IBM Field Engineering

Theory of Operation

(Manual of Instruction)

Restricted Distribution

Specifications contained herein are subject to change without notice. Revisions and/or supplements to this publication will be issued periodically.

System/360 Model 44

Introduction; Functional Units

IBM® Field Engineering

Theory of Operation

(Manual of Instruction)

System/360 Model 44

Introduction, Functional Units

PREFACE

This volume contains an Introduction and a Functional Units description for the IBM 2044 Processing Unit, the processor for the IBM System/360 Model 44. The Introduction gives a general outline to digital computers and computing technique, and defines the relationship of the IBM 2044 to the System/360. Functional Units describes the various parts which form the processing unit.

The manual assumes knowledge of the System/360 as described in IBM System/360 Principles of Operation, Form A22-6821.

This volume is one of the manuals that constitute the IBM Field Engineering Theory of Operation manual for the IBM System/360 Model 44. The other volumes and their form numbers are:

System/360 Model 44, Principles of Operation - Processing Unit, Form Y33-0002: Describes the operation of the instructions for the accelerator and basic machine (other thar floating-point instructions), and program and machine interrupts.

System/360 Model 44, Principles of Operation - Channels, Form Y33-0003: Describes the Common Channel area, the Multiplexor Channel 0 and the High Speed Multiplexor Channel.

These volumes are referenced in other volumes by the main element of their titles.

Reference is also made in these volumes to the following associated manuals:

Field Engineering Theory of Operation (FETO), IBM System/360 Model 44, Floating Point Feature, Form Y33-0005: Gives an introduction to floating-point arithmetic, describes the functional implementation of floating-point arithmetic in the 2044 and details the operation of floating-point instructions.

Field Engineering Theory of Operation (FETO), IBM System/360 Model 44, Single Disk Storage Drive, Form Y33-0006: Gives an introduction to the operation of the control unit and describes in detail the functional parts and the operations that may be performed.

Field Engineering Maintenance Manual (FEMM), IBM System/360 Model 44, Form Y33-0007: Contains information for servicing the 2044 Processing Unit.

Field Engineering Maintenance Diagrams (FEMD), IBM System/360 Model 44, Volume 2, Form Y33-0008: Contains maintenance information in the following categories: Data Flow Charts, Flow Charts, Timing Charts, MAP's.

Other related manuals that describe units used in the System/360 Model 44 are:

Field Engineering Manual of Instruction (FEMI), 1052 Adapter, Form 223-2808.

Field Engineering Maintenance Manual (FEMM) Single Disk Storage/Direct Access, Form Y26-3663.

First Edition

This manual makes obsolete Field Engineering Theory of Operation, System/360 Model 44, Forms Z33-0001-0, Z33-0002-0, and Z33-0004-0.

The manual is written basically to Engineering Change Level 390049 and in some cases anticipates Engineering Change Level 390063. Significant changes or additions to the information in the manual will be covered in subsequent revisions or FE supplements.

This publication was prepared by IBM European Laboratories, Product Publications. A form is provided at the back of this manual for reader's comments. If the form has been removed, comments may be addressed to: IBM Corporation, FE Manuals, Dept. B96, PO Box 390, Poughkeepsie, N.Y. 12602

CONTENTS (continued)

ILLUSTRATIONS

The following abbreviations are used in this manual:

ALS	Arithmetic and Logic Section
ASCII	American Standard Code of Information Interchange
B (field)	Base Address (field)
BOM	Basic Operating Memory
CAW	Channel Address Word
CCW	Channel Command Word
CLA	Carry Look-Ahead
CLI	Compare Logical Immediate
CPU	Central Processing Unit
CSU	Core Storage Unit
CSW	Channel Status Word
EBCDIC	Extended Binary Coded Decimal Interchange Code
EXOR	Exclusive OR
FP	Floating-Point
FPR	Floating-Point Register
GPR	General Purpose Register
Hex	Hexadecimal
HSMPX	High Speed Multiplexor (channel)
HW	Hardware (funnel)
I-Fetch	Instruction-Fetch
IC	Instruction Counter
ILC	Instruction Length Code
I/O	Input/Output
IPL	Initial Program Loading
Irpt	Interrupt
Mc	Megacycle
MPX	Multiplexor (channel)
MSLS	Medium-Speed Local Storage
ns	Nanosecond
op	Operation
OSC	Oscillator
PSW	Program Status Word
R/W	Read/Write
SA	Sense Amplifier
SAR	Storage Address Register
SC	Shift Counter
SCR	Silicon Controlled Rectifier
SDR	Storage Data Register
T/XC	True/Criss-Cross
UCW	Unit Control Word
μs	Microsecond
X (field)	Index (field)

GENERAL COMPUTER FUNDAMENTALS

- Computers process large quantities of data at electronic speeds.
- Paper-work can be automated because of the repetitive actions involved.
- Electronics have made the modern high-speed data processing system possible.

Modern methods of accounting, measuring, and testing generate huge quantities of information that must be processed quickly and accurately. A vast amount of data constantly pours into such places as retail establishments, weather stations, insurance companies, and tax bureaus. Also, our rapidly expanding scientific investigations need faster and faster methods for carrying out increasingly complex calculations. To meet these demands, machines have been developed which can automatically compute, select, and correlate data at electronic speeds.

The automation of paper-work is possible because the actions involved are sufficiently repetitive. The variety of steps necessary in processing business records or in computing scientific problems, for example, is small in comparison with the number of times these steps must be taken. One of the first paper-work machines was the ink stamp, possibly because the operation of applying a date or a name was so obviously repetitive. As additional mechanization was applied to paper-work, machines began to take over the long and painstaking tasks of accounting.

Although accounting applications of business machines require a certain amount of arithmetic, such as accumulating totals and balances, the problem is principally one of processing data. A large amount of information is fed into these machines (input) and a large quantity of information is produced (output). The machines are therefore called data processing machines.

The first data processing machine had to handle information in a series of individual operations. These included punching information into cards, sorting and classifying cards, producing totals and balances, and finally, printing the results. Intermediate results from one machine had to be transferred to another and many human decisions and interventions were necessary for a complete accounting procedure.

When electronic inventions were applied, the rate of calculation was vastly increased compared with that of other machines. But, of more importance, a new technique was introduced. This technique used electronic devices to transport data within the machine and to hold intermediate results.

Full application of these innovations has resulted in the modern computer as we know it today, in which data is fed into one end of the system and final results are presented at the other.

General Computer Configuration

- Five functional sections: input, storage, arithmetic and logic, control, and output.
- Data and instructions must be in a language and format that can be understood by a machine.
- Input may be a card reader or magnetic tape unit.
- Storage is usually magnetic cores.
- Arithmetic and logic section executes instructions (add, multiply, compare etc.).
- Control comes from storage in units of information (instructions).
- Output may include printers, punches, or magnetic tape units.
- A single instruction causes only one operation to be performed.
- Instructions are executed in an automatic sequential fashion.
- A series of instructions is called a program.

A modern computer has five distinct functional sections: input, storage, arithmetic and logic, control and output (Figure 1-1).

A computer must receive directions that are even more specific than those used in a card accounting system because a computer does nothing unless directed to do so by its control section. Therefore, the control section of the computer must receive instructions from people who plan the operations to be performed by the computer. The advantages of using a computer include greatly increased processing speeds, a higher degree of automation and greater flexibility.

All information to be used by the computer must pass through the input section, which interprets information and converts it to a language that the computer understands. The input section may include such items as card readers and magnetic tape units.

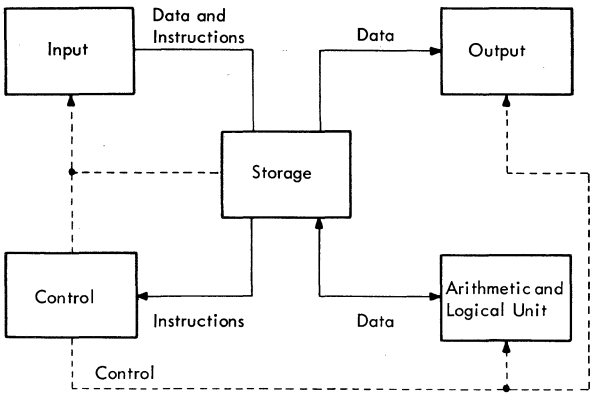


Figure 1-1. Functional Sections of a Computer

The information is directed from the input section to the storage section. Most computers use an information-holding device, composed of magnetic cores, for their main storage unit. This magnetic core storage serves as the source of all information to be used by the computer. Core storage has important advantages over the punched-card storage used in mechanical accounting systems. Most important of these advantages is the high speed at which information may be placed in, or removed from, storage. The highest degree of performance from core storage is realized only if the information is arranged in a specific order. Once the information is located in core storage, it may be called for instantly and in any sequence.

The control section of a computer directs the operation of the entire computer. Unlike the card system, the control section of a computer receives all directions in the form of units of detailed information from core storage. These units of information, which tell the control section what operations are to be performed, are called instructions. The part of information in core storage that is to be operated on is commonly referred to as data. Instructions as well as data must be delivered to storage from the input section. As shown in Figure 1-1, the control section receives instructions from storage and then exerts the necessary controls over all other sections of the computer.

Actual operations are, for the most part, performed in the arithmetic and logic section. The instructions that may be executed by a given computer may include such operations as add, subtract, multiply, divide etc. Also available will be instructions to place the results back into core storage. From core storage, instructions may tell the control section that the information is to be delivered to the output section. The output section may include printers, punches and magnetic tape units.

A single instruction causes only a specific operation to be performed by the computer. A series of instructions pertaining to an entire procedure is called a program. In modern data processing systems the program is stored internally and the system has access to the instructions at electronic speeds. Such programs are called stored programs.

When operating under program control, the computer executes one instruction at a time. After executing one instruction, the computer automatically proceeds to the next instruction. It is important to realize that every computer must be directed by some type of program during every step of its operation.

The generalized stored-program computer, illustrated in Figure 1-1, is operated in the following manner:

1. A program consisting of instructions that direct the computer in every step of its operation is punched into IBM cards. The data to be processed is also punched into cards.
2. The cards are placed in the card reader, and a key (or a series of keys) on the computer console is pressed to tell the control section that the information located in the card reader is to be read into core storage.
3. The control section starts the card reader and delivers the information to the proper locations in core storage. Information contained in the last card tells the control section where to find the next instruction.
4. The control section calls for and decodes the first instruction, to determine what operation to perform and where the data to be processed is located in core storage.
5. The control section causes this data, also located in core storage, to be delivered to the arithmetic and logic section.
6. The arithmetic and logic section performs the operation called for by the control section.
7. After the first instruction has been performed, the control section calls for the next instruction from core storage.
8. The process continues until an instruction is encountered that either tells the machine to stop or causes the results (now located in core storage) to be delivered to an output section.

IBM SYSTEM/360 FUNDAMENTALS

IBM System/360 is a general-purpose system designed for commercial, scientific, communications or control applications. A standard instruction set provides the basic computing function of the system. A decimal feature may be added to this set to provide a commercial instruction set, or a floating-point feature may be added to provide a scientific

instruction set. When the storage protection feature is added to the commercial and scientific features, a universal instruction set is obtained. Direct control and timer features may be added to satisfy requirements for teleprocessing systems to allow load sharing or to satisfy real-time needs.

System/360 can accommodate large quantities of addressable storage. The markedly increased capacities over other presently used storages are provided by the combined use of high-speed storage of medium size and medium-speed storage of large size. Thus, the requirements for both performance and size are satisfied in one system by incorporating a hierarchy of storage units. The design also anticipates future development of greater storage capacities.

System/360 incorporates a standard method for attaching input/output devices differing in function, data rate and access time. An individual System/360 is obtained by selecting the system components most suited to the applications from a wide variety of alternatives in internal performance, functional ability, and input/output (I/O).

Models of System/360 differ in storage speed, width (the amount of data obtained in each instruction access), register width, and capability of simultaneous processing, yet these differences do not affect the logical appearance of System/360 to the programmer. Several Central Processing Units (CPU's) permit a wide range of internal performance. The range is such that the ratio of internal performances between the largest and the smallest model is approximately 50:1 for scientific computation and 15:1 for commercial processing.

Compatibility

All models of System/360 are upward and downward compatible, that is, a program gives identical results on any model. Compatibility allows for ease in systems growth, convenience in systems backup, and simplicity in education.

The compatibility rule has three limitations:

1. The systems facilities used by a program should be the same in each case. Thus, the optional CPU features and the storage capacity, as well as the quantity, type, and priority of I/O equipment, should be equivalent.

2. The program should be independent of the relation of instruction execution times, I/O data rates, access time, and command execution times from machine to machine.

3. The compatibility rule does not apply to detail functions for which neither frequency of occurrence nor usefulness of result warrants identical action in all models. These functions are concerned with the handling of invalid programs and machine malfunctions.

In the case of the Model 44, programs are compatible if they adhere to the limited instruction set.

System Program

Interplay of equipment and program is an essential consideration in System/360. The system is designed to operate with a supervisory program that co-ordinates and executes all I/O instructions, handles exceptional conditions and supervises scheduling and execution of multiple programs.

System/360 provides for efficient switching from one program to another, as well as for the relocation of programs in storage. To the programmer, the supervisory program and the equipment are indistinguishable.

System Alerts

The interrupt system permits the CPU to change state automatically, as a result of conditions arising outside the system in I/O units or in the CPU itself. An interrupt switches the CPU from one program to another by changing not only the instruction address but all essential machine-status information.

A storage protection feature ensures that a program, or data in a defined area of storage, cannot be accidentally overwritten by another program. Main storage may be divided into a maximum of 16 areas, each of which has a code number or 'key' assigned to it. Each Program Status Word (PSW) also has a key, and processing under a particular PSW may be carried out only in the area of main storage defined by the key.

If an attempt is made to address any location outside the area defined by the PSW key, a protection check occurs and this location is not written into. For correct operation, no loss of performance is caused by the protection feature.

Programs are checked for correct instructions and data as they are executed. This policing action identifies and separates program errors and machine errors. Thus, program errors cannot create machine checks since each type of error causes a unique interrupt. In addition to an interrupt due to machine malfunction, the information necessary to identify the error is recorded in a predetermined storage location. This procedure appreciably reduces the mean-time to repair a machine fault. Moreover, operator errors are reduced by minimizing the active manual controls. To reduce accidental operator errors, operator consoles and I/O devices function under control of the system program.

Multi-System Operation

Several models of System/360 can be combined into one multi-system configuration. Three levels of communication between CPU's are available. Largest in capacity, and moderately fast in response, is communication by means of a shared I/O device, for example, a disk file. Faster transmission is obtained by direct connection between the channels of two individual systems. Finally, storage may be shared on some models between the CPU's, making information exchange possible at storage speeds. These modes of communication are supplemented by allowing one CPU to be interrupted by another CPU and by making direct status information available from one CPU to another.

Input/Output

Channels provide the data path and control for I/O devices as they communicate with the CPU. In general, channels operate asynchronously with respect to the CPU and, in some cases, a single data path is made up of several subchannels. When this is the case, the single data path may be shared by several low-speed devices, for example, card readers, punches, printers and terminals. This channel is called a multiplexor channel. Channels that are not made up of several such subchannels can operate at higher speed than the multiplexor channels and are called selector channels. In every case, the amount of data that comes into the channel in parallel from an I/O device is a byte (eight bits). All channels or subchannels operate in the same way and respond to similar I/O instructions and commands.

Each I/O device is connected to one or more channels by an I/O interface. This allows present and future I/O devices to be attached without alteration to the instruction set or channel function. Control units are used where necessary to match the internal connections of the I/O device to the interface. Flexibility is enhanced by optional access to a control unit or device from either of two channels.

Technology

System/360 employs solid-logic integrated components, which in themselves provide advanced equipment reliability. These components are also faster and smaller than previous components and lend themselves to automated fabrication.

In the implementation of the accelerator feature (provision of 16 general-purpose registers in hardware) use is made of monolithic circuits. In this technique several register positions (latches and gating circuits) are included in one silicon chip.

Information Formats

- Data and instructions must be in a format acceptable to the machine.
- Individual units of a coding system are called bits (binary digit information).
- Bits in various combinations represent all characters valid to the system.
- Coding systems used are binary, 4-bit and 8-bit binary-coded decimal, and hexadecimal.

Common practice in computers is to refer to individual units of a coding system as binary digit information or bits. Throughout the computer, components are always in one of two possible states: a line is active or inactive, a latch is on or off, a trigger is set or reset, a magnetic core is magnetized in one direction or the other. Components that operate in this manner (on/off, set/reset) are said to be binary. A latch, when on, represents the presence of a bit; when off, it represents the absence of a bit. Bits in various combinations represent all the characters valid to the system.

The system transmits information between main storage and the CPU in bytes or multiples of bytes. An extra (ninth) bit, the parity or check bit, is transmitted with each byte and carries parity on the bytes. The parity bit cannot be affected by the program; its only effect is to cause an interrupt when a parity error is detected. References to the size of data fields and registers, therefore, exclude the associated parity bits. All storage capacities are expressed in number of bytes provided, regardless of the physical word size used.

Figure 1-2 shows sample information formats. Bytes may be handled separately or grouped together in fields. A halfword is a group of two consecutive bytes and is the basic building block of instructions. A full word is a group of four consecutive bytes; a double word consists of two full words. The location of any group of bytes, or field, is specified by the address of its leftmost byte.

The lengths of fields are either implied by the operation to be performed or stated explicitly as part of the instruction. When the length is implied, the information is said to have a fixed length, which can be either one, two, four or eight bytes. When the length of a field is not implied by the operation code, but is stated explicitly, the information is said to have variable field length. Variable-length operands are variable in length by increments of one byte. The Model 44 uses only fixed-length information.

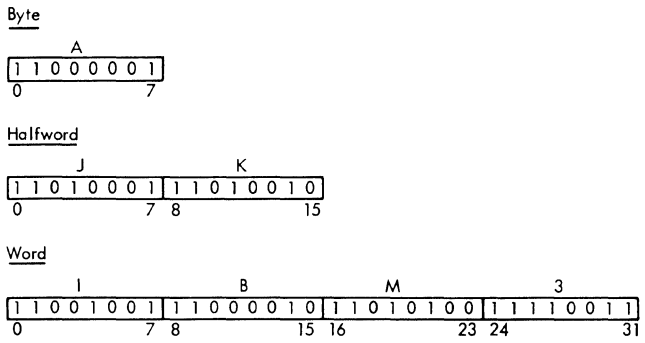


Figure 1-2. Sample Information Formats

Within any program format or any fixed-length operand format, the bits making up the format are consecutively numbered from left to right starting with the number 0.

The System/360 uses four coding systems involving the "bit" principle: binary, 4-bit and 8-bit binary-coded decimal, and hexadecimal.

Binary Coding

- The binary system is a place-value number system with the base 2.
- All numbers are expressed with the two symbols 0 and 1.
- Negative numbers are expressed in two's complement form.
- Binary numbers of various lengths are used within the system.

The binary code uses two symbols, 0 and 1, to represent all quantities. Counting is started in the binary code in the same manner as in the decimal code with 0 for zero and 1 for one. At two in the binary code there are no more symbols to be used, so it becomes necessary to take the same move at two in the binary system that is taken in the decimal system at ten. This move is to place a 1 in the next position to the left and start again with a 0 in the original position. A binary 10 is equivalent in this respect to a 2 in the decimal system. Counting is continued in an analogous manner with a carry to the next higher-order position every time an increment of two is reached instead of every time an increment of ten is reached. Counting in the binary code is as follows:

Binary	Decimal
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8

Although binary numbers generally have more terms than their decimal counterparts (about 3.3 times as many), computation in the binary system is quite simple.

