	Operand 2 key	9–11
010	Operand 1 key	5–7
011	Address key register	0–15
100	See Note	
101	See Note	
110	See Note	
111	See Note	

Note: To avoid future program obsolescence these K-field values should not be used.

If the K-field specifies a specific field within the AKR, the specified field is loaded into bits 13–15 of the register specified in the R-field. Bits 0–12 of the register are set to 0's. If the K-field specifies the entire AKR, the AKR is loaded into the register.

Indicators

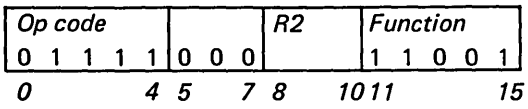
The indicators are not changed.

Program-Check Conditions

Privilege Violate. In the problem state, a privileged instruction is encountered.

Copy Current Level (CPCL)

CPCL reg



The register specified by the R2 field is loaded as follows:

- Bits 0–11 are set to 0's.
- Bits 12–15 are set to the current level. For example, if the current level is 3, bits 14 and 15 are set to 11.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

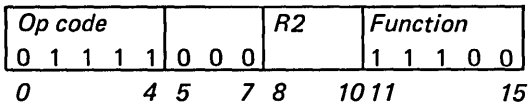
The indicators are not changed.

Program-Check Conditions

Privilege Violate. In the problem state, a privileged instruction is encountered.

Copy Clock (CPCLK)

CPCLK reg



The doubleword value contained in the clock register is set into the registers specified by the R2 field and R2+1 field. The clock value is not changed.

If the R2 field value is 7, registers 7 and 0 are used.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

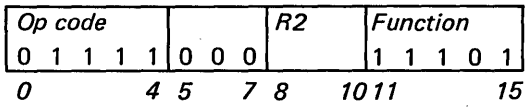
Indicators

The indicators are not changed.

CPCMP—CPCON

Copy Comparator (CPCMP)

CPCMP reg



The doubleword value contained in the comparator register is set into the registers specified by the R2 field and R2+1 field. The comparator value is not changed.

If the R2 field value is 7, registers 7 and 0 are used.

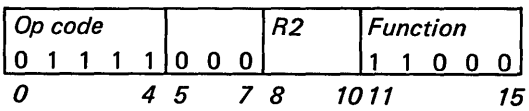
Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

The indicators are not changed.

Copy Console Data Buffer (CPCON)

CPCON reg



The contents of the console data buffer are loaded into the register specified by the R2 field. The contents of the buffer are not changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Carry and Overflow. These indicators are not changed.

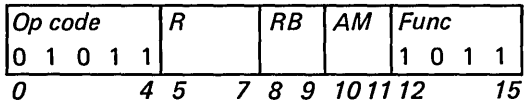
Even, Negative, and Zero. These indicators are changed to reflect the results.

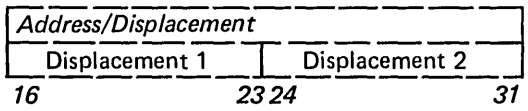
Program-Check Conditions

Privilege Violate. In the problem state, a privileged instruction is encountered.

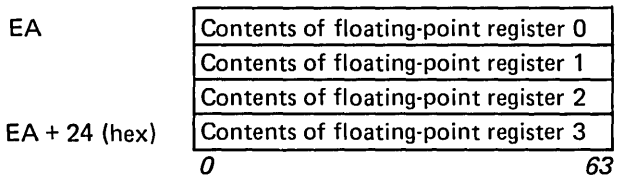
Copy Floating Level Block (CPFLB)

CPFLB reg,addr4

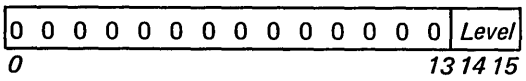




The contents of the floating-point registers (floating level block) for the level specified by the R-field are stored into main storage locations beginning at the specified effective address. All registers remain unchanged. After execution of this instruction, the floating level block appears in main storage as follows:



The general register specified by the R-field has the format:



Bits 0–7, 12, and 13 are not used and must be set to 0's to avoid future code obsolescence. Bits 8–11 must be set to 0's in order to select the floating-point feature. Bits 14 and 15 hold the binary-encoded level of the floating-point level block associated with this operation. For example, 00 for level 0, 01 for level 1, 10 for level 2, and 11 for level 3.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Privilege Violate. In the problem state, a privileged instruction is encountered.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

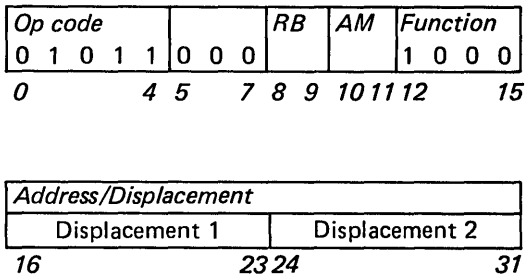
Soft-Exception Trap Conditions

Invalid Function. In the supervisor state, an attempt has been made to execute the instruction when the floating-point feature has not been selected or is not installed. The instruction is terminated.

CPIMR—CPIPF

Copy Interrupt Mask Register (CPIMR)

CPIMR addr4



The contents of the interrupt mask register are stored at the word location in main storage specified by the effective address. The interrupt mask register is not changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

The mask is represented in a bit-significant manner, with bit 0 representing level 0, and so on. (See "Interrupt Masking Facilities" in Chapter 3.) Bits 4–15 are set to 0's.

Indicators

The indicators are not changed.

Program-Check Conditions

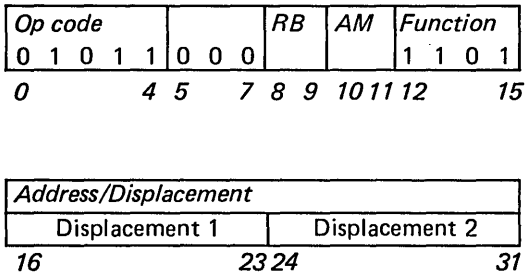
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Privilege Violate. In the problem state, a privileged instruction is encountered.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Copy In-Process Flags (CPIPF)

CPIPF addr4



This instruction permits the supervisor on the current level to inspect the in-process flags of the other levels. The in-process bit, bit 9 of the level status register, is on when a level is active or pending (previously interrupted by a higher level).

The in-process flags for each level are stored at the word location in main storage specified by the effective address. The in-process flags are not changed.

The flags are stored in a bit-significant manner, with bit 0 representing level 0, and so on. Bits 4–15 are set to 0's.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

The indicators are not changed.

Program-Check Conditions

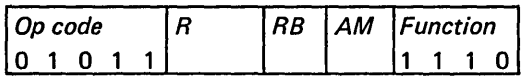
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

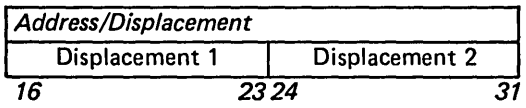
Privilege Violate. In the problem state, a privileged instruction is encountered.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Copy Level Block (CPLB)

CPLB reg,addr4





This instruction stores a level status block (LSB) into 11 words of main storage beginning with the location specified by the effective address. The contents of the LSB and the register specified by the R-field are not changed.

The register specified by the R-field contains the level of the LSB to be stored. This level is placed in bits 14 and 15 of the register. Bits 0–13 are unused and must be set to 0's.

Using this one instruction, the supervisor can copy the information contained in the hardware registers assigned to a program operating on any level. Most instructions are restricted to the registers associated with the current level. After executing a CPLB instruction, the supervisor can:

1. Use the information just stored (for example, the contents of the general registers or the protect key in the LSR).
2. Assign the level to another task by executing a Set Level Block (SELB) instruction that points to a different level status block.

In the second case, the supervisor can restart the preempted program at a later time by executing another SELB instruction that points to the previously stored level status block.

Programming Note: If the AM field equals 01, the contents of the register specified by the RB field are incremented by 2.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Privilege Violate. In the problem state, a privileged instruction is encountered.

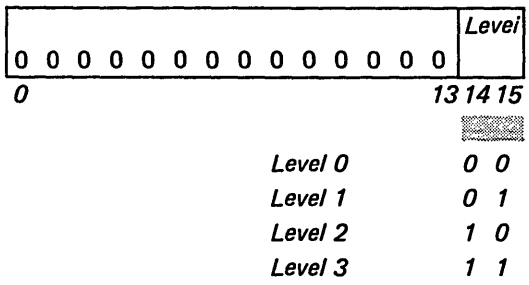
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Level Status Block Format

EA	IAR
	AKR
	LSR
	Register 0
	Register 1
	Register 2
	Register 3
	Register 4
	Register 5
	Register 6
EA+20 (+14 hex)	Register 7

EA=effective address

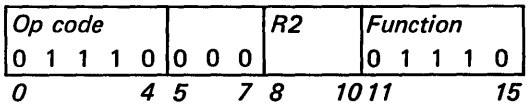
Format of Register Specified by the R-field in CPLB Instruction



CPLSR—CPPSR

Copy Level Status Register (CPLSR)

CPLSR reg



The level status register is loaded into the register specified by the R2 field. The level status register is not changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

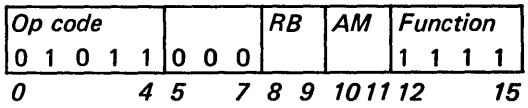
The indicators are not changed.

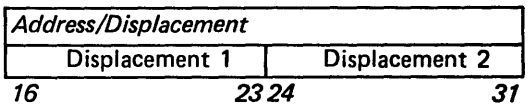
Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Copy Processor Status and Reset (CPPSR)

CPPSR addr4





The contents of the processor status word (PSW) are stored at the word location in main storage specified by the effective address.

This instruction resets bits 0–12 of the PSW. Bits 13–15 are not changed. Refer to "Processor Status Word (PSW)" in Chapter 3 for PSW bit settings.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

The indicators are not changed.

Program-Check Conditions

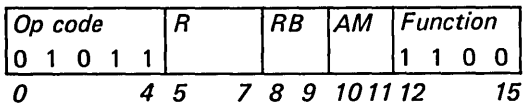
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

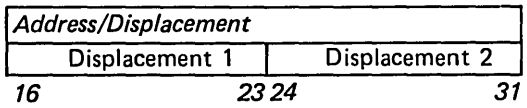
Privilege Violate. In the problem state, a privileged instruction is encountered.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Copy Storage Key (CPSK)

CPSK reg,addr4



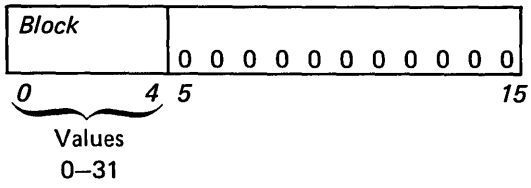


This instruction stores the contents of a storage key register at the byte location in main storage specified by the effective address.

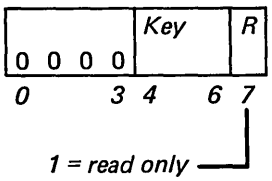
The register specified by the R-field contains the main storage block number for the storage key register to be stored. (A storage key register is associated with every 2048 bytes of storage.) The block number is in bits 0–4 of the register.

Bits 5–15 are not used and must be set to 0's to avoid future code obsolescence.

The format of the register specified by the R-field is:



The format of the byte at the storage location is:



Bits 4–7, the storage key and read-only bit, are the data from the storage key register for the selected main storage block. Bits 0–3 must be set to 0's to avoid future code obsolescence.

The contents of the storage key register are not changed.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

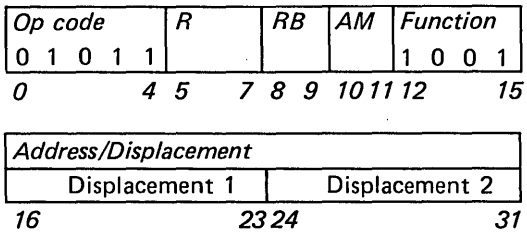
Privilege Violate. In the problem state, a privileged instruction is encountered.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

CPSR

Copy Segmentation Register (CPSR)

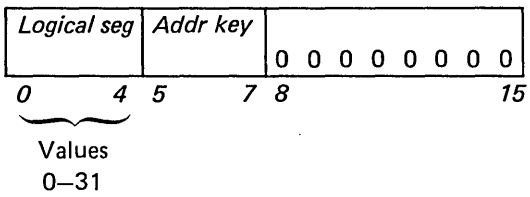
CPSR reg,addr4



This instruction stores the contents of a segmentation register into the doubleword location in main storage specified by the effective address.

The general register specified by the R-field contains the logical address of segmentation register (0–31, decimal) in bits 0–4, and an address key value 000–111 in bits 5–7. Bits 8–15 of the register must be set to 0's.

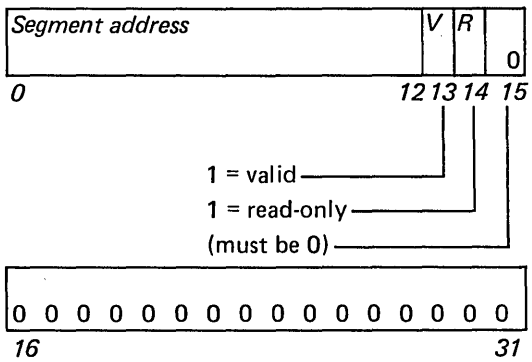
The format of the general register specified by the R-field is:



The logical address of the register selects a specific segmentation register (0–31) in a segmentation stack 0–7.

The address-key field of the register selects a stack 0–7 of the segmentation registers.

The first word of the specified doubleword that is copied from the selected segmentation register has the following format:



The segment address (bits 0–12) contains the high-order bits of the physical address, which is used by the translator to select a 2K-byte block of main storage. Refer to Chapter 5, "Storage Address Relocation Translator" for a description of the translator.

Bit 13, if a 1, signifies that the contents of the segmentation register is valid, and translation can be performed. If an attempt is made to use a segmentation register in which bit 13 is a 0, a program check interrupt occurs, with invalid storage address set in the PSW.

Bit 14, if a 1, signifies that the block is read-only. If an attempt is made to write into the block when bit 14 of the associated segmentation register is a 1 and while in problem state, a program check interrupt occurs, with protect check set in the PSW. When the supervisor state is on a cycle-steal access, bit 14 is ignored. The contents of main storage are not changed.

The second word (bits 16–31) of the specified doubleword must be set to 0's to avoid future code obsolescence.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Function. In the supervisor state, an attempt has been made to execute this instruction when the translator is not enabled.

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

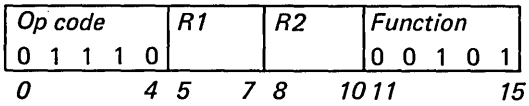
Privilege Violate. In the problem state, a privileged instruction is encountered.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Compare Word (CW)

Register/Register Format

CW reg,reg



The contents of the register specified by the R1 field are subtracted from the contents of the register specified by the R2 field. The contents of both registers are not changed.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

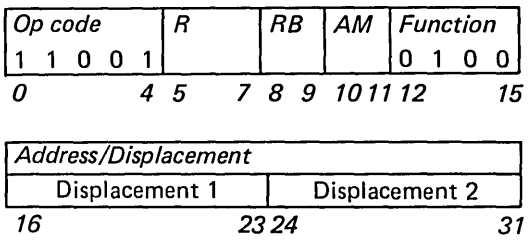
Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Register/Storage Format

CW addr4,reg



The contents of the word in main storage specified by the effective address are subtracted from the contents of the register specified by the R-field. Neither operand is changed.

Bit 12 of the instruction is reserved and must be set to 0 to avoid future code obsolescence.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

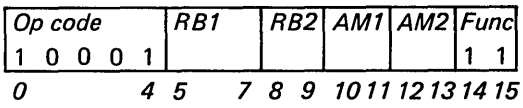
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

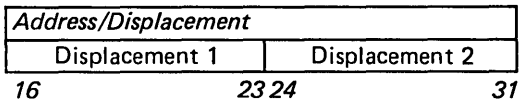
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

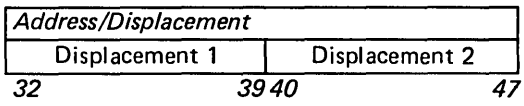
CW

Storage/Storage Format

CW addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. Word operand 1 is subtracted from word operand 2. Neither operand is changed.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

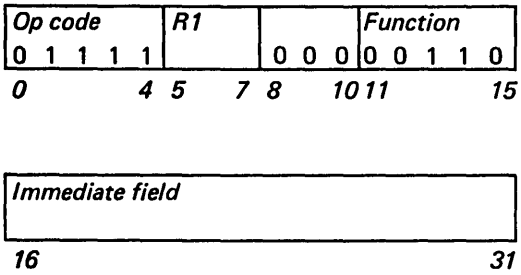
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Compare Word Immediate (CWI)

Register Immediate Long Format

CWI word,reg



The immediate field is subtracted from the contents of the register specified by the R1 field. The contents of the register specified by the R1 field are not changed.

Bits 8–10 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

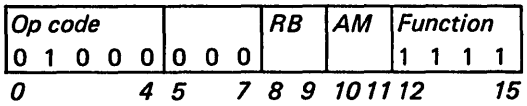
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

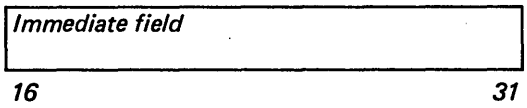
CWI

Storage Immediate Format

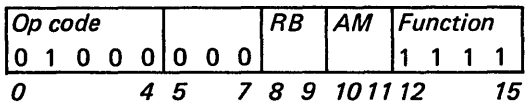
CWI word,addr4

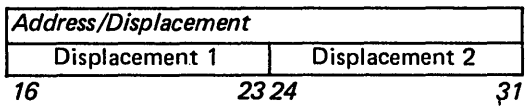
Format without appended word for effective addressing (AM = 00 or 01)





Format with appended word for effective addressing (AM = 10 or 11)





The immediate word is subtracted from the contents of the location specified by the effective address. Neither operand is changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

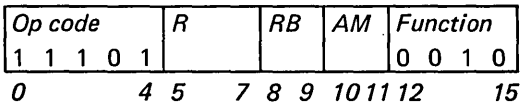
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

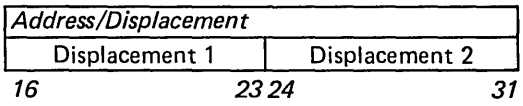
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Divide Byte (DB)

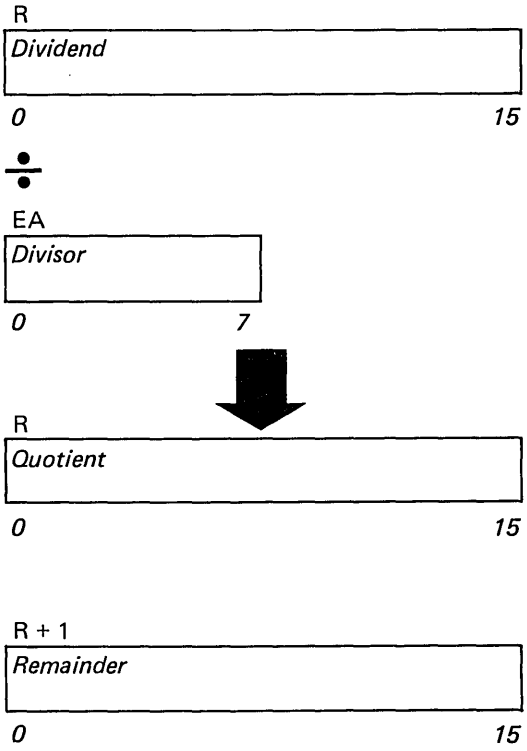
DB addr4,reg





A divide operation is performed between the word dividend contained in the register specified by the R-field and the byte divisor at the location specified by the effective address. The one-word quotient replaces the contents of the specified register while the one-word remainder is placed in the register specified by the R+1 field.

If the R-field specifies register 7, the remainder is placed in register 0.



Indicators

Carry. The carry indicator is cleared, and then set to 1 (together with the overflow indicator) if the overflow was caused by an attempt to divide by 0.

Overflow. The overflow indicator is cleared, and then set to 1 if division by 0 is attempted, or if the quotient cannot be represented in one word. If overflow occurs, the remaining indicators and the contents of the specified register are undefined.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

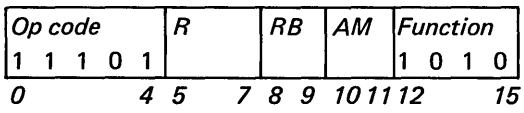
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

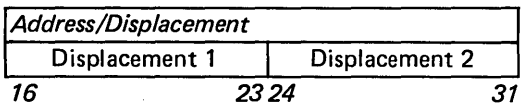
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

DD

Divide Doubleword (DD)

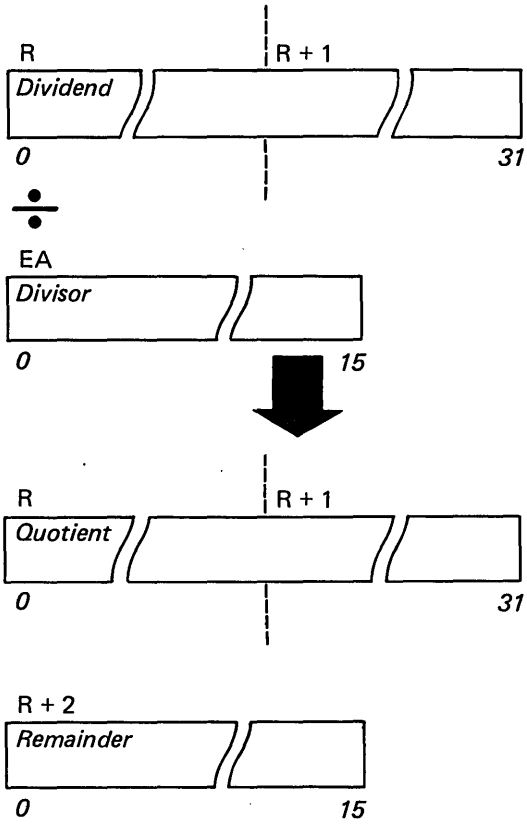
DD addr4,reg





A divide operation is performed between the doubleword dividend contained in the registers, specified by the R-field and R+1 field, and the word divisor at the location specified by the effective address. The doubleword quotient replaces the contents of the specified registers (least-significant word is in the R+1 field). The one-word remainder is placed in the register specified by the R+2 field.

If the R-field value is 6, registers 6, 7, and 0 are used.



Programming Note: If the AM field equals 01, the contents of the register specified by the RB field are incremented by 2.

Indicators

Carry. The carry indicator is cleared, and then set to 1 (together with the overflow indicator) if the overflow was caused by an attempt to divide by 0.

Overflow. The overflow indicator is cleared, and then set to 1 if division by 0 is attempted, or if the quotient cannot be represented in a doubleword. If overflow occurs, the remaining indicators and the contents of the specified registers are undefined.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

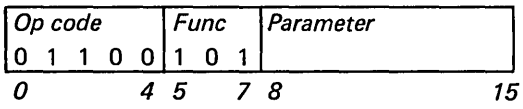
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

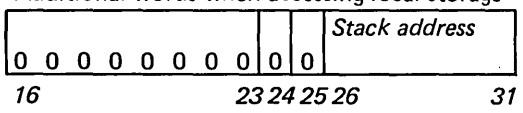
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

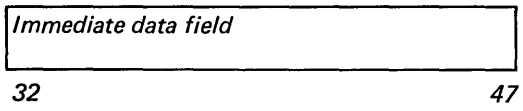
Diagnose (DIAG)

DIAG ubyte



Additional words when accessing local storage



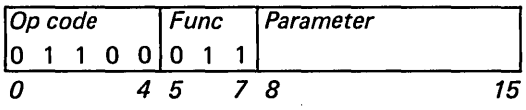


The Diagnose instruction is used for controlling or testing various hardware functions in a machine-dependent manner.

Refer to individual processor publications for information concerning this instruction.

Disable (DIS)

DIS ubyte



The parameter field 1-bits are disabled. The bits in the parameter field have the following significance:

Bit	Significance
8	Not used
9	Not used
10	Not used
11	Not used
12	Storage protect
13	Equate operand spaces (AKR bit 0 set to 0)
14	Translator (PSW bit 14 set to 0)
15	Summary mask (LSR bit 11 set to 0)

Note: Bits not used must be set to 0's to avoid future code obsolescence.

If a Disable instruction immediately follows an Enable summary mask instruction, the interrupt disable function may occur prior to the time that an interrupt can be accepted. Thus, at least one other instruction (for example , no-op) must be executed between the Enable summary mask and Disable instructions to ensure the occurrence of the interrupt.

If parameter bit 14 is set to 1 and the relocation translator is enabled (bit 14 of the PSW is on), then the translator is disabled and bit 14 of the PSW is turned off.

Indicators

The indicators are not changed.

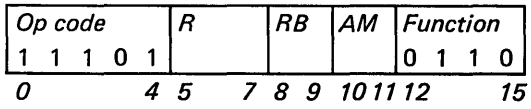
Program-Check Conditions

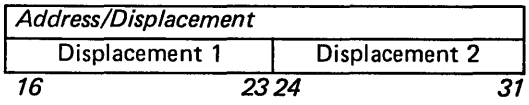
Privilege Violate. In the problem state, a privileged instruction is encountered.

DW

Divide Word (DW)

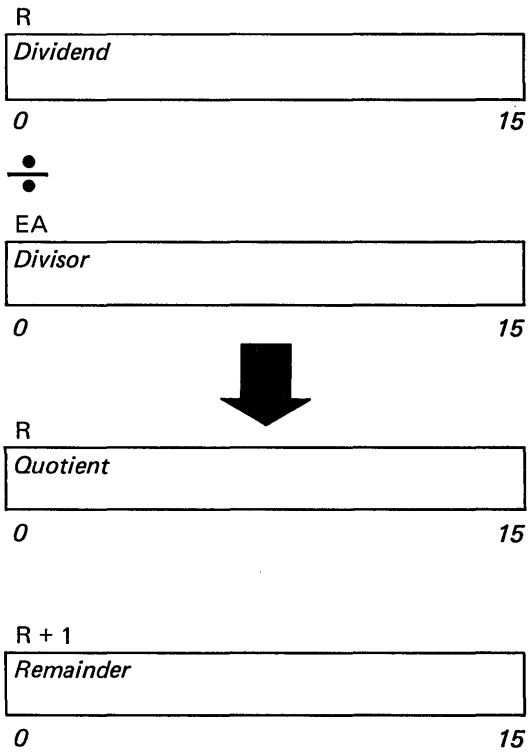
DW addr4,reg





A divide operation is performed between the word dividend contained in the register specified by the R-field and the word divisor at the location specified by the effective address. The one-word quotient replaces the contents of the specified register. The one-word remainder is placed in the register specified by the R+1 field.

If the R-field value is 7, registers 7 and 0 are used.



Indicators

Carry. The carry indicator is cleared, and then set to 1 (together with the overflow indicator) if the overflow was caused by an attempt to divide by 0.

Overflow. The overflow indicator is cleared, and then set to 1 if division by 0 is attempted, if the quotient cannot be represented in one word. If overflow occurs, the remaining indicators and the contents of the specified registers are undefined.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

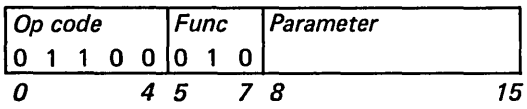
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Enable (EN)

EN ubyte



The parameters field 1-bits are enabled. The bits in the parameter field have the following significance:

Bit	Significance
8	Not used
9	Not used
10	Not used
11	Not used
12	Storage protect
13	Equate operand spaces (AKR bit 0 set to 1)
14	Translator (PSW bit 14 set to 1)
15	Summary mask (LSR bit 11 set to 1)

Note: Bits not used must be set to 0's to avoid future code obsolescence.

If bit 12 is set to 1, the relocation translator (if enabled) is disabled and bit 14 is not checked.

If bit 14 is set to 1 and bit 12 is set to 0, the relocation translator is enabled and the storage protect is disabled.

Indicators

The indicators are not changed.

Program-Check Conditions

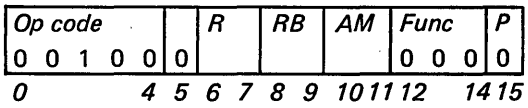
Privilege Violate. In the problem state, a privileged instruction is encountered.

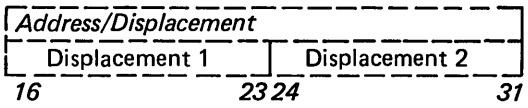
FA

Floating Add (FA)

Storage/Register Format

FA addr4,freg





The 32-bit main storage operand specified by the effective address is algebraically added to the 32-bit operand in the floating-point register specified by the R-field. The result is placed back into the floating-point register specified by the R-field. The main storage operand is not changed. The low-order 32 bits of the specified floating-point register are not changed.

The sign of the sum is determined by the rules of algebra unless all digits of the intermediate-sum fraction are 0's; in this case, the sign is made plus and the result characteristic is forced to 0.

Indicators

Overflow. If an overflow or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. The carry indicator is reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

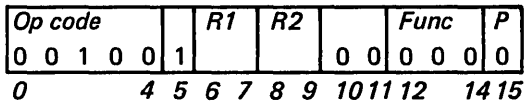
Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Floating-Point Exception. An arithmetic error has been detected. The instruction completes execution.

Register/Register Format

FA freg,freg



The two 32-bit operands contained in the floating-point registers specified by the R1 and R2 fields are added algebraically. The result is placed back into the floating-point register specified by the R2 field. The register specified by the R1 field is unchanged when not equal to the register specified by the R2 field. The low-order 32 bits of the register specified by the R2 field are not changed.

The sign of the sum is determined by the rules of algebra unless all digits of the intermediate-sum fraction are 0's; in this case, the sign is made plus and the result characteristic is forced to 0.

Indicators

Overflow. If an overflow or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. The carry indicator is reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Soft-Exception Trap Conditions

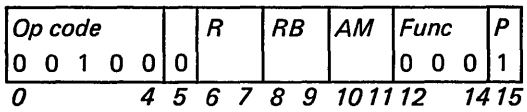
Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is suppressed.

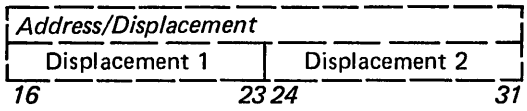
Floating-Point Exception. An arithmetic error has been detected. The instruction completes execution.

Floating Add Double (FAD)

Storage/Register Format

FAD addr4,freg





The 64-bit main storage operand specified by the effective address is algebraically added to the 64-bit operand in the floating-point register specified by the R-field. The result is placed back into the floating-point register specified by the R-field. The main storage operand is not changed.

The sign of the sum is determined by the rules of algebra unless all digits of the intermediate-sum fraction are 0's; in this case, the sign is made plus and the result characteristic is forced to 0.

Indicators

Overflow. If an overflow or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise the indicator is reset.

Carry. The carry indicator is reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

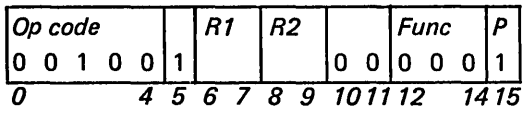
Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Floating-Point Exception. An arithmetic error has been detected. The instruction completes execution.

Register/Register Format

FAD freg,freg



The two 64-bit operands contained in the floating-point registers specified by the R1 and R2 fields are added algrebraically. The result is placed back into the floating-point register specified by the R2 field. The register specified by the R1 field is not unchanged when not equal to the register specified by the R2 field.

The sign of the sum is determined by the rules of algebra unless all digits of the intermediate-sum fraction are 0's; in this case, the sign is made plus and the result characteristic is forced to 0.

Indicators

Overflow. If an overflow or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. The carry indicator is reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Soft-Exception Trap Conditions

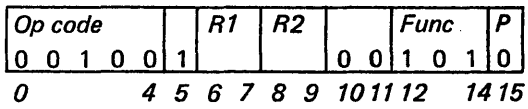
Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is suppressed.

Floating-Point Execution An arithmetic error has been detected. The instruction completes execution.

FC—FCD

Floating Compare (FC)

FC freg,freg



The 32-bit operand contained in the floating-point register specified by the R1 field is algebraically subtracted from the 32-bit operand contained in the floating-point register specified by the R2 field. The contents of both floating-point registers are not changed.

Indicators

Overflow. If an overflow or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the even indicator is reset.

Carry. The carry indicator is reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

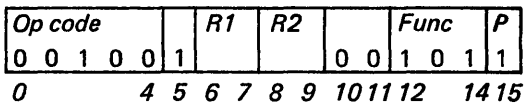
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is suppressed.

Floating Compare Double (FCD)

FCD freg,freg



The 64-bit operand contained in the floating-point register specified by the R1 field is algebraically subtracted from the 64-bit operand contained in the floating-point register specified by the R2 field. The contents of both floating-point registers are not changed.

Indicators

Overflow. If an overflow or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. The carry indicator is reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

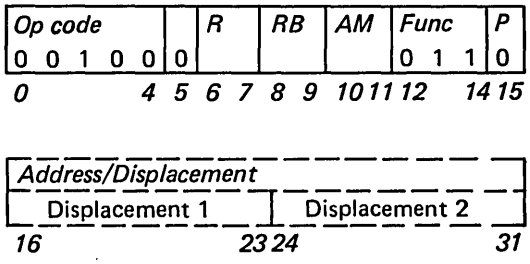
Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is suppressed.

Floating Divide (FD)

Storage/Register Format

FD addr4,freg



The 32-bit dividend contained in the floating-point register specified by the R-field is divided by the 32-bit divisor at the main storage location specified by the effective address. The 32-bit quotient is placed back in the floating-point register specified by the R-field. The low-order 32 bits of the specified floating-point register are not changed. No remainder is perserved. The main storage operand is not changed.

Indicators

Overflow. If a divide check, overflow, or underflow condition occurs, the overflow indicator is set to 1; otherwise the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. If a divide check condition occurs, the carry indicator is set to 1; otherwise the indicator is reset.

Negative and Zero. These indicators are changed to reflect the result unless a divide check condition occurs; in this case, the indicators are left reset to 0.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Soft-Exception Trap Conditions

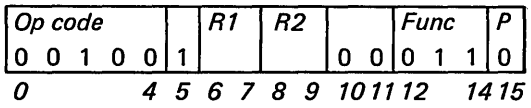
Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Floating-Point Exception. An arithmetic error has been detected.

FD

Register/Register Format

FD freg,freg



The 32-bit dividend contained in the floating-point register specified by the R2 field is divided by the 32-bit divisor contained in the floating-point register specified by the R1 field. The 32-bit quotient is placed back in the floating-point register specified by the R2 field. No remainder is preserved. The low-order 32 bits of the register specified by the R2 field are not changed. The register specified by the R1 field is not changed when not equal to R2.

Indicators

Overflow. If a divide check, overflow, or underflow condition occurs, the overflow indicator is set to 1; otherwise the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. If a divide check condition occurs, the carry indicator is set to 1; otherwise, the indicator is reset.

Negative and Zero. These indicators are changed to reflect the result unless a divide check condition occurs; in this case, the indicators are left reset to 0.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Soft-Exception Trap Conditions

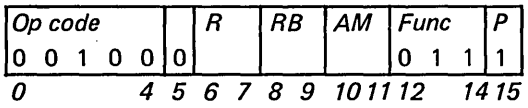
Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is suppressed.

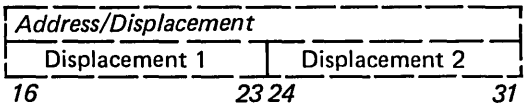
Floating-Point Exception. An arithmetic error has been detected.

Floating Divide Double (FDD)

Storage/Register Format

FDD addr4,freg





The 64-bit dividend contained in the floating-point register specified by the R-field is divided by the 64-bit divisor at the main storage location specified bythe effective address. The 64-bit quotient is placed back in the floating-point register specified by the R-field. No remainder is preserved. The main storage operand is not changed.

Indicators

Overflow. If a divide check, overflow, or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. If a divide check condition occurs, the carry indicator is set to 1; otherwise, the indicator is reset.

Negative and Zero. These indicators are changed to reflect the result unless a divide check condition occurs; in this case, they are left reset to 0.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Soft-Exception Trap Conditions

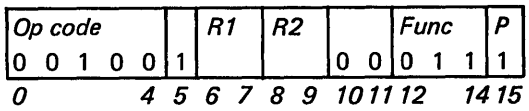
Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Floating-Point Exception. An arithmetic error has been detected.

FDD

Register/Register Format

FDD freg,freg



The 64-bit dividend contained in the floating-point register specified by the R2 field is divided by the 64-bit divisor contained in the floating-point register specified by the R1 field. The 64-bit quotient is placed back in the floating-point register specified by the R2 field. No remainder is preserved. The register specified by the R1 field is not changed when not equal to the R2 field.

Indicators

Overflow. If a divide check, overflow, or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. If a divide check condition occurs, the carry indicator is set to 1; otherwise, the indicator is reset.