Binary Addition

For binary addition, it is necessary to remember the following three rules:

1. Zero plus zero equals zero
2. Zero plus one equals one
3. One plus one equals zero with a carry of one to the next position to the left.

To see how the rules work, consider the addition of 15 and 7, using these numbers expressed in binary notation (15 = 1111; 7 = 111):

	Sixteens	Eights	Fours	Twos	Ones		Decimal
Carries	(1)	(1)	(1)	(1)			
	0	1	1	1	1	=	15
+	0	0	1	1	1	=	7
	1	0	1	1	0	=	22

The resultant sum of the addition contains a 1 in the sixteens, fours, and twos columns, which is the binary representation of 22, the correct sum of 15 plus 7 (16 plus 4 plus 2 equals 22).

Binary Subtraction

The rules for subtraction of binary digits are equally simple:

1. Zero minus zero equals zero
2. One minus one equals zero
3. One minus zero equals one
4. Zero minus one equals one, with one borrowed from the left.

Using the same numbers (15 and 7) as in the addition, the subtraction works as follows:

	Sixteens	Eights	Fours	Twos	Ones		Decimal
Borrows	0	0	0	0	0		
	0	1	1	1	1	=	15
-	0	0	1	1	1	=	7
	0	1	0	0	0	=	8

When subtraction is complete there is a 1 in the eights column only, giving an answer of 8. However, to facilitate machine operation in System/360, all negative binary numbers are expressed in two's complement form and the machine logic performs a subtraction by adding the two's complement. In the following subtract example the -7 decimal is in two's complement and appears as 01001 binary. The two's complement of any binary number is obtained by inverting each bit and then adding a bit in the least-significant position. The machine now adds the two binary values as follows:

	Sixteens	Eights	Fours	Twos	Ones		Decimal
Carries	(1)	(1)	(1)	(1)			
	0	1	1	1	1	=	15
	0	1	0	0	1	=	-7
	0	1	0	0	0	=	8

Thus, the same result is obtained by adding the two's complement of the subtrahend.

Note that in the Model 44, the effective result of the technique used for subtraction is equivalent to the addition of the two's complement. In practice, the implementation of the subtraction differs as explained in "Chapter 2, Functional Units" of this manual.

Binary Multiplication

For ordinary binary multiplication, three rules apply:

1. Zero times zero equals zero
2. Zero times one equals zero; no carries are considered
3. One times one equals one

For the binary multiplication table, all that is necessary when multiplying one number (multiplicand) by another (multiplier) is to examine the multiplier digits one at a time and, each time a 1 is found, add the multiplicand to the result. Each time a 0 is found, no addition is made. Of course, the multiplicand must be shifted for each multiplier digit, but this does not differ from the shifting that is done in the decimal system.

An example of binary multiplication, using 26 (11010) multiplied by 19 (10011), is as follows:

Decimal		Binary
26 = 16 + 8 + 0 + 2 + 0	=	11010
x 19 = 16 + 0 + 0 + 1 + 1	=	10011
		11010
		11010
		00000
		00000
		11010
		111101110

With these rules, the product is arrived at by a series of shifts (according to the number of digits) and adding the multiplicand whenever a 1 is found in the multiplier.

Interpretation of the binary result of the multiplication by using ones, twos, fours etc., results in,

256 + 128 + 64 + 32 + 0 + 8 + 4 + 2 + 0 = 494.

Note that in the Model 44 the pure binary method of multiplication is not used. Instead, a technique known as the 'carry-look-ahead group-of-ones principle' is employed. This makes an economical use of the machine's internal facilities and is summarized as follows:

A group of ones is defined as two or more ones together, scanning from right to left, which may be interspersed with single zeros. A group commences with two ones together and terminates with two zeros together. For example, such a group is 00101011. The multiplication rules are:

If the multiplier digit is a one, and is	
first of a group of ones	subtract
not the first of a group of ones	shift only
an individual one	add
If the multiplier digit is a zero, and is	
a zero within a group of ones	subtract
a zero terminating a group of ones	add
a zero part of a group of zeros	shift only

Binary Division

In the examples of addition, subtraction and multiplication, it has been shown that all of these operations are accomplished by single addition or repetitive addition.

Binary division in the Model 44 is performed by a series of subtractions. The normal rules of division apply:

1. The reduction (subtraction) cycle is successful and a quotient digit is developed if the result of the subtraction has the same sign as the original dividend.
2. The reduction cycle is unsuccessful and no quotient digit is developed if the result of the subtraction has the opposite sign to the original dividend. This condition is called an overdraw.
3. An overdraw must be followed by a correction cycle to add back the divisor and restore the dividend field to the value prior to the unsuccessful reduction. The following example shows the operation of a binary division.

269 = 0 + 256 + 0 + 0 + 0 + 0 + 8 + 4 + 0 + 1	=	0100001101
÷ 7 = 0 + 0 + 4 + 2 + 1	=	00111

A carry-out on the reduction cycle indicates a positive result and, therefore, a quotient digit.

```
Original Dividend      : 0100001101
Divisor                :      00111
Complement of Divisor  :      11001

                                          Carry      Dividend
      Operation                           Out        Field
                                                     0100001101
                                                     11001
 1  Reduction Cycle (Successful)          Yes        000010
 2  Shift Divisor                                     11001
 3  Reduction Cycle (Overdraw)            No         11011
 4  Correction Cycle                                  00111
                                                     000101
 5  Shift Divisor                                     11001
 6  Reduction Cycle (Overdraw)            No         11110
 7  Correction Cycle                                  00111
                                                     001011
 8  Shift Divisor                                     11001
 9  Reduction Cycle (Successful)          Yes        001000
10  Shift Divisor                                     11001
11  Reduction Cycle (Successful)          Yes        000011
12  Shift Divisor                                     11001
13  Reduction Cycle (Overdraw)            No         11100
14  Correction Cycle                                  00111
                                                     00011

Quotient  : 100110  (38)
Remainder : 00011   ( 3)
```

In practice the Model 44 moves the dividend field relative to the divisor, and following an overdraw, combines the correction and subsequent reduction cycle into the one cycle.

These differences are explained further in relation to the "divide instructions" in Principles of Operation - Processing Unit, Form Y33-0002.

Hexadecimal Coding

- Hexadecimal is a place-value system with the base 16.
- Hexadecimal numbers are expressed with 16 different symbols.
- Floating-point arithmetic uses the hexadecimal number system.
- One hexadecimal digit is represented by four binary bits.
- Hexadecimal representation is a convenient shorthand for writing binary numbers.

In the hexadecimal (hex) code, numerical values are expressed in combinations of four bits; however, instead of using the base 10 and indicating the decimal numbers 0 to 9, hex uses the base 16 and indicates the decimal numbers 0 to 15 (Figure 1-3).

Decimal	Binary	Hexadecimal	Decimal	Binary	Hexadecimal
0	0000	0	16	10000	10
1	0001	1	17	10001	11
2	0010	2	18	10010	12
3	0011	3	19	10011	13
4	0100	4	20	10100	14
5	0101	5	21	10101	15
6	0110	6	22	10110	16
7	0111	7	23	10111	17
8	1000	8	24	11000	18
9	1001	9	25	11001	19
10	1010	A	26	11010	1A
11	1011	B	27	11011	1B
12	1100	C	28	11100	1C
13	1101	D	29	11101	1D
14	1110	E	30	11110	1E
15	1111	F	31	11111	1F

Figure 1-3. Hexadecimal Symbol Arrangement

The same move (carry) that is made at two in the binary code and at ten in the decimal code is made at sixteen in the hexadecimal code.

Although the System/360 is a binary system, its input and output can be hexadecimal format. The hexadecimal numbering system is used in the System/360 as a convenient method of representing large binary numbers. The hexadecimal number system uses 16 symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F. The arrangement of symbols to represent the same values in decimal, binary, and hexadecimal is shown in Figure 1-3.

To convert binary numbers to hexadecimal, mark off the binary symbols in groups of four from the radix point position and substitute a hexadecimal symbol for each group of four binary symbols.

```
1100001111100110110100 11. 1001=1100,0011,1110,0110,1101,0011. 1001
  hexadecimal equivalent      = C    3    E    6    D    3    9
```

Decimal Coding

- Every decimal digit is represented by four bits.
- Two decimal numbers occupy one byte in packed decimal format.
- Only the values 0 to 9 are valid.
- In packed decimal format the low-order four bits contain a sign code.
- Negative numbers are carried in true form.

NOTE: The Model 44 does not use decimal arithmetic, and decimal type arithmetic and decimal type instructions are invalid. However, the decimal notation is included here as it is used by other models of System/360.

Decimal arithmetic within System/360 is performed with four-bit binary-coded decimal numbers,

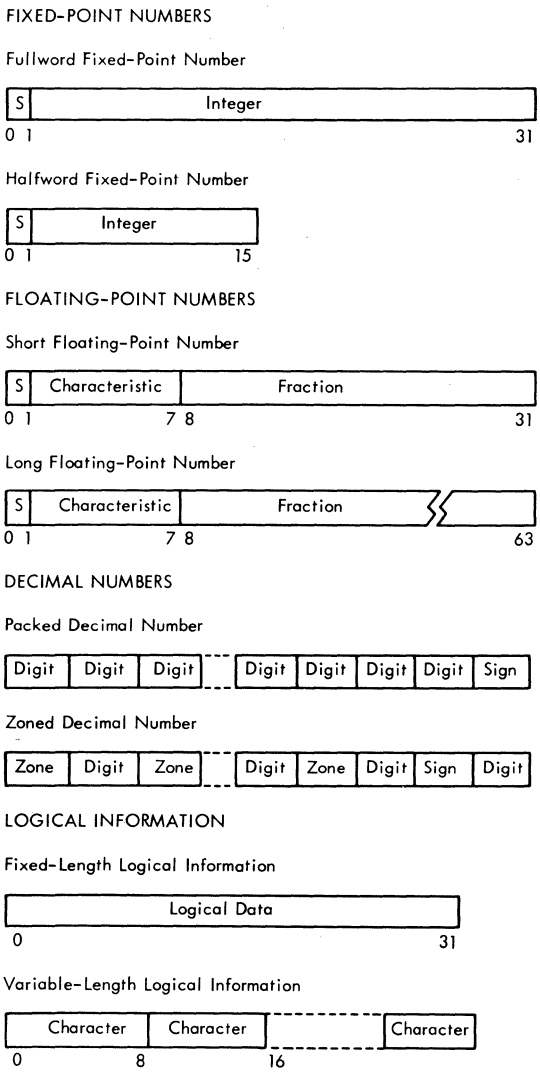


Figure 1-4. Data Format

with two such decimal numbers per byte. This format is called 'packed decimal' (Figure 1-4) with the zoned decimal format and other data formats to be discussed later. In the binary-coded decimal notation, only the numbers 0000 to 1001 (0 to 9 decimal) are valid digit codes. The remaining codes 1010 to 1111 are used to represent the sign as follows:

1011 = + American Standard Code for
1101 = - Information Interchange (ASCII)

1100 = + Extended Binary Coded Decimal
1101 = - Interchange Code (EBCDIC)

The two remaining codes, 1110 and 1111, are regarded as positive in both coding systems. The positioning of the four-bit sign for both packed and unpacked formats is shown in Figure 1-4. Generation of a sign code during decimal arithmetic is governed by the character set preferred, as defined by PSW 1 bit 12.

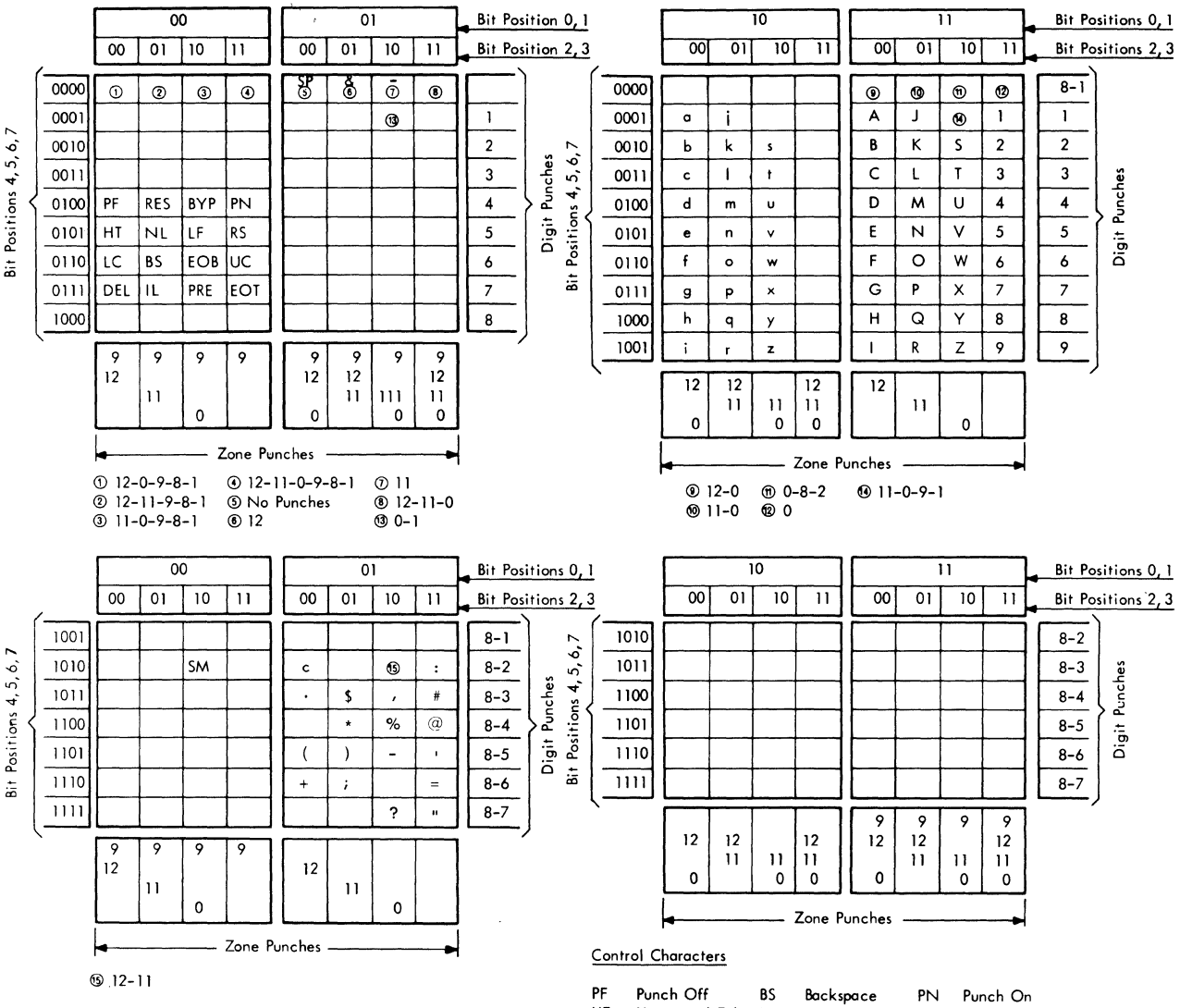
All character-set-sensitive I/O devices assume either one of the two codes, EBCDIC or ASCII. These codes are shown in Figures 1-5 and 1-6 respectively. In both coding systems each character is represented by eight bits (one byte), giving a possible maximum character set of 256. By means of an extended card code, each one of the possible 256 characters may be punched into a card column.

Floating-Point Notation

- Expresses large and small numbers in a shortened form.
- Very small and very large number usually contain many zeros.
- Zeros in large and small numbers usually serve only as 'spacers' (to locate the decimal point).
- As precision is increased, percentage of change in value decreases.
- Data may be either single word (short) or double word (long) format.
- Range of (hexadecimal) numbers obtainable is $.00000000000001 \times 16^{-64}$ to $.FFFFFFFFFFFFFF \times 16^{63}$.
- The sign position of the floating-point data word is the sign of the fraction.
- The exponent is modified by adding 1000000 (64) to the power of 16 required, to position the radix point.

Floating-point notation is a means of expressing large and small numbers in a shortened form. To understand floating-point notation and terminology, the properties of number representation must first be examined.

Two important characteristics exist in any number system: the quantities of unique symbols or words, and the method used to combine these symbols to express increasing quantities. A number system could have a unique symbol for every quantity to be expressed. Such a system would allow the shortest notation of any quantity, but would require an infinite number of symbols. Practical number systems use fewer symbols and combine them to represent increasing quantities. The length of notation (compactness), therefore, depends on the quantity of unique symbols available and the method used to combine them.



Control Characters

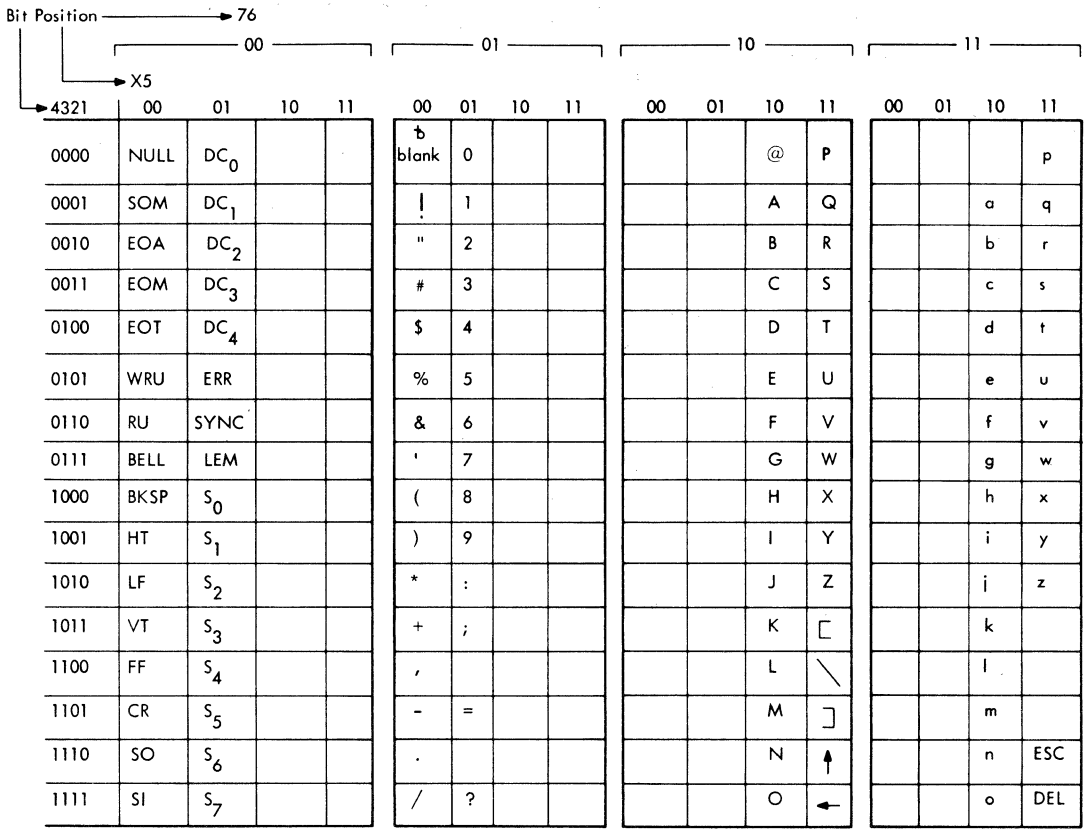
PF	Punch Off	BS	Backspace	PN	Punch On
HT	Horizontal Tab	IL	Idle	RS	Reader Stop
LC	Lower Case	BYP	Bypass	UC	Upper Case
DEL	Delete	LF	Line Feed	EOT	End of Transmission
RES	Restore	EOB	End of Block	SM	Set Mode
NL	New Line	PRE	Prefix	SP	Space

Special Graphic Characters

c	Cent Sign	*	Asterisk	>	Greater-than Sign
.	Period, Decimal Point	)	Right Parenthesis	?	Question Mark
<	Less-than Sign	;	Semicolon	:	Colon
(	Left Parenthesis	¬	Logical NOT	#	Number Sign
+	Plus Sign	-	Minus Sign, Hyphen	@	At Sign
|	Vertical Bar, Logical OR	/	Slash	'	Prime, Apostrophe
&	Ampersand	,	Comma	=	Equal Sign
!	Exclamation Point	%	Percent	"	Quotation Mark
$	Dollar Sign	_	Underscore		

Examples	Type	Bit Pattern Bit Positions 01 23 4567	Hole Pattern: Zone Punches	Hole Pattern: Digit Punches
PF	Control Character	00 00 0100	12-9-4	
%	Special Graphic	01 10 1100	0-8-4	
R	Upper Case	11 01 1001	11-9	
a	Lower Case	10 00 0001	12-0-1	
	Control Character, function not yet assigned	00 11 0000	12-11-0-9-8-1	

Figure 1-5. Extended Binary Coded Decimal Interchange Code



Bit Position → 76 / X5 / 4321	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
0000	NULL	DC_0			b blank	0					@	P				p
0001	SOM	DC_1			!	1					A	Q			a	q
0010	EOA	DC_2			"	2					B	R			b	r
0011	EOM	DC_3			#	3					C	S			c	s
0100	EOT	DC_4			$	4					D	T			d	t
0101	WRU	ERR			%	5					E	U			e	u
0110	RU	SYNC			&	6					F	V			f	v
0111	BELL	LEM			'	7					G	W			g	w
1000	BKSP	S_0			(	8					H	X			h	x
1001	HT	S_1			)	9					I	Y			i	y
1010	LF	S_2			*	:					J	Z			j	z
1011	VT	S_3			+	;					K	[			k	
1100	FF	S_4			,						L	\			l	
1101	CR	S_5			-	=					M	]			m	
1110	SO	S_6			.						N	↑			n	ESC
1111	SI	S_7			/	?					O	←			o	DEL

Figure 1-6. American Standard Code for Information Interchange

Although the decimal number system is the more efficient for the entire range of numbers, the floating-point notation used in the System/360 is more efficient for most very large and very small numbers. It is first necessary to see why the floating-point notation is needed for very small and very large numbers and what characteristics these numbers have that make it possible to create a shortened notation.

Problems in Handling Very Small and Very Large Numbers

Very small and very large numbers are often encountered in scientific calculations. The electron, for example, has a charge of 0.00000000048 cgs (centimeter gram seconds). The sun is approximately 93,000,000 miles away from the earth. It is conceivable that numbers used in scientific calculations may be 50 digits in length. If a computer were designed to process such numbers unaltered, provision would have to be made for 100 digit positions, to allow, for example, multiplication. Such a computer would be expensive to build, and its calculating speed would be low. Thus, there arises the need for a shortened (floating-point) notation for very small and very large numbers.

Very small and very large numbers usually contain many zeros. To understand why this is so, the two cases must be examined separately.

Very Small Numbers: Very small numbers, such as the example showing the charge of an electron (0.00000000048), contain zeros that serve only as "spacers." These spacers are included as part of the number because they locate the position of the decimal point. It is impossible to change any zero in this number without changing the value by a great amount; the least effect would be to change the zero in front of the four to a one (0.00000000148), but this more than triples the number.

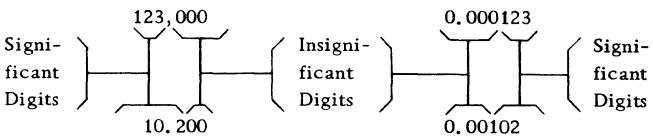
Very Large Numbers: Degrees of Precision: The zeros in very large numbers serve as spacers; however, the number of zeros in most large numbers depends on the instrument used to measure that quantity. Although scientific instruments measure very small and very large quantities, these instruments are limited in the number of digits that can be read accurately. For example, the distance to the sun (93,000,000 miles approximately) may be 92,999,999 miles. To know this, the instrument for measuring the distance would have to distinguish one part in one hundred million, and is unusual to

find such an instrument. Furthermore, with quantities this large, a precise measurement does not have the significance that we normally associate with the term 'precise'. For example, the difference between the figures of 93,000,000 and 92,999,999 is one millionth of one percent. So, as precision is increased, the percentage of change in value becomes so small as to be insignificant. From a practical viewpoint, most of the zeros in large numbers serve only as spacers.

Length of Notation

Scaling: The fact that all very small numbers, and most very large numbers, contain many zeros that act as spacers can be used to advantage in reducing the length of notation for these numbers. For example, a programmer wishes to multiply two 20-digit numbers in a computer built to handle factors of ten digits or less. The first (and still commonly used) method of performing this job is called "scaling." The programmer removes the spacing zeros from the two factors, remembering how many he removes. He then inserts the shortened numbers into the computer, where the multiplication is performed. Finally, he adds back the zeros to the product obtained from the computer. In this case, he adds the same number of zeros to the product that he removed from the two factors: 30 x 20 = 3 x 2 (remember two zeros removed): the product equals 6 with two zeros, or 600.

Significant Digits: Throughout this manual the digits are called significant digits and the spacer zeros are referred to as insignificant digits. The following examples illustrate the meaning of the two terms:



Scientific Notation

Powers of Ten: In the example of scaling that was given, it was necessary for the programmer to remember the number of insignificant digits that he removed. Scientific notation makes this remembering task simpler. The number 20 can be expressed in scientific notation as 2.0 x 10. This gives a method of notation that can be used to record separately the significant and insignificant digits of a number. Note the position of the decimal point in the example: 20.0 = 2.0 x 10. The value of a digit position is always ten times the value of the position immediately to its right, and moving the decimal point to the left divides a number by ten (20.0, 2.0, 0.20). Using this fact scientific notation gives 20.0 = 0.2 x 100; also 20.0 = 0.02 x 1000. As the decimal point is moved to the left, the power of ten that must be used as a multiplier is increased. Using scientific notation, any number can be expressed in many ways:

$20.0 = 20.0 \times 1 = 20.0 \times 10^{0}$ (zero power of ten)

$20.0 = 2.0 \times 10 = 2.0 \times 10^{1}$ (first power of ten)

$20.0 = 0.2 \times 100 = 0.2 \times 10^{2}$ (second power of ten)

$20.0 \times 0.02 \times 1000 = 0.02 \times 10^{3}$ (third power of ten)

Moving the Decimal Point: Scientific notation extends the same idea to allow moving the decimal point to the right. Each movement of the decimal point to the right multiplies a number by ten (20.0, 200.0, 2000.0). In scientific notation, each movement to the right increases the power of ten that must be used as a divisor (20.0 = 200÷10 = 2000÷100). The division is actually accomplished by multiplying a fraction. For example, 20÷10 = 20 x 1/10. The fractions 1/10, 1/100, 1/1000 are denoted by a negative power of ten: $1/10 = 10^{-1}$, $1/100 = 10^{-2}$, $1/1000 = 10^{-3}$. Summarizing the use of the negative power of ten:

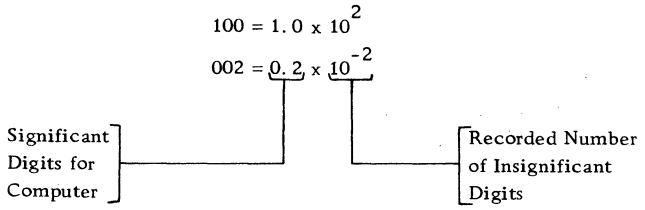
$20 = 200 \times 1/10 = 200 \times 10^{-1}$ (first negative power of ten)

$20 = 2000 \times 1/100 = 2000 \times 10^{-2}$ (second negative power of ten)

$20 = 20000 \times 1/1000 = 20000 \times 10^{-3}$ (third negative power of ten)

The usefulness of the negative powers of ten is more readily understood by applying scientific notation to a small number: $0.0003 = 3.0 \times 10^{-4}$.

It has been shown how scientific notation helps a programmer to remember the two halves of fractions that he scales:



Handling Zeros: When scaling, it is necessary to add zeros to answers obtained from the computer. The following examples show how the addition of zeros is accomplished for each of the four arithmetic operations: multiplication, division, addition and subtraction.

When using scientific notation, multiplication is performed by multiplying the significant digits and adding the exponents (powers of ten) arithmetically:

$$300 \times 150 = 45000$$

$$(3 \times 10^2) \times (15 \times 10^1) = (45 \times 10^3) \longrightarrow \text{manual method}$$

$$\left\{ \begin{array}{ll} 3 \times 15 = 45 & \longrightarrow \text{performed by computer} \\ 10^2 \times 10^1 = 10^{(2+1)} = 10^3 & \longrightarrow \text{added by programmer} \end{array} \right\}$$

$$= (45 \times 10^3)$$

Division calls for dividing significant digits and arithmetically subtracting exponents:

$$300 \div 150 = 2$$

$$(3 \times 10^2) \div (15 \times 10^1) = (0.2 \times 10^1) \text{ or } (2 \times 10^0) \longrightarrow \text{manually}$$

$$\left\{ \begin{array}{ll} 3 \div 15 = 0.2 & \longrightarrow \text{performed by computer} \\ 10^2 \div 10^1 = 10^{(2-1)} = 10^1 & \longrightarrow \text{added by programmer} \end{array} \right\}$$

$$= (0.2 \times 10^1)$$

For addition, the exponent must be made equal by shifting the decimal point; the exponent of the result is then the same as the exponent of the factors:

$$20 + 1 = 21$$

$$(2 \times 10^1) + (.1 \times 10^1)$$

$$\left\{ \begin{array}{lll} 2 + .1 = 2 + .1 = 2.1 & \longrightarrow \text{performed by computer} \\ 10^1 + 10^1 \quad\; 10^1 \quad\; 10^1 & \longrightarrow \text{added by programmer} \end{array} \right\}$$

$$= 2.1 \times 10^1$$

The process of making the exponents equal corresponds to aligning the two factors (aligning the decimal points) in fixed-point arithmetic:

$$2 + .1 = 2.1$$

$$\begin{array}{r} 2.0 \\ \underline{0.1} \\ 2.1 \end{array}$$

Subtraction is similar to addition.

$$20 - 1 = 19$$

$$(2 \times 10^1) - (.1 \times 10^1) = (1.9 \times 10^1)$$

$$\left\{ \begin{array}{lll} 2 - .1 = 2 - .1 = 1.9 & \longrightarrow \text{performed by computer} \\ 10^1 - 10^1 \quad\; 10^1 \quad\; 10^1 & \longrightarrow \text{added by programmer} \end{array} \right\}$$

From these sample calculations the prime disadvantage of scaling becomes apparent: the programmer is required to keep track of the decimal point; he must record the manner in which he scales his factors, then manually process the exponents and re-scale the results obtained from the computer.

Reasons for Floating-Point

Certain applications of the System/360 demand extensive handling of numbers having more than eight digits. In these applications, the numbers may be extremely large, extremely small, or, in some cases, unpredictable. Such situations make fixed-point arithmetic difficult to work with for two reasons:

1. The size of the number is limited by the size of the register.
2. The programmer must keep track of the radix point in all numbers throughout the calculation.

System/360 Floating-Point

Floating-point operation codes (op codes) enable the System/360 to process numbers in scientific notation; both the shortened numbers and their exponents are processed by the computer. In the floating-point operation the radix point of the fraction is initially located and the computer keeps track of it, thereby simplifying the job of scaling.

Terminology

Floating-point instructions perform addition, subtraction, multiplication and division on a type of number representing a numeric shorthand: the floating-point number. Very large, and small, fixed-point numbers are reduced to a hexadecimal fraction and a binary exponent. The fraction and the exponent are operated on separately. Operating on the fraction provides the significant digit result; operating on the exponent places the radix point in the result.

Example:	Fraction		Exponent
3500 may be expressed	0.35	x	10^4
	Fraction		Exponent
570 may be expressed	0.57	x	10^3

The product of these two numbers is then calculated as:

0.1995×10^7 or 1,995,000

System/360 floating-point op codes apply the same principle to numbers either too large or too small to be conveniently handled by fixed-point arithmetic instructions. A large number is reduced to a hexadecimal fraction having a radix point to the left of the high-order digit, and an exponent to the base 16. Similarly a small number is reduced to a hexadecimal fraction having a radix point to the left of the highest order significant digit and an exponent to the base 16.

Data Format

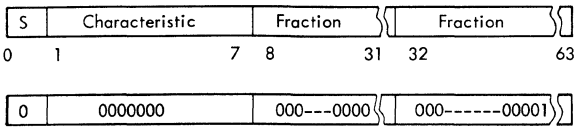
Floating-point information occupies a fixed-length format (Figure 1-4) which may be either a full word or a double word. The first bit in either format is the sign bit (S). The subsequent seven bit positions are occupied by the exponent, while the fraction field may have six or 14 hexadecimal digits (24 or 56 bits).

Sign Handling

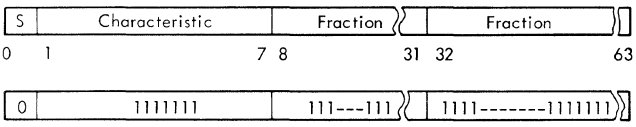
Both the exponent and the sign of the fraction can be either positive or negative. The sign position of the data word is always the sign of the fraction, and the fraction is always expressed in true form. The exponent is modified so that it contains not only the exponent of the floating-point number, but the sign of the exponent also. The modified exponent is a binary number between 0000000 and 1111111, and is now called the characteristic. This characteristic is obtained by adding 1000000 (64) to the power of 16 required, to position the radix point properly. In this notation 1000000 is considered 16^0, 1000001 is 16^1, 1000010 is 16^2, 0111101 is 16^{-3}, and so on, (assuming a fraction of .1).

Capabilities and Limitations

The significant numbers thus obtainable range in magnitude between $.00000000000001 \times 16^{-64}$ to $.FFFFFFFFFFFFFF \times 16^{63}$. The smallest positive significant number would appear in floating-point notation as:



The largest positive number would appear as:



Any floating-point number having a fraction of zero is automatically assigned a characteristic of zero.

All arithmetic operations may be performed with the floating-point op codes.

Normalized Numbers

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 number has a non-zero high-order hexadecimal fraction digit. If one or more high-order fraction digits are zero, the number is said to be un-normalized. The process of normalization consists of shifting the fraction left until the high-order hexadecimal digit is non-zero, and reducing the characteristic by the number of hexadecimal digits shifted. A zero fraction cannot be normalized and its associated characteristic therefore remains unchanged when normalization is called for.

Normalization usually takes place when the intermediate arithmetic result is converted to the final result. This fraction is called post-normalization. In performing multiplication and division, the operands are normalized prior to the arithmetic process. This function is called pre-normalization.

Floating-point operations may be performed with or without normalization. Most operations are post-normalized. Addition and subtraction may be specified either way.

When an operation is performed without normalization, high-order zeros in the result fraction are not eliminated. The result may be normalized or not, depending on the original operands.

In both normalized and un-normalized operations, the initial operands need not be in normalized form. Also, intermediate fraction results are shifted right when an overflow occurs; the intermediate fraction result is usually truncated to the final result length after the shifting, if any. Because normalization applies to hexadecimal digits (represented by four binary bits), the three high-order bits of a normalized number may be zero.

MODEL 44 FUNDAMENTALS

- The CPU of the Model 44 is called the IBM 2044.
- Except for the single-disk storage drive all I/O devices are physically separate from the 2044.
- Input devices feed instructions and data via the channel to the storage.
- Instructions are read out from storage and decoded to provide the system control.
- Data from storage is processed within the main data-flow registers and the result is set back into storage.
- Output devices receive data for recording from storage via the channel.
- All information entering or leaving the computer must pass through storage.
- The functional sections of the 2044 work together to process data.

The Model 44 is designed specifically for small to medium-size scientific applications, advanced data acquisition and process-control applications. The Model 44 contains a large-capacity high-speed storage enabling fast computing speeds with a 32-bit data word. A scientific program run on the Model 50 can be run on the Model 44 up to 60 percent faster. Compatibility with other System/360 Models is maintained, although a limited instruction set is used (decimal arithmetic) and other variable-field-length instructions are excluded. The Model 44 contains, as standard equipment, a single-disk storage drive (capacity 1,088,000 bytes) with its control unit and the console printer/keyboard control unit.

Figure 1-7 shows the functional sections of the 2044. Note that the two functional sections ("input" and "output") in Figure 1-1 have been replaced in Figure 1-7 by the one functional section called channel. The channel is physically located in the central processing unit and its function is the control of the physically separated input and output devices that are connected to it. Thus, all the information from any input device passes through the channel before entering storage, and similarly, all information to an output device from storage must pass through the channel. Input information from a device such as a reader is fed into main storage. This input consists of instructions, specifying the type of operations to be performed, and data on which the operations are performed.

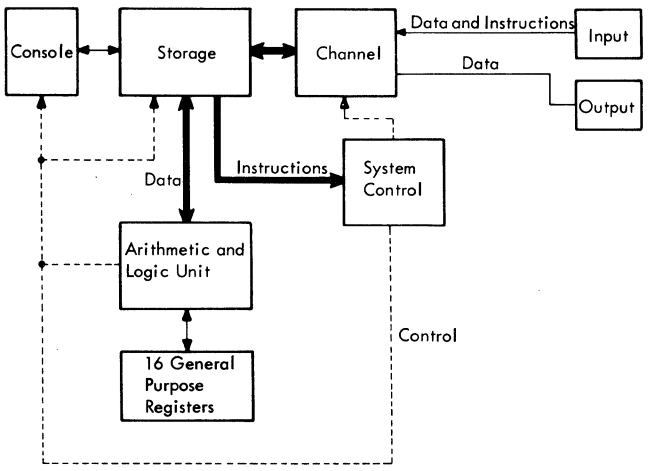


Figure 1-7. Functional Sections of IBM 2044

Core storage accepts both instructions and data, and retains them in a predetermined sequence until required for use during program execution. Each unit of information is stored in a numbered location in core storage, the number being called an address. In System/360, this address is a 24-bit binary-coded address. When required for an operation, the contents of a particular storage location are read out and sent to the appropriate functional unit.

Instructions read out from storage are decoded to provide the controls in the correct sequence to perform the operation as defined in the instruction. The system control functional unit thus controls all other functional units, ensuring their correct operation performing the specified instruction.

The data required in the operation specified by the instruction is then transferred to the arithmetic and logic section of the CPU, where the operation is performed and a result obtained. This result is usually written back into storage or into one of the general-purpose registers (GPR). The 2044 contains 16 GPR, each one word (32 bits) wide, which may contain either a result or data to be operated upon. Thus the GPR may be considered as extra storage.

The result may now be recorded at an output device (for example, a printer) by an instruction (for example, 'write printer'). Such an instruction causes the addressed location of the result in storage to be read out and transferred, via the channel, to the printer. Then, the printer is allowed to print the information.

For operator and customer engineer intervention, console-panel controls allow access to all functional parts of the processing unit.

Thus, a single instruction causes the computer to execute only one operation such as add, subtract, compare or print. Therefore, many instructions must be given to a computer to perform a given task, and this sequence of instructions is called a program.

All functional parts of a computer are interdependent. The control section cannot function without the control information (instructions) received from core storage. The core storage depends on an input device for its supply of instructions and data. The arithmetic and logic unit gets its supply of data from storage and the information on what to do with this data (control) comes from the control section. The functional parts of a computer work together to form a complete system that processes data.