Negative and Zero. These indicators are changed to reflect the result unless a divide check condition occurs; in this case, the indicators are left reset to 0.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Soft-Exception Trap Conditions

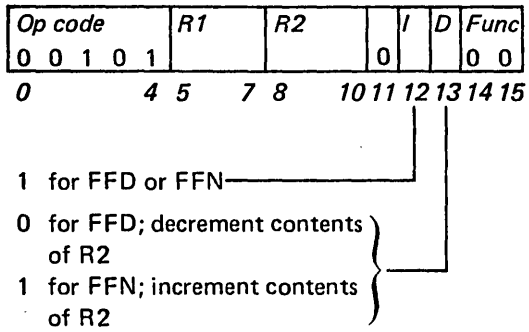
Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is suppressed.

Floating-Point Exception. An arithmetic error has been detected.

Fill Byte Field and Decrement (FFD)

Fill Byte Field and Increment (FFN)

FFD reg,(reg)
FFN reg,(reg)



This instruction fills each byte of a field in main storage with the same bit configuration. Register 7 contains the number of bytes to be filled (field length). If a field length of 0 is specified, the instruction is a no-op. The register specified by the R1 field contains, in bits 8–15, the byte used to fill the field. The register specified by the R2 field contains the starting address of the field in main storage.

After each byte in the field is filled:

1. The address register specified by the R2 field is either incremented or decremented, as determined by bit 13 of the instruction. This permits the field to be filled in either direction.
2. The length count in register 7 is decremented.

The operation ends when the specified field length has been filled (contents of register 7 equal 0). At this time, the address specified by the R2 field has been updated and points to the byte adjacent to the end of the field.

Bits 11 and 15 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

See "Move Byte Field and Decrement (MVFD)" and "Move Byte Field and Increment (MVFN)" for other versions of this machine instruction.

Note: Variable-field-length instructions can be interrupted. When this occurs and the interrupted level resumes operation, the processor treats the incomplete instruction as a new instruction, with the remaining byte count specified in register 7.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result of the last byte moved.

Program-Check Conditions

Invalid Function. Register 7 is specified in the R1 or R2 field of the instruction. The instruction is terminated.

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

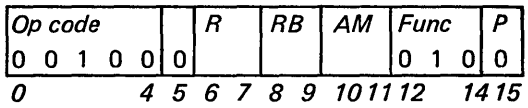
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

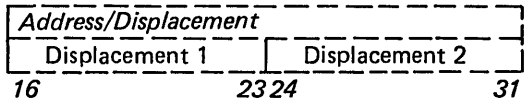
The instruction is terminated.

Floating Multiply (FM)

Storage/Register Format

FM addr4,freg





The 32-bit main storage operand specified by the effective address and the 32-bit operand contained in the floating-point register specified by the R-field are multiplied. The normalized result is placed back into the floating-point register specified by the R-field. The main storage operand is not changed.

The sign of the product is determined by the rules of algebra unless all digits of the product fraction are 0's; in this case, the sign is made plus and the result characteristic is forced to 0.

When either or both operand fractions are 0's, the result is made a true zero.

Indicators

Overflow. If an overflow or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. The carry indicator is reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

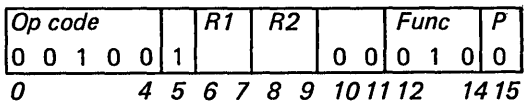
Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Floating-Point Exception. An arithmetic error has been detected.

Register/Register Format

FM freg,freg



The two 32-bit operands contained in the floating-point registers specified by the R1 and R2 fields are multiplied and the normalized result is placed back into the floating-point register specified by the R2 field. The register specified by the R1 field is not changed when not equal to the register specified by the R2 field.

The sign of the product is determined by the rules of algebra unless all digits of the product fraction are 0's; in this case, the sign is made plus and the result characteristic is forced to 0.

When either or both operand fractions are 0's, the result is made a true zero.

Indicators

Overflow. If an overflow or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. The carry indicator is reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is suppressed.

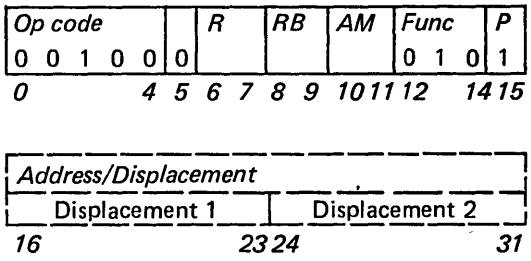
Floating-Point Exception. An arithmetic error has been detected.

FMD

Floating Multiply Double (FMD)

Storage/Register Format

FMD addr4,freg



The 64-bit main storage operand specified by the effective address and the 64-bit operand contained in the floating-point register specified by the R-field are multiplied. The normalized result is placed back into the floating-point register specified by the R-field. The main storage operand is not changed.

The sign of the product is determined by the rules of algebra unless all digits of the intermediate-sum fraction are 0's; in this case, the sign is made plus and the result characteristic is forced to 0.

When either or both operand fractions are 0's, the result is made a true zero.

Indicators

Overflow. If an overflow or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is reset; otherwise, the indicator is reset.

Carry. The carry indicator is reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

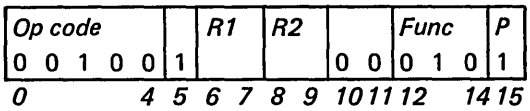
Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Floating-Point Exception. An arithmetic error has been detected.

Register/Register Format

FMD freg,freg



The two 64-bit operands contained in the floating-point registers specified by the R1 and R2 fields are multiplied. The normalized result is placed back into the floating-point register specified by the R2 field. The register specified by the R1 field is unchanged when not equal to the register specified by the R2 field.

The sign of the product is determined by the rules of algebra unless all digits of the intermediate-sum fraction are 0's; in this case, the sign is made plus and the result characteristic is forced to 0.

When either or both operand fractions are 0's, the result is made a true zero.

Indicators

Overflow. If an overflow or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. The carry indicator is reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is suppressed.

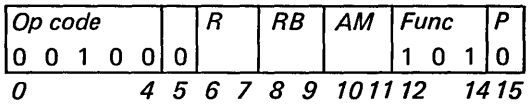
Floating-Point Exception. An arithmetic error has been detected.

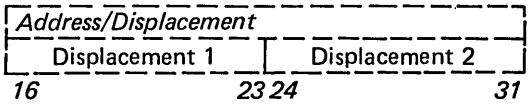
FMV

Floating Move (FMV)

Storage/Register Format

FMV addr4,freg





The 32-bit floating-point operand in the main storage location specified by the effective address is loaded into the floatin-point register specified by the R-field and the current interrupt level. The main storage operand is not changed. The low-order 32 bits of the 64-bit register are set to 0's.

Indicators

Overflow, Even, and Carry. These indicators are reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

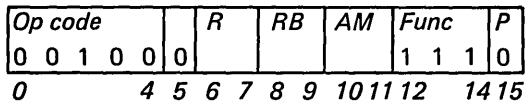
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

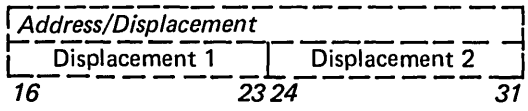
Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Register/Storage Format

FMV freg,addr4





The 32-bit floating-point operand contained in the high-order 32 bits of the floating-point register specified by the R-field is stored in the main storage location specified by the effective address. The register specified by the R-field is not changed.

Indicators

Overflow, Even, and Carry. These indicators are reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

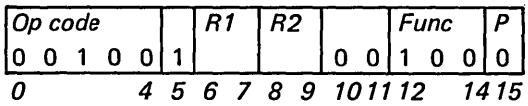
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Register/Register Format

FMV freg,freg



The 32-bit operand contained in the floating-point register specified by the R1 field is moved to the floating-point register specified by the R2 field. The low-order 32 bits of the register specified by the R2 field are set to 0's. The floating-point register specified by the R1 field is unchanged when not equal to the register specified by the R2 field.

Bits 10, 11, and 13 must be set to 0's to avoid future code obsolescence.

Indicators

Overflow, Even, and Carry. These indicators are reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Soft-Exception Trap Conditions

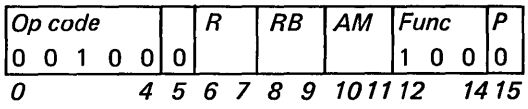
Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is suppressed.

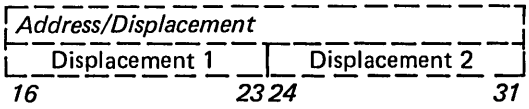
FMVC

Floating Move and Convert (FMVC)

Storage/Register Format

FMVC addr4,freg





The 16-bit signed binary integer in the main storage location specified by the effective address is converted to a 32-bit floating-point number with low-order 0's inserted and then loaded into the floating-point register specified by the R-field and the current interrupt level. The low-order 32 bits of the register are set to 0's. The 64-bit register is normalized with 0's inserted at the low-order positions during normalization. The main storage operand is not changed.

Indicators

Overflow, Even, and Carry. These indicators are reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

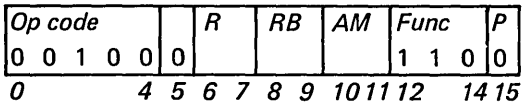
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

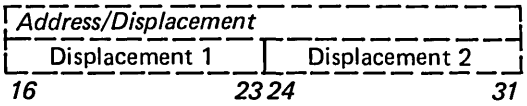
Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Register/Storage Format

FMVC freg,addr4





The 32-bit floating-point operand contained in the high-order 32 bits of the floating-point register specified by the R-field is converted to a signed 16-bit binary integer and stored at the main storage location specified by the effective address. Any fraction remaining after conversion is truncated.The register specified by the R-field is not changed. If the characteristic of the floating-point number is negative, the integer stored is 0.

Indicators

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$; otherwise the indicator is reset.

Even and Carry. These indicators are reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

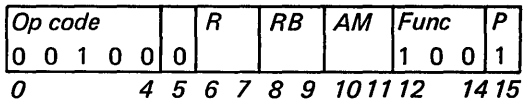
Soft-Exception Trap Conditions

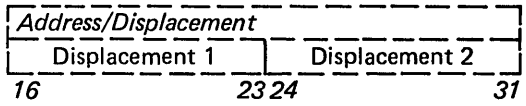
Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Floating Move and Convert Double (FMVCD)

Storage/Register Format

FMVCD addr4,freg





The 32-bit signed binary integer in the main storage location specified by the effective address is converted to a 64-bit floating-point number with low-order 0's inserted and then loaded into the floating-point register specified by the R-field and the current interrupt level. The 64-bit register is normalized with 0's inserted at the low-order positions during normalization. The main storage operand is not changed.

Indicators

Overflow, Even, and Carry. These indicators are reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

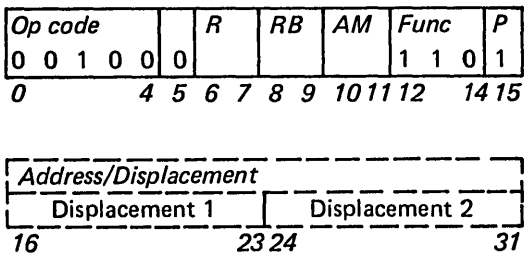
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Register/Storage Format

FMVCD freg,addr4



The 64-bit floating-point operand contained in the floating-point register specified by the R-field is converted to a 32-bit signed binary integer and stored at the main storage location specified by the effective address. Any fraction remaining after conversion is truncated. The register specified by the R-field is not changed. If the characteristic of the floating-point number is negative, the integer stored is 0.

Indicators

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in the doubleword; that is, if the difference is less than -2^{31} or greater than $+2^{31}-1$; otherwise, the indicator is reset.

Even and Carry. These indicators are reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Soft-Exception Trap Conditions

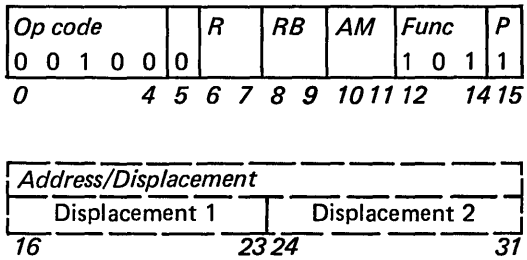
Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

FMVD

Floating Move Double (FMVD)

Storage/Register Format

FMVD addr4,freg



The 64-bit floating-point operand in the main storage location specified by the effective address is loaded into the floating-point register specified by the R-field and the current interrupt level. The main storage operand is not changed.

Indicators

Overflow, Even, and Carry. These indicators are reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

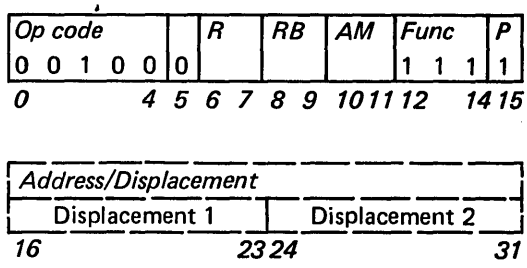
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Register/Storage Format

FMVD freg,addr4



The 64-bit floating-point operand contained in the register specified by the R-field is stored in the main storage location specified by the effective address. The register specified by the R-field is not changed.

Indicators

Overflow, Even, and Carry. These indicators are reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

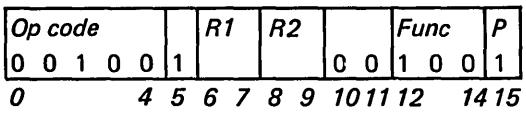
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Register/Register Format

FMVD freg,freg



The 64-bit operand contained in the floating-point register specified by the R1 field is moved to the floating-point register specified by the R2 field. The floating-point register specified by the R1 field is not changed.

Bits 10, 11, and 13 must be set to 0's to avoid future code obsolescence.

Indicators

Overflow, Even, and Carry. These indicators are reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

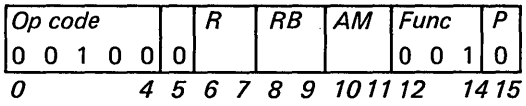
Soft-Exception Trap Conditions

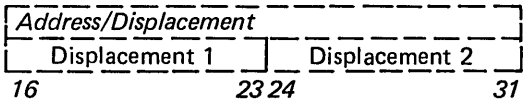
Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is suppressed.

Floating Subtract (FS)

Storage/Register Format

FS addr4,freg





The 32-bit main storage operand specified by the effective address is algebraically subtracted from the 32-bit operand contained in the floating-point register specified by the R-field. The result is placed back into the floating-point register specified by the R-field. The low-order 32 bits of the specified floating-point register are not changed. The main storage operand is not changed.

The sign of the sum is determined by the rules of algebra unless all digits of the intermediate-sum fraction are 0's; in this case, the sign is made plus and the result characteristic is forced to 0.

Indicators

Overflow. If an overflow or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. The carry indicator is reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

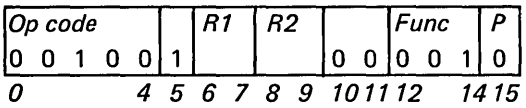
Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Floating-Point Exception. An arithmetic error has been detected.

Register/Register Format

FS freg,freg



The 32-bit operand contained in the floating-point register specified by the R1 field is algebraically subtracted from the 32-bit operand contained in the floating-point register specified by the R2 field. The result is placed back into the floating-point register specified by the R2 field. The low-order 32 bits of the register specified by the R2 field are not changed. The register specified by the R1 field remains unchanged when not equal to the register specified by the R2 field.

The sign of the sum is determined by the rules of algebra unless all digits of the intermediate-sum fraction are 0's; in this case, the sign is made plus and the result characteristic is forced to 0.

Indicators

Overflow. If an overflow or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. The carry indicator is reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Soft-Exception Trap Conditions

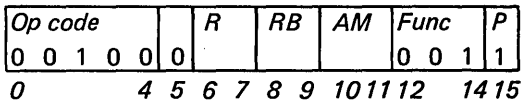
Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is suppressed.

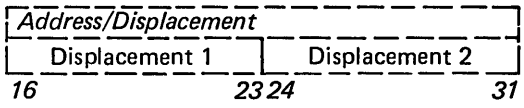
FSD

Floating Subtract Double (FSD)

Storage/Register Format

FSD addr4,freg





The 64-bit main storage operand specified by the effective address is algebraically subtracted from the 64-bit operand contained in the floating-point register specified by the R-field. The result is placed back into the floating-point register specified by the R-field. The main storage operand is not changed.

The sign of the sum is determined by the rules of algebra unless all digits of the intermediate-sum fraction are 0's; in this case, the sign is made plus and the result characteristic is forced to 0.

Indicators

Overflow. If an overflow or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. The carry indicator is reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

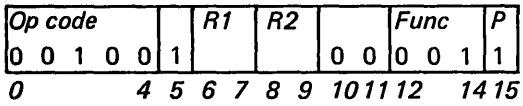
Soft-Exception Trap Conditions

Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

Floating-Point Exception. An arithmetic error has been detected.

Register/Register Format

FSD freg,freg



The 64-bit operand contained in the floating-point register specified by the R1 field is algebraically subtracted from the 64-bit operand contained in the floating-point register specified by the R2 field. The result is placed back into the floating-point register specified by the R2 field. The register specified by the R1 field remains unchanged when not equal to the register specified by the R2 field.

The sign of the sum is determined by the rules of algebra unless all digits of the intermediate-sum fraction are 0's; in this case, the sign is made plus and the result characteristic is forced to 0.

Indicators

Overflow. If an overflow or underflow condition occurs, the overflow indicator is set to 1; otherwise, the indicator is reset.

Even. If an underflow condition occurs, the even indicator is set to 1; otherwise, the indicator is reset.

Carry. The carry indicator is reset.

Negative and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

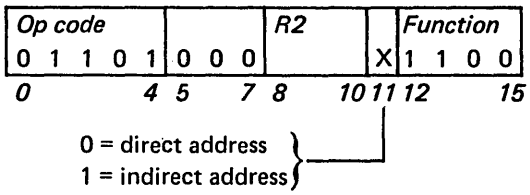
Soft-Exception Trap Conditions

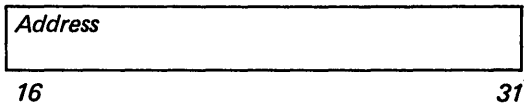
Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is suppressed.

Floating-Point Exception. An arithmetic error has been detected.

Operate I/O (IO)

IO longaddr





Refer to Chapter 4 for a detailed description of the operation of this instruction.

An effective main storage address is generated as follows:

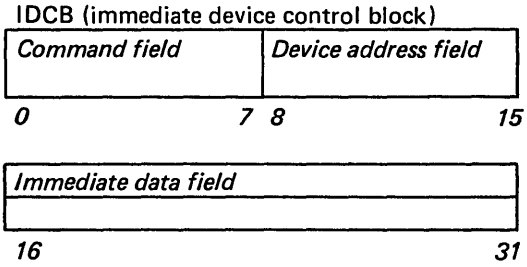
1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals 0, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0 (direct address). The result from step 1 is the effective address.

   Bit 11=1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

The effective address specifies the location of a two-word control block, called the immediate device control block (IDCB). The IDCB contains the command, device address, and a one-word immediate data field.



The immediate data field serves two purposes:

1. For direct program control (DPC) operations, it holds the data transferred to or from the I/O device.
2. For cycle-steal operations, it holds the address of the device control block (DCB).

Indicators

Even, Carry, and Overflow. These indicators are changed to reflect the condition code. See "Branch on Condition Code (BCC)" or "Branch on Not Condition Code (BNCC)" instructions for indicator settings.

Negative and Zero. These indicators are not changed.

Program-Check Conditions

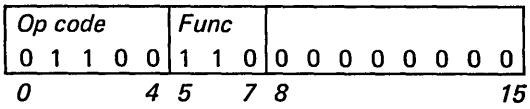
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Privilege Violate. In the problem state, a privileged instruction is encountered.

Specification Check. The effective address or indirect address results in an even-byte boundary violation.

Interchange Operand Keys (IOPK)

IOPK



The contents of the operand 1 key (OP1K) are interchanged with the contents of the operand 2 key (OP2K) in the current address key register.

Bits 8–15 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

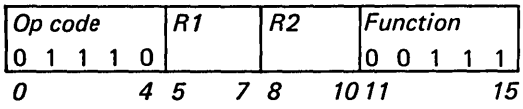
The indicators are not changed.

Program-Check Conditions

Privilege Violate. In the problem state, a privileged instruction is encountered.

Interchange Registers (IR)

IR reg,reg



The contents of the registers specified by the R1 and R2 fields are interchanged.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

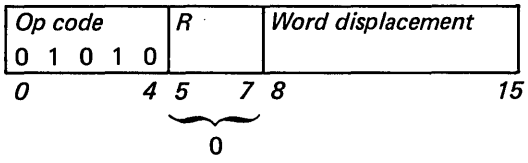
Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

J—JAL

Jump Unconditional (J)

J jdisp
jaddr



Bit 8 (the leftmost bit of the word displacement field) is propagated left seven bit positions and a 0 is appended at the low-order end; this results in a 16-bit word. (Word displacement is converted to a byte displacement.) This value is added to the instruction address register. The new value in the IAR becomes the address of the next instruction to be fetched.

Indicators

The indicators are not changed.

Program-Check Conditions

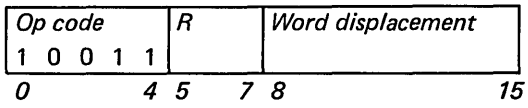
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Specification Check. The effective address or indirect address results in an even-byte boundary violation.

Jump and Link (JAL)

JAL jdisp,reg
jaddr,reg



The updated value of the instruction address register (the location of the next sequential instruction) is stored into the register specified by the R-field. Bit 8 (the leftmost bit of the word displacement field) is propagated left by seven bit positions and a 0 is appended at the low-order end; this results in a 16-bit word. (Word displacement is converted to a byte displacement.) This value is added to the updated contents of the instruction address register, and the result is stored in the instruction address register. This becomes the address of the next instruction to be fetched.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. Branching does not occur, but the storing of the updated instruction address into the register specified by the R-field still occurs.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

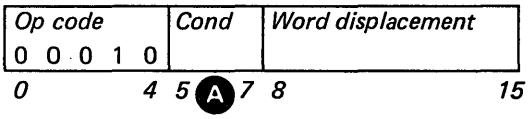
Jump on Condition (JC)

Mnemonic	Operand syntax	Instruction name	Condition field bits (see A)
JC	cond,jdisp cond,jaddr	Jump on Condition	Any value listed below

Extended mnemonic	Operand syntax	Instruction name	Condition field bits (see A)
JE	jdisp jaddr	Jump on Equal	000
JOFF	jdisp jaddr	Jump if Off	000
JZ	jdisp jaddr	Jump on Zero	000
JMIX	jdisp jaddr	Jump if Mixed	001
JP	jdisp jaddr	Jump on Positive	001
JON	jdisp jaddr	Jump if On	010
JN	jdisp jaddr	Jump on Negative	010
JEV	jdisp jaddr	Jump on Even	011
JLT	jdisp jaddr	Jump on Arithmetically Less Than	100
JLE	jdisp jaddr	Jump on Arithmetically Less Than or Equal	101
JLLE	jdisp jaddr	Jump on Logically Less Than or Equal	110
JCY	jdisp jaddr	Jump on Carry	111
JLLT	jdisp jaddr	Jump on Logically Less Than	111

Cond field bits	Extended mnemonics: Jump	Indicators tested: 0 1 2 3 4 / E C O N Z
000	JE, JOFF, JZ	X X X X 1
	JNE, JNOFF, JNZ	X X X X 0
001	JMIX, JP	X X X 0 0
	JNMIX, JNP	X X X X 1 X X X 1 X
010	JN, JON	X X X 1 X
	JNN, JNON	X X X 0 X
011	JEV	1 X X X X
	JNEV	0 X X X X
100	JLT	X X 0 1 X X X 1 0 X
	JGE	X X 1 1 X X X 0 0 X
101	JLE	X X 0 1 X X X 1 0 X X X X X 1
	JGT	X X 1 1 0 X X 0 0 0
110	JLLE	X 1 X X X X X X X 1
	JLGT	X 0 X X 0
111	JCY, JLLT	X 1 X X X
	JLGE, JNCY	X 0 X X X

JC



This instruction tests the condition of the various indicators set by a previously executed instruction (for example, an arithmetic, compare, test bit, or test word type of instruction).

If the condition tested is met, bit 8 (the leftmost bit of the word displacement field) is propagated left by seven bit positions and a 0 is appended at the low-order end; this results in a 16-bit word. (Word displacement is converted to a byte displacement.) This value is added to the updated value of the instruction address register, and becomes the address of the next instruction to be fetched. If the condition tested is not met, the next sequential instruction is fetched.

Indicators

The indicators are not changed.

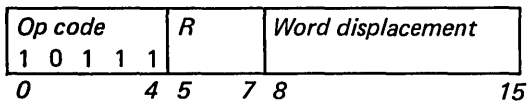
Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Jump on Count (JCT)

JCT jdisp,reg
jaddr,reg

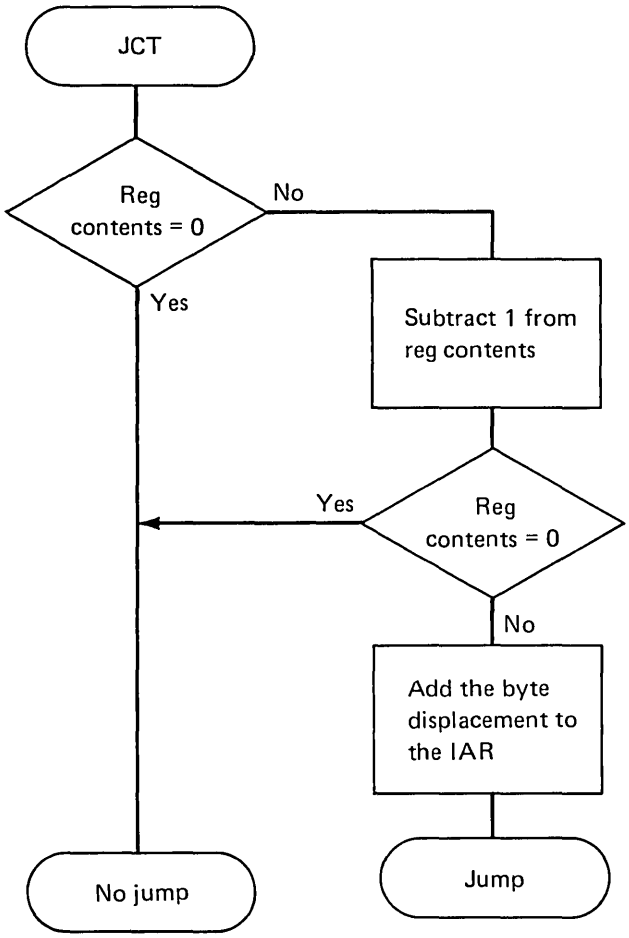


This instruction tests the contents of the register specified by the R-field.

If the register contents are not 0, the contents are decremented by 1. If the register contents are still not 0, the word displacement is converted to a byte displacement and added to the contents of the updated instruction address register (IAR). This value indicates the location of the next instruction to be fetched.

If the register contents are 0 when initially tested, no decrementing occurs. In this case, or when the register contents are 0 after decrementing, the next sequential instruction is fetched.

Note: When the register contents are not 0, the word displacement is converted to a byte displacement as follows: Bit 8 (the leftmost bit the word displacement field) is propagated left by seven bit positions, and a 0 is appended at the low-order end. This results in a 16-bit word that has been doubled in magnitude.



Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. Branching does not occur, but the contents of the register specified by the R-field are still decremented by 1.

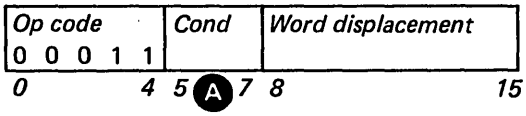
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Jump on Not Condition (JNC)

Mnemonic	Operand syntax	Instruction name	Condition field bits (see A)
JNC	cond,jdisp cond,jaddr	Jump on Not Condition	Any value listed below

Extended mnemonic	Operand syntax	Instruction name	Condition field bits (see A)
JNE	jdisp jaddr	Jump on Not Equal	000
JNOFF	jdisp jaddr	Jump if Not Off	000
JNZ	jdisp jaddr	Jump on Not Zero	000
JNMIX	jdisp jaddr	Jump on Not Mixed	001
JNP	jdisp jaddr	Jump on Not Positive	001
JNON	jdisp jaddr	Jump if Not On	010
JNN	jdisp jaddr	Jump on Not Negative	010
JNEV	jdisp jaddr	Jump on Not Even	011
JGE	jdisp jaddr	Jump on Arithmetically Greater Than or Equal	100
JGT	jdisp jaddr	Jump on Arithmetically Greater Than	101
JLGT	jdisp jaddr	Jump on Logically Greater Than	110
JLGE	jdisp jaddr	Jump on Logically Greater Than or Equal	111
JNCY	jdisp jaddr	Jump on No Carry	111

Cond field bits	Extended mnemonics: Jump	Indicators tested: 0 1 2 3 4 / E C O N Z
000	JE, JOFF, JZ	X X X X 1
	JNE, JNOFF, JNZ	X X X X 0
001	JMIX, JP	X X X 0 0
	JNMIX, JNP	X X X X 1 X X X 1 X
010	JN, JON	X X X 1 X
	JNN, JNON	X X X 0 X
011	JEV	1 X X X X
	JNEV	0 X X X X
100	JLT	X X 0 1 X X X 1 0 X
	JGE	X X 1 1 X X X 0 0 X
101	JLE	X X 0 1 X X X 1 0 X X X X X 1
	JGT	X X 1 1 0 X X 0 0 0
110	JLLE	X 1 X X X X X X X 1
	JLGT	X 0 X X 0
111	JCY, JLLT	X 1 X X X
	JLGE, JNCY	X 0 X X X



This instruction tests the condition of the various indicators set by a previously executed instruction (for example, an arithmetic, compare, test bit, or test word type of instruction.)

If the condition tested is met, bit 8 (the leftmost bit of the word displacement field) is propagated left by seven bit positions and a 0 is appended at the low-order end; resulting in a 16-bit word. (Word displacement is converted to a byte displacement.) This value is added to the updated value of the instruction address register, and becomes the address of the next instruction to be fetched.

If the condition tested is not met, the next sequential instruction is fetched.

Indicators

The indicators are not changed.

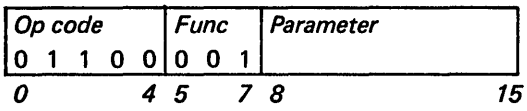
Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Level Exit (LEX)

LEX [ubyte]



When this instruction is executed, the processor exits the current level. The in-process bit (LSR bit 9) for the current level is set to 0. Next, the instruction tests for pending levels or outstanding priority interrupt requests, and the condition of the summary mask (LSR bit 11) for the level to be exited.

If pending levels or outstanding requests exist and the summary mask is enabled, a branch is executed to the address contained in the IAR of the highest pending or requesting level. This level then becomes the current level and processing resumes.

If pending levels or outstanding requests exist and the summary mask is disabled, the priority interrupts are not allowed. The highest pending level becomes the current level and processing resumes.

If no levels are pending, the processor goes to the wait state.

If no levels are pending and no interrupt requests are outstanding, the processor goes to the wait state.

The parameter field can be optionally coded with a one-byte unsigned absolute value or expression. If not coded, the parameter field defaults to 0. The processor ignores the value, but is used as an identifier.

For additional information about level switching, refer to "Program-Controlled Level Switching" in Chapter 3.

Programming Note: When a level is exited by a LEX instruction and processing is to continue on a pending level, one instruction is executed on the pending level prior to sampling for a trace class interrupt.

Indicators

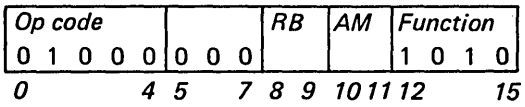
The indicators are not changed.

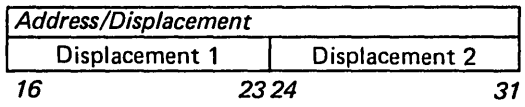
Program-Check Conditions

Privilege Violate. In the problem state, a privileged instruction is encountered.

Load Multiple and Branch (LMB)

LMB addr4





Refer to "Stack Operations" in Chapter 2 for a detailed description of the operation of this instruction. The LMB instruction is used in conjunction with the Store Multiple (STM) instruction described later in this chapter.

The contents of the registers for the current level are loaded from the stack defined by the stack control block pointed to by the effective address. The registers to be loaded are defined by the stack entry previously stored by a STM instruction. The next instruction is fetched from the storage address contained in register 7.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Programming Note: If the AM field equals 01, the contents of the register specified by the RB field are incremented by 2.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

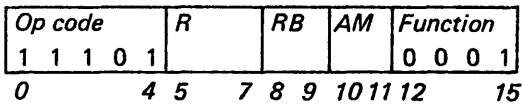
Specification Check. Indirect address, stack control block, stack element, or register 7 results in an even-byte boundary violation.

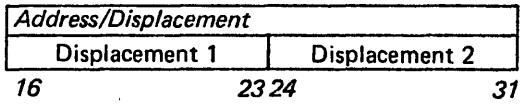
Soft-Exception Trap Conditions

Stack Exception. The stack is empty.

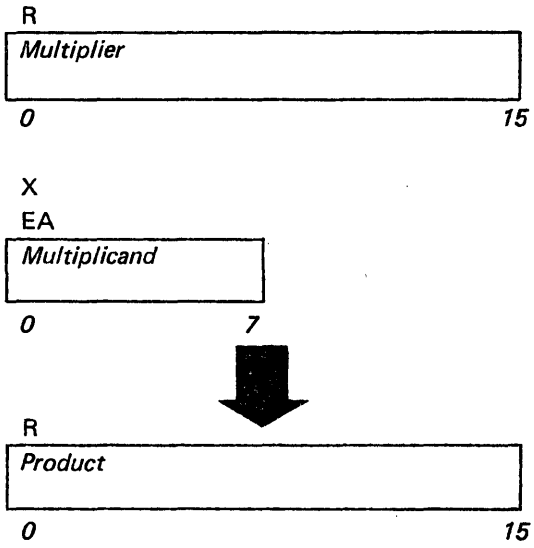
Multiply Byte (MB)

MB addr4,reg





A multiply operation is performed between the word multiplier contained in the register specified by the R-field and the byte multiplicand at the location specified by the effective address. The word product replaces the contents of the register.



Indicators

Carry. The carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the result cannot be represented in 16 bits. If overflow occurs, the contents of the specified register are the least-significant bits of the resulting product.

Even, Negative, and Zero. The indicators are changed to reflect the result.

Program-Check Conditions

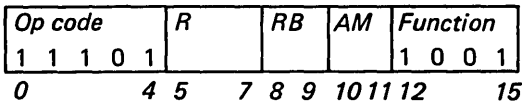
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

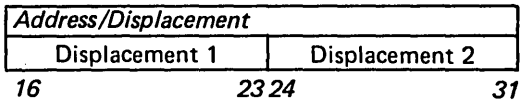
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Multiply Doubleword (MD)

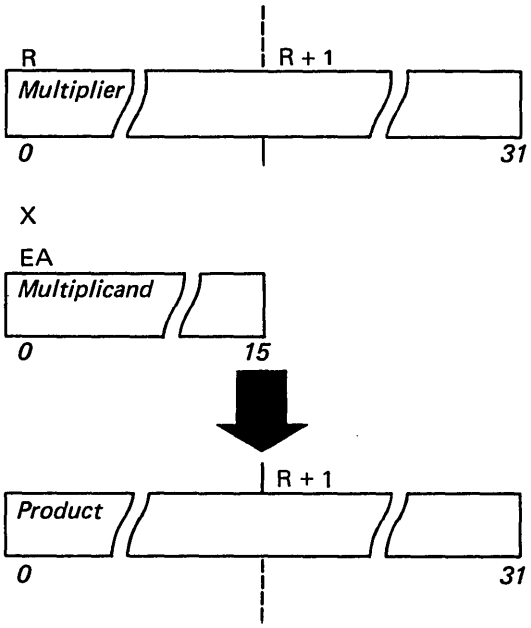
MD addr4,reg





A multiply operation is performed between the doubleword multiplier contained in the registers specified by the R-field and the R+1 field and the word multiplicand at the location specified by the effective address. The doubleword product replaces the contents of the registers with the least-significant word in the R+1 field.

If the R-field value is 7, registers 7 and 0 are used.



Programming Note: If AM=01, the register specified by the RB field is incremented by 2.

Indicators

Carry. The carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the result cannot be represented in 32 bits. If overflow occurs, the contents of the specified registers are the least-significant bits of the resulting product.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

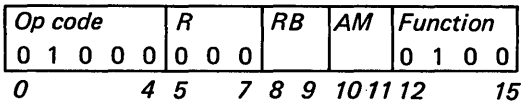
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

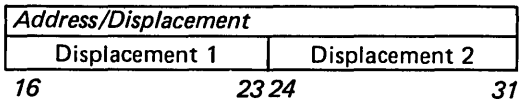
MVA

Move Address (MVA)

Storage/Register Format

MVA addr4,reg





The effective address is loaded into the register specified by the R-field.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the operand loaded into the register specified by the R-field.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

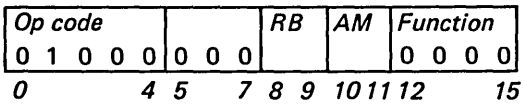
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

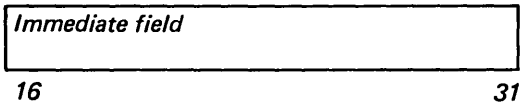
Specification Check. The effective address or indirect address results in an even-byte boundary violation.