MODEL 44 SYSTEM STRUCTURE

The basic structure of the Model 44 consists of a main storage, a CPU and one or more multiplexor channels with I/O devices attached to the channel(s) through control units. A program routine of operations (instructions) is loaded, in the required sequence with the relevant data, into core storage before the system begins to process data. The CPU then proceeds sequentially through the program routine, decoding each instruction and performing the

operation defined before progressing to the next instruction.

The CPU contains the facilities for addressing main storage, for fetching or storing information, for arithmetic and logical processing of data, for sequencing instructions in the required order and for initiating the communication between storage and external devices. The system control section provides the normal CPU control that guides the CPU through the operation necessary to execute the instructions. The CPU contains 16 GPR's for fixed-point operations and, optionally, four floating-point registers (FPR) for floating-point operations.

Instruction Word

- Instructions can be one or two halfwords long.
- Instructions must be located on integral halfword boundaries.

The length of a machine instruction can be one or two halfwords. It is related to the number of storage addresses necessary for performing the operation. An instruction consisting of only one halfword cannot cause any references to main storage. An instruction that is two halfwords long has one operand in main storage and therefore provides one storage address. All instructions must be located in storage on integral halfword boundaries. (An integral halfword boundary is any 24-bit address whose low-order bit is 0.)

Instruction Format

- The first byte of an instruction contains the op code.
- Up to 256 op codes are possible.
- Largest instruction set for Model 44 is 112 op codes.
- There are four instruction formats (RR, RX, RS and SI).

The first halfword of an instruction consists of two parts (Figure 1-8). In the first part, bits 0 to 7 are the op code. Provision is made for up to 256 op codes by using the eight-bit binary format. However, the largest instruction set presently available for the Model 44 consists of 112 op codes.

The second part of the first halfword, bits 8 to 15, may be used as a register specification, a mask, a byte of immediate data, or it may be ignored. Immediate data is held in the instruction format and used as one of the operands.

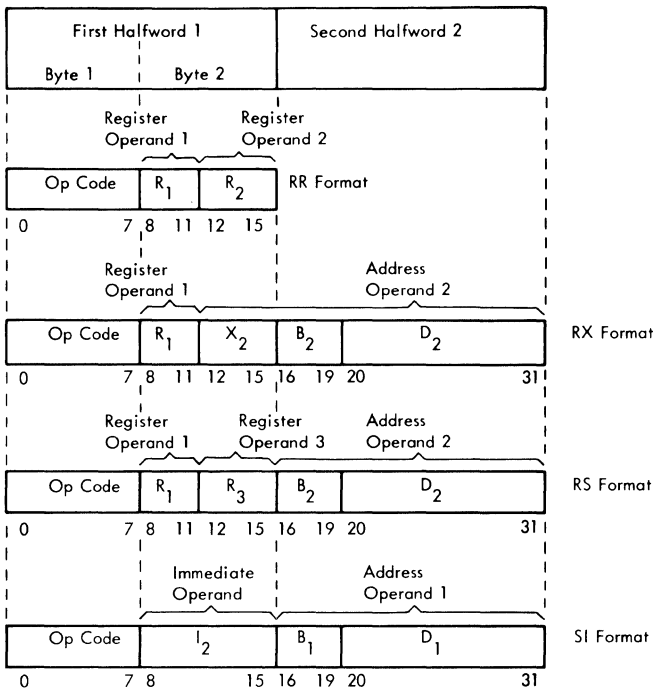


Figure 1-8. Four Basic Instruction Formats for Model 44

The second halfword, when present in the instruction, always has the same format. This format is a four-bit base-address register designation (B), followed by a 12-bit displacement address (D).

The four basic instruction formats are denoted by the format codes RR, RX, RS and SI. The format codes express, in general terms, the operation to be performed. RR denotes a register-to-register operation; RX, a register-to-indexed-storage operation; RS, a register-to-storage operation; and SI, a storage-and-immediate-operand operation.

For purposes of describing the execution of instructions, operands are designated as first and second operands or operand 1 and operand 2. These names refer to the manner in which the operands participate. The operand, to which a field in an instruction format applies, is generally denoted by the number following the code name of the field, for example, R1, B1, D2. The length and format of an instruction are specified by the first two bits of the op code. For example:

Bit Positions (0 and 1)	Instruction Length	Instruction Format
00	One halfword	RR
01	Two halfwords	RX
10	Two halfwords	RS or SI

The second byte is used either as two four-bit fields or as a single eight-bit field. This byte can contain the following information:

Four-bit operand register specification (R1, R2 or R3)

Four-bit index register specification (X2)

Four-bit mask (M1)

Eight-bit byte of immediate data (I2)

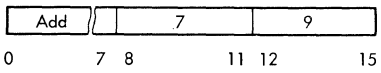
In some instructions a four-bit or the whole second byte of the first halfword is ignored.

The second halfword always has the same format:

Four-bit base register designator (B1 or B2), followed by a 12-bit displacement (D1 or D2).

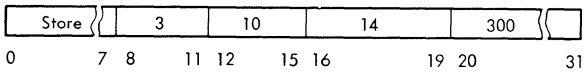
The following two examples show the RR and RX instructions and their related instruction formats:

RR Format



Execution of the Add instruction adds the contents of GPR 9 to the contents of GPR 7 and the sum of the addition is placed in GPR 7.

RX Format



Execution of the store instruction stores the contents of GPR 3 at a main storage location addressed by the sum of 300 and the low-order 24 bits of GPR14 and 10.

Address Generation

- Base address is a 24-bit number contained in the GPR specified by the instruction B field.
- Index address is a 24-bit number contained in the GPR specified by the instruction X field.
- Displacement address is a 12-bit number contained in the instruction.

For addressing purposes, operands can be grouped in three classes: explicitly addressed operands in main storage, immediate operands placed as part of the instruction stream in main storage, and operands located in the general-purpose or floating-point registers.

To permit the ready re-location of program segments and to provide for flexible specifications of input, output, and working areas, all instructions referring to main storage have the capacity of employing a full address.

The address used to refer to main storage is generated from binary numbers.

The base address (B) is a 24-bit number contained in a GPR specified by the program in the B field of the instruction. The B field is included in every address specification. The base address can be used as a means of static re-location of programs and data. In array-type calculations, it can specify the location of an array and, in record-type processing, it can identify the record. The base address provides for addressing the entire main storage. The base address may also be used for indexing purposes.

The index (X) is a 24-bit number contained in a GPR specified by the program in the X field of the instruction. It is included only in the address specified by the RX instruction format. The index can be used to provide the address of an element within an array. Thus, the RX format instructions permit double indexing.

The displacement (D) is a 12-bit number contained in the instruction format, and is included in every address computation. The displacement provides for relative addressing up to 4095 bytes beyond the element or base address. In array-type calculations the displacement may be used to specify one of the many items associated with an element. In the processing of records, the displacement can be used to identify items within a record.

In forming the address, the base address and index are treated as unsigned 24-bit positive binary triggers. The displacement is similarly treated as a 12-bit positive binary trigger. The base address, the index and the displacement are added as 24-bit binary numbers, ignoring overflow. Since every address includes a base, the sum is always 24 bits long. The address bits are numbered 8 to 31, corresponding to the numbering of the base address and index bits in the GPR.

The program may have zeros in the base address, index, or displacement fields. A zero is used to indicate the absence of the corresponding address component. A base or index of zero implies that a zero quantity is to be used in forming the address regardless of the contents of GPR 0. A displacement of zero has no special significance. Initialization, modification, and testing of base addresses and indexes can be carried out by fixed-point instructions, or by branch and link, branch on count, and branch on index instructions.

Instruction Types

- The standard instruction set provides fixed-point, logical, input/output, branching and status switching operations.
- The floating-point instruction set is an optional feature in Model 44 and includes all System/360 floating-point instructions.

The standard instruction set of the Model 44 consists of 64 instructions (Figure 1-9). The standard instruction set allows the system to perform five types of operation:

Fixed-point arithmetic
Logical
Branching
Status Switching
Input/Output

An optional instruction set for floating-point (Figure 1-10) adds 44 instructions to the standard set and permits the system to perform floating-point arithmetic in RR and RX format with both long and short operands.

INSTRUCTION	MNEMONICS			
	RR Format	RX Format	RS Format	SI Format
Add	AR	A	---	---
Add Halfword	---	AH	---	---
Add Logical	ALR	AL	---	---
AND	NR	N	---	NI
Branch and Link	BALR	BAL	---	---
Branch on Condition	BCR	BC	---	---
Branch on Count	BCTR	BCT	---	---
Compare	CR	C	---	---
Compare Halfword	---	CH	---	---
Compare Logical	CLR	CL	---	CLI
Diagnose	---	---	---	---
Divide	DR	D	---	---
Exclusive OR	XR	X	---	XI
Halt I/O	---	---	---	HIO
Insert Character	---	IC	---	---
Load	LR	L	---	---
Load Address	---	LA	---	---
Load and Test	LTR	---	---	---
Load Complement	LCT	---	---	---
Load Halfword	---	LH	---	---
Load Negative	LNR	---	---	---
Load Positive	LPR	---	---	---
Load PSW	---	---	---	LPSW
Move	---	---	---	MVI
Multiply	MR	M	---	---
Multiply Halfword	---	MH	---	---
OR	OR	O	---	OI
Set Program Mask	SPM	---	---	---
Set System Mask	---	---	---	SSM
Shift Left Double	---	---	SLDA	---
Shift Left Double Logical	---	---	SLDL	---
Shift Left Single	---	---	SLA	---
Shift Left Single Logical	---	---	SLL	---
Shift Right Double	---	---	SRDA	---
Shift Right Double Logical	---	---	SRDL	---
Shift Right Single	---	---	SRA	---
Shift Right Single Logical	---	---	SRL	---
Start I/O	---	---	---	SIO
Store	---	ST	---	---
Store Character	---	STC	---	---
Subtract	SR	S	---	---
Subtract Halfword	---	SH	---	---
Subtract Logical	SLR	SL	---	---
Supervisor Call	SVC	---	---	---
Test and Set	---	---	---	TS
Test Channel	---	---	---	TCH
Test I/O	---	---	---	TIO
Test Under Mask	---	---	---	TM

Figure 1-9. Standard Instruction Set for Model 44

INSTRUCTION	MNEMONICS			
	RR Format		RX Format	
	Long	Short	Long	Short
Add Normalized	ADR	AER	AD	AE
Add Un-normalized	AWR	AUR	AW	AU
Compare	CDR	CER	CD	CE
Divide	DDR	DER	DD	DE
Halve	HDR	HER	---	---
Load	LDR	LER	LD	LE
Load and Test	LTDR	LTER	---	---
Load Complement	LCDR	LCER	---	---
Load Negative	LNDR	LNER	---	---
Load Positive	LPDR	LPER	---	---
Multiply	MDR	MER	MD	ME
Store	---	---	STD	STE
Subtract Normalized	SDR	SER	SD	SE
Subtract Un-normalized	SWR	SUR	SW	SU

Figure 1-10. Floating Point Optional Instructions for Model 44

Logical operations are performed by instructions such as: AND, OR, Exclusive OR (EXOR), 'compare logical', 'test under mask', 'insert character', and 'store character'.

The input/output instructions are: 'start I/O', 'test I/O', 'halt I/O', and 'test channel'. These I/O instructions control the I/O devices and the data transfer of information between the CPU and the I/O devices.

The branching instructions provide a means of altering the normal program sequence, either directly and unconditionally, or as a condition dependent upon results obtained during computation. Prevailing machine conditions may also cause the normal program sequence to change when tested by a branch instruction.

Status-switching instructions include: 'load PSW', 'set program mask', 'set system mask', and 'supervisor call'. All these instructions alter the PSW and thus alter the system operation status.

Sequential Instruction Execution

- Instructions are executed sequentially under control of an instruction address register.
- The instruction address is updated as instructions are executed.

Normally, the operation of the CPU is controlled by instructions taken in sequence. An instruction is fetched from a location specified by the current instruction address. The instruction address is then increased by the number of bytes in the instruction to address the next instruction in sequence. The instruction is then executed, and the same steps are repeated, using the new value of the instruction address.

In principle, all halfwords of an instruction are fetched from storage after the preceding operation is completed and before execution of the current operation, even though physical storage-word size and overlap of instruction execution with storage access may cause actual instruction fetching to be different. Thus, it is possible to modify an instruction in storage by the immediately preceding instruction.

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

Branching

- Branching instructions permit out-of-sequence operations.
- Conditional branch instructions allow decision-making.

The normal sequence 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. These tasks are accomplished with branching instructions.

Subroutine linkage permits the introduction of a new instruction address and the preservation of the return address and associated information. In general, decision-making is provided symmetrically by the 'branch on condition' instruction. This instruction inspects a two-bit condition code that reflects the result of a majority of the arithmetic, logical, and I/O operations. Each of these operations can set the code in any of four states, and the conditional branch can specify any selection of these four states as the criterion for branching. For example, the condition code reflects such conditions as non-zero, operand 1 high, overflow, channel busy, zero, etc. Once set, the condition code remains unchanged until modified by an instruction that reflects a different condition code.

The two bits of the condition code provide for four possible settings: 0, 1, 2 and 3. (To avoid confusion and to relate them to machine operations, these settings are commonly referred to by their binary equivalents: 00, 01, 10 and 11.) The specific meaning of any setting is significant only to the operation setting the condition code.

Loop control can be performed by the conditional branch when it tests the outcome of address arithmetic and counting operations. For some particularly frequent combinations of arithmetic and tests, the instructions 'branch on count' and 'branch on index' are provided. These branches, being specialized, provide increased performances for these tasks.

Program Status Word

- Two words (PSW 1 and PSW 2).
- Controls instruction sequencing.
- Holds and indicates system status.

The program status word is a double word containing the information required for proper program execution. The PSW includes the instruction address, condition code, and other fields (Figure 1-11). In general, the PSW is used to control the instruction sequencing and to hold and indicate the status of the system in relation to the program being executed. The active, or controlling, PSW is called the current PSW. By storing the current PSW during an interrupt, the status of the processing unit can be preserved for subsequent inspection. By loading a new PSW, or part of a PSW, the state of the processing unit can be initialized or changed. For example, 'set system mask' instruction introduces a new eight-bit system mask field; the 'load PSW' introduces a completely new PSW; 'set program mask' introduces a new program mask field into the PSW. A summary of the purpose of each field follows.

System Mask: Bits 0 to 7 of PSW 1 are associated with the I/O channels, and external signals are specified in the following table. When the appropriate mask bit is a one, the source can interrupt the CPU. When a mask bit is a zero, the corresponding source cannot interrupt the CPU and the interrupt remains pending.

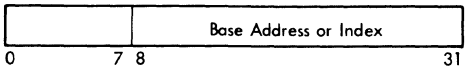
System Mask Bit	Interrupt Source
0	Multiplexor Channel 0
1	High Speed Multiplexor Channel 1
2	High Speed Multiplexor Channel 2
7	External

Protection Key: Bits 8 to 11 of PSW 1 form the CPU protection key.

ASCII (A): When bit 12 of PSW 1 is a one, the codes preferred for the extended ASCII code are generated for decimal results. When bit 12 of PSW 1 is a zero, the codes preferred for EBCDIC are generated. (See Figures 1-6 and 1-7.)

Machine-Check Mask (M): When bit 13 of PSW 1 is a one, any detected machine malfunction causes the machine-check interrupt routine to be initiated.

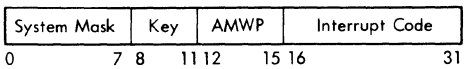
BASE AND INDEX REGISTERS



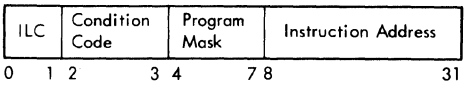
0-7 Ignored
8-31 Base Address or Index

PROGRAM STATUS WORD

(PSW 1)

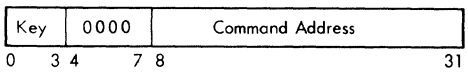


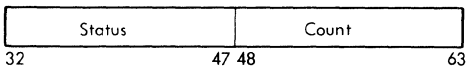
(PSW 2)



PSW 1		PSW 2	
0-7	System Mask	0-1	Instruction Length Code (ILC)
0	Channel 0	2-3	Condition Code
1	Channel 1	4-7	Program Mask
2	Channel 2	4	Fixed-Point Overflow Mask
3	Channel 3	5	Decimal Overflow Mask
4	Channel 4	6	Exponent Underflow Mask
5	Channel 5	7	Significance Mask
6	Channel 6	8-31	Instruction Address
7	External Mask		
8-11	Protection Key		
12	ASCII Mode (A)		
13	Machine Check Mask (M)		
14	Wait State (W)		
15	Problem State (P)		
16-31	Interrupt Code		

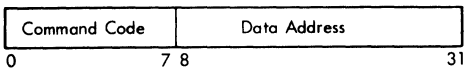
CHANNEL STATUS WORD

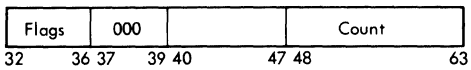




0-3	Protection key	39	Unit Exception
4-7	Zero	40	Program-Control Interrupt
8-31	Command Address	41	Incorrect Length
32-47	Status	42	Program Check
32	Attention	43	Protection Check
33	Status Modifier	44	Channel Data Check
34	Control Unit End	45	Channel Control Check
35	Busy	46	Interface Control Check
36	Channel End	47	Chaining Check
37	Device End	48-63	Count
38	Unit Check		

CHANNEL COMMAND WORD





0-7	Command Code	35	Skip Flag
8-31	Data Address	36	Program-Controlled Interruption Flag
32-36	Command Flags		
32	Chain Data Flag	37-39	Zero
33	Chain Command Flag	40-47	Ignored
34	Suppress Length Indication Flag	48-63	Count

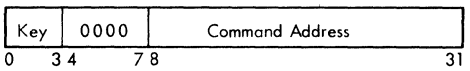
COMMAND CODE ASSIGNMENT

NAMES

Write	CD	CC	SLI	PCI	MMMM	MM01
Read	CD	CC	SLI SKIP	PCI	MMMM	MM10
Read Backward	CE	CC	SLI SKIP	PCI	MMMM	1100
Control	CD	CC	SLI	PCI	MMMM	MM11
Sense	CD	CC	SLI SKIP	PCI	MMMM	0100
Transfer in Channel					XXXX	1000

CD = Chain Data
CC = Chain Command
SLI = Suppress Length Indication
SKIP = Skip
PCI = Program-Control Interrupt

CHANNEL ADDRESS WORD



0-3 Protection
4-7 Zero
8-31 Command Address

Figure 1-11. Control PSW Format

When bit 13 of PSW 1 is a zero, the CPU is masked and any conditions that cause a machine check do not initiate the machine-check interrupt routine. The interrupt condition does not remain pending.

Wait State (W): When bit 14 of PSW 1 is a one, the CPU is in the wait state. When bit 14 of PSW 1 is a zero, the CPU is in the running state.

Problem State (P): When bit 15 of the PSW 1 is a one, the CPU is in the problem program state. When bit 15 of PSW 1 is a zero, the CPU is in the supervisor state.

Interrupt Code: Bits 16 to 31 of PSW 1 identify the cause of an I/O interrupt, program interrupt, supervisor call interrupt or external interrupt. The code is zero when a machine check interrupt occurs.

Instruction Length Code (ILC): The code in bits 0 to 1 of PSW 2 indicates the length, in halfwords, of the last-interpreted instruction when a program interrupt or supervisor call interrupt occurs. The code is unpredictable for I/O external interrupts or machine check interrupts.

Condition Code: Bits 2 and 3 of PSW 2 form the two-bit condition code, giving four possible settings. The meaning of the condition code setting for the different instruction types is given in Appendix G of IBM System/360 Principles of Operation, Form A22-6821.