Storage Immediate Format

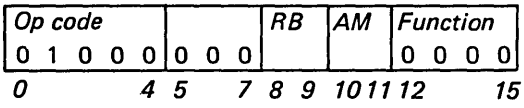
MVA raddr,addr4

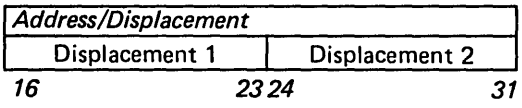
Format without appended word for effective addressing (AM = 00 or 01)

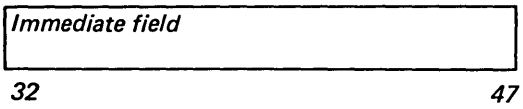




Format with appended word for effective addressing (AM = 10 or 11)







The operand in the immediate field replaces the contents of the location specified by the effective address. The immediate operand is not changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

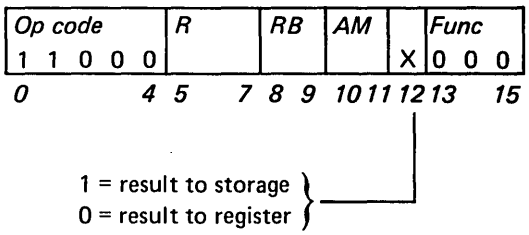
Specification Check. The effective address and indirect address results in an even-byte boundary violation.

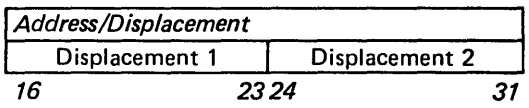
MVB

Move Byte (MVB)

Register/Storage Format

MVB reg,addr4
addr4,reg





A byte is moved between the least-significant byte of the register specified by the R-field and the location specified by the effective address in main storage.

Bit 12 of the instruction specifies the direction of the move:

Bit 12=0. The byte is moved from storage to register. The high-order bit of the byte (sign) is propagated to the eight high-order bits of the register. This permits the Compare Byte Immediate (CBI) instruction to be used for byte compare operations. The operand in storage is not changed.

Bit 12=1. The byte is moved from register to storage. The contents of the register specified by the R-field are not changed.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the operand moved.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

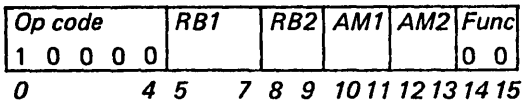
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

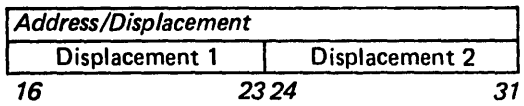
The instruction is terminated.

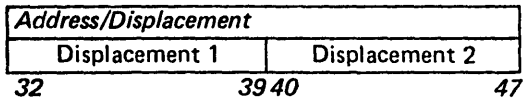
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Storage/Storage Format

MVB addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. A byte is moved from operand 1 to operand 2. Operand 1 is not changed.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the byte moved.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

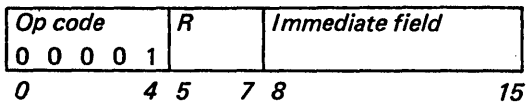
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

MVBI—MVBZ

Move Byte Immediate (MVBI)

MVBI byte,reg



The register specified by the R-field is loaded with the immediate operand.

The immediate field of the instruction forms the operand to be loaded. The immediate field is expanded to a 16-bit operand by propagating the sign bit value through the high-order bit positions. This operand is loaded into the register specified by the R-field.

Indicators

Carry and Overflow. These indicators are not changed.

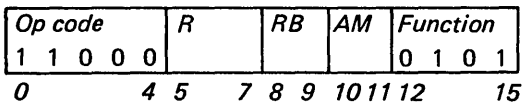
Even, Negative, and Zero. These indicators are changed to reflect the operand loaded into the register.

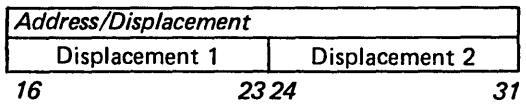
Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Move Byte and Zero (MVBZ)

MVBZ addr4,reg





The byte specified by the effective address is loaded into the least-significant byte of the register specified by the R-field. The high-order bit of the byte (sign) is propagated to the eight high-order bits within the register. The byte specified by the effective address is then set to 0's.

Bit 12 of the instruction is not used and must be set to 0 to avoid future code obsolescence.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the operand loaded into the register.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

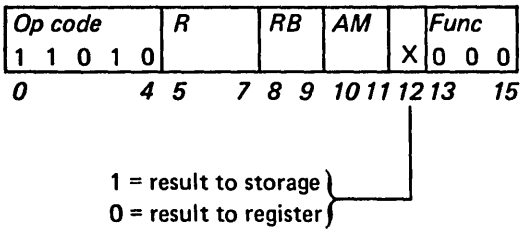
The instruction is terminated.

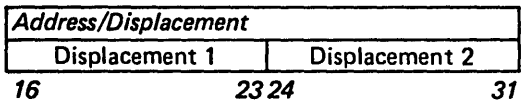
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Move Doubleword (MVD)

Register/Storage Format

MVD addr4,reg
reg,addr4





A doubleword is moved between the contents of the register pair specified by the R-field and the R+1 field and the doubleword location specified by the effective address in main storage. The source operand is not changed.

If the R-field value is 7, registers 7 and 0 are used.

Bit 12 of the instruction specifies the direction of the move.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the operand moved.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

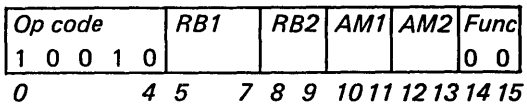
The instruction is terminated.

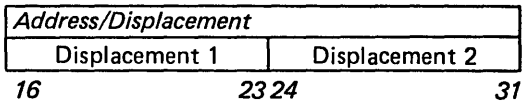
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

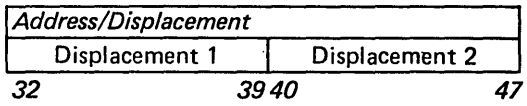
MVD—MVDZ

Storage/Storage Format

MVD addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. A doubleword is moved from operand 1 to operand 2. Operand 1 is not changed.

Note: In case of overlapping operands, operand 1 is fetched in its entirety before operand 2 is stored.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the doubleword moved.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

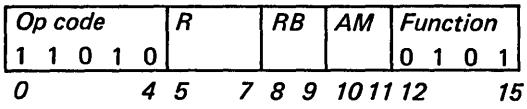
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

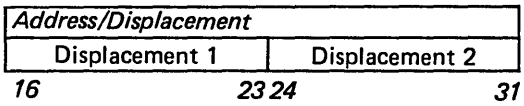
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Move Doubleword and Zero (MVDZ)

MVDZ addr4,reg





The doubleword specified by the effective address is loaded into the register pair specified by the R-field and the R+1 field. The doubleword specified by the effective address is then set to 0.

If the R-field value is 7, registers 7 and 0 are used.

Bit 12 of the instruction is not used and must be set to 0 to avoid future code obsolescence.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the operand loaded.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

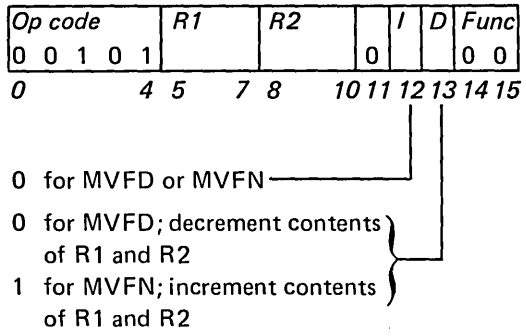
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Move Byte Field and Decrement (MVFD)

Move Byte Field and Increment (MVFN)

MVFD (reg),(reg)
MVFN (reg),(reg)



This instruction moves a specified number of bytes (one byte at a time) from one storage location to another. Register 7 contains the number of bytes to be moved (field length). If a field length of 0 is specified, the instruction is a no-op. The register specified by the R1 field contains the address of operand 1; the register specified by the R2 field contains the address of operand 2. Operand 1 is moved to operand 2.

Note: In case of overlapping operands, operand 1 is fetched in its entirety before operand 2 is stored.

After each byte is moved:

1. The addresses in the R1 and R2 fields are either incremented or decremented, determined by bit 13 of the instruction. This allows the field to be moved in either direction.
2. The length count in register 7 is decremented.

The operation ends when the specified field length has been filled (contents of register 7 equals 0). At this time, the addresses in the R1 and R2 fields have been updated and point to the next operands.

Bits 11 and 15 of the instructions are not used and must be set to 0's to avoid future code obsolescence.

See "Fill Byte Field and Decrement (FFD)" and "Fill Byte Field and Increment (FFN)" for other versions of this machine instruction.

Note: Variable-field-length instructions can be interrupted. When this occurs and the interrupted level resumes operation, the processor treats the incomplete instruction as a new instruction, with the remaining count specified in register 7.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result of the last byte moved.

Program-Check Conditions

Invalid Function. Register 7 is specified in the R1 or R2 field of the instruction. The instruction is terminated.

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

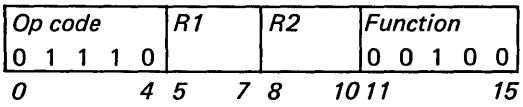
The instruction is terminated.

MVW

Move Word (MVW)

Register/Register Format

MVW reg,reg



The contents of the register specified by the R1 field replace the contents of the register specified by the R2 field. The contents of the register specified by the R1 field are not changed.

Indicators

Carry and Overflow. These indicators are not changed.

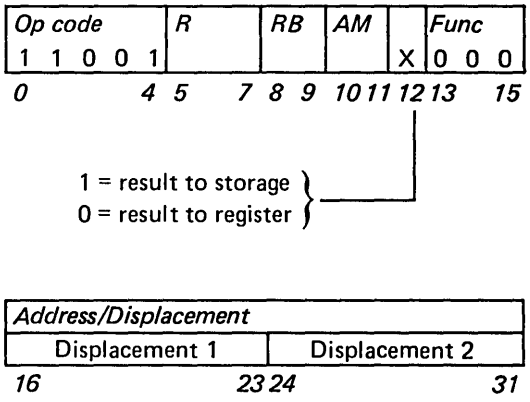
Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Register/Storage Format

MVW reg,addr4
addr4,reg



A word is moved between the contents of the register specified by the R-field and the location specified by the effective address in main storage. The source operand is not changed.

Bit 12 of the instruction specifies the direction of the move.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the operand moved.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

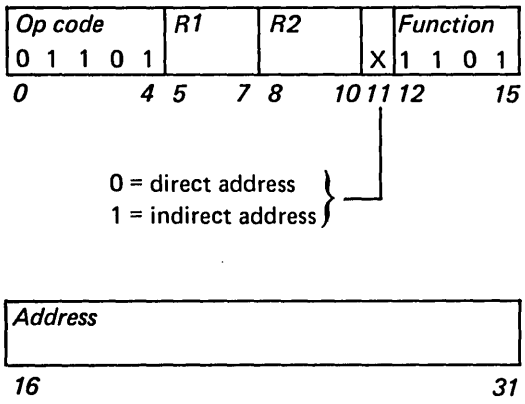
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Register/Storage Long Format

MVW reg,longaddr



The contents of the register specified by the R1 field are stored into the main storage location specified by an effective address. This effective address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals 0, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0 (direct address). The result from step 1 is the effective address.

   Bit 11=1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result stored from the register specified by the R1 field.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, the instruction:

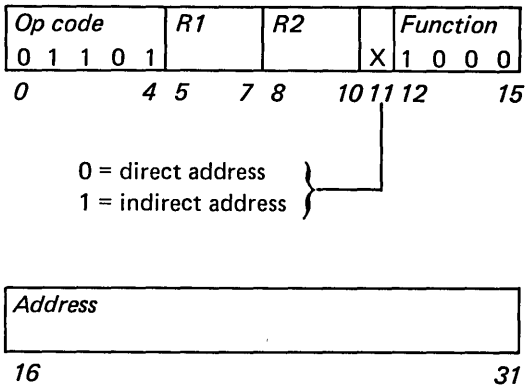
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

Specification Check. The effective address or indirect address results in an even-byte boundary violation.

MVW

Storage/Register Long Format

MVW longaddr,reg



The register specified by the R1 field is loaded with the contents of the main storage location specified by an effective address. This effective address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals 0, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0 (direct address). The result from step 1 is the effective address.

   Bit 11=1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result loaded into the register specified by the R1 field.

Program-Check Conditions

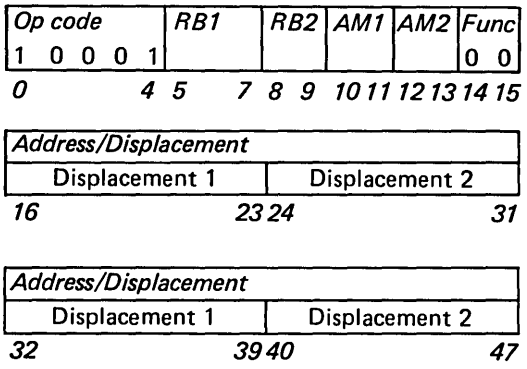
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Specification Check. The effective address or indirect address results in an even-byte boundary violation.

Storage/Storage Format

MVW addr5,addr4



The address arguments generate the effective addresses of two operands in main storage. A word is moved from operand 1 to operand 2. Operand 1 is not changed.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the word moved.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

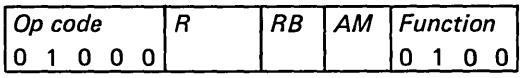
The instruction is terminated.

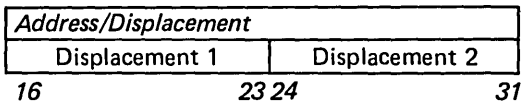
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Move Word Immediate (MVWI)

Storage/Register Format

MVWI word,reg





The effective address value is loaded into the register specified by the R-field. This value is equal to the value of word as specified by the programmer.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the operand loaded into the register specified by the R-field.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

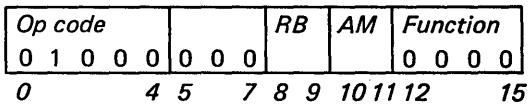
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

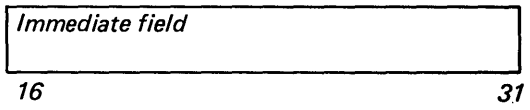
MVWI

Storage Immediate Format

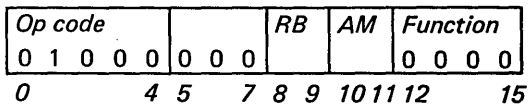
MVWI word,addr4

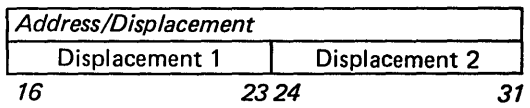
Format without appended word for effective addressing (AM = 00 or 01)

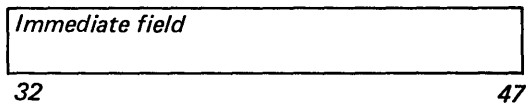




Format with appended word for effective addressing (AM = 10 or 11)







The operand in the immediate field replaces the contents of the location specified by the effective address. The immediate operand is not changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

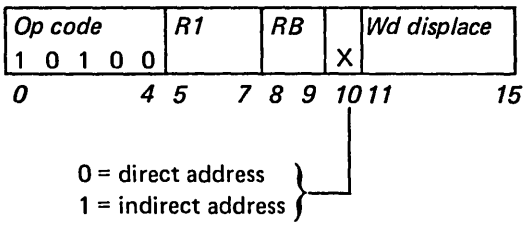
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Move Word Short (MVWS)

Register/Storage Format

MVWS reg,shortaddr



The contents of the register specified by the R1 field are stored into the main storage location specified by the effective address. The contents of the register are not changed.

The effective address is generated as follows:

1. The five-bit unsigned integer (word displacement) is doubled in magnitude (converted to a byte displacement).
2. The result from step 1 is added to the contents of the base register (RB) to form a main storage address.
3. Instruction bit 10 is tested for direct or indirect addressing:

   Bit 10=0 (direct address). The result from step 2 is the effective address.

   Bit 10=1 (indirect address). The result from step 2 is the address of the main storage location that contains the effective address.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the operand stored into main storage.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, the instruction:

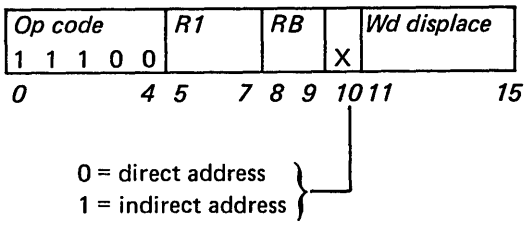
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

Specification Check. The effective address or indirect address results in an even-byte boundary violation.

MVWS

Storage/Register Format

MVWS shortaddr,reg



The contents of the main storage location specified by the effective address are loaded into the register specified by the R1 field. The contents of the main storage location are not changed.

The effective address is generated as follows:

1. The five-bit unsigned integer (word displacement) is doubled in magnitude (converted to a byte displacement).
2. The result from step 1 is added to the contents of the base register (RB) to form a main storage address.
3. Instruction bit 10 is tested for direct or indirect addressing:

   Bit 10=0 (direct address). The result from step 2 is the effective address.

   Bit 10=1 (indirect address). The result from step 2 is the address of the main storage location that contains the effective address.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the operand loaded into the register specified by the R1 field.

Program-Check Conditions

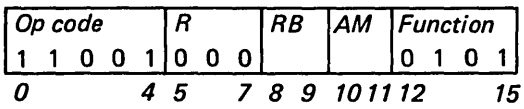
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

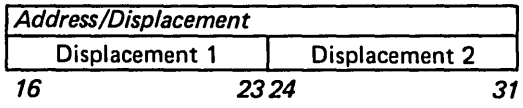
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Specification Check. The effective address or indirect address results in an even-byte boundary violation.

Move Word and Zero (MVWZ)

MVWZ addr4,reg





The word specified by the effective address is loaded into the register specified by the R-field. The word specified by the effective address is then set to 0.

Bit 12 of the instruction is not used and must be set to 0 to avoid future code obsolescence.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

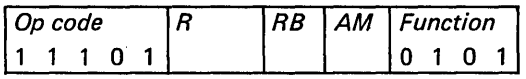
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

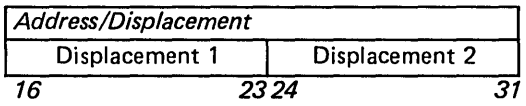
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

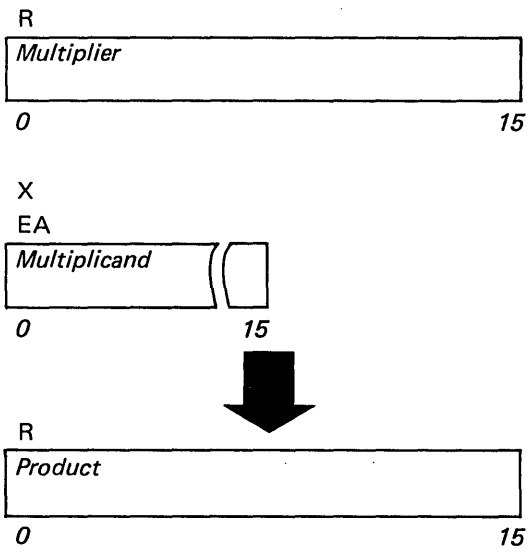
Multiply Word (MW)

MW addr4,reg





A multiply operation is performed between the word multiplier contained in the register specified by the R-field and the word multiplicand at the location specified by the effective address. The word product replaces the contents of the register.



Indicators

Carry. The carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the result cannot be represented in 16 bits. If overflow occurs, the contents of the specified register are the least-significant bits of the result.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

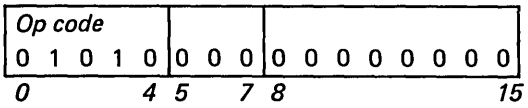
Invalid Storage Address. One or more words the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

No Operation (NOP)

NOP



When bits 5–15 are all 0's, the instruction performs no operation (no-op).

Indicators

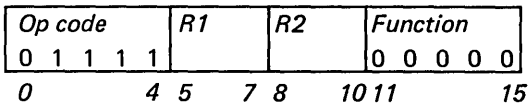
The indicators are not changed.

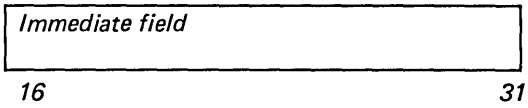
Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

AND Word Immediate (NWI)

NWI word,reg[,reg]





The immediate field is ANDed bit-by-bit with the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed unless the R1 and R2 field specify the same register.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

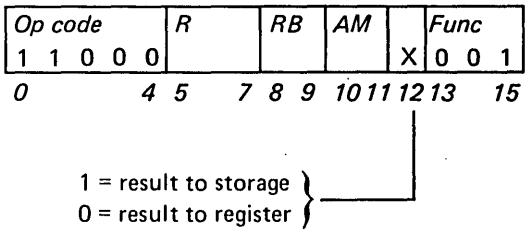
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

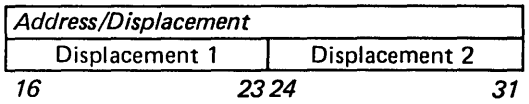
OB

OR Byte (OB)

Register/Storage Format

OB reg,addr4
addr4,reg





A logical OR operation is performed between the least-significant byte of the register specified by the R-field and the location specified by the effective address in main storage. The source operand is not changed. When going from storage to register, bits 0–7 of the register are not changed.

Bit 12 of the instruction specifies the destination of the result.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

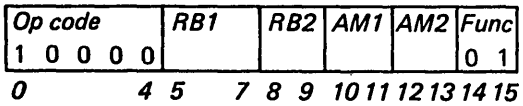
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

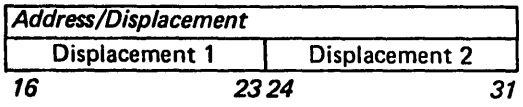
The instruction is terminated.

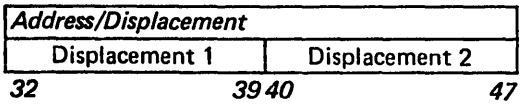
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Storage/Storage Format

OB addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. A one-byte logical OR operation is performed between operand 1 and operand 2. The result replaces operand 2. Operand 1 is not changed.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

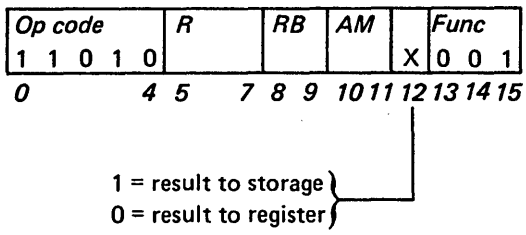
The instruction is terminated.

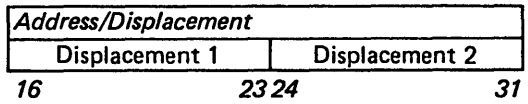
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

OR Doubleword (OD)

Register/Storage Format

OD addr4,reg
reg,addr4





A logical OR operation is performed between the contents of the register pair specified by the R-field and the R+1 field and the doubleword in main storage specified by the effective address. The source operand is not changed.

If the R-field equals 7, register 7 and register 0 are used.

Bit 12 of the instruction specifies the destination of the result.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

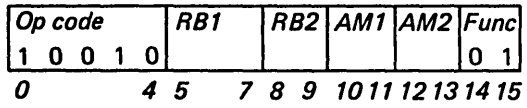
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

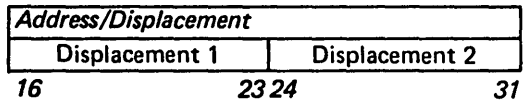
The instruction is terminated.

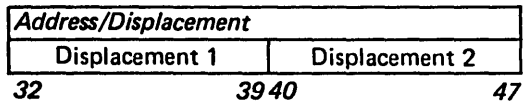
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Storage/Storage Format

OD addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. A doubleword logical OR operation is performed between operand 1 and operand 2. The result replaces operand 2. Operand 1 is not changed.

Note: In case of overlapping operands, operand 1 is fetched in its entirety before operand 2 is stored.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

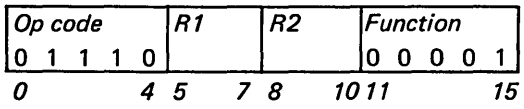
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

OW

OR Word (OW)

Register/Register Format

OW reg,reg



The contents of the register specified by the R1 field are ORed bit-by-bit with the contents of the register specified by the R2 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed.

Indicators

Carry and Overflow. These indicators are not changed.

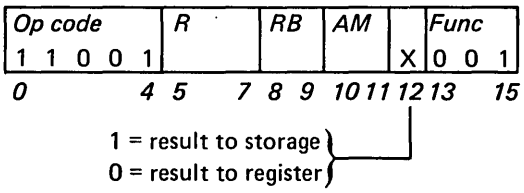
Even, Negative, and Zero. These indicators are changed to reflect the result.

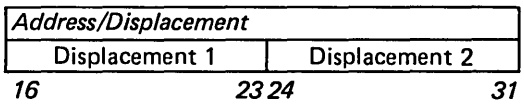
Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Register/Storage Format

OW reg,addr4
addr4,reg





A logical OR operation is performed between the contents of the register specified by the R-field and the location specified by the effective address in main storage. The source operand is not changed.

Bit 12 of the instruction specifies the destination of the result.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

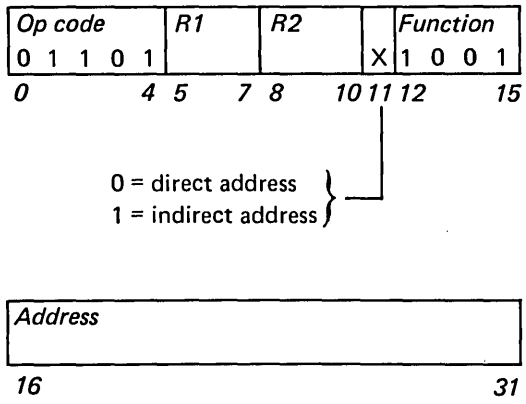
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Storage/Register Long Format

OW longaddr,reg



A logical OR operation is performed between the contents of the main storage location specified by an effective address and the contents of the register specified by the R1 field. The result is placed in the register specified by the R1 field.

The effective main storage address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals 0, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0 (direct address). The result from step 1 is the effective address.

   Bit 11=1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result loaded into the register specified by the R1 field.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

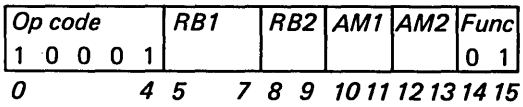
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

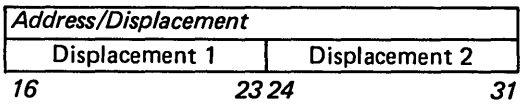
Specification Check. The effective address or indirect address results in an even-byte boundary violation.

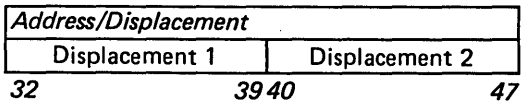
OW

Storage/Storage Format

OW addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. A one word logical OR operation is performed between operand 1 and operand 2. The result replaces operand 2. Operand 1 is not changed.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

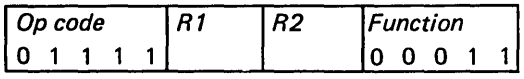
The instruction is terminated.

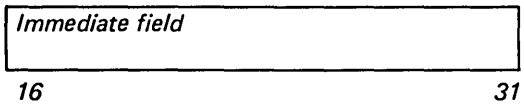
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

OR Word Immediate (OWI)

Register Immediate Long Format

OWI word,reg[,reg]





The immediate field is ORed bit-by-bit with the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed unless the R1 and R2 field specify the same register.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

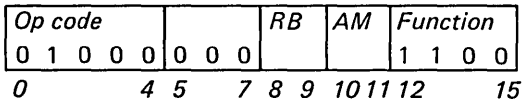
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

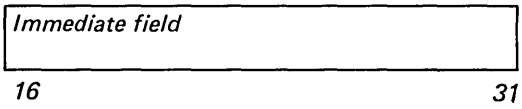
Protect Check. In the problem state, an instruction is fecthed or data is accessed from a storage area not assigned to the current operation.

Storage Immediate Format

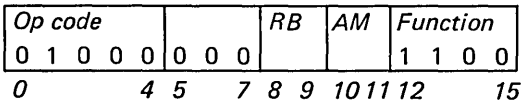
OWI word,addr4

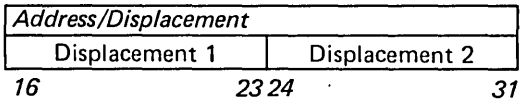
Format without appended word for effective addressing (AM = 00 or 01)

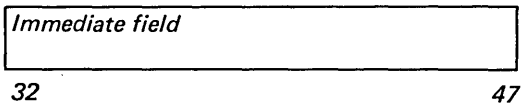




Format with appended word for effective addressing (AM = 10 or 11)







A logical OR operation is performed between the immediate field and the contents of the main storage location specified by the effective address. The result replaces the contents of the storage location. The immediate operand is not changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

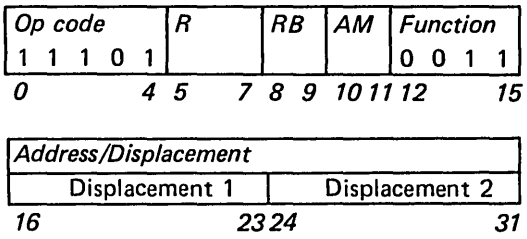
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Pop Byte (PB)

PB addr4,reg



The top element of a byte stack is popped from the stack and loaded into the least-significant byte of the register specified by the R-field. The stack is defined by the stack control block pointed to by the effective address.

Programming Note: If AM equals 01, the register specified by the RB field is incremented by 2.

Refer to "Stack Operations" in Chapter 2 for additional information about the operation of this instruction and the associated stack control block.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

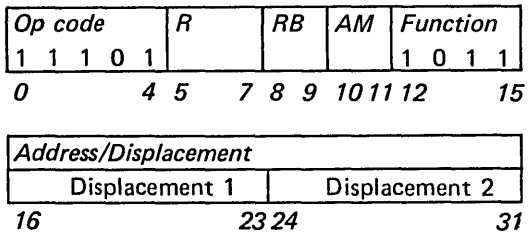
Specification Check. The indirect address or the stack control block results in an even-byte boundary violation.

Soft-Exception Trap Conditions

Stack Exception. The stack is empty.

Pop Doubleword (PD)

PD addr4,reg



The top element of a doubleword stack is popped from the stack and loaded into the register pair specified by the R-field and the R+1 field. The stack is defined by the stack control block pointed to by the effective address.

If the R-field value is 7, registers 7 and 0 are used.

Programming Note: If AM equals 01, the register specified by the RB field is incremented by 2.

Refer to "Stack Operations" in Chapter 2 for additional information about the operation of this instruction and the associated stack control block.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

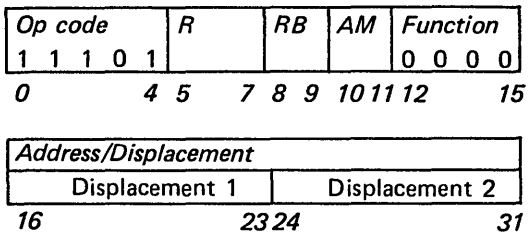
Soft-Exception Trap Conditions

Stack Exception. The stack is empty.

PSB—PSD

Push Byte (PSB)

PSB reg,addr4



The least-significant byte of the register specified by the R-field is pushed into the stack. The stack is defined by the stack control block pointed to by the effective address.

Programming Note: If AM equals 01, the register specified by the RB field is incremented by 2.

Refer to "Stack Operations" in Chapter 2 for additional information about the operation of this instruction and the associated stack control block.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

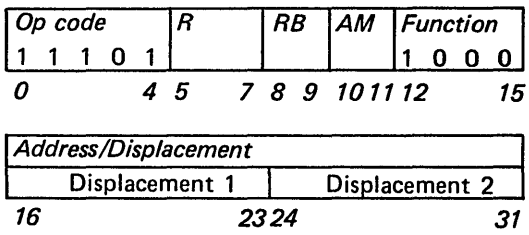
Specification Check. The indirect address or stack control block results in an even-byte boundary violation. The instruction is terminated.

Soft-Exception Trap Conditions

Stack Exception. The stack is full.

Push Doubleword (PSD)

PSD reg,addr4



The doubleword operand contained in the register pair specified by the R-field and the R+1 field is pushed into the stack. The stack is defined by the stack control block pointed to by the effective address.

If the R-field value is 7, registers 7 and 0 are used.

Programming Note: If AM equals 01, the register specified by the RB field is incremented by 2.

Refer to "Stack Operations" in Chapter 2 for additional information about the operation of this instruction and the associated stack control block.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

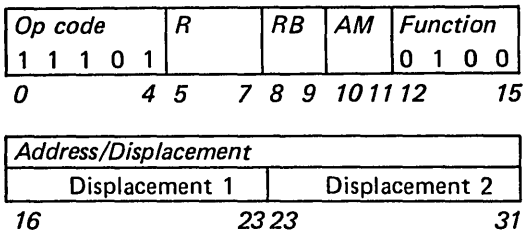
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Soft-Exception Trap Conditions

Stack Exception. The stack is full.

Push Word (PSW)

PSW reg,addr4



The word operand contained in the register specified by the R-field is pushed into the stack. The stack is defined by the stack control block pointed to by the effective address.

Programming Note: If AM equals 01, the register specified by the RB field is incremented by 2.

Refer to "Stack Operations" in Chapter 2 for additional information about the operation of this instruction and the associated stack control block.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

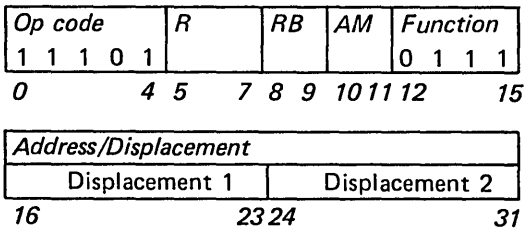
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Soft-Exception Trap Conditions

Stack Exception. The stack is full.

Pop Word (PW)

PW addr4,reg



The top element of a word stack is popped from the stack and loaded into the register specified by the R-field. The stack is defined by the stack control block pointed to by the effective address.

Programming Note: If AM equals 01, the register specified by the RB field is incremented by 2.

Refer to "Stack Operations" in Chapter 2 for additional information about the operation of this instruction and the associated stack control block.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Soft-Exception Trap Conditions

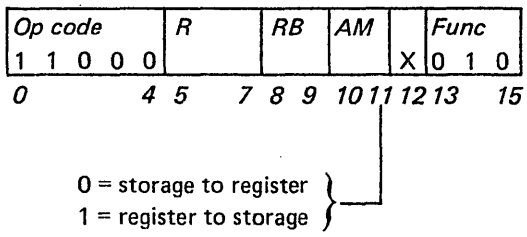
Stack Exception. The stack is empty.

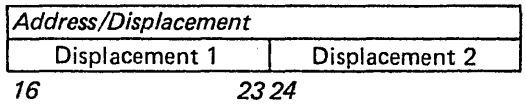
RBTB

Reset Bits Byte (RBTB)

Register/Storage Format

RBTB addr4,reg
reg,addr4





This instruction operates either:

1. Storage to register (instruction bit 12 equals 0), or
2. Register to storage (instruction bit 12 equals 1).

Storage to Register. The specified bits are reset in the least-significant byte of the register specified by the R-field. The bit positions containing 0's correspond to the bit positions containing 1-bits in the main storage byte location specified by the effective address. The remaining bits in the low-order byte of the register are not changed. Bits 0–7 of the register and the storage operand are not changed.

Register to Storage. The specified bits are reset in the main storage byte location specified by the effective address. The bits positions containing 0's correspond to the bit positions containing 1-bits in the least-significant byte of the register specified by the R-field. The remaining bits in the storage location are not changed. The register operand is not changed.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

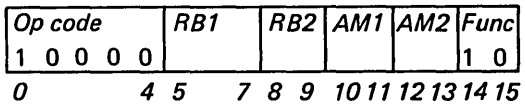
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

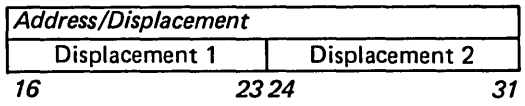
The instruction is terminated.

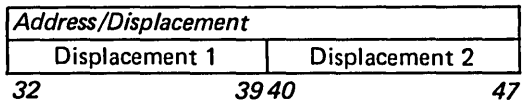
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Storage/Storage Format

RBTB addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. The bit positions containing 1-bits in byte operand 1 determine the bit positions set to 0's in byte operand 2. The remaining bits in operand 2 are not changed. The result replaces operand 2. Operand 1 is not changed.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

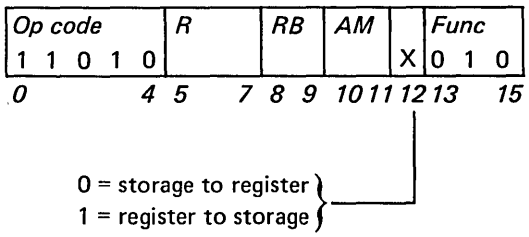
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

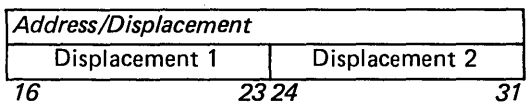
RBTD

Reset Bits Doubleword (RBTD)

Register/Storage Format

RBTD addr4,reg
reg,addr4





This instruction operates either:

1. Storage to register (instruction bit 12 equals 0), or
2. Register to storage (instruction bit 12 equals 1).

Storage to Register. The specified bits are reset in the register pair specified by the R-field and the R+1 field. The bit positions containing 0's correspond to the bit positions containing 1-bits in the doubleword main storage location specified by the effective address. The remaining bits in the register pair are not changed. The storage operand is not changed.

Register to Storage. The specified bits are reset in the doubleword main storage location specified by the effective address. The bit positions containing 0's correspond to the bit positions containing 1-bits in the register pair specified by the R-field and the R+1 field. The remaining bits in the storage operand are not changed. The register operand is not changed.

If the R-field value is 7, registers 7 and 0 are used.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

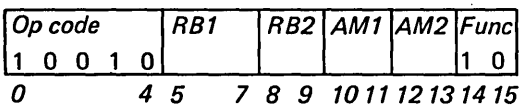
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

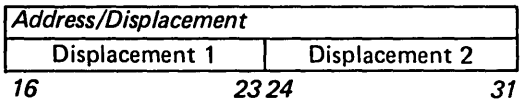
The instruction is terminated.

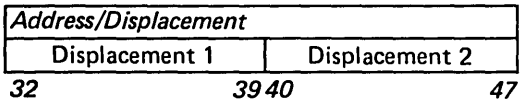
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Storage/Storage Format

RBTD addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. The bit positions containing 1-bits in doubleword operand 1 determine the bit positions set to 0's in doubleword operand 2. The remaining bits in operand 2 are not changed. The result replaces operand 2. Operand 1 is not changed.

Note: In case of overlapping operands, operand 1 is fetched in its entirety before operand 2 is stored.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

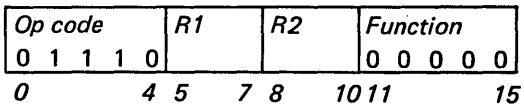
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Reset Bits Word (RBTW)

Register/Register Format

RBTW reg,reg



The bit positions containing 1-bits in the register specified by the R1 field determine the bit positions set to 0's in the register specified by the R2 field. The remaining bits in the register specified by the R2 field are not changed. The contents of the register specified by the R1 field are not changed unless the R1 and R2 field specify the same register.

Indicators

Carry and Overflow. These indicators are not changed.

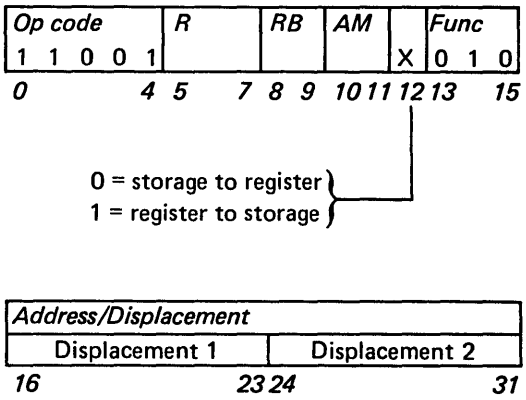
Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Register/Storage Format

RBTW addr4,reg
reg,addr4



This instruction operates either:

1. Storage to register (instruction bit 12 equals 0), or
2. Register to storage (instruction bit 12 equals 1).

Storage to Register. The specified bits are reset in the register specified by the R-field. The bit positions containing 0's correspond to the bit positions containing 1-bits in the main storage word location specified by the effective address. The remaining bits in the register are not changed. The storage operand is not changed.

Register to Storage. The specified bits are reset in the main storage word location specified by the effective address. The bit positions containing 0's correspond to the bit positions containing 1-bits in the register specified by the R-field. The remaining bits in the storage operand are not changed. The register operand is not changed.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

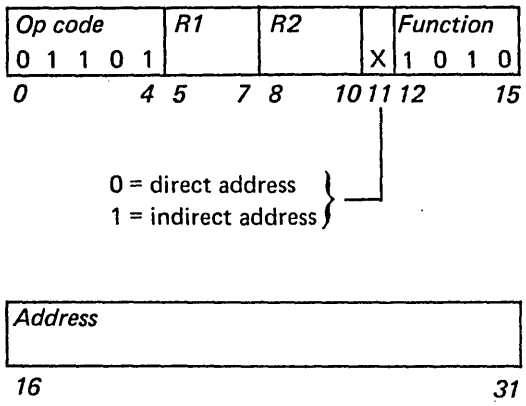
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Storage/Register Long Format

RBTW longaddr,reg



The bit positions containing 1-bits in the main storage word location specified by the effective address determine the bit positions set to 0's in the register specified by the R1 field. The remaining bits in the register specified by the R1 field are not changed. The storage operand is not changed.

The effective address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals 0, no register contributes to the address generation. The contents of the R2 field are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0 (direct address). The result from step 1 is the effective address.

   Bit 11=1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

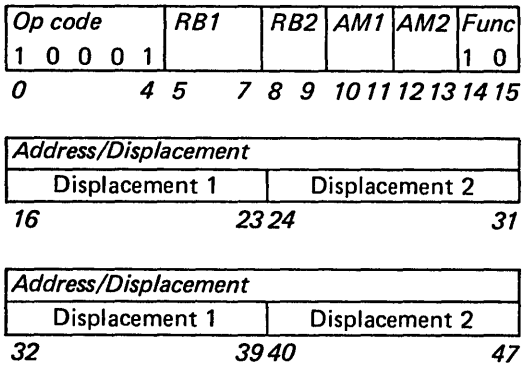
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Specification Check. The effective address or indirect address results in an even-byte boundary violation.

RBTW—RBTWI

Storage/Storage Format

RBTW addr5,addr4



The address arguments generate the effective addresses of two operands in main storage. The bit positions containing 1-bits in word operand 1 determine the bit positions set to 0's in word operand 2. The remaining bits in operand 2 are not changed. The result replaces operand 2. Operand 1 is not changed.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

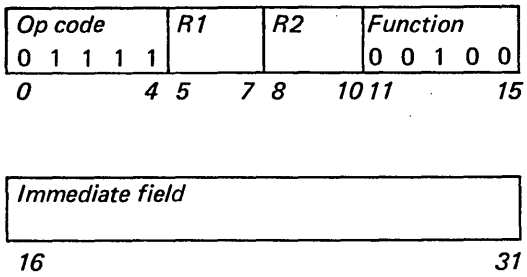
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Reset Bits Word Immediate (RBTWI)

Register Immediate Long Format

RBTWI word,reg[,reg]



The bit positions containing 1-bits in the immediate field are set to 0's in the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field.

The contents of the register specified by the R1 field are not changed unless the R1 and R2 field specify the same register.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

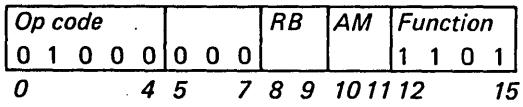
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

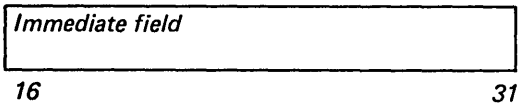
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Storage Immediate Format

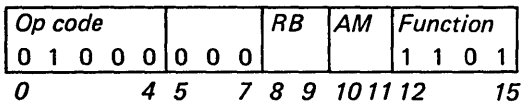
RBTWI word,addr4

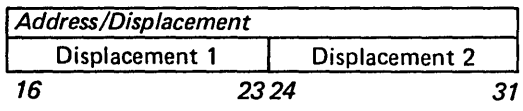
Format without appended word for effective addressing (AM = 00 or 01)

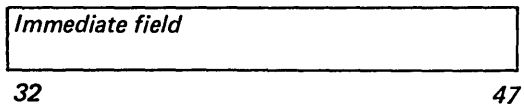




Format with appended word for effective addressing (AM = 10 or 11)







The bit positions containing 1-bits in the immediate field determine the bit positions set to 0's in the main storage location specified by the effective address. The immediate operand and source operand are not changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

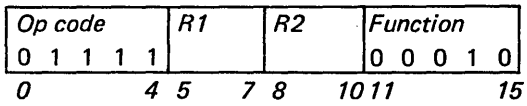
The instruction is terminated.

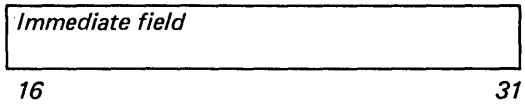
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Subtract Address (SA)

Register Immediate Long Format

SA raddr,reg[,reg]





The immediate field (an address value) is subtracted from the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed unless the R1 and R2 field specify the same register.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

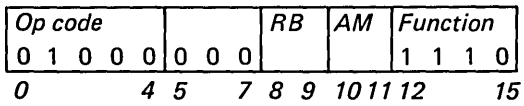
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

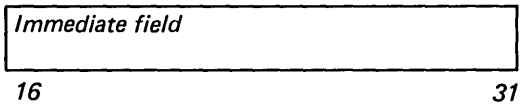
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Storage Immediate Format

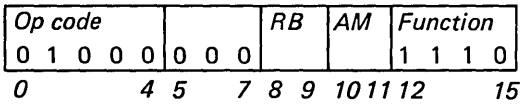
SA raddr,addr4

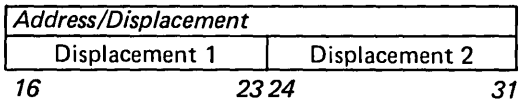
Format without appended word for effective addressing (AM = 00 or 01)

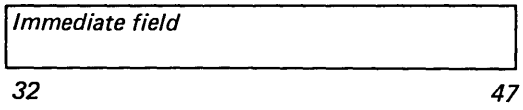




Format with appended word for effective addressing (AM = 10 or 11)







The immediate field (an address value) is subtracted from the contents of the main storage location specified by the effective address. The result replaces the contents of the storage location specified by the effective address. The immediate operand is not changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

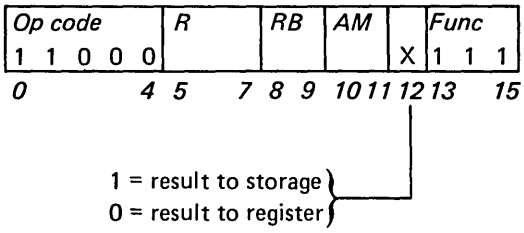
The instruction is terminated.

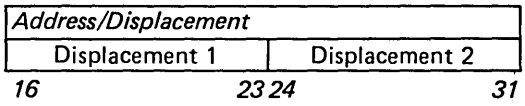
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

SB

Subtract Byte (SB)

SB reg,addr4
addr4,reg





A subtract operation is performed between the least-significant byte of the register specified by the R-field and the location specified by the effective address in main storage. The source operand and high-order byte of the register are not changed.

Bit 12 of the instruction specifies the destination of the result.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the byte, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one byte; that is, if the difference is less than -2^7 or greater than $+2^7-1$.

If an overflow occurs, the result contains the correct low-order eight bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

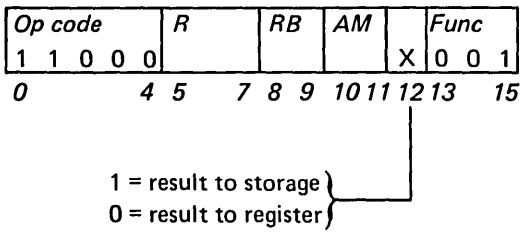
The instruction is terminated.

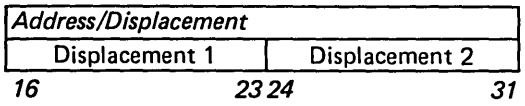
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Set Bits Byte (SBTB)

Register/Storage Format

SBTB reg,addr4
addr4,reg





A logical OR operation is performed between the least-significant byte of the register specified by the R-field and the location specified by the effective address in main storage. The source operand is not changed. When going from storage to register, bits 0–7 of the register are not changed.

Bit 12 of the instruction specifies the destination of the result.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

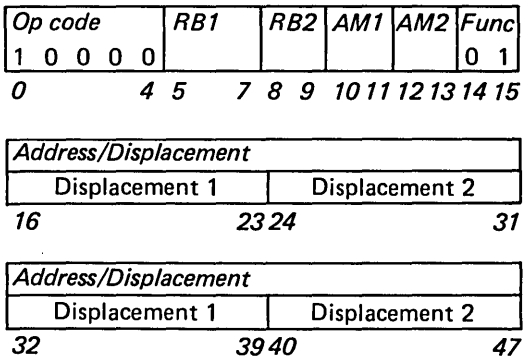
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

SBTB

Storage/Storage Format

SBTB addr5,addr4



The address arguments generate the effective addresses of two operands in main storage. A one byte logical OR operation is performed between operand 1 and operand 2. The result replaces operand 2.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

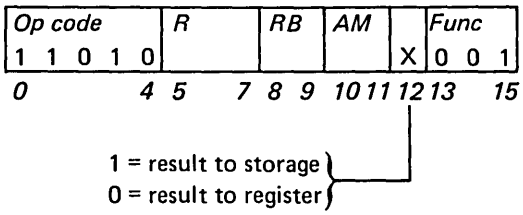
The instruction is terminated.

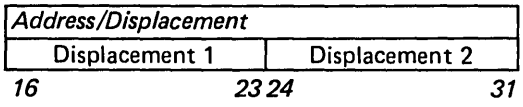
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Set Bits Doubleword (SBTD)

Register/Storage Format

SBTD addr4,reg
reg,addr4





A logical OR operation is performed between the contents of the register pair specified by the R-field and the R+1 field and the doubleword in main storage specified by the effective address. The source operand is not changed.

If the R-field value is 7, registers 7 and 0 are used.

Bit 12 of the instruction specifies the destination of the result.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

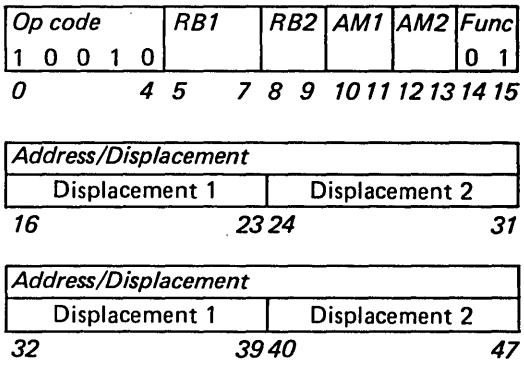
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

SBTD

Storage/Storage Format

SBTD addr5,addr4



The address arguments generate the effective addresses of two operands in main storage. A doubleword logical OR operation is performed between operand 1 and operand 2. The result replaces operand 2. Operand 1 is not changed.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

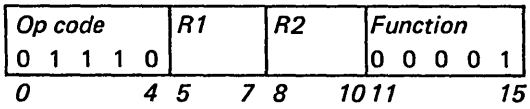
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Set Bits Word (SBTW)

Register/Register Format

SBTW reg,reg



The contents of the register specified by the R1 field are ORed bit-by-bit with the contents of the register specified by the R2 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed.

Indicators

Carry and Overflow. These indicators are not changed.

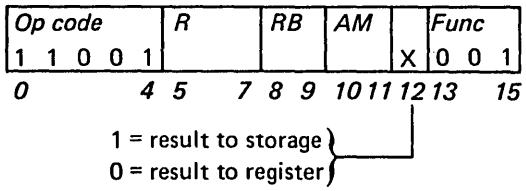
Even, Negative, and Zero. These indicators are changed to reflect the result.

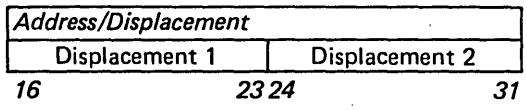
Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Register/Storage Format

SBTW reg,addr4
addr4,reg





A logical OR operation is performed between the contents of the register specified by the R-field and the location specified by the effective address in main storage. The source operand is not changed.

Bit 12 of the instruction specifies the destination of the result.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

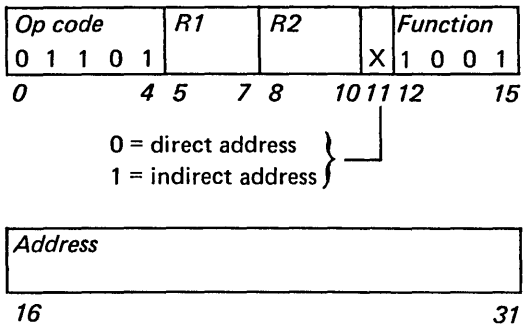
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

SBTW

Storage/Register Long Format

SBTW longaddr,reg



A logical OR operation is performed between the contents of the main storage location specified by an effective address and the contents of the register specified by the R1 field. The result is placed in the register specified by the R1 field.

The effective main storage address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals 0, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0 (direct address). The result from step 1 is the effective address.

   Bit 11=1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

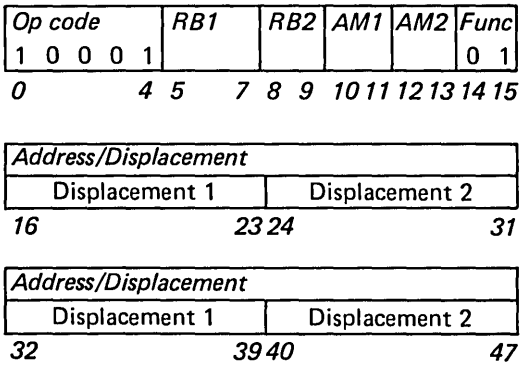
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Specification Check. The effective address or indirect address results in an even-byte boundary violation.

Storage/Storage Format

SBTW addr5,addr4



The address arguments generate the effective addresses of two operands in main storage. A one-word logical OR operation is performed between operand 1 and operand 2. The result replaces operand 2. Operand 1 is not changed.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

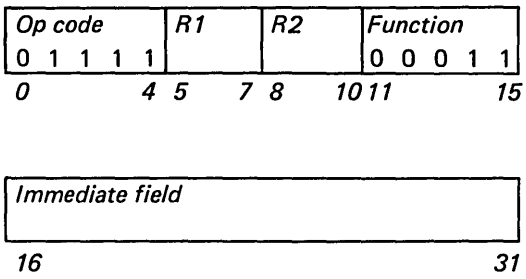
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Set Bits Word Immediate (SBTWI)

Register Immediate Long Format

SBTWI word,reg[,reg]



The immediate field is ORed bit-by-bit with the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed unless the R1 and R2 field specify the same register.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

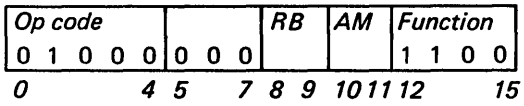
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

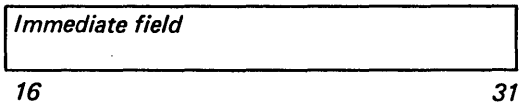
SBTWI

Storage Immediate Format

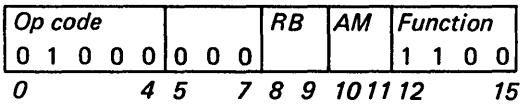
SBTWI word,addr4

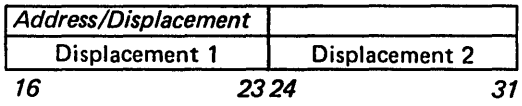
Format without appended word for effective addressing (AM = 00 or 01)

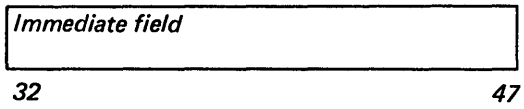




Format with appended word for effective addressing (AM = 10 or 11)







A logical OR operation is performed between the immediate field and the contents of the main storage location specified by the effective address. The result replaces the contents of the storage location. The immediate operand is not changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

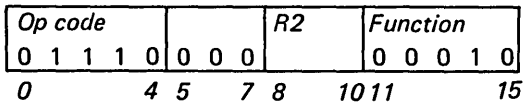
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Subtract Carry Indicator (SCY)

SCY reg



The value of the carry indicator on entry is subtracted from the contents of the register specified by the R2 field. The result is placed in the register specified by the R2 field.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Programming Note. This instruction can be used when subtracting multiple word operands. See "Indicators—Multiple Word Operands" in Chapter 2.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even. The even indicator is not changed.

Negative. The negative indicator is changed to reflect the result.

Zero. If on at entry, the zero indicator is changed to reflect the result. If off at entry, it remains off.

Program-Check Conditions

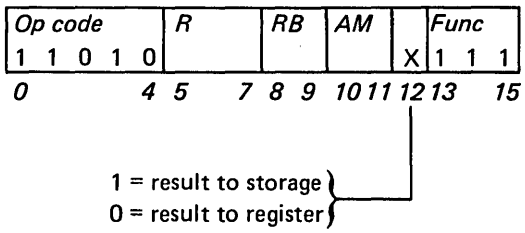
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

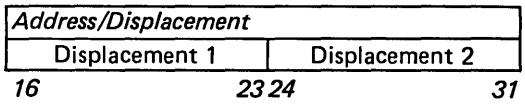
SD

Subtract Doubleword (SD)

Register/Storage Format

SD reg,addr4
addr4,reg





A subtract operation is performed between the register pair specified by the R-field and the R+1 field and the doubleword in main storage specified by the effective address. The source operand is not changed.

If the R-field value is 7, registers 7 and 0 are used.

Bit 12 of the instruction specifies the destination of the result.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the doubleword, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in the doubleword; that is, if the difference is less than -2^{31} or greater than $+2^{31}-1$.

If an overflow occurs, the result contains the correct low-order 32 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

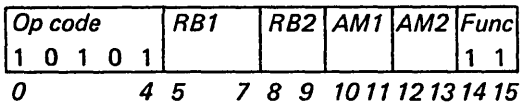
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

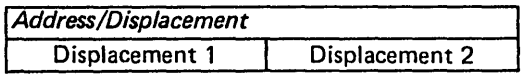
The instruction is terminated.

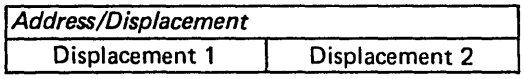
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Storage/Storage Format

SD addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. Doubleword operand 1 is subtracted from doubleword operand 2. The result replaces operand 2. Operand 1 is not changed.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the doubleword, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in the doubleword; that is, if the difference is less than -2^{31} or greater than $+2^{31}-1$.

If an overflow occurs, the result contains the correct low-order 32 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

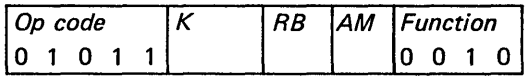
SEAKR

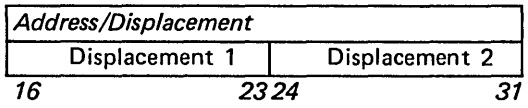
Set Address Key Register (SEAKR)

System Register/Storage Format

Mnemonic	Syntax	Instruction name	K-field
SEAKR	addr4	Set Address Key Register	011

Extended mnemonic	Syntax	Instruction name	K-field
SEISK	addr4	Set Instruction Space Key	000
SEOOK	addr4	Set Operand 1 Key	010
SEOTK	addr4	Set Operand 2 Key	001





The address key register (AKR) field, specified by the K-field, is loaded from the word location in main storage specified by the effective address. The contents of the word in main storage are not changed.

The K-field can specify either a field within the AKR or the entire AKR.

K-field	Address key register field name	Bits
000	Instruction space key	13–15
001	Operand 2 key	9–11
010	Operand 1 key	5–7
011	Address key register	0–15
100	See Note	
101	See Note	
110	See Note	
111	See Note	

Note: To avoid future program obsolescence, these K-field values should not be used.

If the K-field specifies a specific field within the AKR, bits 13–15 from the word location in main storage are loaded into the AKR field. If the K-field specifies the entire AKR, bits 0–15 from the word location in main storage are loaded into the AKR.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

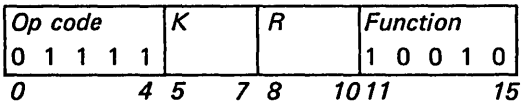
Privilege Violate. In the problem state, a privileged instruction is encountered.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

System Register/Register Format

Mnemonic	Syntax	Instruction name	K-field
SEAKR	reg	Set Address Key Register	011

Extended mnemonic	Syntax	Instruction name	K-field
SEISK	reg	Set Instruction Space Key	000
SEOOK	reg	Set Operand 1 Key	010
SEOTK	reg	Set Operand 2 Key	001



The address key register (AKR) field, specified by the K-field, is loaded from the register specified by the R-field. The contents of the register are not changed.

The K-field can specify either a field within the AKR or the entire AKR.

K-field	Address key register field name	Bits
000	Instruction space key	13–15
001	Operand 2 key	9–11
010	Operand 1 key	5–7
011	Address key register	0–15
100	See Note	
101	See Note	
110	See Note	
111	See Note	

Note: To avoid future program obsolescence, these K-field values should not be used.

If the K-field specifies a specific field within the AKR, bits 13–15 from the register specified by the R-field are loaded into the AKR field. If the K-field specifies the entire AKR, bits 0–15 from the specified register are loaded into the AKR.

Indicators

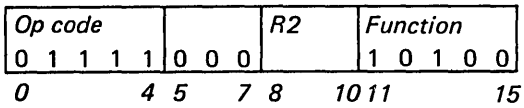
The indicators are not changed.

Program-Check Conditions

Privilege Violate. In the problem state, a privileged instruction is encountered.

Set Clock (SECLK)

SECLK reg



The registers specified by the R2 field and the R2+1 field contain a doubleword value that represents time in milliseconds. This value is set into the clock register. The register specified by the R2 field and the R2+1 field are not changed.

If the R2 field value is 7, registers 7 and 0 are used.

Bits 5–7 of the instruction are not used, and must be set to 0's to avoid future code obsolescence.

Indicators

The indicators are not changed.

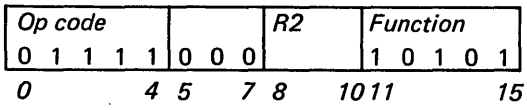
Program-Check Conditions

Privilege Violate. In the problem state, a privileged instruction is encountered.

SECMP—SECON

Set Comparator (SECMP)

SECMP reg



The registers specified by the R2 field and the R2+1 field contain a doubleword value that represents time in milliseconds. This value is set into the comparator register. The register specified by the R2 field and the R2+1 field are not changed.

If the R2 field value is 7, registers 7 and 0 are used.

Bits 5–7 of the instruction are not used, and must be set to 0's to avoid future code obsolescence.

Indicators

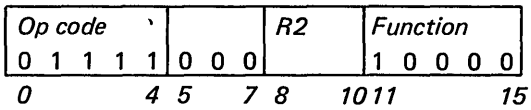
The indicators are not changed.

Program-Check Conditions

Privilege Violate. In the problem state, a privileged instruction is encountered.

Set Console Data Lights (SECON)

SECON reg



The contents of the register specified by the R2 field are stored in the console data lights. The contents of the register are unchanged.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future conde obsolescence.

If the programmer console is not installed, the instruction performs no operation.

Indicators

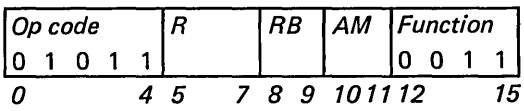
The indicators are not changed.

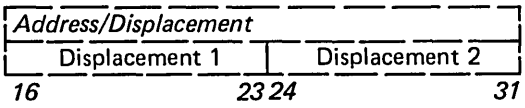
Program-Check Conditions

Privilege Violate. In the problem state, a privileged instruction is encountered.

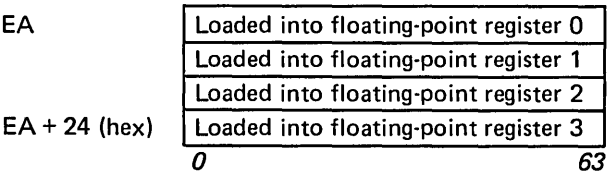
Set Floating Level Block (SEFLB)

SEFLB reg,addr4

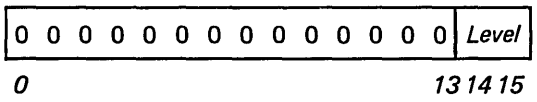




A floating level block in main storage is loaded into the floating-point registers for the level specified by the register specified by the R-field. The generated effective address specifies the beginning address of the floating level block. The contents of main storage and the register specified by the R-field are not changed. The floating level block appears in main storage as follows:



The general register specified by the R-field has the following format:



Bits 0–7, 12, and 13 must be set to 0's to avoid future code obsolescence. Bits 8–11 must be set to 0's to select the floating-point feature. Bits 14 and 15 hold the binary-encoded level of the floating level block associated with this operation. For example, 00 for level 0, 01 for level 1, 10 for level 2, and 11 for level 3.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Privilege Violate. In the problem state, a privileged instruction is encountered.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

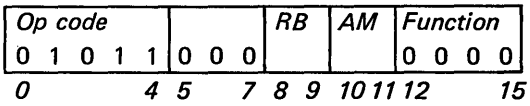
Soft-Exception Trap Conditions

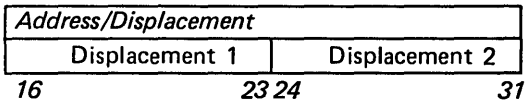
Invalid Function. An attempt has been made to execute a floating-point instruction when the floating-point feature is not installed. The instruction is terminated.

SEIMR—SEIND

Set Interrupt Mask Register (SEIMR)

SEIMR addr4





The contents of the word location in main storage specified by the effective address are loaded into the interrupt mask register. The contents of main storage are not changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

The mask is represented in a bit-significant manner, with bit 0 representing level 0, and so on. (See "Interrupt Masking Facilities" in Chapter 3.) Bit 4–15 are set to 0's.

Indicators

The indicators are not changed.

Program-Check Conditions

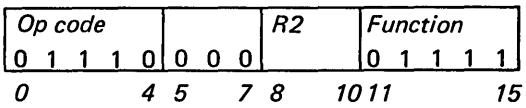
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Privilege Violate. In the problem state, a privileged instruction is encountered.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Set Indicators (SEIND)

SEIND reg



Bits 0–7 of the register specified by the R2 field are loaded into bits 0–7 of the current level status register (indicators). Bits 8–15 of the register specified by the R2 field are ignored. Bits 8–15 of the level status register are not changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

The following table shows the indicator bits of the level status register (LSR):

LSR bit	Indicator
0	Even
1	Carry
2	Overflow
3	Negative
4	Zero

Indicators

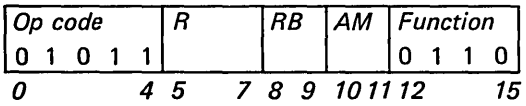
The indicators are changed as specified by the register designated by the R2 field.

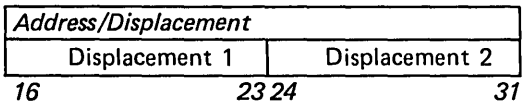
Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Set Level Block (SELB)

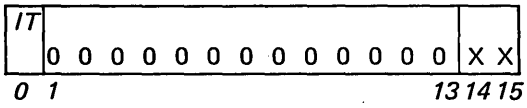
SELB reg,addr4





This instruction loads a level status block (LSB), from 11 words of main storage, into the hardware LSB for a selected level. The beginning location for the main storage LSB is specified by the effective address. The contents of the storage locations are not changed.

The format of the register specified by the R-field is:



Inhibit Trace (IT) Bit

When bit 0 (IT) of the register specified by the R-field and bit 10 (trace bit) of the selected level status register (LSR) are set to 1, the SELB and the instruction pointed to by the IAR within the main storage LSB are executed before trace interrupts are allowed.

If bit 0 is set to 0 and bit 10 in the LSR of the main storage LSB is set to 1, the SELB instruction is executed and trace interrupts are allowed.

The main storage LSB contains data produced prior to the execution of the SELB instruction. When the instruction is executed, the data in the main storage LSB replaces the chosen priority-level hardware LSB data.

Bits 1–13 of the register are not used and must be set to 0's to avoid future code obsolescence.

Bits 14–15 of the register specified by the R-field specify the selected level.

Bits	**14**	**15**
Level 0	0	0
Level 1	0	1
Level 2	1	0
Level 3	1	1

Refer to "Program-Controlled Level Switching" in Chapter 3 for further information.

Programming Notes:

1. When trace is enabled, double tracing can be prevented by exiting the trace routine with a SELB instruction that has the IT bit set to 1.
2. Trace interrupts are inhibited on a level exit when the SELB sets the current level in-process bit to 0 and the trace bit to 1.
3. The current level AKR and LSR registers are not changed except when bits 14 and 15 of the specified R-field register select the same level.
4. If the AM field equals 01, the contents of the register specified by the RB field are incremented by 2.

Execution of the SELB instruction can cause the processor to change levels. Also, the processor may exit supervisor state. For additional information about the processor action when executing this instruction, refer to "Program-Controlled Level Switching" in Chapter 3.

Indicators

The indicators are not changed if the specified level is other than the current level.

SELB

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

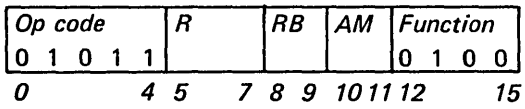
Protect Check. The instruction loads an LSB that causes the processor to enter the problem state. The instruction defined by the LSB is accessed from a storage area not assigned to the current operation. The instruction pointed to by the hardware LSB is suppressed. The SELB instruction is terminated.

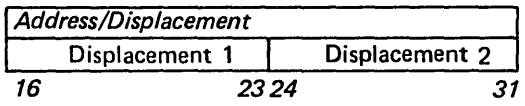
Privilege Violate. In the problem state, a privileged instruction is encountered.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Set Storage Key (SESK)

SESK reg,addr4



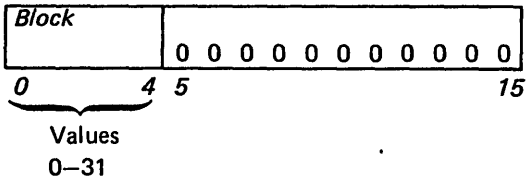


This instruction loads a storage key register with the contents of the byte location in main storage specified by the effective address.

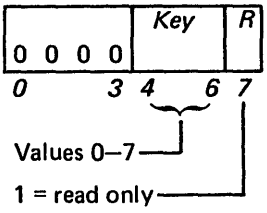
The register specified by the R-field contains the main storage block number for the storage key register to be loaded. (A storage key register is associated with every 2048 bytes of storage.) The block number is binary encoded in bits 0–4 of the register.

Bits 5–15 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

The format of the register specified by the R-field is:



The format of the byte at the storage location is:



Bits 0–3 are not used and must be set to 0's to avoid future code obsolescence. Bits 4–7 are the storage key and read-only bit for the selected storage block.

The contents of the storage location are not changed.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

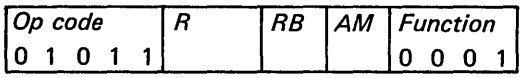
Privilege Violate. In the problem state, a privileged instruction is encountered.

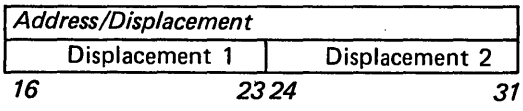
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

SESR

Set Segmentation Register (SESR)

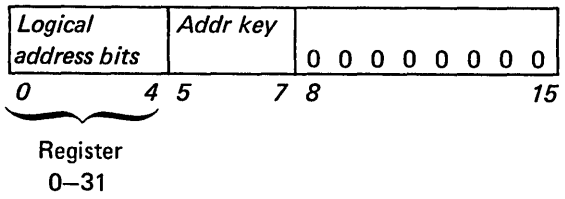
SESR reg,addr4





This instruction loads a segmentation register with the contents of the doubleword location in main storage specified by the effective address.

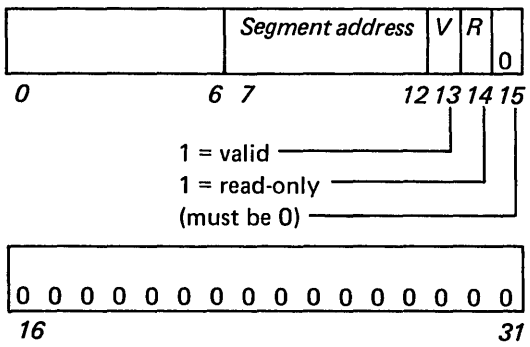
The general register specified by the R-field must contain the logical address of a segmentation register (0–31, decimal) in bits 0–4, and an address key value of 000 to 111 in bits 5–7. Bits 8–15 of the register must be set to 0's. The format of the general register specified by the R-field is:



The logical address of the register selects a specific segmentation register (0–31) in a segmentation register stack 0–7.