Program Mask: Bits 4 to 7 of PSW 2 are the four program mask bits. Each bit is associated with a program exception as specified in Figure 1-11. When the mask bit is a one, the exception results in a program interrupt. When the mask bit is zero, no interrupt occurs.

Instruction Address: Bits 8 to 31 of PSW 2 are the instruction address. This address specifies the leftmost byte position of the next instruction.

Interrupts

- Five types of interrupt are: machine check, program check, supervisor call, external, I/O.
- Each interrupt type has two associated PSW locations in main storage: old and new.
- An interrupt is taken only when not masked for the interrupt type.
- Priority in acceptance of interrupts is: machine check (internal), program or supervisor call, external, machine check (external), I/O.
- Interrupt action consists of an exchange of PSW's.
- Interrupt handling is determined by program.

The interrupt system allows the CPU to change state as a result of conditions external to the system, in I/O units, or in the CPU itself. Five classes of interrupt conditions are possible: I/O, program, supervisor call, external, and machine check.

Each class has two related PSW's, called old and new, in unique main-storage locations (Figure 1-12). In all classes, an interrupt involves storing the current PSW in its old position and making the PSW at the new position the current PSW. The old PSW holds all necessary status information of the system existing at the time of the interrupt. If, at the conclusion of the interrupt routine, there is an instruction to make the old PSW the current PSW, the system is restored to the state prior to the interrupt and the interrupted routine continues.

Address		Length	
Decimal	Binary		
0	0000 0000	Doubleword	Initial Program Loading PSW
8	0000 1000	Doubleword	Initial Program Loading CCW1
16	0001 0000	Doubleword	Initial Program Loading CCW2
24	0001 1000	Doubleword	External Old PSW
32	0010 0000	Doubleword	Supervisor Call Old PSW
40	0010 1000	Doubleword	Program Old PSW
48	0011 0000	Doubleword	Machine Check Old PSW
56	0011 1000	Doubleword	Input/Output Old PSW
64	0100 0000	Doubleword	Channel Status Word
72	0100 1000	Word	Channel Address Word
76	0100 1100	Word	Unused
80	0101 0000	Word	Timer
84	0101 0100	Word	Unused
88	0101 1000	Doubleword	External New PSW
96	0110 0000	Doubleword	Supervisor Call New PSW
104	0110 1000	Doubleword	Program New PSW
112	0111 0000	Doubleword	Machine Check New PSW
120	0111 1000	Doubleword	Input/Output New PSW

Figure 1-12. Permanent Storage Assignments

Interrupts are taken only when the CPU is interruptible for the interrupt source. The system mask, program mask, and machine-check mask bits in the PSW may be used to mask certain interrupts. When masked off, an interrupt either remains pending or is ignored. The system mask may keep I/O and external interrupts pending, the program mask may cause four of the 15 program interrupts to be ignored, and the machine-check mask may cause machine-check interrupts to be ignored. Other interrupts cannot be masked off.

An interrupt always takes place after one instruction execution is finished and before a new instruction execution is started. However, the occurrence of an interrupt may affect the execution of the current instruction. To permit proper programmed action following an interrupt, the cause of the interrupt is identified and provision is made to locate the last-executed instruction.

Input/Output Interrupt

An I/O interrupt provides a means for the CPU to respond to conditions in the channels and I/O units.

An I/O interrupt can occur only when the related channel is not masked. The address of the channel and I/O unit involved are recorded in the old PSW. Further information concerning the I/O action is preserved in the Channel Status Word (CSW) that is stored during the interrupt.

Program Interrupt

Unusual conditions encountered in a program create program interrupts. These conditions include

incorrect operands and operand specifications as well as exceptional results. The interrupt code of the PSW identifies the interrupt cause. This interrupt cause is recorded in binary-coded form in bit positions 28 to 31 of the old PSW 1. In System/360 there are 15 possible sources of a program interrupt. Of these, 11 are applicable to the Model 44. The significance of each interrupt follows.

Invalid Operation: Occurs when an operation code is unassigned or the operation code is not available on the particular model.

Privileged Operation: Occurs when the CPU is operating in the problem program state and a privileged instruction is given.

Execute: Occurs when the instruction designated by an execute instruction is another execute instruction.

Addressing: Occurs when an address specifies any part of data, an instruction, or a control word outside the available main storage area for the particular installation.

Specification: Occurs when

1. A data word, instruction word, or control word address does not specify an integral boundary for the unit of information. A unit of information is 32 or 64 bits for floating-point data, 16 bits for instructions, and 64 bits for control words.
2. The R1 field of an instruction specifies an odd register address for a pair of general-purpose registers that contain a 64-bit operand.
3. A floating-point register address other than 0, 2, 4 or 6 is specified.
4. Bits 8 to 11 of PSW 1 are non-zero; the protection key must be zero when the protection feature is not installed.

Fixed-Point Overflow: Occurs when a high-order carry is generated or high-order significant bits are lost in fixed-point add, subtract, shift, or sign-control operations.

Fixed-Point Divide: Occurs when a quotient exceeds the register size in fixed-point divide. For division by zero, the instruction is suppressed.

Exponent Overflow: Occurs in floating-point add, subtract, multiply and divide when the resultant exponent exceeds 127.

Exponent Underflow: Occurs in floating-point add, subtract, multiply and divide when the resultant exponent is less than zero. The operation is completed by making the result a true zero.

Significance: Occurs when the result is an all-zero fraction in floating-point add or subtract.

Floating-Point Divide: Occurs when an attempt is made to divide by a floating-point number with an all-zero fraction.

Supervisor-Call Interrupt

This interrupt occurs as a result of execution of the 'supervisor-call' instruction. Eight bits from the instruction format are placed in the interrupt code of the old PSW, permitting a message to be associated with the interrupt. A major use for the 'supervisor-call' instruction is to switch from the problem state to the supervisor state. This interrupt may also be used for other modes of status switching.

External Interrupt

The external interrupt provides the means by which the CPU responds to signals from the interrupt key on the system control panel, from the timer, and from the external signals. An external interrupt can occur only when the system mask bit 7 is a one. The source of the interrupt is identified by the interrupt code in bits 24 to 31 of PSW 1. The remainder of the interrupt code, bits 16 to 23, is made zero.

Machine-Check Interrupt

The occurrence of a machine check (if not masked off) terminates the current instruction, and causes the machine-check interrupt. A machine check cannot be caused by invalid data or instructions.

Note that differentiation is made between an internal machine-check interrupt, which occurs as a result of CPU errors, and an external machine-check interrupt, which occurs as a result of channel errors. However, there is only one pair of main storage fixed locations (old and new) for both types of machine check.

Priority of Interrupts

During the execution of an instruction, several interrupt requests may occur simultaneously which are accepted in the following predetermined order:

Internal Machine Check (CPU)
Program or Supervisor Call
External
External Machine Check (Channels)
Input/Output

The program and supervisor-call interrupts are mutually exclusive and cannot occur at the same time.

When more than one interrupt-cause requests service, the action consists of storing the old PSW and fetching the new PSW belonging to the interrupt that is taken first. This new PSW subsequently is stored without any instruction execution and the next interrupt PSW is fetched. This process continues until no more interrupts are to be serviced. When the last interrupt request has been serviced, instruction execution is resumed, using the PSW last fetched. The order of execution for the interrupt subroutines is, therefore, the reverse of the order in which PSW's are fetched.

Thus, the most important interrupts (I/O, external machine check, external, program, or supervisor call) are actually serviced first. The internal machine check, when it occurs, does not allow any other interrupts to be taken.

Program States

Over-all CPU status is determined by four alternative types of program state, each of which can be changed independently to its opposite, and most of which are indicated by a bit or bits in the PSW. The program states are named: 'stopped or operating', 'running or waiting', 'masked or interruptible', and 'supervisor or problem'. These states differ in the way they affect the CPU functions and the manner in which their status is indicated and switched. All program states are independent of each other in their functions, indication, and status switching.

Stopped or Operating State: The stopped state is entered and left by manual procedure. Instructions are not executed, interrupts are not accepted, and the timer is not updated. In the operating state, the CPU is capable of executing instructions and being interrupted.

Running or Waiting State: In the running state, instruction fetching execution proceeds in the normal manner. The waiting state is normally entered by the program to await an interrupt, for example, an I/O interrupt or operator intervention from the console. In the waiting state, no instructions are processed, the timer is updated, and I/O and external interrupts are accepted unless masked. Running or waiting state is determined by the setting of bit 14 in PSW 1.

Masked or Interruptible State: The CPU may be interruptible or masked for the system, program, and machine interrupts. When the CPU is interruptible for a class of interrupts, these interrupts are accepted. When the CPU is masked, the system interrupts remain pending. The interruptible states of the CPU are changed by changing the mask bits of the PSW.

Supervisor or Problem State: In the problem state, all I/O instructions and a group of control instructions are invalid. In the supervisor state, all instructions are valid. The choice of problem or supervisor state is determined by bit 15 of PSW 1.

Input/Output

- All information transfer in input/output operations is between main storage and the I/O device.
- Control of each I/O device attached to a system is provided by control units.
- Control units respond to a standard set of signals from the channel.
- The channel transmits and receives the standard set of signals over an I/O interface.
- The channel has access to main storage and holds the control information necessary for an I/O operation.

Input/output operations involve the transfer of information to or from main storage and an I/O device. I/O devices include such equipment as card reader/punches, magnetic tape units, disk storage, drum storage, typewriter-keyboard devices, printers, teleprocessing devices, and process control equipment.

Many I/O devices function with an external document, such as punched cards or a reel of magnetic tape. Some I/O devices handle only electrical signals, such as those found in process-control networks. In either case, I/O device operation is regulated by a control unit. The control-unit function may be housed with the I/O device, as is the case with a printer, or a separate control unit may be used. In all cases, the control-unit function provides the logical and buffering capabilities necessary to operate the associated I/O device. From the programming point of view, most control-unit functions merge with I/O device functions although each control unit functions only with the I/O device for which it is designed.

To enable the CPU to control a wide variety of I/O devices, all control units are designed to respond to a standard set of signals from the channel. This control-unit-to-channel connection is called the I/O interface. It enables the CPU to handle all I/O operations with only four instructions.

Channels

- A channel can be regarded as an independent computer for processing I/O instructions.
- The CPU program specifies only the I/O operation to be performed; execution of the operation is then performed solely by the channel.
- Channels may be physically separated from the CPU or they may share the CPU facilities.

Channels are the connecting link between the CPU and the I/O interface and thus between main storage and control units. Each channel has facilities for:

Accepting I/O instructions from the CPU
Addressing devices specified by I/O instructions
Fetching channel-control information from main storage
Decoding control information
Testing control information for validity
Executing control information
Providing control signals to the I/O interface
Accepting control-response signals from the I/O interface
Buffering data transfers
Checking parity of bytes transferred
Counting the number of bytes transferred
Accepting status information from I/O devices
Maintaining channel-status information
Sending requested status information to main storage
Sequencing interrupt requests from I/O devices
Signalling interrupts to the CPU

A channel may be an independent unit, complete with necessary logical and storage capabilities, or it may share CPU facilities and be physically integrated with the CPU as in the 2044. In either case, channel functions are identical.

System/360 has two types of channels: multiplexor and selector. The channel facility necessary to sustain an operation with an I/O device is called a subchannel. The selector channel has one subchannel; the multiplexor channel has multiple subchannels.

Channels have two modes of operation: burst and byte. (Byte mode is sometimes referred to as 'multiplex mode'.)

In burst mode, all channel facilities are monopolized for the duration of data transfer to or from a particular I/O device. The selector channel functions only in burst mode.

The multiplexor channel functions in both burst mode and in byte mode. In the byte mode, the multiplexor channel sustains simultaneous I/O operations on several subchannels. Bytes of data are interleaved together and then routed to or from the selected I/O devices and to or from the desired locations in main storage.

The Model 44 has only the multiplexor channel and this is described later under the heading "Model 44 Channels."

Input/Output Instructions

The System/360 uses only four I/O instructions:

Start I/O
Test Channel
Test I/O
Halt I/O

Input/output instructions can be executed only while the CPU is in the supervisor state.

All I/O operations are initiated by the start I/O instruction. If the channel facilities are free, the start I/O is accepted and the CPU continues its program. The channel independently selects the I/O device specified by the instruction.

Start I/O: Initiates an I/O operation. The address part of the instruction specifies the channel and the I/O device.

Test Channel: Sets the condition code in the PSW to indicate the state of the channel addressed by the instruction. The condition code then indicates channel available, interrupt condition in channel, channel working, or channel not operational.

Test I/O: Sets the condition code in the PSW to indicate the state of the addressed channel, subchannel, and I/O device. A CSW is stored in main storage location 40 hex.

Halt I/O: Terminates a channel operation.

Channel Address Word

Successful execution of a start I/O causes the channel to fetch a Channel Address Word (CAW) from main storage location 48 hex. The CAW specifies the byte location in main storage where the channel program begins.

Figure 1-11 shows the format for the CAW. Bits 0 to 3 specify the storage protection key that will apply to the I/O operation. (In Model 44 bits 0 to 3 are ignored since the protection feature is not yet available.) Bits 4 to 7 must contain zeros. Bits 8 to 31 specify the location of the first Channel Command Word (CCW).

Channel Command Word

The byte location specified by the CAW is the first of eight bytes of information that the channel fetches

from main storage. These 64 bits of information are called a CCW.

One or more CCW's make up the channel program that directs channel operations. If more than one CCW is to be used, each one points to the next CCW to be fetched, except for the last one, which identifies itself as the last in the chain.

Six channel commands are provided (Figure 1-11)

Read	Control
Write	Sense
Read Backward	Transfer In Channel

Read

The read command causes a read operation from the selected I/O device and defines the area in main storage to be used.

Write

The write command causes a write operation on the selected I/O device and defines the data in main storage to be written.

Read Backward

The read backward command causes a read operation in which the external data medium is moved in a backward direction if so designed. Bytes read backward are placed in descending main storage locations.

Control

The control command contains information used to control the selected I/O device. This control information is called an order. Orders are peculiar to the particular I/O device in use; orders can specify such functions as rewinding a tape unit, searching for a particular track in disk storage, or line skipping on a printer.

Sense

The sense command specifies the first main storage location to which status information is transferred from the selected control unit. This sense data may be one or more bytes long. It provides detailed information concerning the selected I/O device, such as a stacker-full condition of a card reader or a file-protected condition of a reel of magnetic tape on a tape unit. Sense data have a significance peculiar to the I/O device involved.

Transfer In Channel

The transfer-in-channel command specifies the location of the next CCW to be used by the channel, whenever the programmer wants to break the existing chain of CCW's and cause the channel to begin fetching a new chain of CCW's from a different area in main storage.

External documents, such as punched cards or magnetic tape, may carry CCW's that the channel can use to govern reading of the external document being read. These are known as self-describing records.

Input/Output Termination

Input/output operations normally terminate with the 'device end' signal and 'channel end' conditions, and an interrupt signal to the CPU.

A command can be rejected during execution of a start I/O, however, by a busy condition, program check, etc. The rejection of the command is indicated by the condition code in the PSW, and the details of the conditions that precluded initiation of the I/O operation are provided in the CSW stored when the command is rejected.

Channel Status Word

The CSW at main storage location 40 hex provides information about the termination of an I/O operation. It is formed or reformed by a start I/O or test I/O instruction, or by an I/O interrupt (Figure 1-11).

Input/Output Interrupts

Input/output interrupts are caused by termination of an I/O operation or by operator intervention at the I/O device. Input/output interrupts enable the CPU to provide appropriate programmed response to conditions that occur in the I/O devices or channels.

Input/output interrupts have two priority sequences, one for the I/O devices attached to a channel, and another for channel interrupts. A channel establishes interrupt priority for its associated I/O devices before initiating an I/O interrupt signal to the CPU. Conditions responsible for I/O interrupt requests are preserved in the I/O devices or channels until they are accepted by the CPU.

Model 44 Channels

- The two types of channel available are the multiplexor channel and the high speed multiplexor channel.
- At least one channel is mandatory with each system, but the channel type is optional.
- Subchannels of the multiplexor channel are in the extension of main storage.
- SLT registers form the subchannel for the high speed multiplexor channel.

The Model 44 uses the System/360 I/O interface. All System/360 I/O instructions, and channel commands and functions are provided. All I/O operations overlap with processing.

The I/O control section consists of two parts, both physically integrated with the CPU: the multiplexor (MPX) channel (commonly referred to as MPX 0), and the one or two high speed multiplexor (HSMPX) channels. At least one channel must be installed, but the channel types are optional.

In the MPX channel, the information associated with a subchannel is kept in the extension of the CPU main storage; in the HSMPX channels, this information is held in the SLT registers associated with the high speed channels. One set of SLT registers is provided for each HSMPX subchannel. (See "Subchannels.")

On any given channel operating in byte mode, the number of I/O devices that can transfer data simultaneously depends on:

1. The speed of the devices;
2. The maximum aggregate data transfer rate of the channel;
3. The number of subchannels constituting the channel;
4. The number and type of I/O devices attached (via control units) to the channel.

Rates for individual channels are reduced by operations on other channels as well as by the use of data chaining, the Transfer In Channel (TIC) command, and Program Control Interrupts (PCI).

Subchannels

In the case of the MPX channel, the number of subchannels depends on the capacity of main storage.

Bytes	Subchannels
32,768 (32K)	32
65,536 (64K)	64
131,072 (128K)	64

Each HSMPX channel equips the Model 44 with at least one high-speed subchannel; up to three additional high-speed subchannels can be installed in each HSMPX channel. With only one subchannel the HSMPX channel is virtually a selector channel. Thus, the system maximum is eight HSMPX subchannels and 64 MPX subchannels.

Control Units and Devices

Up to eight control units can be attached to the MPX channel, and two control units can be attached for each HSMPX subchannel. Thus, the system maximum is 24 control units: 16 attached to high-speed subchannels (eight on each HSMPX channel) and eight attached to the MPX channel. In the following examples the console printer-keyboard and single-disk storage drive(s) require two control unit positions and two subchannels on the MPX channel, or two control unit positions and one subchannel on the HSMPX channel.

Example 1: If 63 MPX subchannels are in use for I/O devices or communication lines that are controlled via seven control units, the subchannel 64 is still available to attach a control unit (such as a tape control).

Example 2: An IBM 2702 Transmission Control, to control 31 lines would tie up 31 MPX subchannels. This would leave up to 33 subchannels or seven control units, whichever limit is reached first, for the use of other I/O devices attached via the MPX channel.

In either case, the HSMPX channel capabilities remain available: up to eight control units for I/O devices per HSMPX channel. A system maximum of 72 devices (one for every MPX and HSMPX subchannel) may, if permitted by device and channel speeds, transfer data simultaneously.

Shared Subchannels

As in other models of the System/360, up to eight subchannels on the MPX channel can be shared by I/O devices. The capability for sharing provides for the control of as many as 16 devices, such as tape units, via each shared subchannel. Only one device at a time may be involved in data transfer on a shared subchannel, but, in the meantime, other devices on the same subchannel can be engaged in free-running operations, such as tape rewinding. Every HSMPX subchannel has the same shared and free-running capabilities as the eight MPX subchannels that can be shared.

Addressing

The addresses of shared and non-shared subchannels and devices are as detailed in the "Input/Output Device Addressing" section of IBM System/360 Principles of Operation, Form A22-6821.