The address-key field of the register selects a stack 0–7 of the segmentation registers.

The first word of the specified doubleword that is loaded into the selected segmentation register has the following format:



The segment address (bits 7–12) contains the six high-order bits of the physical address, which is used by the translator to select a 2K-byte block of main storage.

Bit 13, if a 1, signifies that the contents of the segmentation register are valid, and translation can be performed. If an attempt is made to use a segmentation register with bit 13 set to 0, a program-check interrupt occurs with invalid storage address set in the PSW.

Bit 14, if a 1, signifies that the block is read-only. If an attempt is made to write into the block when bit 14 of the associated segmentation register is a 1 and while in problem state, a program-check interrupt occurs, with protect check set in the PSW. When in supervisor state or on a cycle-steal access, bit 14 is ignored. The contents of main storage can be changed.

The second word (bits 16–31) of the specified doubleword must be set to 0 to avoid future code obsolescence.

Chapter 5 describes the relocation translator and relocation addressing. Refer to "Storage Mapping" in Chapter 5 for an example of loading segmentation registers.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Function. In the supervisor state, an attempt has been made to execute this instruction when the translator is not enabled.

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

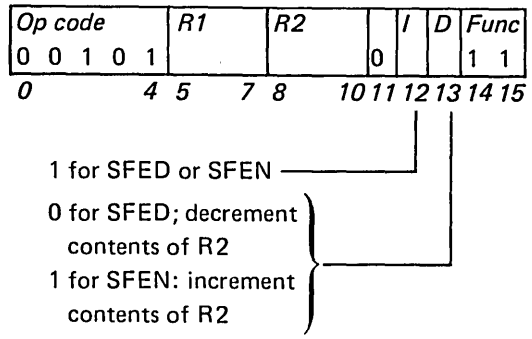
Privilege Violate. In the problem state, a privileged instruction is encountered.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Scan Byte Field Equal and Decrement (SFED)

Scan Byte Field Equal and Increment (SFEN)

SFED reg,(reg)
SFEN reg,(reg)



This instruction compares a field in main storage against a single byte contained in a register. Register 7 contains the number of bytes to be compared. This number is decremented after each byte is compared.

The register specified by the R1 field contains, in bits 8–15, the single byte of operand 1. The register specified by the R2 field contains the starting address of operand 2. Operand 1 is subtracted from operand 2, but neither operand is changed.

After each byte is compared, the address in the R2 field is incremented or decremented (as determined by bit 13 of the instruction). The operation terminates when either:

1. An equal condition is detected, or
2. The number of bytes specified in register 7 has been compared.

When an equal condition occurs, the address in the register specified by the R2 field points to the next operand to be compared, but the count in register 7 is not updated.

Bit 11 of the instruction is not used and must be set to 0 to avoid future code obsolescence.

See "Compare Byte Field Equal and Decrement (CFED)" and "Compare Byte Field Equal and Increment (CFEN)" for other versions of this machine instruction.

Notes:

1. Variable-field-length instructions can be interrupted. When this occurs and the interrupted level resumes operation, the processor treats the incomplete instruction as a new instruction, with the remaining byte count specified in register 7.
2. If the specified count in register 7 is 0, the instruction performs no operation (no-op).

Indicators

Carry. If a borrow is detected out of the high-order bit position of the byte, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one byte; that is, if the difference is less than -2^7 or greater than $+2^7-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result of the subtract operation.

Program-Check Conditions

Invalid Function. Register 7 is specified in the R1 or R2 field of the instruction. The instruction is terminated.

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

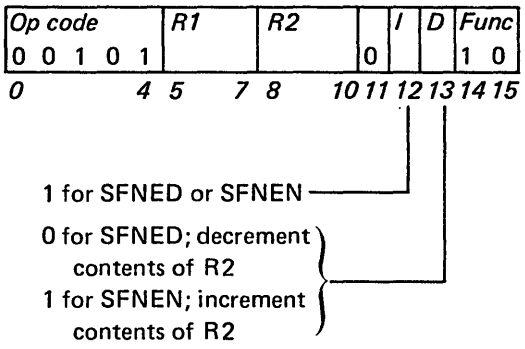
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

SFNED—SFNEN

Scan Byte Field Not Equal and Decrement (SFNED)

Scan Byte Field Not Equal and Increment (SFNEN)

SFNED reg,(reg)
SFNEN reg,(reg)



This instruction compares a field in main storage against a single byte contained in a register. Register 7 contains the number of bytes to be compared. This number is decremented after each byte is compared.

The register specified by the R1 field contains, in bits 8–15, the single byte of operand 1. The register specified by the R2 field contains the starting address of operand 2. Operand 1 is subtracted from operand 2, but neither operand is changed.

After each byte is compared, the address in the R2 field is incremented or decremented (as determined by bit 13 of the instruction). The operation terminates when either:

1. An unequal condition is detected, or
2. The number of bytes specified in register 7 has been compared.

When an unequal condition occurs, the address in the register specified by the R2 field points to the next operand to be compared, but the count in register 7 is not updated.

Bit 11 of the instruction is not used and must be set to 0 to avoid future code obsolescence.

See "Compare Byte Field Not Equal and Decrement (CFNED)" and "Compare Byte Field Not Equal and Increment (CFNEN)" for other versions of this machine instruction.

Notes:

1. Variable-field-length instructions can be interrupted. When this occurs and the interrupted level resumes operation, the processor treats the incomplete instruction as a new instruction, with the remaining byte count specified in register 7.
2. If the specified count in register 7 is 0, the instruction performs no operation (no-op).

Indicators

Carry. If a borrow is detected out of the high-order bit position of the byte, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one byte; that is, if the difference is less than -2^7 or greater than $+2^7-1$.

Even, Negative, and Zero. These indicators are changed to reflect the result of the subtract operation.

Program-Check Conditions

Invalid Function. Register 7 is specified in the R1 or R2 field of the instruction. The instruction is terminated.

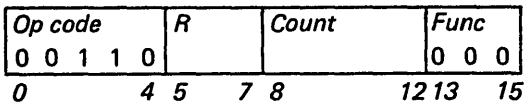
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Shift Left Circular (SLC)

Immediate Count Format

SLC cnt16,reg



The bits in the register specified by the R-field are shifted left by the number of bit positions specified in the count field. The bits shifted out of the high-order bit (bit 0) reenter at the low-order bit (bit 15). If the shift count is 0, no shifting occurs.

Although the register to be shifted contains only 16 bits, shift count values of 0–31 may be specified. Shift counts greater than 16 lengthen the execution time and provide an effective shift of modulo 16.

Indicators

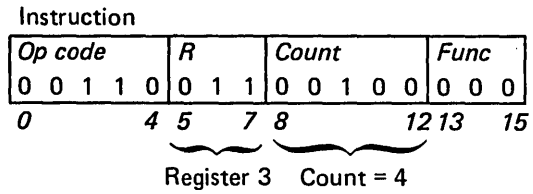
Carry and Overflow. These indicators are not changed.

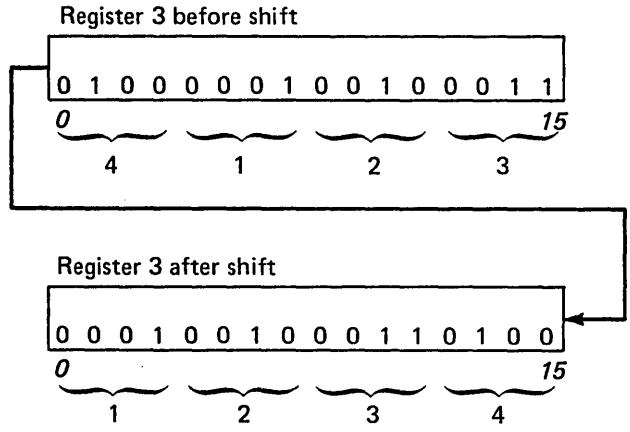
Even, Negative, and Zero. These indicators are changed to reflect the final contents of the register.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Example:

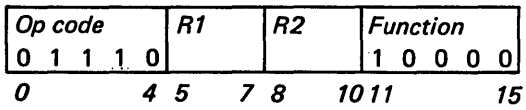




SLC

Count in Register Format

SLC reg,reg



Note: In the assembler syntax, operand 1 is the register that contains the shift count. Operand 2 is the register that is shifted.

The bits in the register specified by the R1 field are shifted left by the number of bits specified by the shift count. This count is obtained from bits 8–15 of the register specified by the R2 field.

The contents of the register specified by the R2 field are not changed unless the R1 and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the register to be shifted contains only 16 bits, shift count values of 0–255 may be specified. Shift counts greater than 16 lengthen the execution time and provide an effective shift of modulo 16.

Indicators

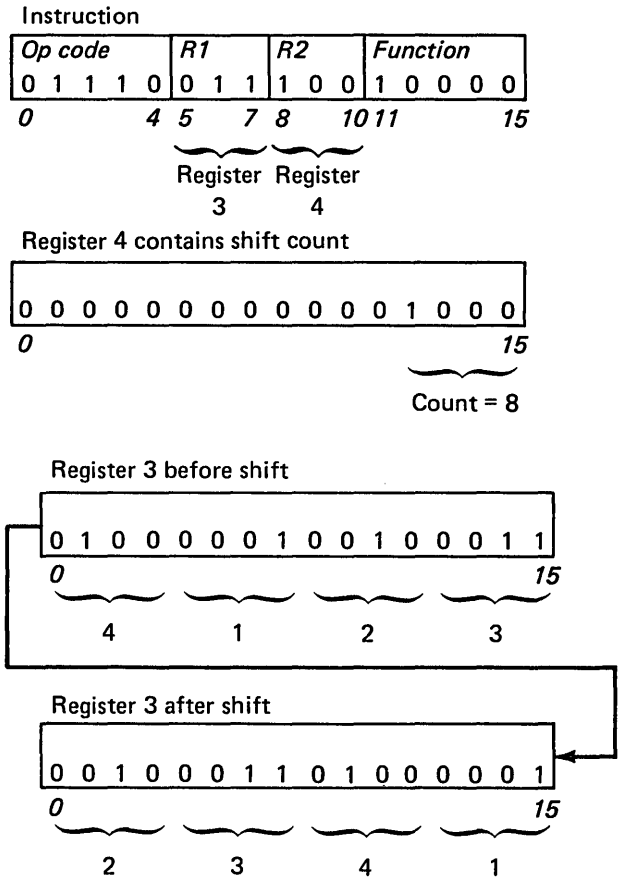
Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the final contents of the register specified by the R1 field.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

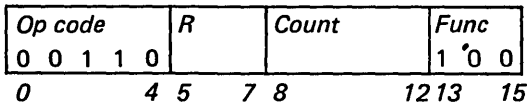
Example:



Shift Left Circular Double (SLCD)

Immediate Count Format

SLCD cnt31,reg



The bits in the register pair specified by the R-field and the R+1 field are shifted left by the number of bit positions specified in the count field.

Within the register pair, the register specified by the R-field contains the high-order word (bits 0–15); the register specified by the R+1 field contains the low-order word (bits 16–31). The bits shifted out of the high-order bit (bit 0) reenter at the low-order bit (bit 31). If the shift count is 0, no shifting occurs.

If the R-field value is 7, registers 7 and 0 are used for the register pair.

Indicators

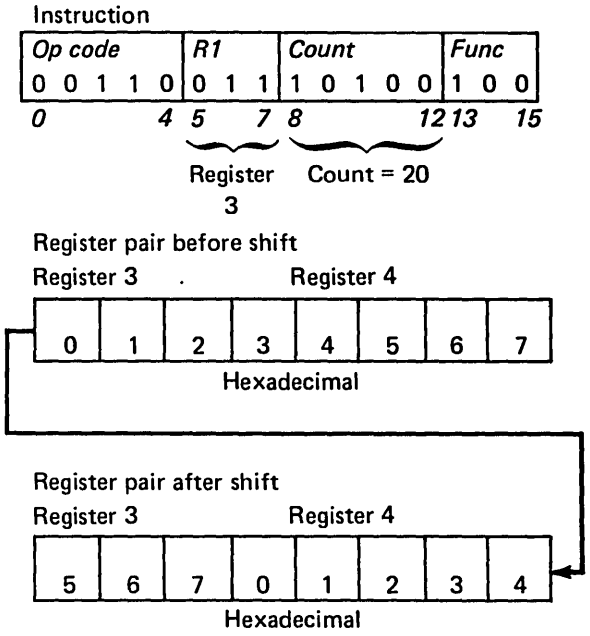
Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the final contents of the two registers.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

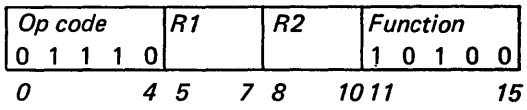
Example:



SLCD

Count in Register Format

SLCD reg,reg



Note: In the assembler syntax, operand 1 is the register that contains the shift count. Operand 2 is the register that is shifted.

The bits in the register pair specified by the R1 field and the R1+1 field are shifted left by the number of bits specified by the shift count. This count is obtained from bits 8–15 of the register specified by the R2 field.

Within the register pair, the register specified by the R1 field contains the high-order word (bits 0–15); the register specified by the R1+1 field contains the low-order word (bits 16–31). The bits shifted out of the high-order bit (bit 0) reenter at the low-order bit (bit 31). If the shift count is 0, no shifting occurs.

If the R1 field value is 7, registers 7 and 0 are used for the register pair.

The contents of the register specified by the R2 field are not changed unless the R1 (or R1+1) and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the registers to be shifted represent 32 bits, shift count values of 0–255 may be specified. Shift-count values greater than 32 lengthen the execution time and provide an effective shift of modulo 32.

Indicators

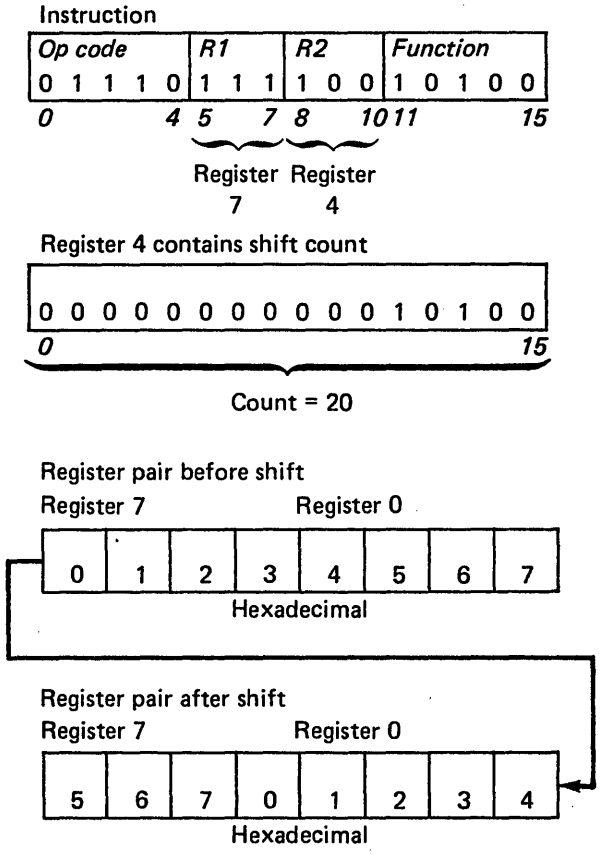
Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the final contents of the two registers.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

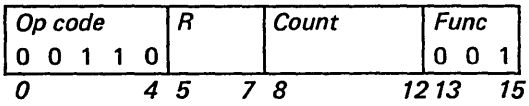
Example:



Shift Left Logical (SLL)

Immediate Count Format

SLL cnt16,reg



The bits in the register specified by the R-field are shifted left by the number of bit positions specified in the count field. The vacated low-order bit positions of the register are set to 0's. If the shift count is 0, no shifting occurs.

Although the register to be shifted contains only 16 bits, shift count values of 0–31 may be specified. Shift counts greater than 17 lengthen the execution time and provide an effective shift of 17.

Indicators

Carry. The carry indicator is set to reflect the last bit shifted out of bit 0. If the count is 0, the carry indicator is reset.

Overflow. The overflow indicator is reset, and then set to a 1 if the most-significant bit in the register (bit 0) has changed during the operation.

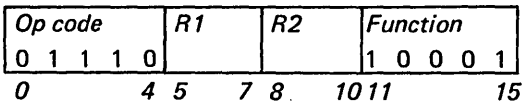
Even, Negative, and Zero. These indicators are changed to reflect the final contents of the register.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Count in Register Format

SLL reg,reg



Note: In the assembler syntax, operand 1 is the register that contains the shift count. Operand 2 is the register that is shifted.

The bits in the register specified by the R1 field are shifted left by the number of bits specified by the shift count. This count is obtained from bits 8–15 of the register specified by the R2 field. The vacated low-order bits of the register specified by the R1 field are set to 0's. If the shift count is 0, no shifting occurs.

The contents of the register specified by the R2 field are not changed unless the R1 and R2-fields specify the same register. In this case, the register contents are shifted as specified.

Although the register shifted contains only 16 bits, shift count values of 0–255 may be specified. Shift counts greater than 17 lengthen the execution time and provide an effective shift of 17.

Indicators

Carry. The carry indicator is set to reflect the last bit shifted out of bit 0. If the count is 0, the carry indicator is reset.

Overflow. The overflow indicator is reset, and then set to a 1 if the most-significant bit in the register (bit 0) has changed during the operation.

Even, Negative, and Zero. These indicators are changed to reflect the final contents of the register specified by the R1 field.

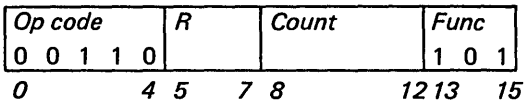
Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Shift Left Logical Double (SLLD)

Immediate Count Format

SLLD cnt31,reg



The bits in the register pair specified by the R-field and the R+1 field are shifted left by the number of bit positions specified in the count field. The vacated low-order bits of the register pair are set to 0's.

Within the register pair, the register specified by the R-field contains the high-order word (bits 0–15); the register specified by the R+1 field contains the low-order word (bits 16–31). If the shift count is 0, no shifting occurs.

If the R-field value is 7, registers 7 and 0 are used for the register pair.

Indicators

Carry. The carry indicator is set to reflect the last bit shifted out of bit 0.

Overflow. The overflow indicator is cleared, and then set to a 1 if the most-significant bit in the register pair (bit 0) has changed during the operation.

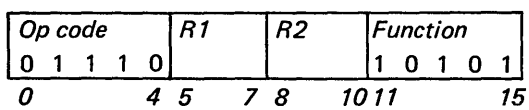
Even, Negative, and Zero. These indicators are changed to reflect the final contents of the two registers.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Count in Register Format

SLLD reg,reg



Note: In the assembler syntax, operand 1 is the register that contains the shift count. Operand 2 is the register that is shifted.

The bits in the register pair specified by the R1 field and the R1+1 field are shifted left by the number of bit positions specified by the shift count. This count is obtained from bits 8–15 of the register specified by the R2 field. The vacated low-order bit positions of the register pair are set to 0's.

Within the register pair, the register specified by the R1 field contains the high-order word (bits 0–15); the register specified by the R1+1 field contains the low-order word (bits 16–31). If the shift count is 0, no shifting occurs.

If the R1 field value is 7, registers 7 and 0 are used for the register pair.

The contents of the register specified by the R2 field are note changed unless the R1 (or R1+1) and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the registers to be shifted represent 32 bits, shift count values of 0–255 may be specified. Shift counts greater than 33 lengthen the execution time and provide an effective shift of 33.

Indicators

Carry. The carry indicator is set to reflect the last bit shifted out of bit 0.

Overflow. The overflow indicator is reset, and then set to a 1 if the most-significant bit in the register pair (bit 0) has changed during the operation.

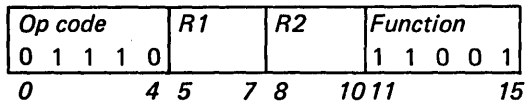
Even, Negative, and Zero. These indicators are changed to reflect the final contents of the two registers.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Shift Left and Test (SLT)

SLT reg,reg



Note: In the assembler syntax, operand 1 is the register that contains the shift count. Operand 2 is the register that is shifted.

The bits in the register specified by the R1 field are shifted left. The vacated low-order bit positions of the register are set to 0's.

Shifting continues until either one of the following occurs:

1. The number of bits specified by the shift count have been shifted. This count is obtained from bits 8–15 of the register specified by the R2 field. If the shift count is 0, no shifting occurs.
2. A 1-bit is shifted from the high-order bit (bit 0) to the carry indicator. In this case, the remaining shift count is loaded into bits 8–15 of the register specified by the R2 field.

Bits 0–7 of the register specified by the R2 field are not changed; these bits must be set to 0's to avoid future code obsolescence.

If the R1 and R2 fields specify the same register, the bits in the register are shifted as specified and, when shifting is complete, the remaining shift count replaces the shifted result.

Although the register to be shifted contains only 16 bits, shift count values of 0–255 may be specified.

Indicators

Carry. The carry indicator is set to reflect the last bit shifted out of bit 0 of the register specified by the R1 field. If the count is 0, the carry indicator is reset.

Overflow. The overflow indicator is reset, and then set to a 1 if the most-significant bit (bit 0) in the register specified by the R1 field has changed during the operation.

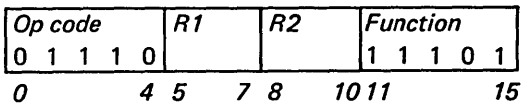
Even, Negative, and Zero. These indicators are changed to reflect the final contents of bits 8–15 of the register specified by the R2 field.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Shift Left and Test Double (SLTD)

SLTD reg,reg



Note: In the assembler syntax, operand 1 is the register that contains the shift count. Operand 2 is the register that is shifted.

The bits in the register pair specified by the R1 field and the R1+1 field are shifted left. The vacated low-order bit positions of the register pair are set to 0's.

Shifting continues until either one of the following occurs:

1. The number of bits specified by the shift count have been shifted. This count is obtained from bits 8–15 of the register specified by the R2 field. If the shift count is 0. no shifting occurs.
2. A 1-bit is shifted from the high-order bit to the carry indicator. In this case, the remaining shift count is loaded into bits 8–15 of the register specified by the R2 field.

Bits 0–7 of the register specified by the R2 field are not changed; these bits must be set to 0's to avoid future code obsolescence.

Within the register pair, the register specified by the R1 field contains the high-order word (bits 0–15); the register specified by the R1+1 field contains the low-order word (bits 16–31).

If the R1 field value is 7, registers 7 and 0 are used for the register pair.

If the R1 (or R1+1) and R2 fields specify the same register, the bits in the register are shifted as specified and, when shifting is complete, the remaining shift count replaces the shifted result.

Although the registers to be shifted contain only 32 bits, shift count values of 0–255 may be specified.

Indicators

Carry. The carry indicator is set to reflect the last bit shifted out of bit 0 of the register specified by the R1 field. If the count is 0, the carry indicator is reset.

Overflow. The overflow indicator is reset, and then set to a 1 if the most-significant bit (bit 0) in the register specified by the R1 field has changed during the operation.

Even, Negative, and Zero. These indicators are changed to reflect the final contents of bits 8–15 of the register specified by the R2 field.

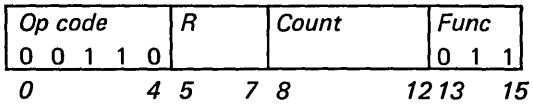
Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Shift Right Arithmetic (SRA)

Immediate Count Format

SRA cnt16,reg



The bits in the register specified by the R-field are shifted right by the number of bit positions specified in the count field. The value of the sign (the high-order bit) is entered into the vacated high-order bit positions of the register specified by the R-field. If the shift count is 0, no shifting occurs.

Although the register to be shifted contains only 16 bits, shift count values of 0–31 may be specified. Shift counts greater than 16 lengthen the execution time and provide an effective shift of 16.

Indicators

Carry and Overflow. These indicators are not changed.

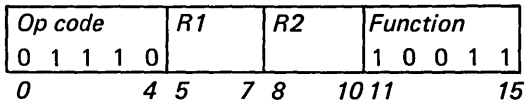
Even, Negative, and Zero. These indicators are changed to reflect the final contents of the register.

Program-Check Condition

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Count in Register Format

SRA reg,reg



Note: In the assembler syntax, operand 1 is the register that contains the shift count. Operand 2 is the register that is shifted.

The bits in the register specified by the R1 field are shifted right by the number of bit positions specified by the shift count. This count is obtained from bits 8–15 of the register specified by the R2 field. The value of the sign (the high-order bit) is entered into the vacated high-order bit positions of the register specified by the R1 field. If the shift count is 0, no shifting occurs.

The contents of the register specified by the R2 field are not changed unless the R1 and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the register to be shifted is 16 bits, shift count values of 0–255 may be specified. Shift counts greater than 16 lengthen the execution time and provide an effective shift of 16.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the final contents of the register specified by the R1 field.

Program-Check Conditions

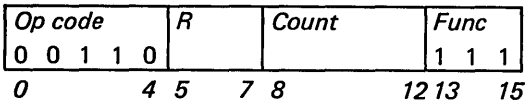
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

SRAD

Shift Right Arithmetic Double (SRAD)

Immediate Count Format

SRAD cnt31,reg



The bits in the register pair specified by the R-field and the R+1 field are shifted right by the number of bit positions specified in the count field. The value of the sign (the high-order bit) is entered into the vacated high-order bit positions of the register pair.

Within the register pair, the register specified by the R-field contains the high-order word (bits 0–15); the register specified by the R+1 field contains the low-order word (bits 16–31). If the shift count is 0, no shifting occurs.

If the R-field value is 7, registers 7 and 0 are used for the register pair.

Indicators

Carry and Overflow. These indicators are not changed.

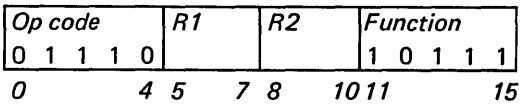
Even, Negative, and Zero. These indicators are changed to reflect the final contents of the register pair.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Count in Register Format

SRAD reg,reg



Note: In the assembler syntax, operand 1 is the register that contains the shift count. Operand 2 is the register that is shifted.

The bits in the register pair specified by the R1 field and the R1+1 field are shifted right by the number of bit positions specified by the shift count. This count is obtained from bits 8–15 of the register specified by the R2 field. The value of the sign (the high-order bit) is entered into the vacated high-order bit positions of the register pair.

Within the register pair, the register specified by the R1 field contains the high-order word (bits 0–15); the register specified by the R1+1 field contains the low-order word (bits 16–31). If the shift count is 0, no shifting occurs.

If the R-field value is 7, registers 7 and 0 are used for the register pair.

The contents of the register specified by the R2 field are not changed unless the R1 (or R1+1) and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the registers to be shifted represent 32 bits, shift count values of 0–255 may be specified. Shift counts greater than 32 lengthen the execution time and provide an effective shift of 32.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the final contents of the register pair.

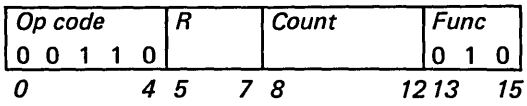
Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Shift Right Logical (SRL)

Immediate Count Format

SRL cnt16,reg



The bits in the register specified by the R-field are shifted right by the number of bit positions specified in the count field. The vacated high-order bit positions of the register are set to 0's. If the shift count is 0, no shifting occurs.

Although the register to be shifted contains only 16 bits, shift count values of 0–31 may be specified. Shift counts greater than 16 lengthen the execution time and provide an effective shift of 16.

Indicators

Carry and Overflow. These indicators are not changed.

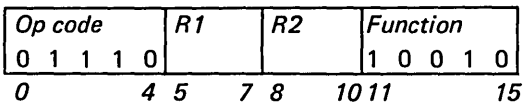
Even, Negative, and Zero. These indicators are changed to reflect the final contents of the register.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Count in Register Format

SRL reg,reg



Note: In the assembler syntax, operand 1 is the register that contains the shift count. Operand 2 is the register that is shifted.

The bits in the register specified by the R1 field are shifted right by the number of bit positions specified by the shift count. This count is obtained from bits 8–15 of the register specified by the R2 field. The vacated high-order bit positions of the register specified by the R1 field are set to 0's. If the shift count is 0, no shifting occurs.

The contents of the register specified by the R2 field are not changed unless the R1 and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the register to be shifted contains only 16 bits, shift count values of 0–255 may be specified. Shift counts greater than 16 lengthen the execution time and provide an effective shift of 16.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the final contents of the register specified by the R1 field.

Program-Check Conditions

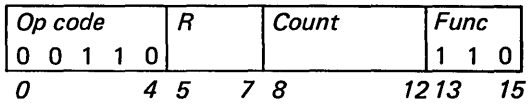
Protect Check. In the problem state, an instruction is fecthed or data is accessed from a storage area not assigned to the current operation.

SRLD

Shift Right Logical Double (SRLD)

Immediate Count Format

SRLD cnt31,reg



The bits in the register pair specified by the R-field and the R+1 field are shifted right by the number of bit positions specified in the count field. The vacated high-order bits of the register pair are set to 0's.

Within the register pair, the register specified by the R-field contains the high-order word (bits 0–15); the register specified by the R+1 field contains the low-order word (bit 16–31). If the shift count is 0, no shifting occurs. If the R-field value is 7, registers 7 and 0 are used for the register pair.

Indicators

Carry and Overflow. These indicators are not changed.

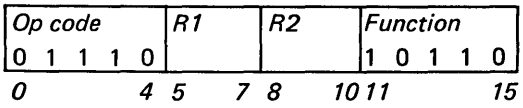
Even, Negative, and Zero. These indicators are changed to reflect the final contents of the register pair.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Count in Register Format

SRLD reg,reg



Note: In the assembler syntax, operand 1 is the register that contains the shift count. Operand 2 is the register that is shifted.

The bits in the register pair specified by the R1 field and the R1+1 field are shifted right by the number of bit positions specified by the shift count. This count is obtained from bits 8–15 of the register specified by the R2 field. The vacated high-order bits of the register pair are set to 0's.

Within the register pair, the register specified by the R1 field contains the high-order word (bits 0–15); the register specified by the R1+1 field contains the low-order word (bits 16–31). If the shift count is 0, no shifting occurs. If the R1 field value is 7, registers 7 and 0 are used for the register pair.

The contents of the register specified by the R2 field are not changed unless the R1 (or R1+1) and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the registers to be shifted represent 32 bits, shift count values of 0–255 may be specified. Shift counts greater than 32 lengthen the execution time and provide an effective shift of 32.

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the final contents of the register pair.

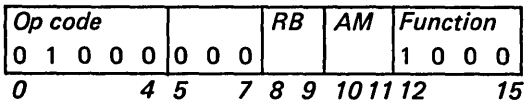
Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

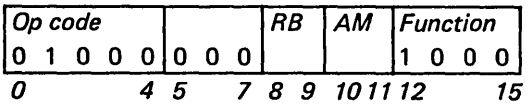
Store Multiple (STM)

STM reg,addr4[,abcnt]

Format without appended word for effective addressing (AM = 00 or 01)



Format with appended word for effective addressing (AM = 10 or 11)



The STM instruction stores the contents of a specified number of registers for the current level into a stack. This stack is defined by the stack control block pointed to by the effective address.

The RL field specifies the last register to be stored. Register 7 is stored first; then register 0 through the register specified by the RL field. For example, if the RL field specifies register 2, STM stores registers 7, 0, 1, and 2. If the RL field specifies register 7, only register 7 is stored.

The N-field specifies the number of words to be allocated in the stack as a dynamic work area. A value of 0 is valid.

The new top element address of the stack (incremented by 2) is loaded into the last register stored; that is, the register specified by the RL field. This address points to the low storage end of the dynamic work area (or the last register stored if N=0).

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Programming Note: If the AM field equals 01, the contents of the register specified by the RB field are incremented by 2.

Refer to "Stack Operations" in Chapter 2 for additional information about the operation of this instruction. The STM instruction is used in conjunction with the Load Multiple and Branch (LMB) instruction described previously in this chapter.

Indicators

The indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The indirect address, stack control block, or stack element results in an even-byte boundary violation. The instruction is terminated.

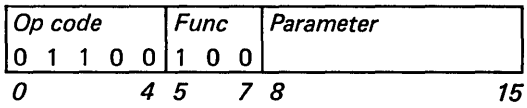
Soft-Exception Trap Conditions

Stack Exception. The stack is full or cannot contain the number of words to be stored.

STOP—SVC

Stop (STOP)

STOP [ubyte]



This instruction is executed only when the programmer console is installed and the Mode switch is in the Diagnostic position. Otherwise, this instruction performs no operation (no-op). The processor enters the stop state following execution of this instruction.

The parameter field can be optionally coded with a one-byte unsigned absolute value or expression. If not coded, the parameter field defaults to 0. The processor ignores the value, but is used as an identifier.

Indicators

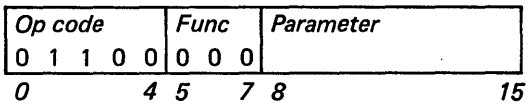
The indicators are not changed.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Supervisor Call (SVC)

SVC ubyte



The instruction address register (IAR) is incremented by 2; the current level status block (LSB) is stored, using an address key of 0, starting at the main storage location specified by the contents of the SVC LSB pointer that resides in main storage location 0010 (hexadecimal). The instruction also causes the following events:

- The summary mask (LSR bit 11) is disabled.
- Supervisor state (LSR bit 8) is set to 1.
- Trace (LSR bit 10) is set to 0.
- Equate operand spaces (AKR bit 0) is set to 0.
- Operand 2 key contents are loaded into the operand 1 key.
- The operand 2 key and the instruction space key are then set to 0's.

The parameter field (bits 8–15) is under control of the programming system. This field is loaded into the low-order byte of register 1. The high-order byte of register 1 is set to 0.

Subsequently, the contents of main storage location 0012 hexadecimal (SVC start instruction address) are loaded into the instruction address register, and become the address of the next instruction to be fetched.

Execution of this instruction causes a class interrupt. See "Class Interrupts" in Chapter 3 for additional information.

Indicators

The indicators are not changed.

Program-Check Conditions

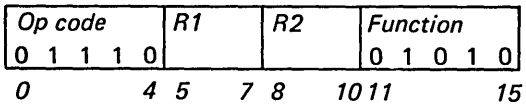
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Specification Check. The LSB pointer or SIA location in the reserved main storage location specifies an odd storage address. The instruction is terminated.

Subtract Word (SW)

Register/Register Format

SW reg,reg



The contents of the register specified by the R1 field are subtracted from the contents of the register specified by the R2 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed unless the R1 and R2 fields specify the same register.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the register, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

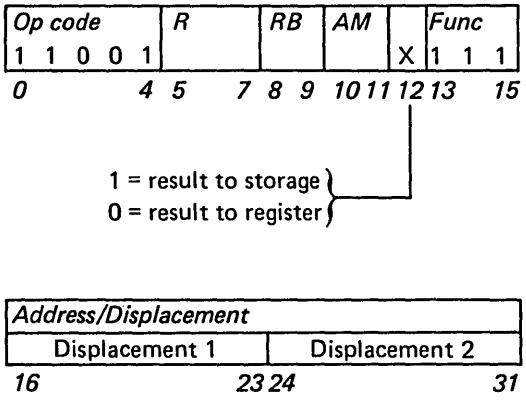
Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Register/Storage Format

SW reg,addr4
addr4,reg



A subtract operation is performed between the register specified by the R-field and the location specified by the effective address in main storage. The source operand is not changed.

Bit 12 of the instruction specifies the destination of the result.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

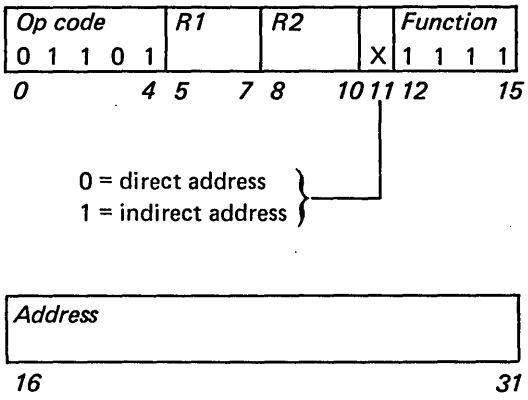
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Storage/Register Long Format

SW longaddr,reg



The contents of the main storage word location specified by an effective address are subtracted from the contents of the register specified by the R1 field. The result is placed in the register specified by the R1 field.

The effective main storage address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals 0, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0 (direct address). The result from step 1 is the effective address.

   Bit 11=1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

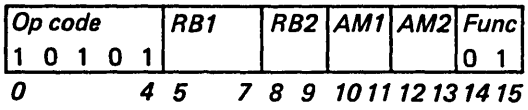
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

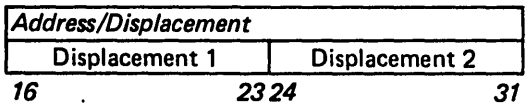
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

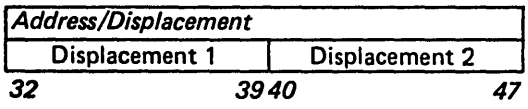
Specification Check. The effective address or indirect address results in an even-byte boundary violation.

Storage/Storage Format

SW addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. Word operand 1 is subtracted from word operand 2. The result replaces operand 2. Operand 1 is not changed.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

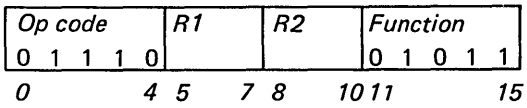
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

SWCY

Subtract Word With Carry (SWCY)

SWCY reg,reg



If the carry indicator is on at entry (denoting a borrow), a positive 1 is subtracted from the contents of the register specified by the R2 field. The contents of the R1 field are then subtracted from the intermediate result. If the carry indicator is off at entry, the contents of the R1 field are subtracted from the contents of the register specified by the R2 field. The contents of the register specified by the R1 field are not changed unless the R1 and R2 fields specify the same register. The final result replaces the contents of the register specified by the R2 field.

Programming Note: This instruction can be used when subtracting multiple word operands. See "Indicators—Multiple Word Operands" in Chapter 2.

Indicators

Carry. If a borrow is detected out of the high-order position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even. The even indicator is not changed.

Negative. The negative indicator is changed to reflect the result.

Zero. If on at entry, the zero indicator is changed to reflect the result. If off at entry, it remains off.

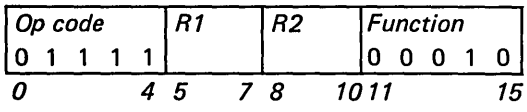
Program-Check Conditions

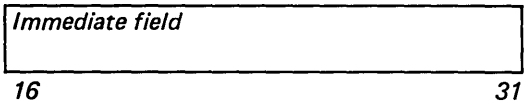
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Subtract Word Immediate (SWI)

Register Immediate Long Format

SWI word,reg[,reg]





The immediate field is subtracted from the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed unless the R1 and R2 fields specify the same register.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

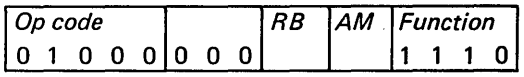
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

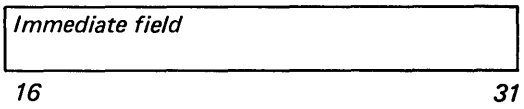
SWI

Storage Immediate Format

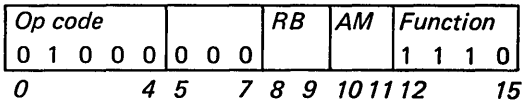
SWI word,addr4

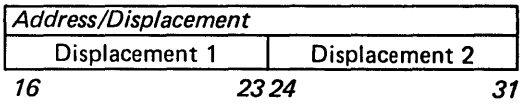
Format without appended word for effective addressing (AM = 00 or 01)

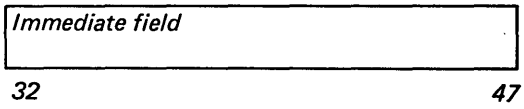




Format with appended word for effective addressing (AM = 10 or 11)







The immediate field is subtracted from the contents of the main storage location specified by the effective address. The result replaces the contents of the storage location specified by the effective address. The immediate operand is not changed.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Carry. If a borrow is detected out of the high-order bit position of the word, the carry indicator is set to 1. If no borrow is detected, the carry indicator is reset.

Overflow. The overflow indicator is cleared, and then set to 1 if the difference cannot be represented in one word; that is, if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

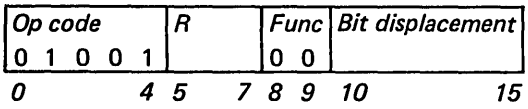
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Test Bit (TBT)

TBT (reg,bitdisp)



The bit displacement is added to the byte address contained in the register specified by the R-field to form an effective bit address. The bit displacement field is an unsigned six-bit binary integer.

The bit at the effective bit address is tested. If the bit is 0, the zero indicator is set to 1. If the bit is 1, the negative indicator is set to 1.

Indicators

Zero and Negative. These indicators are reset to 0's, and then set to reflect the result of the preceding test.

Even, Carry, and Overflow. These indicators are not changed.

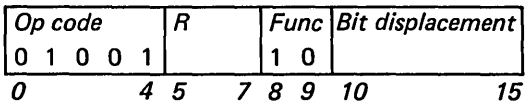
Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, an instruction is fecthed or data is accessed from a storage area not assigned to the current operation.

Test Bit and Reset (TBTR)

TBTR (reg,bitdisp)



The bit displacement is added to the byte address contained in the register specified by the R-field to form an effective bit address. The bit displacement field is an unsigned six-bit binary integer.

The bit at the effective bit address is tested. If the bit is 0, the zero indicator is set to 1. If the bit is 1, the negative indicator is set to 1. Following this test, the addressed bit is unconditionally set to 0.

Indicators

Zero and Negative. These indicators are reset to 0's, and then set to reflect the result of the preceding test.

Even, Carry, and Overflow. These indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

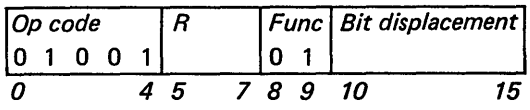
Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is suppressed.
- Attempts to change an operand in a storage area assigned as read-only. The instruction is terminated.

TBTS—TBTV

Test Bit and Set (TBTS)

TBTS (reg,bitdisp)



The bit displacement is added to the byte address contained in the register specified by the R-field to form an effective bit address. The bit displacement field is an unsigned six-bit binary integer.

The bit at the effective bit address is tested. If the bit is 0, the zero indicator is set to 1. If the bit is 1, the negative indicator is set to 1. Following this test, the addressed bit is unconditionally set to 1.

Indicators

Zero and Negative. These indicators are reset to 0's, and then set to reflect the result of the preceding test.

Even, Carry, and Overflow. These indicators are not changed.

Program-Check Conditions

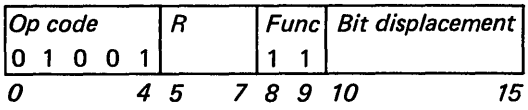
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is suppressed.
- Attempts to change an operand in a storage area assigned as read-only. The instruction is terminated.

Test Bit and Invert (TBTV)

TBTV (reg,bitdisp)



The bit displacement is added to the byte address contained in the register specified by the R-field to form an effective bit address. The bit displacement field is an unsigned six-bit binary integer.

The bit at the effective bit address is tested. If the bit is 0, the zero indicator is set to 1. If the bit is 1, the negative indicator is set to 1. Following this test, the addressed bit is unconditionally inverted.

Indicators

Zero and Negative. These indicators are reset to 0's, and then set to reflect the result of the preceding test.

Even, Carry, and Overflow. These indicators are not changed.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

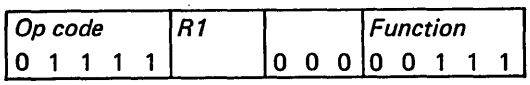
Protect Check. In the problem state, the instruction:

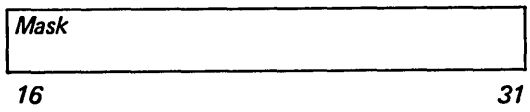
- Is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is suppressed.
- Attempts to change an operand in a storage area assigned as read-only. The instruction is terminated.

Test Word Immediate (TWI)

Register Immediate Long Format

TWI word,reg





The contents of the register specified by the R1 field are tested against the mask contained in the immediate word of the instruction. The contents of the register specified by the R1 field are not changed.

Mask field bits set to 1 select the bits to be tested in the register.

Example:

Mask	0000	0000	0111	1100
Register	0000	0000	0011	0101
Selected bits			011	01

Result: Zero and negative indicators remain 0's, (selected bits combination of 1's and 0's remain 0's).

The selected bits are tested. If all the mask bits or selected bits are 0's, the zero indicator is set to 1. If the selected bits are 1's, the negative indicator is set to 1. If the selected bits are a combination of 1's and 0's, both indicators remain 0's.

Bits 8–10 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Zero and Negative. These indicators are reset to 0's, and then set to reflect the result of the mask test.

Even, Carry, and Overflow. These indicators are not changed.

Program-Check Conditions

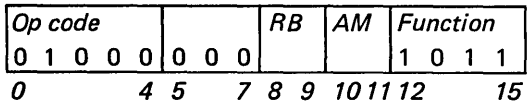
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

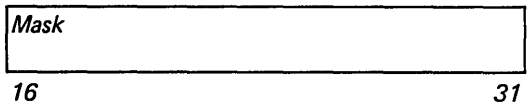
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Storage Immediate Format

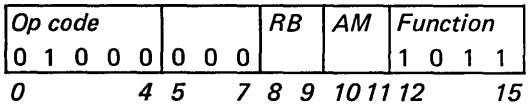
TWI word,addr4

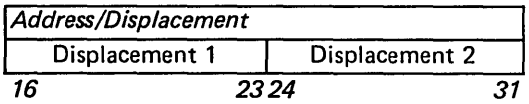
Format without appended word for effective addressing (AM = 00 or 01)

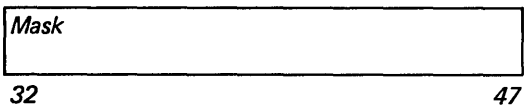




Format with appended word for effective addressing (AM = 10 or 11)







TWI—VR

The contents of the storage location specified by the effective address are tested against the mask field of the instruction. Neither operand is changed.

Mask field bits set to 1 select the bits to be tested in the effective address storage location.

Example:

Mask	0000	0000	0000	1110
Storage contents	0000	0000	0101	1110
Selected bits				111

Result: Negative indicator set to 1 (selected bits all 1's).

The selected bits are tested. If all the mask bits or selected bits are 0's, the zero indicator is set to 1. If the selected bits are 1's, the negative indicator is set to 1. If the selected bits are a combination of 1's and 0's, both indicators remain 0's.

Bits 5–7 of the instruction are not used and must be set to 0's to avoid future code obsolescence.

Indicators

Zero and Negative. These indicators are reset to 0's, and then set to reflect the result of the mask test.

Even, Carry, and Overflow. These indicators are not changed.

Program-Check Conditions

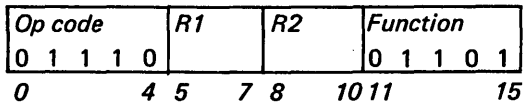
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation. The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Invert Register (VR)

VR reg[,reg]



The contents of the register specified by the R1 field are 1's complemented. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed.

Indicators

Carry and Overflow. These indicators are not changed.

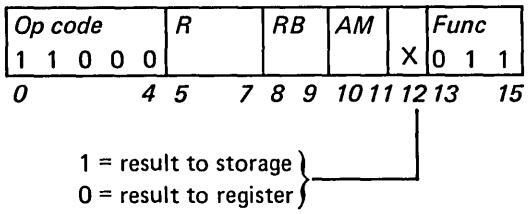
Even, Negative, and Zero. These indicators are changed to reflect the result.

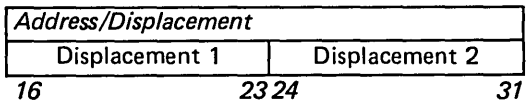
Program-Check Conditions

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Exclusive OR Byte (XB)

XB reg,addr4
addr4,reg





A logical Exclusive OR operation is performed between the least-significant byte of the register specified by the R-field and the main storage location specified by the effective address. The source operand is not changed. When going from storage to register, bits 0–7 of the register are not changed.

Bit 12 of the instruction specifies the destination of the result.

Example of Exclusive OR Byte:

Register contents	0000	1010	1100	0011
Storage operand			0110	0101
Result			1010	0110

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result of the Exclusive OR operation.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

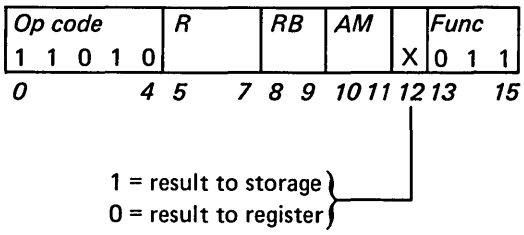
The instruction is terminated.

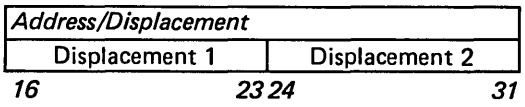
Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

XD

Exclusive OR Doubleword (XD)

XD reg,addr4
addr4,reg





A logical Exclusive OR operation is performed between the contents of the register pair specified by the R-field and the R+1 field and the doubleword in main storage specified by the effective address. The source operand is not changed.

If the R-field value is 7, registers 7 and 0 are used as the register pair.

Bit 12 of the instruction specifies the destination of the result.

Example of Exclusive OR Doubleword:

Register pair contents	0000 0000 1010 1100 0000 0000 1110 1111
Storage operand	0000 0000 1101 0011 0000 0000 1101 0000
Result	0000 0000 0111 1111 0000 0000 0011 1111

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

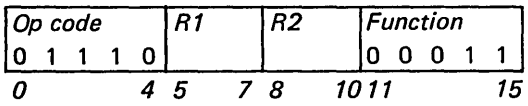
The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Exclusive OR Word (XW)

Register/Register Format

XW reg,reg



The contents of the register specified by the R1 field are Exclusive ORed bit-by-bit with the contents of the register specified by the R2 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed unless the R1 and R2 fields specify the same register.

Example of Exclusive OR Word:

Register contents (R1)	1111 0000 1010 0000
Register contents (R2)	0011 1111 0111 1111
Result	1100 1111 1101 1111

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

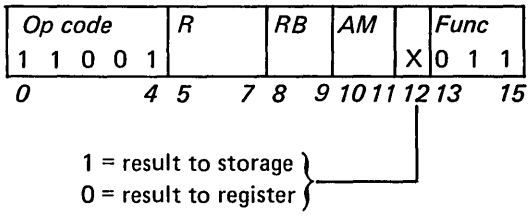
Program-Check Conditions

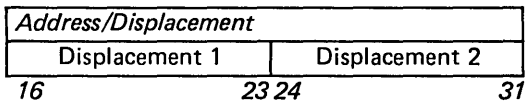
Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

XW

Register/Storage Format

XW reg,addr4
addr4,reg





A logical Exclusive OR operation is performed between the contents of the register specified by the R-field and the main storage location specified by the effective address. The source operand is not changed.

Bit 12 of the instruction specifies the destination of the result.

Example of Exclusive OR Word:

Register contents (R)	1111 0000 1010 0000
Storage operand	0011 1111 0111 1111
Result	1100 1111 1101 1111

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system. The instruction is terminated.

Protect Check. In the problem state, the instruction:

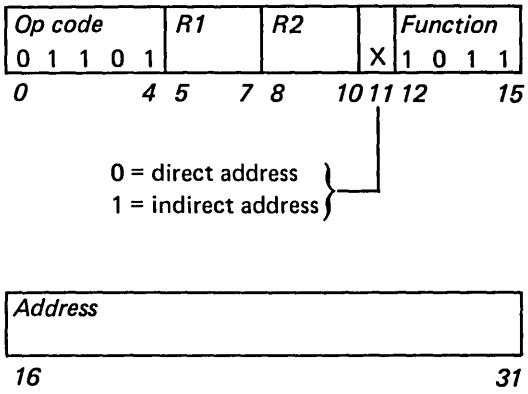
- Is fetched or data is accessed from a storage area not assigned to the current operation.
- Attempts to change an operand in a storage area assigned as read-only.

The instruction is terminated.

Specification Check. The effective address or indirect address results in an even-byte boundary violation. The instruction is terminated.

Storage/Register Long Format

XW longaddr,reg



A logical Exclusive OR operation is performed between the contents of the register specified by the R1 field and the contents of the main storage word location specified by the effective address. The result is placed in the register specified by the R1 field.

The effective main storage address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals 0, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0 (direct address). The result from step 1 is the effective address.

   Bit 11=1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Example of Exclusive OR Word:

Register contents (R1)	1111 0000 1010 0000
Storage operand	0011 1111 0111 1111
Result	1100 1111 1101 1111

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result loaded into the register specified by the R1 field.

Program-Check Conditions

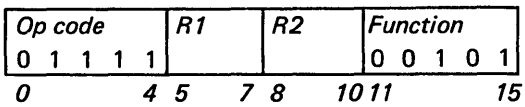
Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

Specification Check. The effective address or indirect address results in an even-byte boundary violation.

Exclusive OR Word Immediate (XWI)

XWI word,reg[,reg]



The immediate field is Exclusive ORed bit-by-bit with the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are not changed unless the R1 and R2 fields specify the same register.

Example of Exclusive OR Word:

Register contents (R1)	1111 0000 1010 0000
Immediate operand	0011 1111 0111 1111
Result	1100 1111 1101 1111

Indicators

Carry and Overflow. These indicators are not changed.

Even, Negative, and Zero. These indicators are changed to reflect the result.

Program-Check Conditions

Invalid Storage Address. One or more words of the instruction or the effective address are outside the installed storage size of the system.

Protect Check. In the problem state, an instruction is fetched or data is accessed from a storage area not assigned to the current operation.

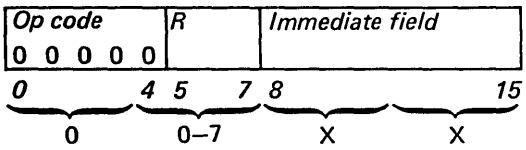
Appendix A. Instruction Formats

The following instruction formats are shown in ascending sequence based on operation code. Bits 0–4 of the first instruction word comprise the operation code field. Bit combinations and their hexadecimal representations are shown for each operation code.

Some instructions contain a function field that modifies the operation code to form individual instructions within a group. Each chart shows the function field bit combinations in hexadecimal and in ascending sequence. The assembler mnemonic, assembler syntax, and instruction name are listed for the individual instructions. The asterisk (*) shown with the assembler syntax indicates indirect addressing.

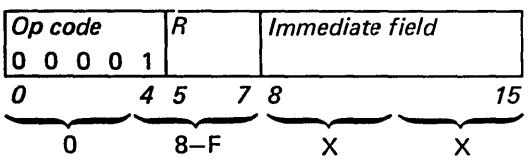
Refer to "Effective-Address Generation" in Chapter 2 for a description of the Address Mode (AM) appended words.

0xxx



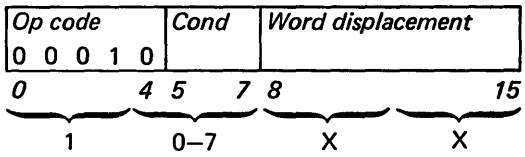
0	0–7	X	X	ABI	byte,reg	Add Byte Immediate

0xxx



0	8–F	X	X	MVBI	byte,reg	Move Byte Immediate

1xxx

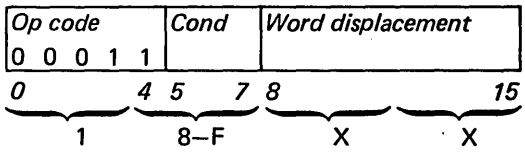


1	0–7	X	X

JC	cond,jdisp	Jump on Condition
JC	cond,jaddr	Jump on Condition

Extended mnemonics:
JCY, JE, JEV, JLE, JLLE, JLLT, JLT, JMIX
JN, JOFF, JON, JP, JZ

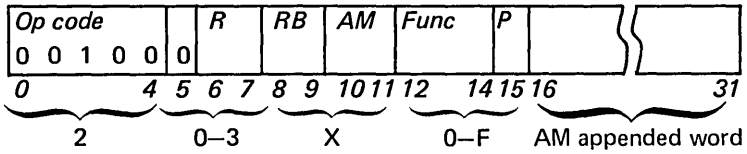
1xxx



1	8–F	X	X

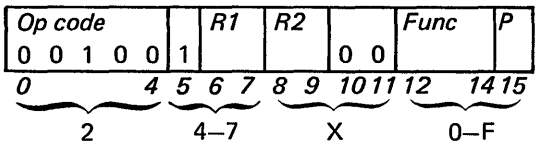
JNC	cond,jdisp	Jump on Not Condition
JNC	cond,jaddr	Jump on Not Condition

Extended mnemonics:
JGE, JGT, JLGE, JLGT, JNCY, JNE, JNEV,
JNMIX, JNN, JNOFF, JNON, JNP, JNZ



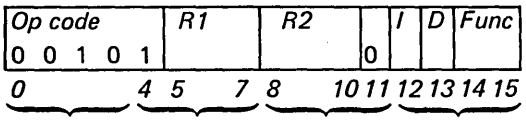
2	0–3	X				
			0	FA	addr4,freg	Floating Add
			1	FAD	addr4,freg	Floating Add Double
			2	FS	addr4,freg	Floating Subtract
			3	FSD	addr4,freg	Floating Subtract Double
			4	FM	addr4,freg	Floating Multiply
			5	FMD	addr4,freg	Floating Multiply Double
			6	FD	addr4,freg	Floating Divide
			7	FDD	addr4,freg	Floating Divide Double
			8	FMVC	addr4,freg	Floating Move and Convert
			9	FMVCD	addr4,freg	Floating Move and Convert Double
			A	FMV	addr4,freg	Floating Move
			B	FMVD	addr4,freg	Floating Move Double
			C	FMVC	freg,addr4	Floating Move and Convert
			D	FMVCD	freg,addr4	Floating Move and Convert Double
			E	FMV	freg,addr4	Floating Move
			F	FMVD	freg,addr4	Floating Move Double

2XXX



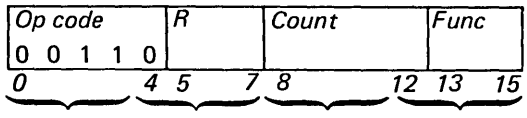
2	4–7	X				
			0	FA	freg,freg	Floating Add
			1	FAD	freg,freg	Floating Add Double
			2	FS	freg,freg	Floating Subtract
			3	FSD	freg,freg	Floating Subtract Double
			4	FM	freg,freg	Floating Multiply
			5	FMD	freg,freg	Floating Multiply Double
			6	FD	freg,freg	Floating Divide
			7	FDD	freg,freg	Floating Divide Double
			8	FMV	freg,freg	Floating Move
			9	FMVD	freg,freg	Floating Move Double
			A	FC	freg,freg	Floating Compare
			B	FCD	freg,freg	Floating Compare Double
			C	(invalid)		Executes FMV
			D	(invalid)		Executes FMVD
			E	(invalid)		Indicators are reset
			F	(invalid)		Indicators are reset

2xxx



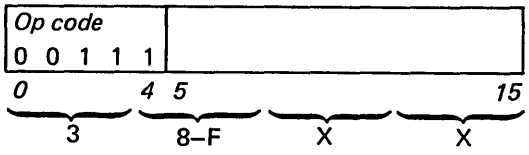
2	8–E				
		0	MVFD	(reg),(reg)	Move Byte Field and Decrement
		1	(invalid)		
		2	CFNED	(reg),(reg)	Compare Byte Field Not Equal and Decrement
		3	CFED	(reg),(reg)	Compare Byte Field Equal and Decrement
		4	MVFN	(reg),(reg)	Move Byte Field and Increment
		5	(invalid)		
		6	CFNEN	(reg),(reg)	Compare Byte Field Not Equal and Increment
		7	CFEN	(reg),(reg)	Compare Byte Field Equal and Increment
		8	FFD	reg,(reg)	Fill Byte Field and Decrement
		9	(invalid)		
		A	SFNED	reg,(reg)	Scan Byte Field Not Equal and Decrement
		B	SFED	reg,(reg)	Scan Byte Field Equal and Decrement
		C	FFN	reg,(reg)	Fill Byte Field and Increment
		D	(invalid)		
		E	SFNEN	reg,(reg)	Scan Byte Field Not Equal and Increment
		F	SFEN	reg,(reg)	Scan Byte Field Equal and Increment

3xxx



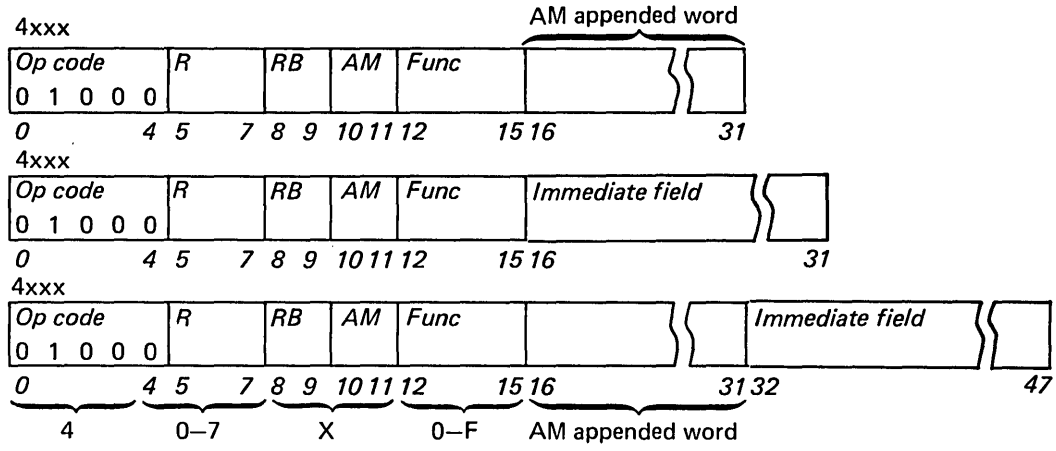
3	0–7	X			
		0,8	SLC	cnt16,reg	Shift Left Circular
		1,9	SLL	cnt16,reg	Shift Left Logical
		2,A	SRL	cnt16,reg	Shift Right Logical
		3,B	SRA	cnt16,reg	Shift Right Arithmetic
		4,C	SLCD	cnt31,reg	Shift Left Circular Double
		5,D	SLLD	cnt31,reg	Shift Left Logical Double
		6,E	SRLD	cnt31,reg	Shift Right Logical Double
		7,F	SRAD	cnt31,reg	Shift Right Arithmetic Double

3xxx



3	8–F	X	X	Invalid operation code (program-check condition)

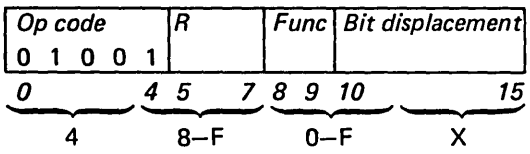
4xxx



4	0–7	X			
			0	MVA	raddr,addr4	Move Address
			0	MVWI	word,addr4	Move Word Immediate
			1	(invalid)		
			2	(invalid)		
			3	(invalid)		
			4	MVA	addr4,reg	Move Address (see Note)
			4	MVWI	word,reg	Move Word Immediate (see Note)
			5	(invalid)		
			6	(invalid)		
			7	(invalid)		
			8	STM	reg,addr4[,abcnt]	Store Multiple
			9	AWI	word,addr4	Add Word Immediate
			9	AA	raddr,addr4	Add Address
			A	LMB	addr4	Load Multiple and Branch (see Note)
			B	TWI	word,addr4	Test Word Immediate
			C	OWI	word,addr4	OR Word Immediate
			C	SBTWI	word,addr4	Set Bits Word Immediate
			D	RBTWI	word,addr4	Reset Bits Word Immediate
			E	SWI	word,addr4	Subtract Word Immediate
			E	SA	raddr,addr4	Subtract Address
			F	CWI	word,addr4	Compare Word Immediate
			F	CA	raddr,addr4	Compare Address

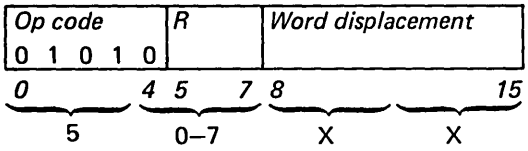
Note: Use format without immediate field.

4xxx



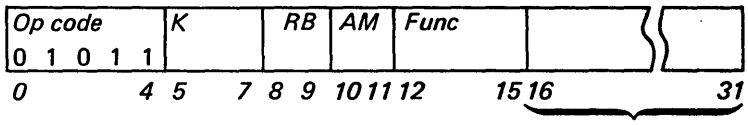
4	8–F	0–3	X	TBT	(reg,bitdisp)	Test Bit
		4–7		TBTS	(reg,bitdisp)	Test Bit and Set
		8–B		TBTR	(reg,bitdisp)	Test Bit and Reset
		C–F		TBTV	(reg,bitdisp)	Test Bit and Invert

5xxx

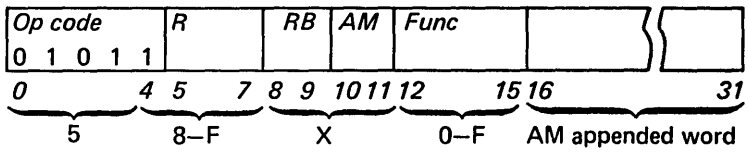


5	0	0	0	NOP	No Operation	
	0	X	X	J	jdisp	Jump Unconditional
				J	jaddr	Jump Unconditional
	1–7	X	X	BXS	(reg^{1-7},jdisp)	Branch Indexed Short
				BXS	(reg^{1-7})	Branch Indexed Short
				BXS	addr	Branch Indexed Short

5xxx



5xxx

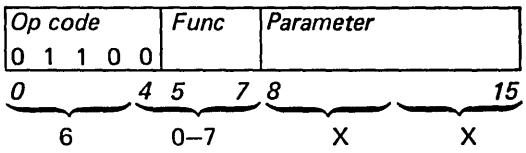


5	8–F	X				
			0	SEIMR	addr4	Set Interrupt Mask Register
			1	SESR	reg,addr4	Set Segmentation Register
			2	SEAKR	addr4	Set Address Key Register (Note 1)
			3	SEFLB	reg,addr4	Set Floating Level Block
			4	SESK	reg,addr4	Set Storage Key
			5	(invalid)		
			6	SELB	reg,addr4	Set Level Status Block
			7	(invalid)		
			8	CPIMR	addr4	Copy Interrupt Mask Register
			9	CPSR	reg,addr4	Copy Segmentation Register
			A	CPAKR	addr4	Copy Address Key Register (Note 2)
			B	CPFLB	reg,addr4	Copy Floating Level Block
			C	CPSK	reg,addr4	Copy Storage Key
			D	CPIPF	addr4	Copy In-Process Flags
			E	CPLB	reg,addr4	Copy Level Block
			F	CPPSR	addr4	Copy Processor Status and Reset

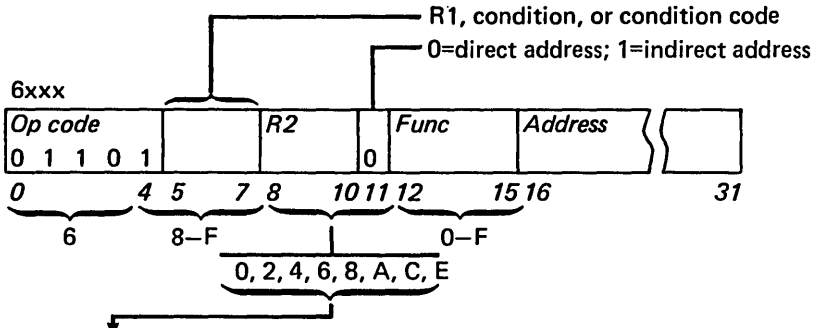
Notes:

1. Use format with K-field.
   Extended mnemonics: SEISK, SEOTK, SEOOK.
2. Use format with K-field.
   Extended mnemonics: CPISK, CPOTK, CPOOK.

6xxx



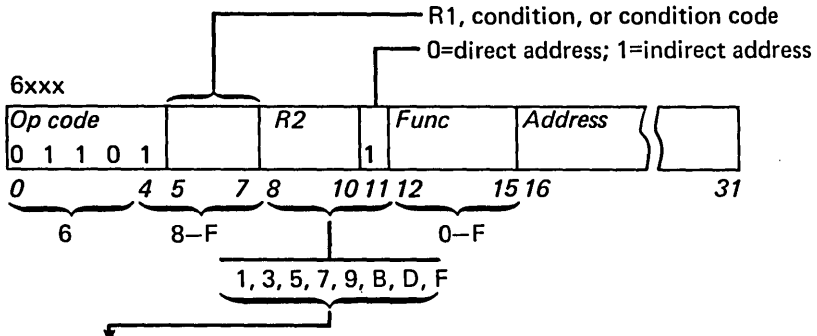
6	0	X	X	SVC ubyte — Supervisor Call
	1			LEX [ubyte] — Level Exit
	2			EN ubyte — Enable
	3			DIS ubyte — Disable
	4			STOP [ubyte] — Stop
	5			DIAG ubyte — Diagnose
	6			IOPK — Interchange Operand Keys
	7			(invalid)



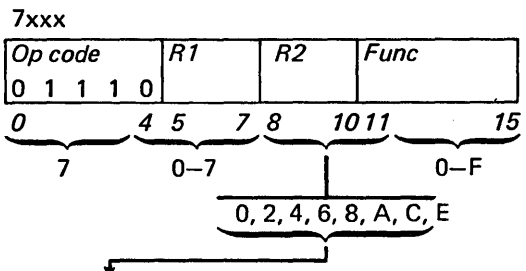
6	8–F				
		0	BC	cond,longaddr	Branch on Condition (Note 1)
		1	BNC	cond,longaddr	Branch on Not Condition (Note 2)
		2	B	longaddr	Branch Unconditional (Note 3)
		3	BAL	longaddr,reg	Branch and Link (Note 4)
		4	BCC	cond,longaddr	Branch on Condition Code (Note 5)
		5	BNCC	cond,longaddr	Branch on Not Condition Code (Note 6)
		6	BOV	longaddr	Branch on Overflow
		7	BNOV	longaddr	Branch on Not Overflow
		8	MVW	longaddr,reg	Move Word
		9	OW	longaddr,reg	OR Word
		9	SBTW	longaddr,reg	Set Bits Word
		A	RBTW	longaddr,reg	Reset Bits Word
		B	XW	longaddr,reg	Exclusive OR Word
		C	IO	longaddr	Operate I/O
		D	MVW	reg,longaddr	Move Word
		E	AW	longaddr,reg	Add Word
		F	SW	longaddr,reg	Subtract Word

Notes:

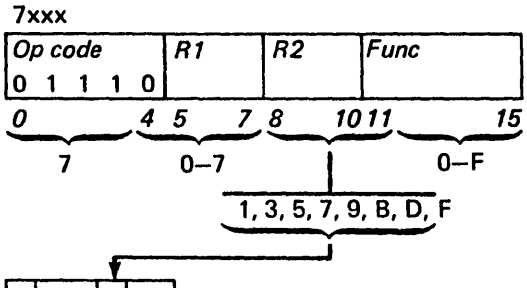
1. Extended mnemonics: BCY, BE, BEV, BLE, BLLE, BLLT, BLT, BMIX, BN, BOFF, BON, BP, BZ.
2. Extended mnemonics: BGE, BGT, BLGE, BLGT, BNCY, BNE, BNEV, BNMIX, BNN, BNOFF, BNON, BNP, BNZ.
3. Extended mnemonic: BX.
4. Extended mnemonic: BALX.
5. Extended mnemonic: BNER.
6. Extended mnemonic: BER.