Of the 11 bits of addressing that specify the channel, subchannel, and device to be used in an I/O operation, the three high-order bits are assigned to specify the channel as follows:

000 - MPX channel 0
001 - HSMPX channel 1
010 - HSMPX channel 2

The eight low-order bits specify the subchannel and the device:

1. For shared MPX subchannels, indicated by 1 in the high-order position: 1 xxx yyyy (xxx specifies one of the eight subchannels that can be shared; yyyy specifies one of the 16 devices using subchannel and control unit xxx).
2. For non-shared MPX subchannels, indicated by 0 in the high-order position: 0 xxx xxxx (xxx xxxx specifies one of the non-shared subchannels, and its control unit and device, up to 64 subchannels, depending on main storage size, less the number of subchannels being used as shared subchannels).
3. For HSMPX subchannels, all having the shared capability: 1 hhc xxxx (hh specifies one of the four HSMPX subchannels on that channel, c specifies one of the two control units on the subchannel, and xxxx specifies one of up to 16 devices).

Burst Mode

In burst mode, either type of multiplexor channel (MPX or HSMPX) operates as a selector channel that has only one subchannel. Only one device at a time communicates with the CPU via the particular channel, but communication is at a higher maximum data rate than in the byte mode and (in the case of the MPX channel) the interference with CPU operation, per byte of I/O data transferred, is reduced.

Byte Mode

In byte mode, each channel can concurrently sustain one data-transferring I/O operation per subchannel. provided that the aggregate data rate does not exceed the capacity of the channel. Thus, the MPX channel may be capable of simultaneous communication with as many I/O devices as can be attached to it (up to 64, via no more than eight control units). Also each HSMPX channel may be capable of simultaneous communication with as many I/O devices as there are subchannels in that channel (up to four).

MODEL 44 DATA FLOW AND CONTROL

Both the main storage section and the arithmetic and logic section are controlled by signals generated in the system control section (Figure 1-13). These three sections are the principal components of the CPU.

Sixteen general-purpose registers are also provided for temporary storage of information. These are located in the extension storage or accommodated in hardware registers if the accelerator feature is installed.

Input/output communication with the CPU is achieved through channels which provide their own controls, data handling and means of addressing. (See Principles of Operation, Channels, Form Y33-0003.)

The console provides the operator or customer engineer with a means of addressing and loading data into main storage.

A more detailed data flow is shown in the FEMD IBM System/360 Model 44, Volume 2, Form Y33-0008. Main storage general-purpose registers, Arithmetic and Logic Section (ALS) registers and all data paths are one fullword (32 bits) wide. The CPU basic cycle-time is 250 nanoseconds. This basic cycle-time is made possible by the extensive use of high-speed circuitry. In addition, machine speed is helped by the physical layout which keeps connection paths as short as possible.

Main and Extension Storage and Associated Components

Core Storage

The core storage array consists of drivers, amplifiers, etc. The area referred to as main storage is the part that is addressable by the programmer. Extension storage is the part of the storage that is used for temporary storage of channel control information, for the least-significant halves of the FPR's (long precision) and for the sixteen GPR's when the accelerator feature is not installed. Both main storage and extension storage are physically packed in the same core array. The complete storage is in two sections, each two bytes wide. A storage access always reads out both sections, that is four bytes or one fullword.

Storage size ranges from 32K bytes to 128K bytes packaged within the main frame. A further 128K bytes may be held in a 'blister', or attachment, to the main frame.

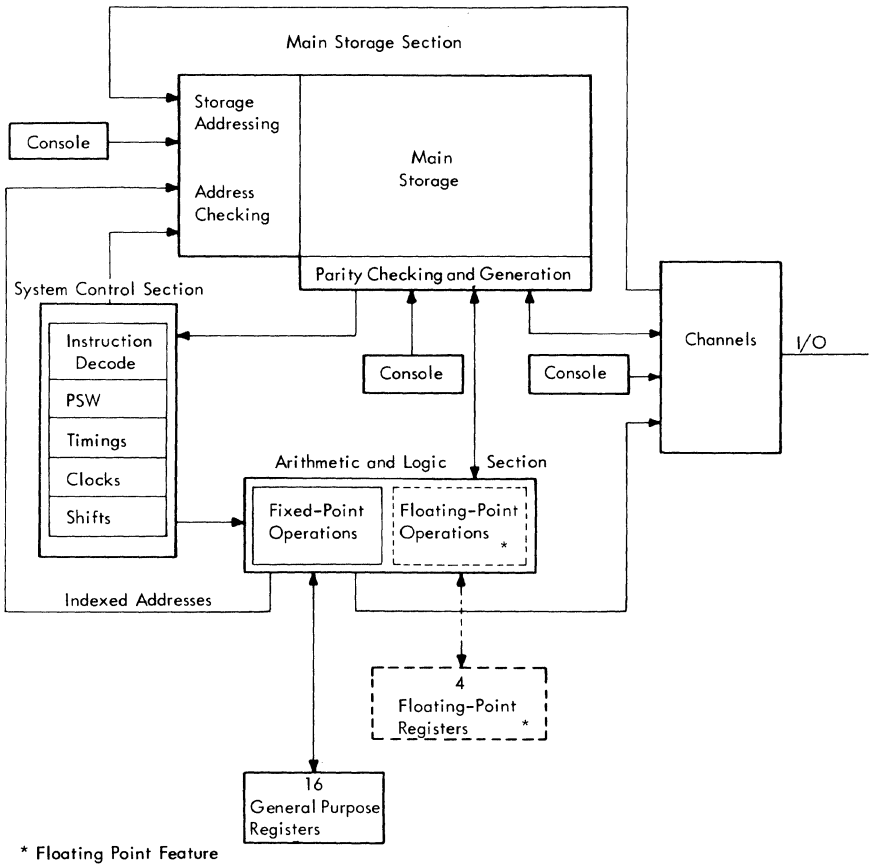


Figure 1-13. Simplified Data Flow

Storage Address Registers

Each of the two sections of storage has a Storage Address Register (SAR 1 or SAR 2) associated with it. Each SAR may be set from the 'instruction counter', the console, the 'address generate' circuits (for operand addresses) and the channels.

The addresses are full word addresses (the two least-significant bits are not used). In order to handle halfword instructions the contents of SAR 1 and SAR 2 least-significant positions may differ.

IC + 2

When it is required to address an instruction at a halfword boundary, it is necessary to address one word with SAR 2, to obtain the least-significant two bytes of a fullword (although these are the most-significant two bytes of the instruction), and to address the word at the next higher address with SAR 1 (since this contains the least-significant two bytes of the instruction if it is a full word instruction).

This manipulation of the address is carried out by the unit call IC + 2, the output of which goes to SAR 1.

The actual manipulation of the two halves of the word, so that they are properly aligned in the data flow, is accomplished by the true/criss-cross unit.

Storage Data Register

All data read into or out of main or extension storage passes through the Storage Data Register (SDR). Data from storage is parity checked at the SDR and correct parity is generated for data to be stored.

The two bytes from each half of storage are read out of their respective halves of the SDR to form a full word. Similarly, two bytes from each half of the SDR may be written back into their respective storage sections.

Communication of data between CPU and channels is made via the SDR.

True/Criss-Cross

The two halves of the SDR (the most-significant two bytes and the least-significant two bytes may be in the wrong position for processing. Interchange of the positions of the two halves (if required) is carried out by the true/criss-cross controlled by bit 30

of the address (this is one of the bits not used in the SAR's).

The true/criss-cross may also be used to block the propagation of one of the halfwords.

Arithmetic and Logic Section

The main flow of data, after leaving the true/criss-cross, is to the ABC funnel and then to the A, B and C registers. Data always enters the B and C registers unless the registers are specifically deconditioned. Data enters the A register only if it is specifically conditioned. Entry to the BX register is through the B register. Each register is 32 bits wide.

The B register is the most important register of the ALS. Arithmetic and logic operations are performed between the B register and the A register. The C register is used as a back-up to the B register. The BX register is used with the B register when a 64-bit register is required (for multiply and divide operations), and also as an auxiliary address register. A path is provided from the BX register direct to the instruction counter in PSW 2.

There are two methods of data entry to the registers. These are the normal additive method, where a 1 overwritten on an existing 1 gives an output of 1, and the binary method where a 1 overwritten on an existing 1 turns the trigger off to give a 0.

The A register always uses the additive method. The B register uses either method depending on the source of the data. The C register uses the binary method, and the BX register the normal method, the data coming from the B register.

The arithmetic functions are achieved by using the register capabilities outlined previously, together with the circuitry to provide the correct carries. The A and C register contents are analysed in the Carry Look Ahead (CLA) unit, and the appropriate carries entered into the B register through the ABC funnel.

System Control Section

- Three registers are used for control functions:
  1. PSW
  2. Shift
  3. Instruction

- Timing is achieved by five clocks:
  1. Read/Write
  2. Compute
  3. I/O
  4. Interface
  5. Basic (used for all functions)

- Cycle and sequence controls are used respectively for the read/write and compute clock to define the application for a particular clock cycle.

Control Registers

The PSW contains information for correct program operation. The part of the PSW in the latches containing bits 0 to 31 is known as PSW 1, and the part in the latches containing bits 32 to 63 as PSW 2.

The PSW is loaded from one of 11 positions in main storage depending on the reason for the change. For example, location 000 supplies the Initial Program Loading (IPL) PSW and location 030 hex supplies the 'machine check old PSW'.

The shift counter register holds the number of shifts that are required to be carried out by the B and BX registers. As each shift is carried out, the value held in the counter is decreased by an amount proportional to the number of places shifted. The counter is loaded via the B register.

One of the primary controls of the 2044 is the output from the 'instruction decode' area. The instruction register contains the operation code and the addresses of registers (GPR's) in which the operands are contained, or which contain data to form a main storage address. The instruction register is loaded from main storage via the SDR and the true/criss-cross. The register is made up as follows:

op code (8 bits), Ra, Rb and Rc (4 bits each).

Timing

A computer moves and operates on data according to a predetermined pattern. The sequence in which these patterns are performed is decided by programming. The sequence of events inside each pattern is decided by logical decisions and timing pulses.

It must be possible to define each individual pulse during the performance of an operation.

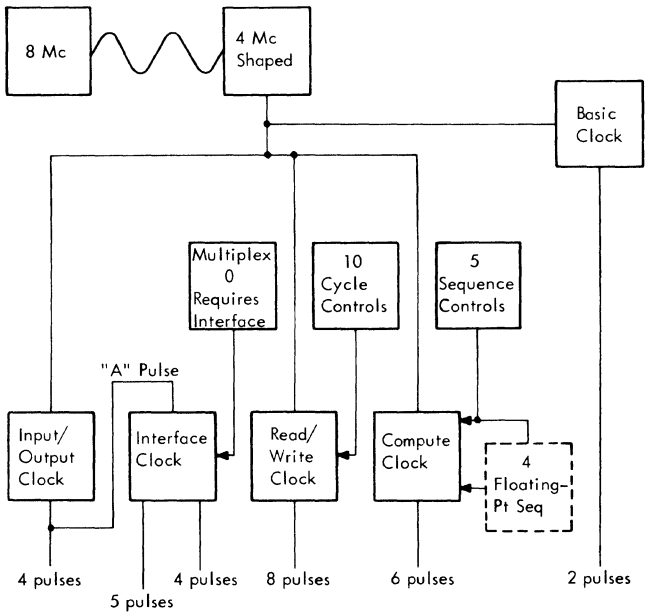


Figure 1-14. Block Diagram of Timing

Pulses are generated by a free-running oscillator, the output of which is put through circuitry to shape and time the pulses to the requirements of the computer.

These shaped pulses are used to drive the ring of latches or triggers which constitute a clock. The time required for all latches or triggers in a ring to be switched on and off once is defined as a clock cycle. The output of one of the latches or triggers is a specific clock pulse.

An eight-megacycle oscillator provides pulses that are shaped and divided to produce square-wave four-megacycle pulses (Figure 1-14). These pulses of 250 nanoseconds cycle time are the clock-driving pulses from which all clocks are derived.

Produced directly from these pulses are the basic clock pulses CP 1 and CP 2. These pulses are always available if power is good. The other four clocks, read/write, compute, I/O and interface, are normally used to gate or define the use of these clock pulses. The read/write clock is used for all main or extension storage accesses. The compute clock is used for register-to-register functions and register display for all CPU registers. All channels use the I/O clock for channel registers. The interface clock is used by the multplexor channel 0 only.

The I/O clock is continuously running once power is good. The interface clock is started whenever multiplexor channel 0 uses the interface either for input or output. To control the read/write clock there are ten cycle controls, one for each type of storage access.

The compute clock is similarly controlled by five sequence controls. During each sequence a similar set of registers is used. Each of these five sequences may be further defined by four sequences designed specially for floating-point operations.

ARITHMETIC AND LOGIC SECTION AND REGISTERS

To avoid duplication, and for simplicity, this section of Functional Units is preceded with an explanation of the Arithmetic and Logic Section (ALS). The individual components performing the arithmetic and logic function are described after the introduction. For further details on timing and complete operations, refer to FEMD IBM System/360 Model 44, Volume 2, Form Y33-0008 and Principles of Operation - Processing Unit, Form Y33-0002.

INTRODUCTION

- The ALS does not physically exist as a discrete unit.
- All components and data paths are a full word wide.
- No parity bits are carried around the main data flow.
- The analysis of the operation (op) code determines the correct handling of operands.
- Additions are performed between A and B registers; the results are developed in the B register.
- Subtraction is the addition of the two's complement of the second operand.
- Multiply and divide functions are combinations of successive addition or subtraction and shifting.

Add operations, including development of the effective address and timer updating, are achieved by using several separate components as shown by the ALS data flow in Figure 2-1. Each of these components has established functions as described later.

Most ALS operations are essentially variations of the basic addition principle.

Addition

Principles of Addition

When an addition is performed with pencil and paper, the carries are mentally developed as the calculation progresses from right to left. See Example 1.

Example 1. Add hexadecimal (hex) 15 + 39 = 4E

11	1	← Carries
0001	0101	
0011	1001	
0100	1110	

The machine uses a different method for the same operation. Initially, both operands are Exclusive OR'ed (EXOR) in the B and C registers (A in Example 2). Then, carries which are decided by the Carry Look-Ahead (CLA) circuits as shown at B in Example 2, are EXOR'ed with the preceding intermediate result in the B register. The final sum (C in Example 2) is obtained in the B register after completion of these two steps.

Example 2. Add hex 15 + 39 = 4E

	0001	0101
	0011	1001
A	0010	1100
B	0110	0010
C	0100	1110

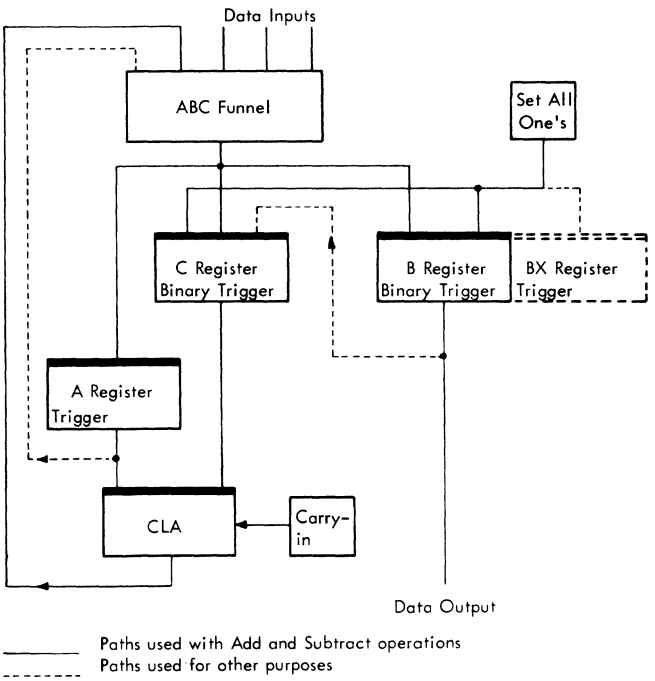


Figure 2-1. Simplified ALS Data Flow

Add Operation

Figure 2-2 is a representation of the machine operation of adding 15 hex and 39 hex.

The A, B and C registers are cleared. Operand 2 enters the B and C registers via the ABC funnel and subsequently operand 1 enters the A, B and C registers. When the fetching of both operands is completed the A register contains operand 1, and the B and C registers (because of their EXOR properties) contain the half sum of the operands. This half sum is an initial addition of the two operands with no provision made for the carries.

The final sum is produced by EXOR'ing the half sum of the operands with the CLA output in the B register. During this period the A register is not conditioned and the C register is deconditioned to prevent feedback to the CLA.

The CLA circuits are fed from the A and C registers. The principles of carry generation are the same as those adopted for ordinary pencil and paper addition operation. If it is considered that the situation is the same as that after the first step in Example 2 (previously described), the process is as follows:

Operand 1 is in the A register, while the half sum is in the C register (which reflects the state of the B register).

If there is a bit in any A register position and there is no bit in the corresponding position of the C register, it indicates that two bits have been entered in that position of the C and B registers and a carry must be generated to the next high-order position.

If there is already a bit in the next high-order position of the C (and B) register, the carry must be propagated upwards to the next high-order position, and so on.

This process represents the two possible ways of forming a carry. The carry bit changes the state of the position into which it enters, since it enters through the ABC funnel end of B register and is EXOR'ed.

The final sum is obtained in the B register and gated to the appropriate addressed unit, i.e. Storage Data Register (SDR), Storage-Address Register (SAR), General-Purpose Register (GPR) or Floating-Point Register (FPR). Since the B register is the communication link between the ALS and the remainder of the CPU, no further handling is needed.

During effective address computation, the displacement can be considered as operand 2 and the contents of the base or index register as operand 1. In the case of double indexing, the first sum developed (with the index register) can be considered as operand 2 for the second sum, and the base register contents as operand 1.

Before the second addition takes place the C register (which was deconditioned during the CLA gating) does not reflect the content of the B register. This condition is corrected by gating the B register into the C register, as shown by the dotted line between the B and C registers in Figure 2-1.

Subtraction

Principles of Subtraction

Since the ALS cannot perform a subtraction operation, subtraction is achieved by adding the two's complement of operand 2 to operand 1.

Two's complementing is achieved by inverting each bit of the operand and by adding one in the least-significant position.

Example 3.

Hex 98 - 37 = 61			Hex 98 + (-37) = 61	
1001	1000	equivalent	1001	1000
-0011	0111	to	+1100	1001
=0110	0001		carry ← 0110	0001

	A Register OR Triggers	B Register EXOR Binary Triggers	C Register EXOR Binary Triggers
1 Reset A, B and C registers	0000 0000	0000 0000	0000 0000
2 Gate operand 2 in B and C registers	—	0011 1001	0011 1001
3 Gate operand 1 in A, B and C registers	0001 0101	(0001 0101)	(0001 0101)
Contents of the registers are now	0001 0101	0010 1100	0010 1100
4 Gate CLA output to B register	—	(0110 0010)	—
Contents of the registers are now The result is obtained in B register	0001 0101	0100 1110	0010 1100

Figure 2-2. Add Example (Hex 15 + 39 = 4E)

Subtract Operation

Figure 2-3 is a representation of the machine operation of subtracting 37 hex from 98 hex.

The A register is reset before the operand is fetched and the B and C registers are set to all ones. Operand 2 is EXOR'ed into the B and C registers via the ABC funnel; at this point the B and C registers each contain the one's complement of operand 2. Operand 1 is then gated via the ABC funnel into the A, B and C registers.

The contents of the A and C registers are analyzed in the CLA using the method described for the Add operation. In this case, however, a carry-in is forced into the least-significant CLA position to produce the two's complement.

The final result is produced by EXOR'ing the CLA output with the content of the B register.

Multiply and Divide

Multiply and divide operations (and most of the floating-point operations) involve double-word operands (products, dividends, long-precision fractions, etc.). To accommodate these double-word operands, the B register is extended to the right by the BX register, both units then constituting a register, 64 bits wide, with facilities for left and right shifting. The abilities of the B register are maintained and utilized in the multiply and divide operations. Principles and the full description of the multiply and divide operations are given in Principles of Operation - Processing Unit, Form Y33-0002.