6	8–F	0	BC	cond,longaddr*	Branch on Condition
		1	BNC	cond,longaddr*	Branch on Not Condition
		2	B	longaddr*	Branch Unconditional
		3	BAL	longaddr*,reg	Branch and Link
		4	BCC	cond,longaddr*	Branch on Condition Code
		5	BNCC	cond,longaddr*	Branch on Not Condition Code
		6	BOV	longaddr*	Branch on Overflow
		7	BNOV	longaddr*	Branch on Not Overflow
		8	MVW	longaddr*,reg	Move Word
		9	OW	longaddr*,reg	OR Word
		9	SBTW	longaddr*,reg	Set Bits Word
		A	RBTW	longaddr*,reg	Reset Bits Word
		B	XW	longaddr*,reg	Exclusive OR Word
		C	IO	longaddr*	Operate I/O
		D	MVW	reg,longaddr*	Move Word
		E	AW	longaddr*,reg	Add Word
		F	SW	longaddr*,reg	Subtract Word

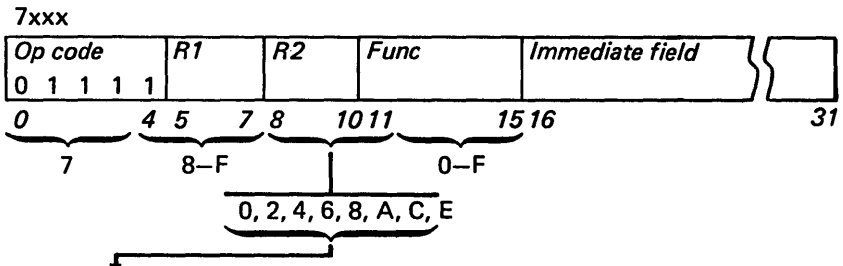


7	0–7				
		0	RBTW	reg,reg	Reset Bits Word
		1	OW	reg,reg	OR Word
		1	SBTW	reg,reg	Set Bits Word
		2	SCY	reg	Subtract Carry Indicator
		3	XW	reg,reg	Exclusive OR Word
		4	MVW	reg,reg	Move Word
		5	CW	reg,reg	Compare Word
		6	CMR	reg[,reg]	Complement Register
		7	IR	reg,reg	Interchange Registers
		8	AW	reg,reg	Add Word
		9	AWCY	reg,reg	Add Word With Carry
		A	SW	reg,reg	Subtract Word
		B	SWCY	reg,reg	Subtract Word With Carry
		C	ACY	reg	Add Carry Register
		D	VR	reg[,reg]	Invert Register
		E	CPLSR	reg	Copy Level Status Register
		F	SEIND	reg	Set Indicators



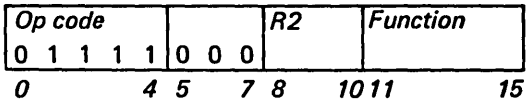
7	0–7				
		0	SLC	reg,reg	Shift Left Circular
		1	SLL	reg,reg	Shift Left Logical
		2	SRL	reg,reg	Shift Right Logical
		3	SRA	reg,reg	Shift Right Arithmetic
		4	SLCD	reg,reg	Shift Left Circular Double
		5	SLLD	reg,reg	Shift Left Logical Double
		6	SRLD	reg,reg	Shift Right Logical Double
		7	SRAD	reg,reg	Shift Right Arithmetic Double
		8	(invalid)		
		9	SLT	reg,reg	Shift Left and Test
		A	(invalid)		
		B	(invalid)		
		C	(invalid)		
		D	SLTD	reg,reg	Shift Left and Test Double
		E	(invalid)		
		F	(invalid)		

7XXX

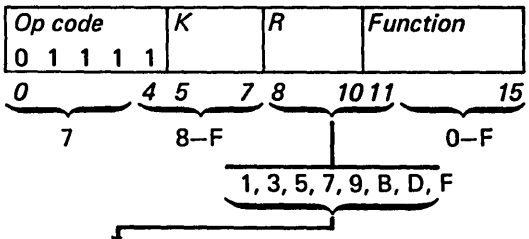


7	8–F					
		0	NWI	word,reg[,reg]	AND Word Immediate
		1	AWI	word,reg[,reg]	Add Word Immediate
		1	AA	raddr,reg[,reg]	Add Address
		2	SWI	word,reg[,reg]	Subtract Word Immediate
		2	SA	raddr,reg[,reg]	Subtract Address
		3	OWI	word,reg[,reg]	OR Word Immediate
		3	SBTWI	word,reg[,reg]	Set Bits Word Immediate
		4	RBTWI	word,reg[,reg]	Reset Bits Word Immediate
		5	XWI	word,reg[,reg]	Exclusive OR Word Immediate
		6	CWI	word,reg	Compare Word Immediate
		6	CA	raddr,reg	Compare Address
		7	TWI	word,reg	Test Word Immediate
		8	(invalid)		
		9	(invalid)		
		A	(invalid)		
		B	(invalid)		
		C	(invalid)		
		D	(invalid)		
		E	(invalid)		
		F	(invalid)		

7xxx



7xxx

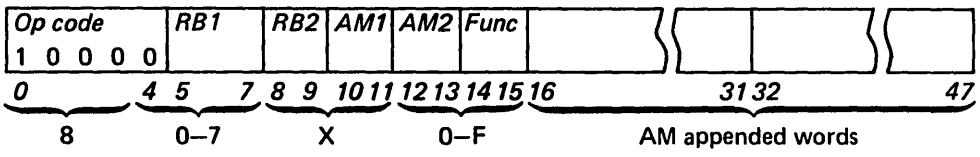


7	8–F	0	SECON	reg	Set Console Data Lights
		1	(invalid)		
		2	SEAKR	reg	Set Address Key Register (Note 1)
		3	(invalid)		
		4	SECLK	reg	Set Clock
		5	SECMP	reg	Set Comparator
		6	(invalid)		
		7	(invalid)		
		8	CPCON	reg	Copy Console Data Buffer
		9	CPCL	reg	Copy Current Level
		A	CPAKR	reg	Copy Address Key Register (Note 2)
		B	(invalid)		
		C	CPCLK	reg	Copy Clock
		D	CPCMP	reg	Copy Comparator
		E	(invalid)		
		F	(invalid)		

Notes:

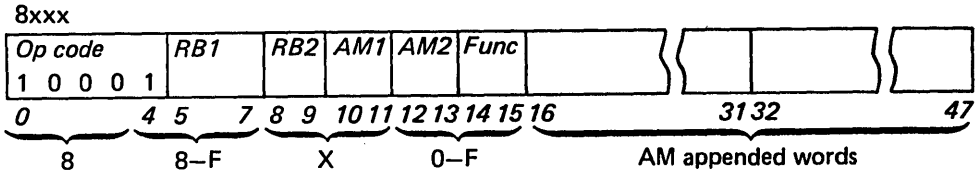
1. Use format with K-field.
   Extended mnemonics: SEISK, SEOTK, SEOOK.
2. Use format with K-field.
   Extended mnemonics: CPISK, CPOTK, CPOOK.

8xxx



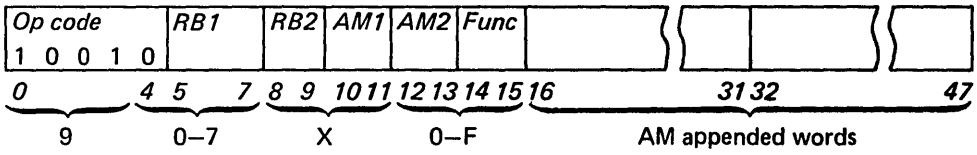
8	0–7	X	0,4 8,C	MVB	addr5,addr4	Move Byte
			1,5 9,D	OB	addr5,addr4	OR Byte
			1,5 9,D	SBTB	addr5,addr4	Set Bits Byte
			2,6 A,E	RBTB	addr5,addr4	Reset Bits Byte
			3,7 B,F	CB	addr5,addr4	Compare Byte

8xxx



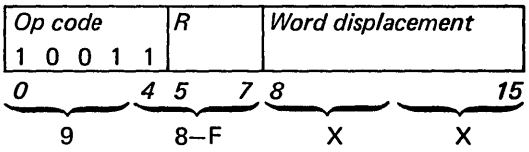
8	8–F	X	0,4 8,C	MVW	addr5,addr4	Move Word
			1,5 9,D	OW	addr5,addr4	OR Word
			1,5 9,D	SBTW	addr5,addr4	Set Bits Word
			2,6 A,E	RBTW	addr5,addr4	Reset Bits Word
			3,7 B,F	CW	addr5,addr4	Compare Word

9xxx

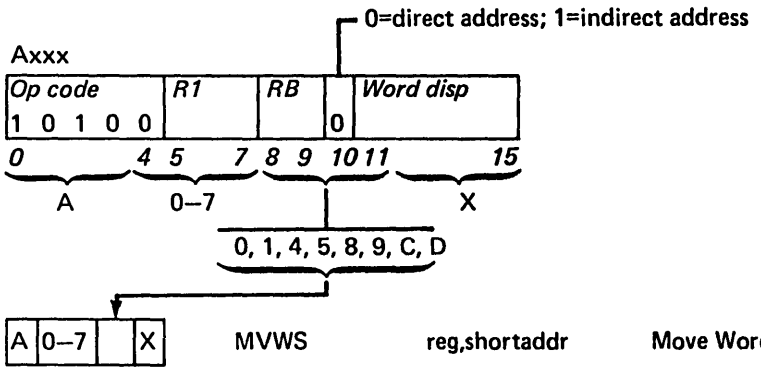


9	0–7	X	0,4 8,C	MVD	addr5,addr4	Move Doubleword
			1,5 9,D	OD	addr5,addr4	OR Doubleword
			1,5 9,D	SBTD	addr5,addr4	Set Bits Doubleword
			2,6 A,E	RBTD	addr5,addr4	Reset Bits Doubleword
			3,7 B,F	CD	addr5,addr4	Compare Doubleword

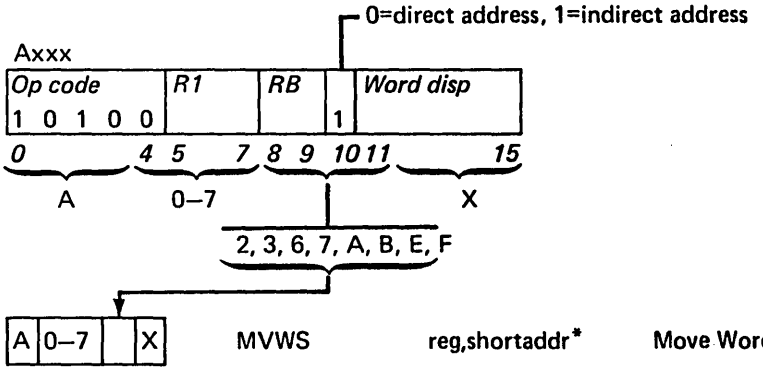
9xxx



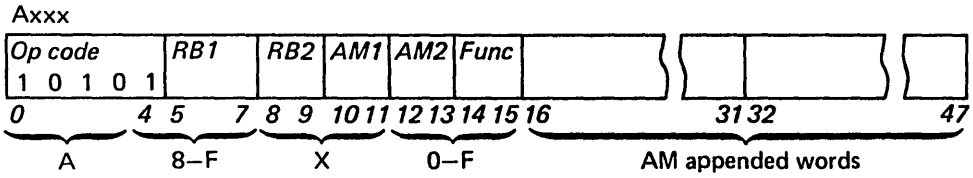
9	8–F	X	X	JAL	jdisp,reg	Jump and Link
				JAL	jaddr,reg	Jump and Link



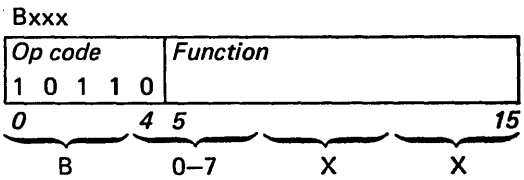
MVWS reg,shortaddr Move Word Short



MVWS reg,shortaddr* Move Word Short

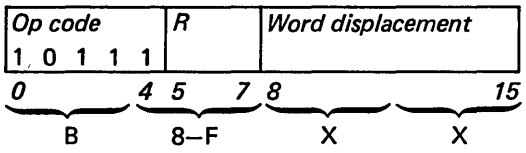


A	8–F	X	0,4 8,C	AW	addr5,addr4 Add Word
			1,5 9,D	SW	addr5,addr4 Subtract Word
			2,6 A,E	AD	addr5,addr4 Add Doubleword
			3,7 B,F	SD	addr5,addr4 Subtract Doubleword



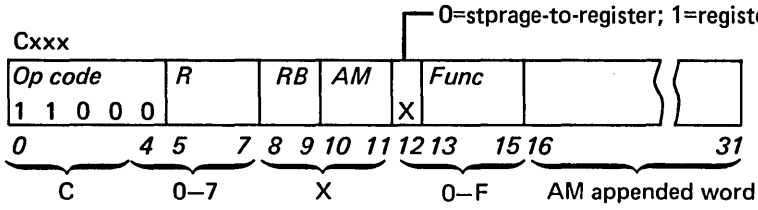
B	0–7	X	X	Invalid operation code (program-check condition)

Bxxx

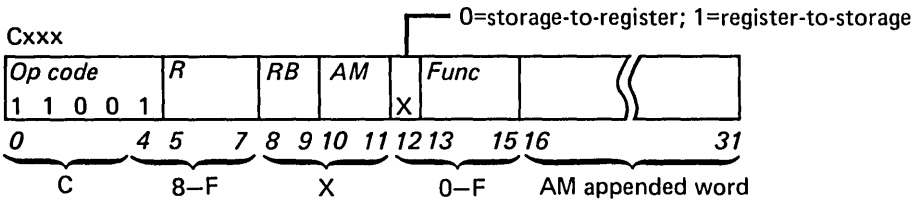


B	8–F	X	X			
				JCT	jdisp,reg	Jump on Count
				JCT	jaddr,reg	Jump on Count

Cxxx

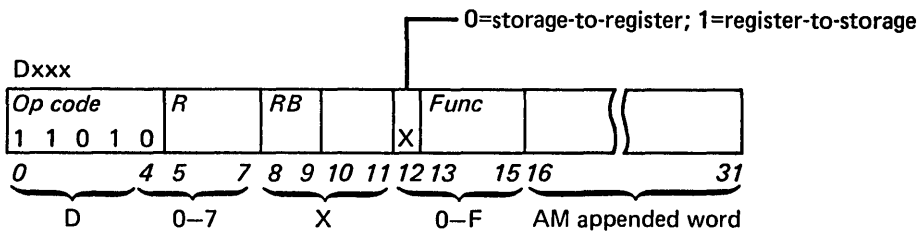


C	0–7	X				
			0	MVB	addr4,reg	Move Byte
			1	OB	addr4,reg	OR Byte
			1	SBTB	addr4,reg	Set Bits Byte
			2	RBTB	addr4,reg	Reset Bits Byte
			3	XB	addr4,reg	Exclusive OR byte
			4	CB	addr4,reg	Compare Byte
			5	MVBZ	addr4,reg	Move Byte and Zero
			6	AB	addr4,reg	Add Byte
			7	SB	addr4,reg	Subtract Byte
			8	MVB	reg,addr4	Move Byte
			9	OB	reg,addr4	OR Byte
			9	SBTB	reg,addr4	Set Bits Byte
			A	RBTB	reg,addr4	Reset Bits Byte
			B	XB	reg,addr4	Exclusive OR Byte
			C	(invalid)		
			D	(invalid)		
			E	AB	reg,addr4	Add Byte
			F	SB	reg,addr4	Subtract Byte

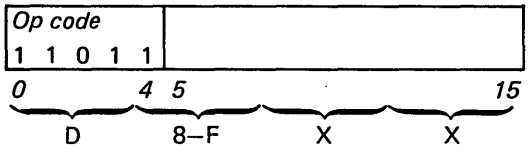


C	8–F	X				
			0	MVW	addr4,reg	Move Word
			1	OW	addr4,reg	OR Word
			1	SBTW	addr4,reg	Set Bits Word
			2	RBTW	addr4,reg	Reset Bits Word
			3	XW	addr4,reg	Exclusive OR Word
			4	CW	addr4,reg	Compare Word
			5	MVWZ	addr4,reg	Move Word and Zero
			6	AW	addr4,reg	Add Word
			7	SW	addr4,reg	Subtract Word
			8	MVW	reg,addr4	Move Word
			9	OW	reg,addr4	OR Word
			9	SBTW	reg,addr4	Set Bits Word
			A	RBTW	reg,addr4	Reset Bits Word
			B	XW	reg,addr4	Exclusive OR Word
			C	(invalid)		
			D	(invalid)		
			E	AW	reg,addr4	Add Word
			F	SW	reg,addr4	Subtract Word

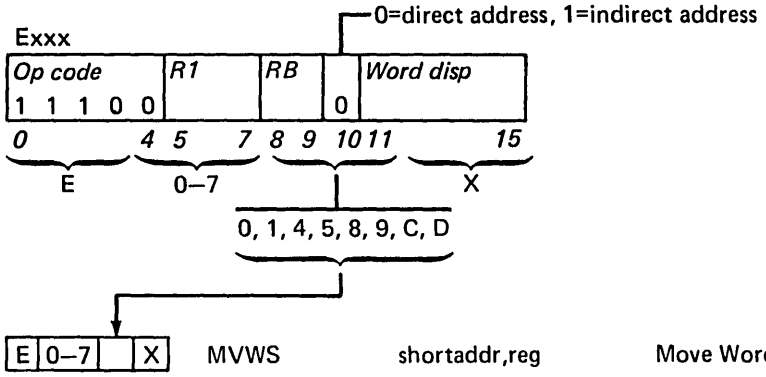
DXXX



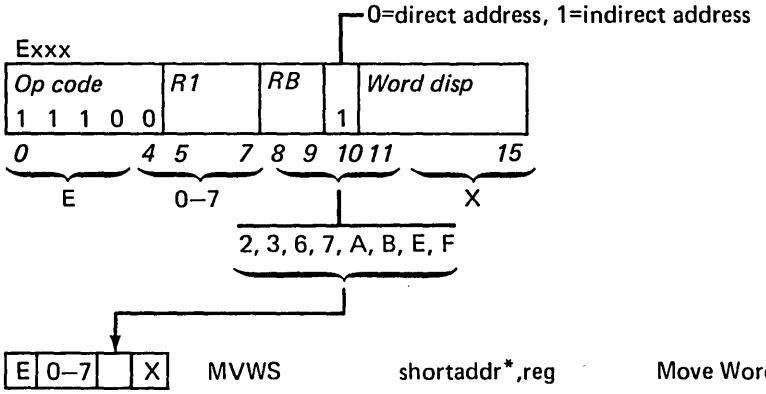
D	0–7	X				
			0	MVD	addr4,reg	Move Doubleword
			1	OD	addr4.reg	OR Doubleword
			1	SBTD	addr4,reg	Set Bits Doubleword
			2	RBTD	addr4,reg	Reset Bits Doubleword
			3	XD	addr4,reg	Exclusive OR Doubleword
			4	CD	addr4,reg	Compare Doubleword
			5	MVDZ	addr4,reg	Move Doubleword and Zero
			6	AD	addr4,reg	Add Doubleword
			7	SD	addr4,reg	Subtract Doubleword
			8	MVD	reg,addr4	Move Doubleword
			9	OD	reg,addr4	OR Doubleword
			9	SBTD	reg,addr4	Set Bits Doubleword
			A	RBTD	reg,addr4	Reset Bits Doubleword
			B	XD	reg,addr4	Exclusive OR Doubleword
			C	(invalid)		
			D	(invalid)		
			E	AD	reg,addr4	Add Doubleword
			F	SD	reg,addr4	Subtract Doubleword



D	8–F	X	X	
				Invalid operation code (program-check condition)



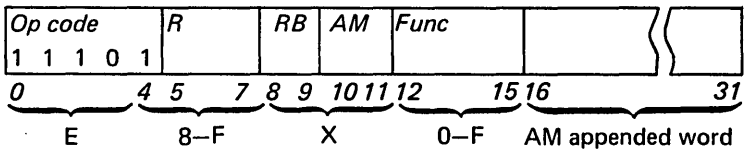
MVWS shortaddr,reg Move Word Short



MVWS shortaddr*,reg Move Word Short

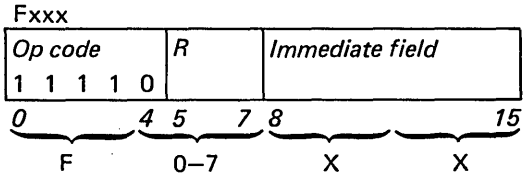
EXXX—FXXX

Exxx



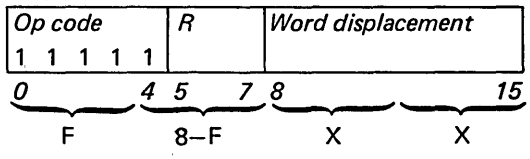
E	8–F	X				
			0	PSB	reg,addr4	Push Byte
			1	MB	addr4,reg	Multiply Byte
			2	DB	addr4,reg	Divide Byte
			3	PB	addr4,reg	Pop Byte
			4	PSW	reg,addr4	Push Word
			5	MW	addr4,reg	Multiply Word
			6	DW	addr4,reg	Divide Word
			7	PW	addr4,reg	Pop Word
			8	PSD	reg,addr4	Push Doubleword
			9	MD	addr4,reg	Multiply Doubleword
			A	DD	addr4,reg	Divide Doubleword
			B	PD	addr4,reg	Pop Doubleword
			C	(invalid)		
			D	(invalid)		
			E	(invalid)		
			F	(invalid)		

Fxxx



F	0–7	X	X			
				CBI	byte,reg	Compare Byte Immediate

Fxxx



F	8–F	X	X			
				BALS	(reg,jdisp)*	Branch and Link Short
				BALS	(reg)*	Branch and Link Short
				BALS	addr*	Branch and Link Short

Appendix B. Assembler Syntax

Coding Notes

1. Data flow modifies a field from left to right.
2. Registers used in effective address calculations are always in parentheses.
3. An address specification followed by an asterisk (*) indicates indirect addressing. The contents of the storage location at the generated address form the effective address.
4. The (reg) + format indicates that, after use, the contents of reg are increased by the number of bytes addressed by the instruction.
5. AM indicates address mode.
6. The parameter field in brackets [ ] can be optionally coded. If the field is not coded, it defaults.

Legend for Machine-Instruction Operands

abcnt	An absolute value or expression representing the size of a work storage area to be allocated by the Store Multiple (STM) instruction. The value coded must be an even number in the range 0–16382.	
addr	An address value. Code an absolute or relocatable expression in the range 0–65535.	
addr4	An address value coded in one of the following forms:	
	(reg^{0-3})	The effective address is the contents of the register reg^{0-3}. (AM=00)
	(reg^{0-3})+	The effective address is the contents of the register reg^{0-3}. After the contents are used by an instruction, they are increased by the number of bytes addressed by the instruction. (AM=01)
	addr	The effective address is the value of addr, unless the instruction and addr are within the range of the same USING statement. If they are, the assembler computes the effective address as a displacement (−32768 to +32767 or 0 to 65535) from the base register, which must be reg^{1-3}. (AM=10)
	addr*	The effective address is the contents of storage at the address defined by addr, unless the instruction and addr are within the domain and range of the same USING statement. If they are, the assembler computes the effective address as the contents of storage at the address defined by a displacement (0–255) from the base register, which must be reg^{1-3}. (AM=11)
	(reg^{1-3},waddr)	The effective address is the contents of the register reg^{1-3}, added to the value of waddr. (AM=10)
	disp1(reg^{1-3}, disp2)*	The effective address is calculated as follows: The contents of the register reg^{1-3} are added to the value of the displacement disp2 to form an address. The contents of that storage location are added to the value of disp1 to form the effective address. (AM=11)

	disp(reg[1-3])*	The effective address is the contents of storage at the address defined by the contents of reg[1-3], added to the value of disp. (AM=11)
	(reg[1-3])*	The effective address is the contents of storage at the address defined by the contents of reg[1-3]. (AM=11)
	(reg[1-3],disp)*	The contents of reg[1-3] are added to disp, forming an address. The contents of storage at that address form the effective address. (AM=11)

For byte addressing, the effective address can be even or odd. For word or doubleword addressing, the effective address must be even.

addr5 An address value that you code in one of the following forms:

(reg)	The effective address is the contents of the register reg. (AM=00)
(reg)+	The effective address is the contents of the register reg. After the contents are used by an instruction, they are increased by the number of bytes addressed by the instruction. (AM=01)
addr	The effective address is the value of addr unless the instruction and addr are within the domain and range of the same USING statement. If they are, the assembler computes the effective address as a displacement (−32768 to +32767 or 0 to 65535) from the base register, which must be reg[1-7]. (AM=10)
addr*	The effective address is the contents of storage at the address defined by addr unless the instruction and addr are within the domain and range of the same USING statement. If they are, the assembler computes the effective address as the contents of storage at the address defined by a displacement (0–255) from the base register, which must be reg[1-7]. (AM=11)
(reg[1-7],waddr)	The effective address is the contents of reg[1-7], added to the value of waddr. (AM=10)
disp1(reg[1-7], disp2)*	The effective address is calculated as follows: The contents of the register reg[1-7] are added to the value of the displacement disp2 to form an address. The contents of that storage location are added to the value of disp1 to form the effective address. (AM=11)
disp(reg[1-7])*	The effective address is the contents of storage at the address defined by the contents of reg[1-7] added to the value of disp. (AM=11)
(reg[1-7])*	The effective address is the contents of storage at the address defined by the contents of reg[1-7]. (AM=11)
(reg[1-7],disp)*	The contents of reg[1-7] are added to disp, forming an address. The contents of storage at that address form the effective address. (AM=11)

For byte addressing, the effective address can be even or odd. For word or doubleword addressing, the effective address must be even.

bitdisp A displacement into a bit field. Code an absolute value or expression in the range 0–63.

byte A byte value. Code an absolute value or expression in the range −128 to +127 or 0 to 255.

cnt16 A single-word (one register) shift count. Code an absolute value or expression in the range 0–16.

cnt31 A doubleword (register pair) shift count. Code an absolute value of expression in the range 0–31.

cond A condition-code value. Code an absolute value or expression in the range 0–7.

disp A byte-address displacement. Code an absolute value or expression in the range 0–255.

freq A floating-point register. Code either a predefined symbol (FR0–FR3) or a symbol that is equated to the desired register number (0, 1, 2, or 3). Symbols are equated with EQUR statements, which must precede the instruction using the register symbol.

jaddr The address of an instruction that is within −256 to +254 bytes of the byte following a jump instruction. Code a relocatable expression.

jdisp A displacement from the byte following a jump instruction. Code an absolute value or expression in the range −256 to +254.

longaddr An address value that you code in one of the following forms:

addr	The effective address is the value of addr, unless the instruction and addr are within the domain and range of the same USING statement. If they are, the assembler computes the effective address as a displacement (−32768 to +32767 or 0 to 65535) from the base register, which must be reg^{1-7}.
addr*	The effective address is the contents of storage at the address defined by addr, unless the instruction and addr are within the domain and range of the same USING statement. If they are, the assembler computes the effective address as the contents of storage at the address defined by a displacement (−32768 to +32767 or 0 to 65535) from the base register, which must be reg^{1-7}.
(reg^{1-7},waddr)	The effective address is the contents of reg^{1-7}, added to the value of waddr.
(reg^{1-7},waddr)*	The contents of the reg^{1-7}, plus waddr, form an address. The contents of storage at that location form the effective address.
(reg^{1-7})	The effective address is the contents of the register reg^{1-7}.
(reg^{1-7})*	The effective address is the contents of storage at the address defined by the contents of reg^{1-7}.

raddr An address value. Code a relocatable expression in the range 0–65535.

reg A general-purpose register. Code either a predefined register symbol (R0–R7) or a symbol that is equated to the desired register number (0, 1, 2, 3, 4, 5, 6, or 7). Symbols are equated with EQUR statements, which must precede the instruction using the register symbol.

reg^{0-3} A general-purpose register. Code either a predefined register symbol (R0–R3) or a symbol that is equated to the desired register number (0, 1, 2, or 3). Symbols are equated with EQUR statements, which must precede the instruction using the register symbol.

reg^{1-3} A general-purpose register. Code either a predefined register symbol (R1–R3) or a symbol that is equated to the desired register number (1, 2, or 3). Symbols are equated with EQUR statements, which must precede the instruction using the register symbol.

reg^{1-7} A general-purpose register. Code either a predefined register symbol (R1–R7) or a symbol that is equated to the desired register number (1, 2, 3, 4, 5, 6, or 7). Symbols are equated with EQUR statements, which must precede the instructions using the register symbol.

shortaddr An address value that you code in one of the following forms:

(reg^{0-3},wdisp)	The effective address is the value of wdisp added to the contents of reg^{0-3}.
(reg^{0-3},wdisp)*	The effective address is the contents of storage at the address defined by the value of wdisp added to the contents of reg^{0-3}.
(reg^{0-3})	The effective address is the contents of (reg^{0-3}).
(reg^{0-3})*	The effective address is the contents of storage at the address defined by the contents of reg^{0-3}.

	addr	To use this form, the instruction and addr must be in the domain and range of the same USING statement. The assembler computes a displacement (0–62) and register combination that refers to the requested location.
	addr*	Same as addr, except that the assembler computes the effective address as the contents of storage at the address defined by a displacement (0–62) and register combination.
	Note: For addr and addr*, the base register must be reg^{0-3}.	
ubyte	An unsigned byte value or mask. Code an absolute value or expression in the range 0–255.	
vcon	An ordinary symbol that is defined externally from the current source program.	
waddr	A one-word address value. Code an absolute or relocatable expression in the range −32768 to +32767 or 0 to 65535.	
wdisp	An even-byte address displacement. Code an absolute value or expression in the range 0–62.	
word	A word value. Code an absolute value or expression in the range −32768 to +32767 or 0 to 65535.	

Appendix C. Number Systems and Conversion Tables

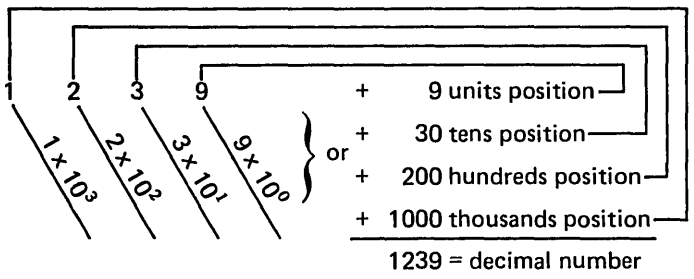
Binary and Hexadecimal Number Systems

Binary Number System

The binary number system, which is used in Series/1, uses a base of 2. The base 2 number system can be compared with the base 10 (decimal) number system.

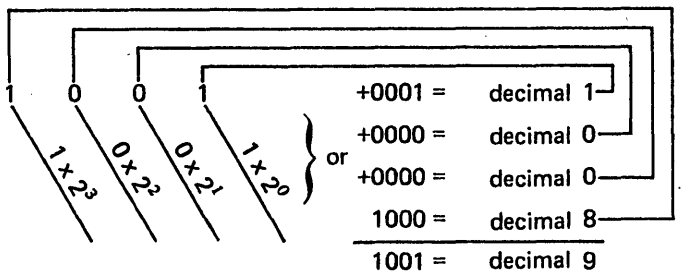
Decimal number	Binary number
0	= 0
1	= 1
2	= 10
3	= 11
4	= 100
5	= 101
6	= 110
7	= 111
8	= 1000
9	= 1001

Example of a decimal number:



As shown above, the decimal number system allows counting from 0–9 in each position; that is, the units, the tens, the hundreds, etc. Beyond 9, a carry into the next higher position is generated. The binary system allows counting from 0–1 in each position, with a carry generated when 1 is incremented. Register displays on the Series/1 console are in binary: a bit light on is a 1; a bit light off is a 0.

Example of a binary number:



Hexadecimal Number System

Binary numbers require about three times as many places as decimal numbers to express the equivalent number. This is not a problem to the computer; however, when talking and writing, or communicating with the computer, these binary numbers are bulky. A long string of 1's and 0's cannot be transmitted effectively from one individual to another; some shorthand method is necessary.

The hexadecimal number system fills this need. Because of the simple relationship of hexadecimal to binary, numbers can be converted from one system to another by inspection. The base or radix of the hexadecimal system is 16, which requires 16 symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. The letters A, B, C, D, E, and F represent the base-10 number values of 10, 11, 12, 13, 14, and 15, respectively.

Four adjacent binary positions are equivalent to one hexadecimal position. The following table compares values of the three number systems:

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

At the value F, all 16 symbols have been used, and a carry to the next higher position of the hexadecimal number is necessary. For example, for the value F, a carry results in the value 10_{16}. The following table compares the values 16_{10} to 21_{10} with binary and hexadecimal equivalents:

Decimal	Binary	Hexadecimal
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
19	0001 0011	13
20	0001 0100	14
21	0001 0101	15

—and so on—

The internal circuitry of the computer processes only binary. An operator can interpret the lights on the computer console as a hexadecimal value (for example, 0001 1110 0001 0011 as 1E13). The hexadecimal value 1E13 is easier to state than a string of 1's and 0's.

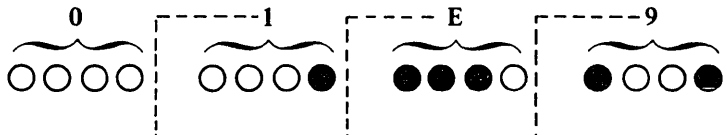
Hexadecimal—Decimal Conversion Tables

The table in this appendix allows direct conversion of decimal and hexadecimal number in these ranges:

Hexadecimal	Decimal
000 to FFF	0000 to 4095

For numbers outside the range of the table, add the following values to the table figures:

Hexadecimal	Decimal
1000	4096
2000	8192
3000	12288
4000	16384
5000	20480
6000	24576
7000	28672
8000	32768



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00_	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
01_	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
02_	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
03_	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
04_	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
05_	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
06_	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
07_	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
08_	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
09_	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A_	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B_	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C_	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D_	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E_	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F_	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255
10_	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
11_	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
12_	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
13_	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
14_	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
15_	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
16_	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
17_	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
18_	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
19_	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A_	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B_	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C_	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D_	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E_	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F_	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20_	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
21_	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
22_	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
23_	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
24_	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
25_	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
26_	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
27_	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
28_	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
29_	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A_	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B_	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C_	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D_	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E_	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F_	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
30_	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
31_	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
32_	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
33_	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
34_	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
35_	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
36_	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
37_	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
38_	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
39_	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A_	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B_	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C_	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D_	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E_	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F_	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
40_	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
41_	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
42_	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
43_	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
44_	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
45_	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
46_	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
47_	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
48_	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
49_	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A_	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B_	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C_	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D_	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E_	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F_	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
50_	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
51_	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
52_	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
53_	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
54_	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
55_	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
56_	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
57_	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
58_	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
59_	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A_	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B_	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C_	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D_	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E_	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F_	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
60_	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
61_	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
62_	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
63_	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
64_	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
65_	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
66_	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
67_	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
68_	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
69_	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A_	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B_	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C_	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D_	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E_	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F_	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791
70_	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
71_	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
72_	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
73_	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
74_	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
75_	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
76_	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
77_	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
78_	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
79_	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A_	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B_	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C_	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D_	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E_	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F_	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
80_	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
81_	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
82_	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
83_	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
84_	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
85_	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
86_	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
87_	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
88_	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
89_	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A_	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B_	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C_	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D_	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E_	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F_	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
90_	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
91_	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
92_	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
93_	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
94_	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
95_	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
96_	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
97_	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
98_	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
99_	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A_	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B_	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C_	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D_	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E_	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F_	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A0_	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A1_	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A2_	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A3_	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A4_	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A5_	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A6_	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A7_	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A8_	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A9_	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA_	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB_	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC_	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD_	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE_	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF_	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B0_	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B1_	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B2_	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B3_	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B4_	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B5_	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B6_	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B7_	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B8_	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B9_	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA_	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB_	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC_	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD_	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE_	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF_	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C0_	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C1_	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C2_	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C3_	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C4_	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C5_	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C6_	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C7_	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C8_	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C9_	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA_	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB_	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC_	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD_	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE_	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF_	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
D0_	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D1_	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D2_	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D3_	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D4_	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D5_	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D6_	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D7_	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D8_	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D9_	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA_	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB_	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC_	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD_	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE_	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF_	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E0_	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E1_	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E2_	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E3_	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E4_	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E5_	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E6_	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E7_	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E8_	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E9_	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA_	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB_	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC_	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED_	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE_	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF_	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F0_	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F1_	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F2_	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F3_	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F4_	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F5_	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F6_	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F7_	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F8_	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F9_	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA_	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB_	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC_	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD_	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE_	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF_	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

Powers of Two Table

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.0625
32	5	0.03125
64	6	0.01562 5
128	7	0.00781 25
256	8	0.00390 625
512	9	0.00195 3125
1,024	10	0.00097 65625
2,048	11	0.00048 82812 5
4,096	12	0.00024 41406 25
8,192	13	0.00012 20703 125
16,384	14	0.00006 10351 5625
32,768	15	0.00003 05175 78125
65,536	16	0.00001 52587 89062 5
131,072	17	0.00000 76293 94531 25
262,144	18	0.00000 38146 97265 625
524,288	19	0.00000 19073 48632 8125
1,048,576	20	0.00000 09536 74316 40625
2,097,152	21	0.00000 04768 37158 20312 5
4,194,304	22	0.00000 02384 18579 10156 25
8,388,608	23	0.00000 01192 09289 55078 125
16,777,216	24	0.00000 00596 04644 77539 0625
33,554,432	25	0.00000 00298 02322 38769 53125
67,108,864	26	0.00000 00149 01161 19384 76562 5
134,217,728	27	0.00000 00074 50580 59692 38281 25
268,435,456	28	0.00000 00037 25290 29846 19140 625
536,870,912	29	0.00000 00018 62645 14923 09570 3125
1,073,741,824	30	0.00000 00009 31322 57461 54785 15625
2,147,483,648	31	0.00000 00004 65661 28730 77392 57812 5
4,294,967,296	32	0.00000 00002 32830 64365 38696 28906 25
8,589,934,592	33	0.00000 00001 16415 32182 69348 14453 125
17,179,869,184	34	0.00000 00000 58207 66091 34674 07226 5625
34,359,738,368	35	0.00000 00000 29103 83045 67337 03613 28125
68,719,476,736	36	0.00000 00000 14551 91522 83668 51806 64062 5
137,438,953,472	37	0.00000 00000 07275 95761 41834 25903 32031 25
274,877,906,944	38	0.00000 00000 03637 97880 70917 12951 66015 625
549,755,813,888	39	0.00000 00000 01818 98940 35458 56475 83007 8125
1,099,511,627,776	40	0.00000 00000 00909 49470 17729 28237 91503 90625
2,199,023,255,552	41	0.00000 00000 00454 74735 08864 64118 95751 95312 5
4,398,046,511,104	42	0.00000 00000 00227 37367 54432 32059 47875 97656 25
8,796,093,022,208	43	0.00000 00000 00113 68683 77216 16029 73937 98828 125
17,592,186,044,416	44	0.00000 00000 00056 84341 88608 08014 86968 99414 0625
35,184,372,088,832	45	0.00000 00000 00028 42170 94304 04007 43484 49707 03125
70,368,744,177,664	46	0.00000 00000 00014 21085 47152 02003 71742 24853 51562 5
140,737,488,355,328	47	0.00000 00000 00007 10542 73576 01001 85871 12426 75781 25
281,474,976,710,656	48	0.00000 00000 00003 55271 36788 00500 92935 56213 37890 625
562,949,953,421,312	49	0.00000 00000 00001 77635 68394 00250 46467 78106 68945 3125
1,125,899,906,842,624	50	0.00000 00000 00000 88817 84197 00125 23233 89053 34472 65625
2,251,799,813,685,248	51	0.00000 00000 00000 44408 92098 50062 61616 94526 67236 32812 5
4,503,599,627,370,496	52	0.00000 00000 00000 22204 46049 25031 30808 47263 33618 16406 25
9,007,199,254,740,992	53	0.00000 00000 00000 11102 23024 62515 65404 23631 66809 08203 125
18,014,398,509,481,984	54	0.00000 00000 00000 05551 11512 31257 82702 11815 83404 54101 5625
36,028,797,018,963,968	55	0.00000 00000 00000 02775 55756 15628 91351 05907 91702 27050 78125
72,057,594,037,927,936	56	0.00000 00000 00000 01387 77878 07814 45675 52953 95851 13525 39062 5
144,115,188,075,855,872	57	0.00000 00000 00000 00693 88939 03907 22837 76476 97925 56762 69531 25
288,230,376,151,711,744	58	0.00000 00000 00000 00346 94469 51953 61418 88238 48962 78381 34765 625
576,460,752,303,423,488	59	0.00000 00000 00000 00173 47234 75976 80709 44119 24481 39190 67382 8125
1,152,921,504,606,846,976	60	0.00000 00000 00000 00086 73617 37988 40354 72059 62240 69595 33691 40625
2,305,843,009,213,693,952	61	0.00000 00000 00000 00043 36808 68994 20177 36029 81120 34797 66845 70312 5
4,611,686,018,427,387,904	62	0.00000 00000 00000 00021 68404 34497 10088 68014 90560 17398 83422 85156 25
9,223,372,036,854,775,808	63	0.00000 00000 00000 00010 84202 17248 55044 34007 45280 08699 41711 42578 125
18,446,744,073,709,551,616	64	0.00000 00000 00000 00005 42101 08624 27522 17003 72640 04349 70855 71289 0625

Powers of Two Table

2^n	n
18,446,744,073,709,551,616	64
36,893,488,147,419,103,232	65
73,786,976,294,838,206,464	66
147,573,952,589,676,412,928	67
295,147,905,179,352,825,856	68
590,295,810,358,705,651,712	69
1,180,591,620,717,411,303,424	70
2,361,183,241,434,822,606,848	71
4,722,366,482,869,645,213,696	72
9,444,732,965,739,290,427,392	73
18,889,465,931,478,580,854,784	74
37,778,931,862,957,161,709,568	75
75,557,863,725,914,323,419,136	76
151,115,727,451,828,646,838,272	77
302,231,454,903,657,293,676,544	78
604,462,909,807,314,587,353,088	79
1,208,925,819,614,629,174,706,176	80
2,417,851,639,229,258,349,412,352	81
4,835,703,278,458,516,698,824,704	82
9,671,406,556,917,033,397,649,408	83
19,342,813,113,834,066,795,298,816	84
38,685,626,227,668,133,590,597,632	85
77,371,252,455,336,267,181,195,264	86
154,742,504,910,672,534,362,390,528	87
309,485,009,821,345,068,724,781,056	88
618,970,019,642,690,137,449,562,112	89
1,237,940,039,285,380,274,899,124,224	90
2,475,880,078,570,760,549,798,248,448	91
4,951,760,157,141,521,099,596,496,896	92
9,903,520,314,283,042,199,192,993,792	93
19,807,040,628,566,084,398,385,987,584	94
39,614,081,257,132,168,796,771,975,168	95
79,228,162,514,264,337,593,543,950,336	96
158,456,325,028,528,675,187,087,900,672	97
316,912,650,057,057,350,374,175,801,344	98
633,825,300,114,114,700,748,351,602,688	99
1,267,650,600,228,229,401,496,703,205,376	100
2,535,301,200,456,458,802,993,406,410,752	101
5,070,602,400,912,917,605,986,812,821,504	102
10,141,204,801,825,835,211,973,625,643,008	103
20,282,409,603,651,670,423,947,251,286,016	104
40,564,819,207,303,340,847,894,502,572,032	105
81,129,638,414,606,681,695,789,005,144,064	106
162,259,276,829,213,363,391,578,010,288,128	107
324,518,553,658,426,726,783,156,020,576,256	108
649,037,107,316,853,453,566,312,041,152,512	109
1,298,074,214,633,706,907,132,624,082,305,024	110
2,596,148,429,267,413,814,265,248,164,610,048	111
5,192,296,858,534,827,628,530,496,329,220,096	112
10,384,593,717,069,655,257,060,992,658,440,192	113
20,769,187,434,139,310,514,121,985,316,880,384	114
41,538,374,868,278,621,028,243,970,633,760,768	115
83,076,749,736,557,242,056,487,941,267,521,536	116
166,153,499,473,114,484,112,975,882,535,043,072	117
332,306,998,946,228,968,225,951,765,070,086,144	118
664,613,997,892,457,936,451,903,530,140,172,288	119
1,329,227,995,784,915,872,903,807,060,280,344,576	120
2,658,455,991,569,831,745,807,614,120,560,689,152	121
5,316,911,983,139,663,491,615,228,241,121,378,304	122
10,633,823,966,279,326,983,230,456,482,242,756,608	123
21,267,647,932,558,653,966,460,912,964,485,513,216	124
42,535,295,865,117,307,932,921,825,928,971,026,432	125
85,070,591,730,234,615,865,843,651,857,942,052,864	126
170,141,183,460,469,231,731,687,303,715,884,105,728	127
340,282,366,920,938,463,463,374,607,431,768,211,456	128

Appendix D. Character Codes

Decimal	Hex	Binary	EBCDIC	ASCII	Eight-bit data inter-change	PTTC/EBCD	PTTC/ correspondence
0	00	0000 0000	NUL	NUL	NUL		
1	01	0001	SOH	SOH	NUL	space	space
2	02	0010	STX	STX		1	1,]
3	03	0011	ETX	ETX	@		
4	04	0100	PF	EOT		2	2
5	05	0101	HT	ENQ	space		
6	06	0110	LC	ACK			
7	07	0111	DEL	BEL		3	3
8	08	1000		BS		4	5
9	09	1001	RLF	HT			
10	0A	1010	SMM	LF	P (even parity)		
11	0B	1011	VT	VT	P (odd parity)	5	7
12	0C	1100	FF	FF	0 (even parity)		
13	0D	1101	CR	CR	0 (odd parity)	6	6
14	0E	1110	SO	SO		7	8
15	0F	1111	SI	SI			
16	10	0001 0000	DLE	DLE		8	4
17	11	0001	DC1	DC1			
18	12	0010	DC2	DC2	H (even parity)		
19	13	0011	TM	DC3	H (odd parity)	9	0
20	14	0100	RES	DC4	((even parity)		
21	15	0101	NL	NAK	((odd parity)	0	z
22	16	0110	BS	SYN		Ⓓ (EOA)	Ⓓ (EOA),9
23	17	0111	IL	ETB			
24	18	1000	CAN	CAN			
25	19	1001	EM	EM			
26	1A	1010	CC	SUB			
27	1B	1011	CU1	ESC	X		
28	1C	1100	IFS	FS		uppercase	uppercase
29	1D	1101	IGS	GS	8		≅
30	1E	1110	IRS	RS			
31	1F	1111	IUS	US		Ⓒ (EOT)	Ⓒ (EOT)
32	20	0010 0000	DS	space		@	t
33	21	0001	SOS	!	EOT		
34	22	0010	FS	”	D (even parity)		
35	23	0011		#	D (odd parity)	/	x
36	24	0100	BYP	$	S (even parity)		
37	25	0101	LF	%	S (odd parity)	s	n
38	26	0110	ETB	&		t	u
39	27	0111	ESC	’			

Decimal	Hex	Binary	EBCDIC	ASCII	Eight-bit data inter-change	PTTC/EBCD	PTTC/ correspondence
40	28	0010 1000		(			
41	29	1001		)		u	e
42	2A	1010	SM	*		v	d
43	2B	1011	CU2	+	T		
44	2C	1100		,		w	k
45	2D	1101	ENQ	-	4		
46	2E	1110	ACK	.			
47	2F	1111	BEL	/		x	c
48	30	0011 0000		0	form feed		
49	31	0001		1	form feed	y	l
50	32	0010	SYN	2		z	h
51	33	0011		3	L		
52	34	0100	PN	4			
53	35	0101	RS	5	,		
54	36	0110	UC	6			
55	37	0111	EOT	7		Ⓢ (SOA), comma	b
56	38	1000		8			
57	39	1001		9			
58	3A	1010		:	\ (even parity)		
59	3B	1011	CU3	;	\ (odd parity)	index	index
60	3C	1100	DC4	<	< (even parity)		
61	3D	1101	NAK	=	< (odd parity)	Ⓑ (EOB)	
62	3E	1110		>			
63	3F	1111	SUB	?			
64	40	0100 0000	space	@		N, -	!
65	41	0001		A	EOA		
66	42	0010		B	B (even parity)		
67	43	0011		C	B (odd parity)	i	m
68	44	0100		D	" (even parity)		
69	45	0101		E	" (odd parity)	k	
70	46	0110		F		l	v
71	47	0111		G			
72	48	1000		H			
73	49	1001		I		m	,
74	4A	1010	¢	J		n	r
75	4B	1011	.	K	R		
76	4C	1100	<	L		o	i
77	4D	1101	(	M	2		
78	4E	1110	+	N			
79	4F	1111	]	O		p	a
80	50	0101 0000	&	P	line feed		
81	51	0001		Q	line feed	q	o
82	52	0010		R		r	s
83	53	0011		S	J		
84	54	0100		T			
85	55	0101		U	*		

Decimal	Hex	Binary	EBCDIC	ASCII	Eight-bit data inter-change	PTTC/EBCD	PTTC/ correspondence
86	56	0101 0110		V			
87	57	0111		W		$	w
88	58	1000		X			
89	59	1001		Y			
90	5A	1010	!	Z	Z (even parity)		
91	5B	1011	$	[	Z (odd parity)	CRLF	CRLF
92	5C	1100	*	\	: (even parity)		
93	5D	1101	)	]	: (odd parity)	backspace	backspace
94	5E	1110	;	Λ		idle	idle
95	5F	1111	¬	–			
96	60	0110 0000	-		ACK		
97	61	0001	/	a		&	j
98	62	0010		b		a	g
99	63	0011		c	F		
100	64	0100		d		b	
101	65	0101		e	&		
102	66	0110		f			
103	67	0111		g		c	f
104	68	1000		h		d	p
105	69	1001		i			
106	6A	1010	¦	j	V (even parity)		
107	6B	1011	,	k	V (odd parity)	e	
108	6C	1100	%	l	6 (even parity)		
109	6D	1101	–	m	6 (odd parity)	f	q
110	6E	1110	>	n		g	comma
111	6F	1111	?	o			
112	70	0111 0000		p		h	/
113	71	0001		q	shift out		
114	72	0010		r	N (even parity)		
115	73	0011		s	N (odd parity)	i	y
116	74	0100		t	. (even parity)		
117	75	0101		u	. (odd parity)		
118	76	0110		v		Ⓨ, period	-
119	77	0111		w			
120	78	1000		x			
121	79	1001		y			
122	7A	1010	:	z		horiz tab	tab
123	7B	1011	#	{	↑		
124	7C	1100	@	|		lowercase	lowercase
125	7D	1101	'	}	>		
126	7E	1110	=	~			
127	7F	1111	"	DEL		delete	
128	80	1000 0000					
129	81	0001	a		SOM	space	space
130	82	0010	b		A (even parity)	=	±, [
131	83	0011	c		A (odd parity)		

Decimal	Hex	Binary	EBCDIC	ASCII	Eight-bit data inter-change	PTTC/EBCD	PTTC/ correspondence
132	84	1000 0100	d		! (even parity)	<	@
133	85	0101	e		! (odd parity)		
134	86	0110	f				
135	87	0111	g			;	#
136	88	1000	h		X-ON	:	%
137	89	1001	i				
138	8A	1010					
139	8B	1011			Q	%	&
140	8C	1100					
141	8D	1101			1	'	¢
142	8E	1110				>	*
143	8F	1111					
144	90	1001 0000			horiz tab	*	$
145	91	0001	j		horiz tab		
146	92	0010	k				
147	93	0011	l		I	(	)
148	94	0100	m				
149	95	0101	n		)	)	Z
150	96	0110	o			Ⓓ (EOA), "	(
151	97	0111	p				
152	98	1000	q				
153	99	1001	r				
154	9A	1010			Y (even parity)		
155	9B	1011			Y (odd parity)		
156	9C	1100			9 (even parity)	uppercase	uppercase
157	9D	1101			9 (odd parity)		
158	9E	1110					
159	9F	1111				Ⓒ (EOT)	Ⓒ (EOT)
160	A0	1010 0000			WRU (even)	¢	T
161	A1	0001	~		WRU (odd)		
162	A2	0010	s				
163	A3	0011	t		E	?	X
164	A4	0100	u				
165	A5	0101	v		%	S	N
166	A6	0110	w			T	U
167	A7	0111	x				
168	A8	1000	y				
169	A9	1001	z			U	E
170	AA	1010			U (even parity)	V	D
171	AB	1011			U (odd parity)		
172	AC	1100			5 (even parity)	W	K
173	AD	1101			5 (odd parity)		
174	AE	1110					
175	AF	1111				X	C
176	B0	1011 0000					
177	B1	0001			return	Y	L

Decimal	Hex	Binary	EBCDIC	ASCII	Eight-bit data interchange	PTTC/EBCD	PTTC/ correspondence
178	B2	1011 0010			M (even parity)	Z	H
179	B3	0011			M (odd parity)		
180	B4	0100			- (even parity)		
181	B5	0101			- (odd parity)		
182	B6	0110					
183	B7	0111				Ⓢ (SOA), |	B
184	B8	1000					
185	B9	1001					
186	BA	1010					
187	BB	1011			]	index	index
188	BC	1100					
189	BD	1101			=	Ⓑ (EOB)	
190	BE	1110					
191	BF	1111					
192	C0	1100 0000	{		EOM (even)	Ⓝ, –	
193	C1	0001	A		EOM (odd)		
194	C2	0010	B				
195	C3	0011	C		C	J	M
196	C4	0100	D				
197	C5	0101	E		#	K	
198	C6	0110	F			L	V
199	C7	0111	G				
200	C8	1000	H				
201	C9	1001	I		X-OFF	M	"
202	CA	1010			S (even parity)	N	R
203	CB	1011			S (odd parity)		
204	CC	1100	⌋		3 (even parity)	O	I
205	CD	1101			3 (odd parity)		
206	CE	1110	⌉				
207	CF	1111				P	A
208	D0	1101 0000	}				
209	D1	0001	J		vertical tab	Q	O
210	D2	0010	K		K (even parity)	R	S
211	D3	0011	L		K (odd parity)		
212	D4	0100	M		+ (even parity)		
213	D5	0101	N		+ (odd parity)		
214	D6	0110	O				
215	D7	0111	P			!	W
216	D8	1000	Q				
217	D9	1001	R				
218	DA	1010					
219	DB	1011			[	CRLF	CRLF
220	DC	1100					
221	DD	1101			;	backspace	backspace
222	DE	1110				idle	idle
223	DF	1111			PAD		

Decimal	Hex	Binary	EBCDIC	ASCII	Eight-bit data inter-change	PTTC/EBCD	PTTC/ correspondence
224	E0	1110 0000	\				
225	E1	0001			bell	+	J
226	E2	0010	S		G (even parity)	A	G
227	E3	0011	T		G (odd parity)		
228	E4	0100	U		, (even parity)	B	+
229	E5	0101	V		, (odd parity)		
230	E6	0110	W				
231	E7	0111	X			C	F
232	E8	1000	Y			D	P
233	E9	1001	Z				
234	EA	1010					
235	EB	1011			W	E	
236	EC	1100	⊣				
237	ED	1101			7	F	Q
238	EE	1110				G	comma
239	EF	1111					
240	F0	1111 0000	0		shift in (even)	H	?
241	F1	0001	1		shift in (odd)		
242	F2	0010	2				
243	F3	0011	3		O	I	Y
244	F4	0100	4				
245	F5	0101	5		/		
246	F6	0110	6			Ⓨ, ¬	—
247	F7	0111	7				
248	F8	1000	8				
249	F9	1001	9				
250	FA	1010	LVM		⇐(even parity)	horiz tab	tab
251	FB	1011			⇐(odd parity)		
252	FC	1100			? (even parity)	lowercase	lowercase
253	FD	1101			? (odd parity)		
254	FE	1110					
255	FF	1111			delete / rub out	delete	

Appendix E. Carry and Overflow Indicators

Carry Indicator Setting

The carry indicator is used to signal overflow of the result when operands are presented as *unsigned* numbers. *The machine does not regard the numbers as either signed or unsigned, but performs the designated operation (add or subtract) on the values presented. The programmer must interpret the condition of the result for the numbers involved.* The machine detects the carry condition during the operation in two ways:

1. Add operation—when a carry out of the high-order bit position of the result operand occurs.
2. Subtract operation—when a borrow beyond the high-order bit position of the result operand occurs.

Add Operation Examples

A four-bit operand size is used in the following examples. Note that the unsigned number range for this operand is 0–15. No other unsigned number values may be represented for this size operand.

- Addition (carry indicator is not set)

Desired operation: 6 + 9 = 15

Machine operation:

Augend	0110
Addend	1001
Result	1111

High-order bit carry = 0

The result fits as an unsigned number. The carry indicator is not set (C=0).

- Addition (carry indicator is set)

Desired operation: 15 + 1 = 16

Machine operation:

Augend	1111
Addend	0001
Result	0000

High-order bit carry = 1

The result does not fit as an unsigned number. The carry indicator is set (C=1).

- Addition (carry indicator is set)

Desired operation: 15 + 15 = 30

Machine operation:

Augend	1111
Addend	1111
Result	1110

High-order bit carry = 1

Result does not fit as an unsigned number. The carry indicator is set (C=1).

Note: The result of adding the two largest numbers can be contained in the operand size and the carry indicator. The carry indicator represents the most significant bit.

Subtract Operation Examples

The processor performs subtraction by using the complement addition method. The second operand is complemented (two's complement) and an add operation is performed. This is actually a three-way add operation between the minuend, the subtrahend (one's complement), and a constant of one. To provide the correct carry (borrow) indication for the subtraction, the carry result of the complement add operation must be inverted to determine the carry indicator setting. The following examples use a four-bit operand with an unsigned number range of 0–15.

- Subtract (carry indicator is not set)

Desired operation: 15 − 1 = 14

Machine operation:			
	Minuend	1111	
	Subtrahend	1110	one's complement
	Constant	1	for two's complement
	Result	1110	

High-order bit carry = 1 — invert for carry indicator

The result fits as an unsigned number. The carry indicator is not set (C=0).

Note: The carry indicator setting (C=0) for this subtract operation was determined by inverting the complement-add carry.

- Subtract (carry indicator is not set)

Desired operation: 15 − 15 = 0

Machine operation:			
	Minuend	1111	
	Subtrahend	0000	one's complement
	Constant	1	for two's complement
	Result	0000	

High-order bit carry = 1 — invert for carry indicator

The result fits as an unsigned number. The carry indicator is not set (C=0).

- Subtract (carry indicator is set)

  The following two examples show the case of a negative result (subtrahend greater than minuend). This negative result cannot be represented in the operand width because all operand bits are used to represent the unsigned number. To flag this condition, the carry indicator is set.

  Example 1.

Desired operation:	0 − 1 = − 1		
Machine operation:	Minuend	0000	
	Subtrahend	1110	one's complement
	Constant	1	for two's complement
	Result	1111	
High-order bit carry = 0		invert for carry indicator	

The result does not fit as an unsigned number. The carry indicator is set (C=1).

Example 2.

Desired operation:	0 − 15 = −15		
Machine operation:	Minuend	0000	
	Subtrahend	0000	one's complement
	Constant	1	for two's complement
	Result	0001	
High-order bit carry = 0		invert for carry indicator	

The result does not fit as an unsigned number. The carry indicator is set (C=1).

Note: When a negative result occurs on a subtract operation, the values may be useful to the programmer. The carry indicator and the result form a signed number. The carry indicator is the sign and the result is the number in two's complement form (see Figure F-4).

Overflow Indicator Setting

The overflow indicator is used to signal overflow of the result when the operands are presented as *signed* numbers. The machine does not regard the numbers as either signed or unsigned, but performs the designated operation (add or subtract) on the values presented. The programmer must interpret the condition of the result for the number representation involved. The machine detects this condition by inspection of any carry into and out of the high-order bit (sign position) of the result operand during the operation. The overflow indicator is set (O = 1) for the two cases where the carries disagree:

1. A carry into, but no carry out of the sign position.
2. No carry into, but a carry out of the sign position.

The overflow indicator is not set (O = 0) for the remaining two cases where the carries agree:

1. A carry into and out of the sign position.
2. No carry into and no carry out of the sign position.

Examples

A four-bit operand size is used in the following examples. Note that the signed number range for a four-bit operand is −8 to +7. No other signed number values may be represented.

- Addition (overflow indicator is not set)

Desired operation: +5 + (+2) = +7

Machine operation:

Augend	0101	
Addend	0010	
Result	0111	

Carry into sign position = 0

Carry out of sign position = 0 carries agree

The result fits as a signed number. The overflow indicator is not set (O = 0).

Desired operation: −4 + (−4) = −8

Machine operation:

Augend	1100	two's complement
Addend	1100	two's complement
Result	1000	two's complement

Carry into sign position = 1

Carry out of sign position = 1 carries agree

The result fits as a signed number. The overflow indicator is not set (O = 0).

- Addition (overflow indicator is set)

Desired operation: +4 + (+4) = +8

Machine operation:

Augend	0100	
Addend	0100	
Result	1000	

Carry into sign position = 1

Carry out of sign position = 0 carries disagree

The result does not fit as a signed number. The overflow indicator is set (O = 1).

Desired operation: −4 + (−5) = −9

Machine operation:

Augend	1100	two's complement
Addend	1011	two's complement
Result	0111	

Carry into sign position = 0

Carry out of sign position = 1 carries disagree

The result does not fit as a signed number. The overflow indicator is set (O = 1).

- Subtraction (overflow indicator is not set)

Desired operation: +7 − (+2) = +5

Machine operation:			
	Minuend	0111	
	Subtrahend	1101	one's complement
	Constant	1	for two's complement
	Result	0101	

Carry into sign position = 1

Carry out of sign position = 1 carries agree

The result fits as a signed number. The overflow indicator is not set (O = 0).

Desired operation: +5 − (−1) = +6

Note: A −1 is equal to 1111.

Machine operation:			
	Minuend	0101	
	Subtrahend	0000	one's complement
	Constant	1	for two's complement
	Result	0110	

Carry into sign position = 0

Carry out of sign position = 0 carries agree

The result fits as a signed number. The overflow indicator is not set (O = 0).

- Subtraction (overflow indicator is set).

Desired operation: +7 − (−2) = +9

Note: A −2 is equal to 1110.

Machine operation:			
	Minuend	0111	
	Subtrahend	0001	one's complement
	Constant	1	for two's complement
	Result	1001	

Carry into sign position = 1

Carry out of sign position = 0 carries disagree

The result does not fit as a signed number. The overflow indicator is set (O=1).

Desired operation: −3 − (+6) = −9

Machine operation:			
	Minuend	1101	two's complement
	Subtrahend	1001	one's complement
	Constant	1	for two's complement
	Result	0111	

Carry into sign position = 0

Carry out of sign position = 1 carries disagree

The result does not fit as a signed number. The overflow indicator is set (O = 1).

Unsigned Numbers

For unsigned addition and subtraction, the carry indicator signals that:

1. On an add instruction, a carry out of the high-order bit position has occurred (result exceeds result operand size). The carry indicator and the resulting operand together form a valid result of which the carry indicator is the most significant bit.
2. On a subtract operation, a borrow beyond the high-order bit position has occurred. A borrow during a subtract operation is defined as either of the following:
   a. No carry is generated out of the high-order bit position when a two's complement of the subtrahend and add is performed to accomplish the subtract operation.
   b. The most significant digit of the minuend must be made larger to generate a difference of 0 or 1 when subtracting the most significant digit of the subtrahend (for example, 1 subtracted from 0).

When a borrow is signalled on a subtract operation, the result is in two's complement form.

The overflow indicator provides no useful information about unsigned operations.

Figure E-1 shows how the carry and overflow indicators are set for an add operation when using 16-bit operands. Figure E-2 provides the same information for a subtract operation.

UNSIGNED NUMBERS

ADD OPERATION—All possible results (16-bit example)

Indicators		Result value		
Overflow	**Carry**	**Hexadecimal**	**Decimal**	
(Note 1)		0000	0	
		•		
		•		
		7FFF	32767	16-bit
		8000	32768	representable
		•		range
		•		
		FFFE	65534	
		FFFF	65535	
	1	0000	65536	
	↓	•		
	↓	•		17-bit range
	↓	7FFF	98303	using carry
	↓	8000	98304	bit
	↓	•		(Note 2)
	↓	•		
	1	FFFE	131070	

Notes:

1. The overflow indicator may be set; however, it provides no useful information.
2. With the carry indicator on, the result and carry form a valid 17-bit unsigned number of which the carry is the most significant bit.

Figure E-1. All possible results of an add operation regarding the operands as unsigned 16-bit numbers

UNSIGNED NUMBERS

SUBTRACT OPERATION—All possible results (16-bit example)

Indicators		Result value		
Overflow	**Carry**	**Hexadecimal**	**Decimal**	
(Note 1)	1	0001	−65535	
		•		
		•		
		•		17-bit
		7FFF	−32769	negative
		8000	−32768	range
		8001	−32767	(Note 2)
		•		
		•		
		•		
	1	FFFF	−1	
		0000	0	
		0001	+1	
		•		
		•		16-bit
		7FFF	+32767	represent-
		8000	+32768	able range
		•		
		•		
		•		
		FFFF	+65535	

Notes:

1. The overflow indictor may be set; however, it provides no useful information.
2. With the carry indicator (borrow) on, the result and carry indicator form a valid 17-bit negative number, of which the carry is the sign and the result is the magnitude in normal two's complement form.

Figure E-2. All possible results of a subtract operation regarding the operands as unsigned 16-bit numbers

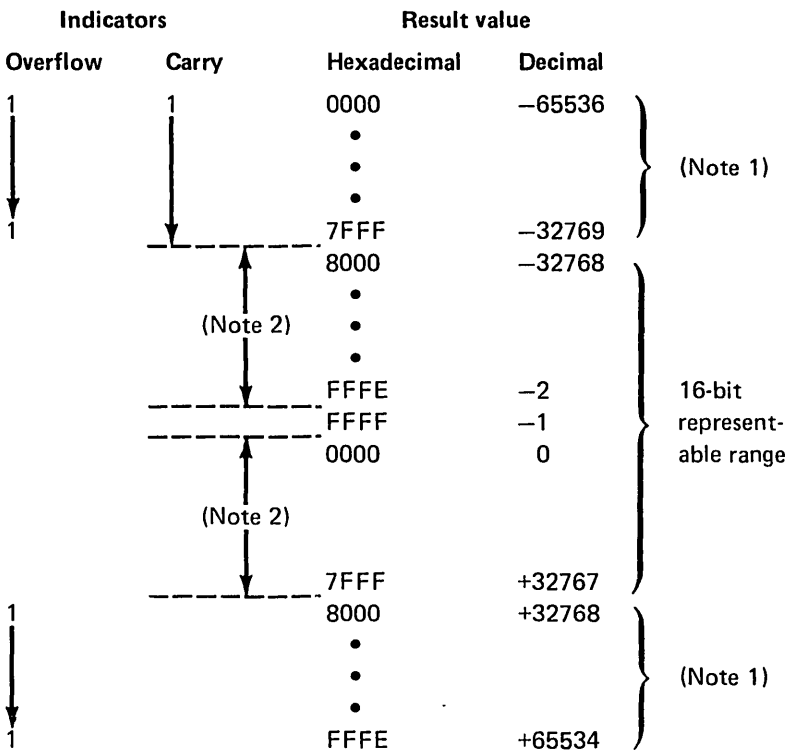
Signed Numbers

For signed addition and subtraction, the overflow indicator signals a result that exceeds the representation capability of the system for the result operand size. When overflow is indicated, the carry indicator and the resulting operand together form a valid result with the carry indicator being the most significant bit. For addition, the carry indicator is the sign (high-order bit) of this result. For subtraction, the carry indicator is the complement of the sign (high-order bit) of the result. A negative result appears in two's complement form. When no overflow is indicated, the carry indicator provides no information about the result.

Figure E-3 shows how the carry and overflow indicators are set for an add operation when using 16-bit operands. Figure E-4 provides the same information for a subtract operation.

SIGNED NUMBERS

ADD OPERATION—All possible results (16-bit example)



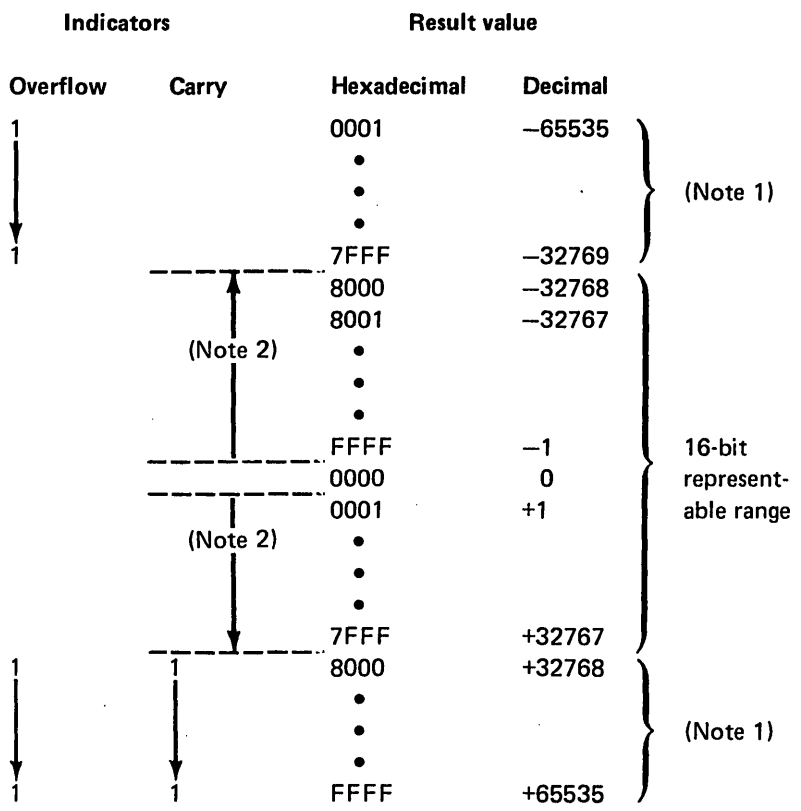
Notes:

1. When overflow occurs, the carry indicator and the result together form a valid 17-bit signed number, of which the carry is the sign, and the result is the magnitude. A negative result is in two's complement form. When no overflow occurs, no useful information is provided by the carry indicator.
2. The carry indicator may be on or off depending on the operands.

Figure E-3. All possible results of an add operation regarding the operands as signed 16-bit numbers

SIGNED NUMBERS

SUBTRACT OPERATION—All possible results (16-bit example)



Notes:

1. When overflow occurs, the carry indicator and the result form a valid 17-bit signed number, of which the carry is the complement of the correct sign, and the result is the magnitude. A negative result is in two's complement form. When no overflow occurs, no useful information is provided by the carry indicator.
2. The carry indicator may be on or off depending on the operands.

Figure E-4. All possible results of a subtract operation regarding the operands as signed 16-bit numbers

Appendix F. Reference Information

This appendix contains the following reference information:

- Address key register (AKR)
- Condition codes
- General registers
- Interrupt status byte
- Level status register (LSR)
- Processor status word (PSW)

Address Key Register (AKR)

Bits	Contents
0	Equate operand spaces
1	(not used, always 0)
2	(not used, always 0)
3	(not used, always 0)
4	(not used, always 0)
5	Operand-1 key (bit 0)
6	Operand-1 key (bit 1)
7	Operand-1 key (bit 2)
8	(not used, always 0)
9	Operand-2 key (bit 0)
10	Operand-2 key (bit 1)
11	Operand-2 key (bit 2)
12	(not used, always 0)
13	Instruction space key (bit 0)
14	Instruction space key (bit 1)
15	Instruction space key (bit 2)

Condition Codes

I/O Instruction Condition Codes

These codes are reported during execution of an Operate I/O instruction:

Condition code (CC) value	LSR position			Reported by	Meaning
	Even	Carry	Overflow		
0	0	0	0	Channel	Device not attached
1	0	0	1	Device	Busy
2	0	1	0	Device	Busy after reset
3	0	1	1	Chan/dev	Command reject
4	1	0	0	Device	Intervention required
5	1	0	1	Chan/dev	Interface data check
6	1	1	0	Controller	Controller busy
7	1	1	1	Chan/dev	Satisfactory

Interrupt Condition Codes

These condition codes are reported by the device or controller during priority interrupt acceptance:

Condition code (CC) value	LSR position			Reported by	Meaning
	Even	Carry	Overflow		
0	0	0	0	Controller	Controller end
1	0	0	1	Device	Program-controlled interrupt (PCI)
2	0	1	0	Device	Exception
3	0	1	1	Device	Device end
4	1	0	0	Device	Attention
5	1	0	1	Device	Attention and PCI
6	1	1	0	Device	Attention and exception
7	1	1	1	Device	Attention and device end

General Registers

R- or RB-field value*	Register selected
000	Register 0
001	Register 1
010	Register 2
011	Register 3
100	Register 4
101	Register 5
110	Register 6
111	Register 7

*The RB field sometimes contains only the two low-order bits. In this case, registers 4–7 cannot be specified.

Interrupt Status Byte (ISB)

DPC Devices

Bits	Contents
0	Device status available
1	Delayed command reject
2	Device-dependent
3	Device-dependent
4	Device-dependent
5	Device-dependent
6	Device-dependent
7	Device-dependent

Cycle-Steal Devices

Bits	Contents
0	Device status available
1	Delayed command reject
2	Incorrect-length record
3	DCB specification check
4	Storage data check
5	Invalid storage address
6	Protect check
7	Interface data check

Level Status Register (LSR)

Bit	Contents
0	Even indicator
1	Carry indicator
2	Overflow indicator
3	Negative result indicator
4	Zero result indicator
5	(not used, always 0)
6	(not used, always 0)
7	(not used, always 0)
8	Supervisor state
9	In-process
10	Trace
11	Summary mask
12	(not used, always 0)
13	(not used, always 0)
14	(not used, always 0)
15	(not used, always 0)

Processor Status Word (PSW)

Bit	Contents
0	Specification check
1	Invalid storage address
2	Privilege violate
3	Protect check
4	Invalid function
5	Floating-point
6	Stack exception
7	(not used, always 0)
8	Storage parity check
9	(not used, always 0)
10	CPU control check
11	I/O check
12	Sequence indicator
13	Auto IPL
14	Translator enabled
15	Power/thermal warning

Index

READER'S COMMENT FORM

GA34-0152-0

IBM Series/1 Principles of Operation

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or the IBM branch office serving your locality.

Corrections or clarifications needed:

Page Comment

Cut or Fold Along Line

Please indicate your name and address in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Reader's Comment Form

Cut Along Line

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development, Dept 27T
P.O. Box 1328
Boca Raton, Florida 33432

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

GA34-0152-0
Printed in U.S.A.

READER'S COMMENT FORM

GA34-0152-0

IBM Series/1 Principles of Operation

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or the IBM branch office serving your locality.

Corrections or clarifications needed:

Page Comment

Cut or Fold Along Line

Please indicate your name and address in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Reader's Comment Form

Cut Along Line

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development, Dept 27T
P.O. Box 1328
Boca Raton, Florida 33432

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

GA34-0152-0
Printed in U.S.A.

READER'S COMMENT FORM

GA34-0152-0

IBM Series/1 Principles of Operation

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or the IBM branch office serving your locality.

Corrections or clarifications needed:

Page Comment

Cut or Fold Along Line

Please indicate your name and address in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Reader's Comment Form

Cut Along Line

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development, Dept 27T
P.O. Box 1328
Boca Raton, Florida 33432

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

GA34-0152-0
Printed in U.S.A.

GA34-0152-0

GA34-0152-0
Printed in U.S.A.