ARITHMETIC AND LOGIC FUNCTION COMPONENTS

A Register

- Used for ALS functions (a part of the carry generation) and data handling.
- 32 bits wide.
- Performs an OR function on data from the ABC funnel.
- Each position is a multi-input trigger.
- Content can be displayed on the console.

Use

The A register, which is 32 bits wide, participates in effective-address computing and handles data in arithmetic and logical operations. The register is also involved in floating-point (FP) operations, as detailed in the Functional Units section of the Floating-Point Feature, Form Y33-0005. The sign bit (position 00 of the register) is used for sign analysis with certain instructions as described subsequently.

Input

The input to the A register is received from the ABC funnel or, if the floating-point feature is installed, from the AX register. The connections to a typical A register position are shown in Figure 2-4.

	A Register OR Triggers	B Register EXOR Binary Triggers	C Register EXOR Binary Triggers
1 Reset A register, set B and C registers	0000 0000	1111 1111	1111 1111
2 Gate operand 2 in B and C registers	——	(0011 0111)	(0011 0111)
Contents of the registers are now	0000 0000	1100 1000	1100 1000
3 Gate operand 1 in A, B and C registers	1001 1000	(1001 1000)	(1001 1000)
Contents of registers are now	1001 1000	0101 0000	0101 0000
4 Gate CLA output (with carry-in forced) to B register	——	(0011 0001)	——
Contents of the registers are now The result is obtained in B register	1001 1000	0110 0001	0101 0000

Figure 2-3. Subtract Example (Hex 98 - 37 = 61)

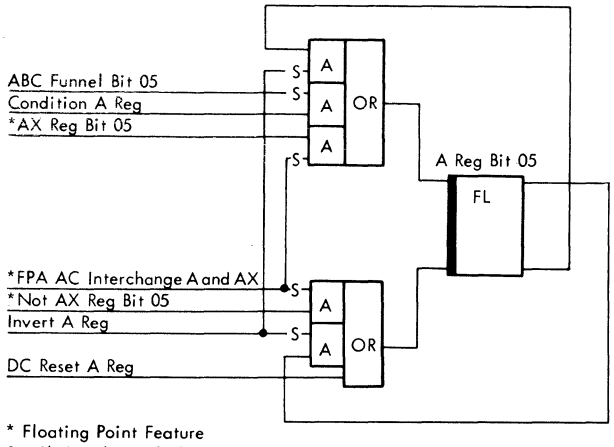


Figure 2-4. Typical A Register Position

Output

The 32 positions of the A register may be gated to the corresponding positions of the ABC funnel or the AX register, and are also connected to the CLA. With the floating-point feature installed, positions 00 to 07 may be gated to the exponent funnel.

Positions 00 and 08 are connected to the condition code circuitry in the Program Status Word 2 (PSW 2).

Position 00 is associated with the load, shift, multiply and divide operations and with the update timer request for sign inspection.

Controls

The timing of the following controls differs from instruction to instruction and with the extent of instruction progress. Refer to the timing diagrams in FEMD IBM System/360 Model 44, Volume 2, Form Y33-0008, for details of the different timings.

DC Reset A Register: This signal clears all positions of the A register. It is possible however, for floating-point requirements, to clear only the eight most-significant positions (bits 00 to 07, characteristic).

Condition A Register; The input of the A register is normally deconditioned; therefore, to receive data from the ABC funnel, the A register must be conditioned by the 'condition A register' signal. The A register is always conditioned if the B register is deconditioned.

Invert A Register: This signal may be required in multiply and divide operations as a step in the process of complementing the A register. The signal can be disabled for floating-point operations.

Interchange A and AX: This signal is provided for floating-point arithmetic purposes and is used to display the AX register.

Description

Each position of the A register is a multi-input trigger as shown in Figure 2-4. Details of the operation of multi-input triggers are given in Appendix B2 of FEMM IBM System/360 Model 44, Form Y33-0007. Most of the "act" (shift pulse) signals are supplied by the circuit type described in Appendix B1 of the same manual.

Special Aspects

As the component circuit is effectively a latch, the A register performs an OR function on data entering from the ABC funnel. The contents of the A register can be displayed on the console.

AX Register

- Installed as part of the floating-point feature for temporary storage of A register contents with a return path to the A register.
- Is 32 bits wide.
- Controlled by floating-point instructions only.
- Each position is a multi-input trigger.
- Contents can be displayed indirectly on the console by operating the CPU 2 switch.

Use

The AX register, which is 32 bits wide, is a slave of the A register and is used to provide temporary storage for the contents of the A register. The AX register is always used when long-precision operands are being handled and is sometimes used in the execution of operations concerned with short-precision operands.

Input and Output

Input to the AX register is received from the A register by interchange, and output is fed to the A register only.

Controls

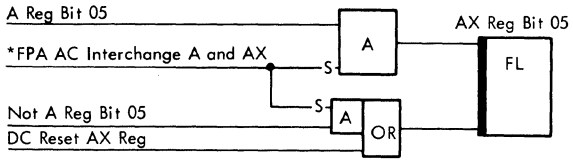
The timing of the following controls differs from instruction to instruction and with the extent of instruction progress. Details of the different timings are given in the timing charts in FEMD IBM System/360 Model 44, Volume 2, Form Y33-0008.

DC Reset AX Register: This signal clears all the 32-bit positions of the register.

Interchange A and AX: This signal interchanges the contents of the A and AX registers.

Description

Each position of the AX register is a multi-input trigger as shown in Figure 2-5. Circuit details of the multi-input triggers are given in Appendix B of FEMM IBM System/360 Model 44, Form Y33-0007.



*Floating Point Feature
S = Shift Pulse with the Occurrence of the Condition

Figure 2-5. Typical AX Register Position

Special Aspects

The contents of the AX register cannot be directly displayed on the console. For display, the contents are first interchanged with those of the A register when the CPU 2 switch is set. When the A register content is then displayed , the actual display is of the AX register contents.

B Register

- Has multiple uses in every instruction and has arithmetic and logical functions.
- Is the path from the ALS to the SDR.
- Is 32 bits wide.
- Performs an EXOR function (binary-triggered) on data received from the ABC funnel.
- Each position is a multi-input trigger and can be displayed on the console.

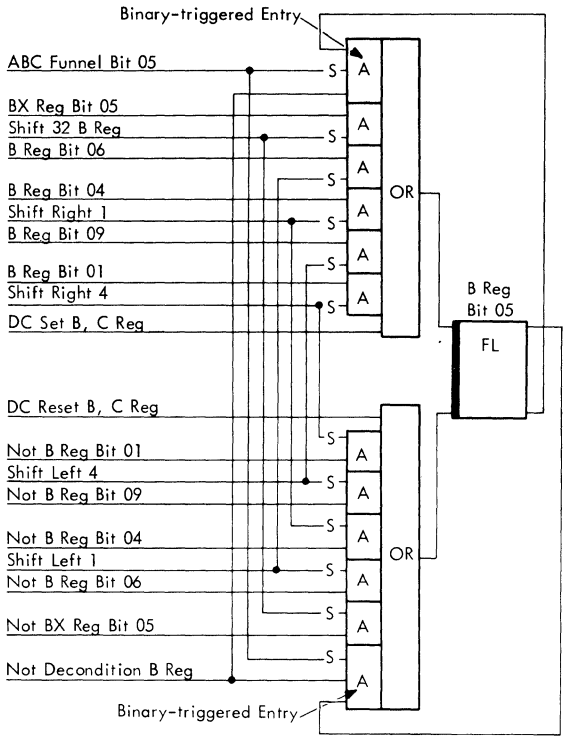
Use

The B register, which is 32 bits wide, is used in the execution of all instructions, largely because it is the only path between the ALS and SDR. The results of arithmetic operations are developed in the B register. The B register performs program shift and automatic shifting in association with byte handling, multiply and divide operations and floating-point instructions. Shift pulses effectively couple the BX register to the left or right of the B register.

The B register is tested for an all-zero content, to check for zero operands (divide) or zero results. This test is the result of a zero in several fields (such as bits 4 to 7, FP use and bits 8 to 11, FP and non-zero protection key use) covering the whole register.

Input

Inputs to the B register are derived from the ABC funnel, from corresponding positions of the BX register and from the B or BX register positions in accordance with the shift operation being performed. A typical B register position, showing input connections, is shown in Figure 2-6.



S = Shift Pulse with the Occurrence of the Condition

Figure 2-6. Typical B Register Position

Output

To simplify description, the B register outputs (as shown in Figure 2-7) are grouped together and designated by code numbers. The functions of the groups (preceded by the corresponding code number) are:

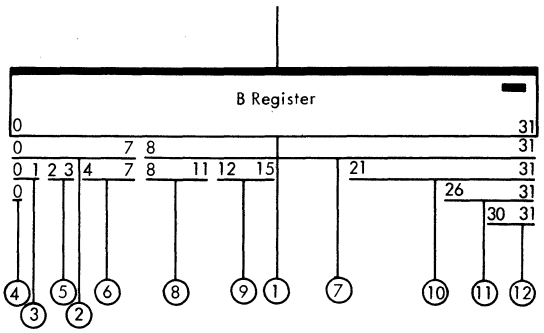


Figure 2-7. B Register Output

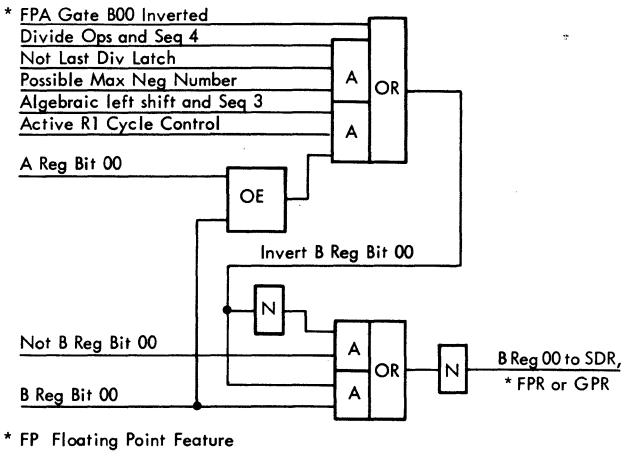


Figure 2-8. B Register Sign-Correction Gating

1. All Positions: The outputs of the 32 positions are distributed as follows:
   - To the corresponding positions of the C register for direct setting and binary gating.
   - To the corresponding positions of the SDR.
   - To the corresponding positions of the BX register.
   - To the B register zero detection circuit.
   - To the B and BX register positions corresponding to the shift pulse.
   - To the corresponding positions of the FPR's and GPR's.

2. Positions 00 to 07: These outputs are used in conjunction with the 'test under mask' operation and the 'system mask' bits (corresponding positions of PSW 1).

3. Positions 00 and 01: These positions are connected to the shift overflow detection circuitry.

4. Position 00: This position is connected to the following circuits:
   - Sign analysis for condition code setting during load and arithmetic operations and 'update timer' interrupt requests.
   - Control for shift-right-one insertion and floating-point multiply and divide operations (B register extension bits).

The content of this position can be gated with inversion for sign correction purposes. See Figure 2-8.

5 and 6. Positions 02 to 07: These positions are connected to the corresponding positions of PSW 2. Bits 2 and 3 form the condition code and bits 4 to 7 form the program mask. Bits 4 to 7 are also analyzed for zero by FP-function overflow circuits (shift-right-four request).

7. Positions 08 to 31: These positions are fed to SAR 2. Positions 08 to 14 are compressed into SAR 2 position 14, and position 31 gives the even/odd address control.

8. Positions 08 to 11: These positions are fed to the floating-point ones and zero detection circuitry.

9. Positions 12 to 15: These positions are fed to the corresponding positions of the PSW 1 (machine state latches).

10. Positions 21 to 31: Bits 21 to 23 are fed to the channel decode and bits 24 to 31 to the channel address.

11. Positions 26 to 31: These positions are fed to the shift counter positions 00 to 05.

12. Positions 30 and 31: These positions are fed to the "parallel" positions, 30 and 31, of the BX register for use in multiply and divide operations.

Controls

The timing of the following controls differs from instruction to instruction and with the extent of instruction progress. Details of the different timings are given in FEMD IBM System/360 Model 44 Volume 2, Form Y33-0008.

DC Reset B, C Registers: This signal clears all 32 positions of the register. It is possible, however, for address-loading and floating-point purposes, to clear only the first byte (positions 00 to 07).

DC Set B, C Registers: This signal is used to set all ones into the 32 positions of the register.

Decondition B Register: In its normal state, the input of the B register is conditioned to receive data from the ABC funnel. This signal deconditions the input of the register and prevents data (from the ABC funnel) from being set into it.

Shift 32: These pulses, which are derived from the line 'interchange B and BX', control interchange of the B and BX registers. However, with the loading of PSW 2, the B register is set into the BX register without disturbing the contents of the B register.

Shift-Right One and Shift-Left-One: These pulses are used in the multiply and divide operations, and to perform the shift instructions to shift one bit position right or left. The pulses always control the B and BX registers as a single double-word register, the BX register being coupled to the right of the B register, even though the BX register is not involved in the operation.

Shift-Right-Four and Shift-Left-Four: These pulses are provided for character-handling operations and for floating-point arithmetic purposes to shift four bit positions right or left. The pulses normally control the B and BX registers as a single double-word register, the BX register being coupled to the right of the B register, even though the BX register is not involved in the operation. The shift-right-four pulse loops the registers. That is, the bits shifted from the rightmost four positions (BX register bits 28 to 31) are coupled to the leftmost four positions (B register bits 00 to 03). This process is used for floating-point long-precision add, subtract and compare operations only, to shift into the lower-order fraction (in the B register), digits from the high-order fraction (in the BX register), during the matching of the exponents: at that time, the shift-right-four link between the low-order end of the B register and the high-order end of the BX register is suppressed. Low-order fraction hex digits are shifted out and lost.

Description

Each position of the B register is a multi-input trigger as shown in Figure 2-6. The contents of the B register can be displayed on the console.

Special Aspects

Because each position of the B register is a binary-coupled multi-input trigger, an EXOR function is performed on all data entering through the ABC funnel. The DC setting of this register is one of the steps in the two's complementing of data.

The B register is tested for a zero result in some operations (condition code setting, loading PSW 1 and floating-point arithmetic).

As shown in Figure 2-8, a sign-correction circuit exists in the path from the B register to the SDR.

The B register bit 00 can be gated true or inverted, according to the analysis shown in Figure 2-8, when the results of the following operations are stored:

Arithmetic left-shift (to give the result its original operand sign);
Divide (to store the maximum negative number);
The majority of floating-point operations.

B Register Floating-Point Extension Positions

When the fraction digits of floating-point operands are used in floating-point multiply and divide, no provision is made to hold the sign of the partial results in the B register, because this register can contain up to 32 bits of data.

With the floating point feature, both the sign of the B register partial result and the data bit shifted to and from the B register bit 00 are held in two extra B register bit positions. The positions are called the 'B register extension' bit and the 'B register extension sign' bit.

The 'B register extension' bit is used on FP multiply to hold the data-bit which is right-shifted into the B register bit 00 position during multiply cycles. The 'B register extension' bit is also used on FP divide cycles to hold the data-bit left-shifted out of the B register bit 00 position during the divide cycles.

The 'B register extension sign' bit is used to hold the sign of the partial results in the B and BX registers during the FP multiply and divide. This bit and the arithmetic carry-out of this position are used in the control of the FP multiply and divide operations in the same manner as the B register bit 00 and C0 are used in the fixed-point multiply and divide operations.

Use of the 'B register extension' bit and 'extension sign' bit is discussed in greater detail in the "Floating-Point Common Multiply" and "Common Divide" sections in Floating Point Feature, Form Y33-0005.

BX Register

- Used for double-word handling operations, for loading PSW 2 bits 8 to 31, for loading the floating-point scratch register, as temporary storage for B register contents, and to develop the quotient during divide operations.
- Width 32 bits, plus two low-order parallel positions and one dummy position.
- Each position is a multi-input trigger.
- Contents are indirectly displayed on the console by interchanging with the B register.
- The parallel positions 30 and 31 are displayed separately from the remaining positions.

Use

The BX register is provided for handling double words in multiply, divide, and floating-point operations. The register is also used during store operations to preserve the effective address and the operand. The BX register is the only path to the instruction counter (bits 08 to 31 of PSW 2). In divide operations, the quotient is formed in the BX register; in multiply operations, the multiplier is analyzed in the BX register.

Input

The BX register is 32 bits wide. The 32 bits are filled from the B register by the B and BX register interchange. Inputs are also derived from positions in the B register and the BX register, depending upon the shift signals. Positions 30 and 31 can receive an input from the quotient bit formation circuit.

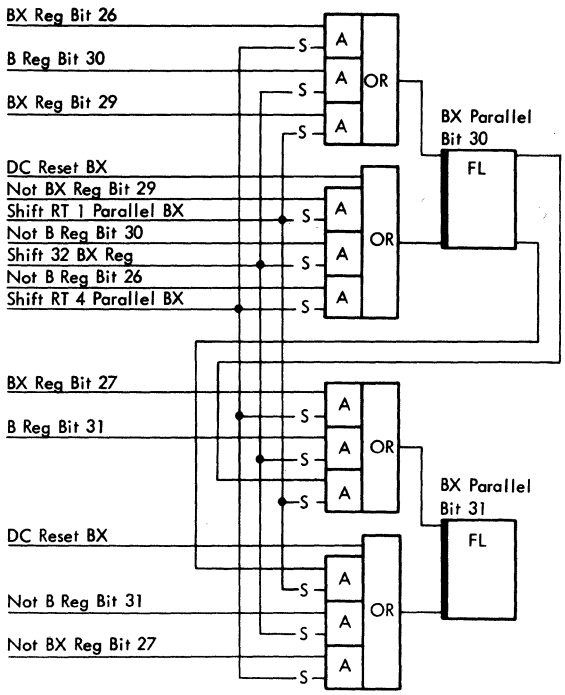
The BX parallel positions receive inputs from the corresponding B register positions and participate in the shift-right-one and shift-right-four operations only. See "Parallel Positions" and Figure 2-9.

Output

The 32 positions can be fed to:

The floating-point scratch register;
The corresponding positions of the B register (shift 32, B register);
The B and BX register positions corresponding to the shift pulses.

Bit positions 08 to 31 are fed to the instruction counter (PSW 2 corresponding bits). Position 00 is used in fixed-point divide control ('possible zero remainder'). Position 30 is fed to the true/criss-cross control circuits for 'store character' or 'store half-word' operations. Parallel positions go to the multiply circuits.



S = Shift Pulse

Figure 2-9. BX Register Parallel Bits 30 and 31

Controls

The timing of the following controls differs from instruction to instruction and with the extent of instruction progress. Detailed timing charts are given in the IBM System/360 Model 44, Volume 2, Form Y33-0008.

Reset BX Register: This signal is provided to clear all 32 positions of the BX register. For floating-point purposes, however, it is possible to reset the first byte (exponent) only, positions 04 to 31 only (preservation of the guard digit), or positions 24 to 31 only. In these cases the line name in the ALD states the bit positions affected.

For the BX parallel positions, see Figure 2-9 (negative logic).

Set BX Register: This signal is used in conjunction with the 'branch on count' instruction and for floating-point purposes.

Shift 32: This pulse is used to interchange the contents of B and BX registers.

Shifts, Right and Left, One and Four: These pulses operate in the same way as previously described for the B register. The BX parallel bits participate in the shift-right-one and shift-right-four.

Description

Each of the 32 positions of the BX register is a multi-input trigger. Figure 2-6 may be considered as a typical BX register position if the binary funnel-input is ignored and if the B and BX labels are interchanged.

Parallel Positions

The two BX parallel positions are also multi-input triggers as shown in Figure 2-9. These positions do not always reflect the state of the normal BX positions 30 and 31 but, during multiply operations, their contents do correspond.

The parallel positions are set from the corresponding positions of the B register at the same time as the remainder of the BX register. They participate in shift-right-one and shift-right-four operations only. The output of these positions is called 'BX bit XX for multiply'.

Special Aspects

An extra bit position BX 32 is incorporated (but not used) in the register because of the mechanical considerations in packaging 32 bits. However, owing to internal connections, BX register bit position 31 feeds to, and is fed from, the bit position 32. On the shift-right-one operation, bit 31 is put into bit position 32 and is considered lost, although on shift left one operation position 32 is put into position 31. This is because shift left one operation cannot follow shift right one operation without the BX register being reset.

Display

The contents of the BX register cannot be displayed directly on the console. To display them, the B and BX registers are first interchanged by operating the CPU 2 switch. When the B register is then displayed on the CPU display roller, the actual display is of the contents of the BX register.

The BX parallel positions 30 and 31 are displayed on the console by another setting of the CPU display roller.

C Register

- Is a slave of the B Register.
- Is part of the carry-generation system.
- Is 32 bits wide.
- Performs an EXOR function on data from the ABC funnel.
- Each position is a multi-input trigger and can be displayed on the console.

Use

The C register is used to reflect (for carry purposes) the contents of the B register, and to feed the CLA.

Input and Output

Inputs to the C register are derived from ABC funnel and from the B register, while output is fed to the CLA only.

Controls

The timing of the following controls differs from instruction to instruction and with the extent of instruction progress. Details of the control timings are given in the timing charts in FEMD IBM System/360 Model 44, Volume 2, Form Y33-0008.

Reset C Register: This control clears all 32 positions of the register. As this signal is the same as that which clears the B register, however, provision is made for clearing only the first byte (bits 00 to 07).

Set C Register: This signal sets ones in all 32 positions of the register for complementing purposes, as in the B register.

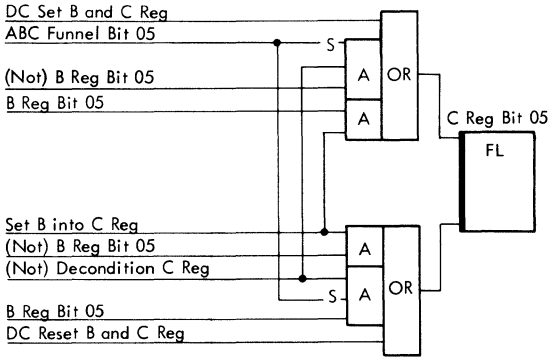
Decondition C Register: In its normal state, the C register is conditioned to receive data from the ABC funnel; the 'decondition C register' signal therefore specifically prevents the entry of data. Note that because of the switching speed of its binary entry, the C register is deconditioned whenever the CLA is gated to the ABC funnel, to avoid feedback to the CLA.

Set B into C Register: This facility is provided to adjust the C register contents to the B register contents after the CLA operation, and to prepare for a following addition as explained in "Introduction" in this Chapter.

Description

Each position of the C register is a multi-input trigger, as shown in Figure 2-10. The binary-triggered input to the C register is exceptional in that the binary gates are generated by another register (B register), as opposed to being self-gated. However, the B and C register contents are always the same whenever the C register output is used (CLA gating). When the C register content is not used, the difference between the B and C register contents is not significant.

The content of the C Register can be directly displayed on the console.



S = Shift Pulse with the Occurrence of the Condition

Figure 2-10. Typical C Register Position

Carry Look-Ahead

- Generates and distributes the carries.
- Boolean expression for CLA function is: $Kn = A.\overline{C} + C.Kn - 1$.
- Is 32 bits wide.
- High-speed logic.
- Output is controlled by the instruction.
- The two high-order outputs are analyzed for condition code setting.
- A carry is forced in the low-order position for complementing purposes by the subtract trigger.

Use

The carry look-ahead, which is 32 bits wide, is part of the arithmetic and logic section and furnishes the carries, when and where they are needed, during the progress of operations.

Input and Output

Inputs to the CLA are fed directly from the A and C registers.

The outputs from the CLA are gated to the corresponding positions of the ABC funnel. The entry is usually to the B register only, where an EXOR operation is performed with the original B register contents. Each of the two most-significant outputs of the CLA is used to set a latch (C0 and C1), the analysis of which is used in many operations to set the condition code.

Controls

There are no controls on the inputs to the CLA but gating-out of the CLA contents is controlled. The timing of these controls differs from instruction to instruction and in accordance with the extent of instruction progress. Details of the timings are given in the timing charts in FEMD IBM System/360 Model 44, Volume 2, Form Y33-0008.

Description

The CLA is a 32-bit-wide block of unlatched high-speed logic. The reasons for the existence of the CLA and its action during an operation are given in the Introduction to this section of the Chapter.

Carries are produced by comparing the corresponding bit positions of the A and C registers according to the following rules:

1. A one bit in the A register and a zero bit in the C register generate a carry-out to the next high-order bit position (Boolean expression: $A.\overline{C}$ = K out).
2. A one bit in the C register and a carry-in to the position propagate the carry to the next high-order bit position (Boolean expression: C.K in = K out).

Propagation is thus from right to left.

Figure 2-11 shows the circuitry associated with the four least-significant positions of the CLA. Such a block of four is repeated eight times to form the 32-bits-wide CLA logic.

Generation and propagation circuits exist for each position of the CLA. In addition, the most-significant position of each block of four collects the carries leaving the block, thus avoiding the need for the carry to ripple through any other position inside the block: this avoids the accumulation of transition times in too many CLA positions and speeds the availability of the output at the ABC funnel. Therefore, in the worst case (namely, a carry generated in the lowest block of four having to ripple up to the CLA bit 00), ten logic circuits (instead of 31) must be traversed.

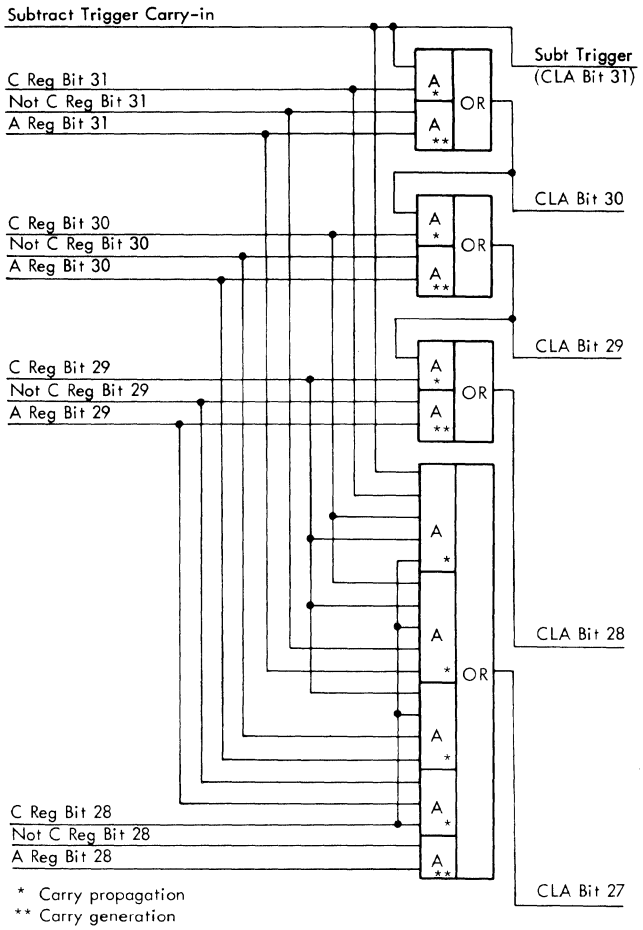


Figure 2-11. CLA Four Least-Significant Positions

Bit positions 01 of both the A and C registers produce a CLA bit 00 output which is gated to the ABC funnel and is used also to set the C1 latch (used for condition code setting).

Bit positions 00 of both the A and C registers produce a CLA output ('adder carry-out') which is not fed to the ABC funnel, but is used only to set the C0 latch. The state of this latch is used later for floating-point purposes and for condition code setting.

Special Aspects

No CLA Bit 31: As shown in the ALD's, CLA bit 30 is the least-significant CLA position and is produced from the A and C register bit 31 positions. Nevertheless, a carry-in can be forced to the CLA for two's complementing purposes (during any subtraction operation). This carry-in should be termed 'CLA bit 31' but, being produced only by the subtract trigger, it is called 'subtract trigger carry-in'.

Negative Logic: In the ALD's, succeeding blocks of logic alternate in sign and in function; therefore a 'plus AND' equals a 'minus OR' and vice versa.

ABC Funnel and Hardware Funnel

- The ABC funnel collects information directed to A, B or C registers.
- The Hardware (HW) funnel is used as an extension of the ABC funnel.
- Both funnels are 32 bits wide.
- Unlatched logic.
- Controlled by the instruction.
- Use of negative logic in the HW funnel permits the timer value generation.

The ABC funnel collects all information which is to be handled by the A, B and C registers. It can also be considered as part of the arithmetic and logic section data flow and is involved in the 'halfword expansion' operation.

Input and Output

Inputs and outputs associated with the ABC funnel are shown in Figure 2-12. Note that in the figure reference is made to two detailed Figures, 2-13 and 2-14.

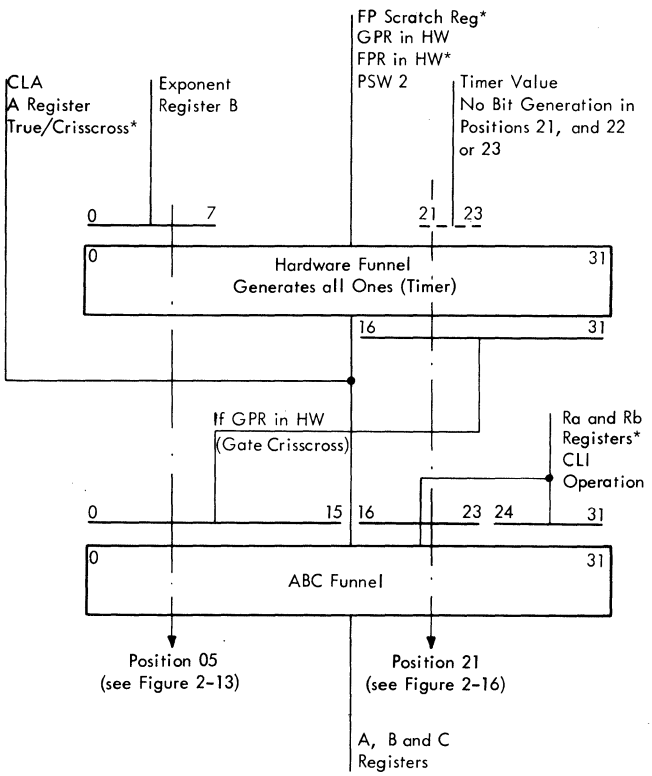
Funnel Controls

Positions 05 and 21 are given as detailed examples in Figure 2-14.

The timing of the following controls differs from instruction to instruction and with the extent of instruction progress. Details of the timing are given in IBM System/360 Model 44, Volume 2, Form Y33-0008.

ABC Funnel Controls

Refer to Figures 2-12, 2-13 and 2-14 for the ABC funnel controls described below.



* Can be gated with Selected Parts of Word

Figure 2-12. ABC Funnel Inputs and Outputs

Set True/Criss-Cross to Funnel: Can be applied as follows:

To positions 00 to 31: To gate the fullword to the registers.

To positions 16 to 31: To gate a halfword operand.

To positions 00 to 15: When expansion of a halfword negative operand is required, and for 'test and set' operations. For information on the former, see Introduction to "Fixed-Point Arithmetic" in Principles of Operation - Processing Unit, Form Y33-0002.

To positions 20 to 31: To gate the displacement during RX format I-fetch.

To positions 16 to 23, or 24 to 31: With even and odd address decoding respectively in the SI format ' compare logical' operations.

To positions 00 to 07, 00 to 15, 00 to 23, depending on the floating-point precision switch setting on the console (8, 10, and 12 respectively) when fetching the floating-point long operand second word. Note that position 14 gates the positions 00 to 31.

Set CLA to ABC Funnel: Gates the 32 positions of the CLA for use during arithmetic and logic operations.

Set A Register to ABC Funnel: Gates the 32 positions of the A register for use in the ALS and for other purposes.

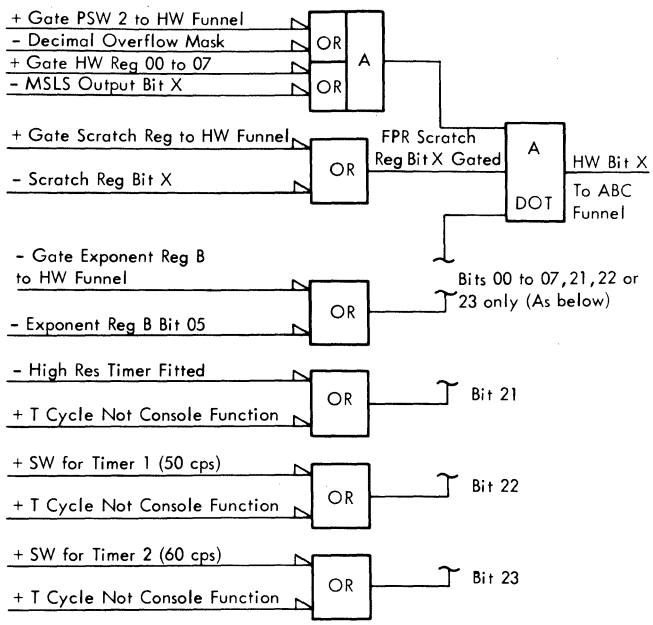


Figure 2-13. Hardware Funnel Controls

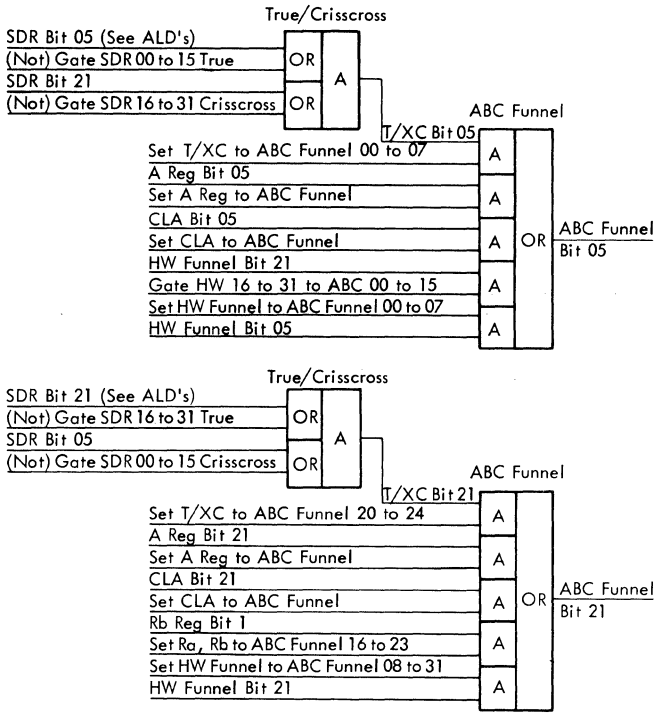


Figure 2-14. Two Positions of the ABC Funnel

Gate HW 16 to 31 to ABC 00 to 15: This facility exists only with the accelerator feature incorporated and provides the means to criss-cross a halfword from a GPR for halfword or byte storage.

Set Ra, Rb, Registers to ABC Funnel: This is provided to bring the immediate operand into the appropriate place for CLI (compare) operation, without using shift cycles.

Set HW Funnel to ABC Funnel: This gates the HW funnel to the ABC funnel. It can select positions 00 to 31 (full word), 00 to 07 (byte 0), or 08 to 31 (bytes 1, 2 and 3) while gating floating-point registers or the floating-point scratch register.

Hardware Funnel Controls

The HW funnel output is one of the inputs to the ABC funnel. The output line for each bit position of the HW funnel is derived from a dot-AND circuit as shown in Figure 2-13.

The normal condition of all inputs to the dot-AND is positive, as both the 'floating-point scratch register' gated bit and the 'exponent register B' gated bit lines are positive in their normal state; i.e., these lines assume the status of the corresponding register only if it is gated to its output bus.

The four controls which determine the output of the HW funnel are:

'Gate PSW 2 to HW funnel'
'Gate hardware register to HW funnel'.
'Gate scratch register to HW funnel'.
'Gate exponent register B to HW funnel' (bits 00 to 07).

Each of these controls generates the 'gate HW funnel to ABC funnel' signal.

The 'gate hardware register to HW funnel' control is interlocked with the other controls so that it is degated if any of the remaining controls are active.

Figure 2-13 shows that when one of the above controls is made active, the corresponding bit governs the input (and thus output) of the dot-AND.

In the normal static condition (no gates active), the HW funnel output of each position is active. This output of all ones is used for generating the timer-decrement constant used in the 'update-interval-timer' operation; for this operation, the HW funnel output in bit positions 21 and either 22 or 23 (depending on the line frequency) is forced to zero during the timer update cycle.

Logic and Description

Refer to Figures 2-12, 2-13 and 2-14 for logic and description.

The ABC funnel is 32 positions wide. Each position is similar to those shown in Figure 2-14 and is implemented in a combination of medium-speed circuitry (HW funnel and true/criss-cross) and high-speed circuitry (ABC funnel proper).

INSTRUCTION REGISTERS

- Instruction registers are the Op register and the Ra, Rb and Rc registers.
- They are used to store the first 20 bits of the instruction during the address indexing and the execution.
- The Op register originates signals and conditions appropriate to the operation.
- The Ra, Rb and Rc registers originate GPR and FPR addresses.
- The Ra and Rb register contents can be used directly for operations CLI, TM, BCT, BCTR and SVC.
- All instruction register contents are displayed on the console.

Use

The Op register is used to store the instruction code. The Ra, Rb and Rc registers are used to hold information associated with the addressing of operands and indexes for use during I-fetch indexing cycles and during instruction execution.

The output of the Op register is decoded to produce the signals that set up the cycle and sequence latches and the appropriate conditions for controlling the required operation.

The Ra, Rb and Rc registers hold respectively the R1, R2 or X2 and B2 fields of the instruction. Their output is used to generate the GPR and FPR addresses that are loaded into the SAR's (extension storage addressing) or that are used to read from or write into hardware registers. The content of the Ra and Rb registers can also be used:

As an immediate operand in the SI format, 'compare logical' operation (CLI),

As a mask in the 'test under mask' (TM) operation and in the RX or RR formats 'branch on count' operation (BCT or BCTR respectively), or

As an interrupt code in the 'supervisor-call' (SVC) operation.

Description

The Op register is eight bits wide and the Ra, Rb and Rc registers are each four bits wide. Each register position is a high-speed latch. The logic of a typical register position is shown in Figure 2-15. The contents of all instruction registers are displayable on the console by selecting the appropriate roller switch setting.

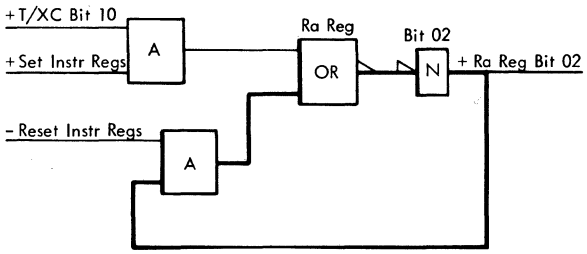


Figure 2-15. Typical Instruction Register Position

Input and Output

The inputs to the instruction register are from the SDR via the true/criss-cross. The outputs are distributed as follows:

Bits 00 and 01 (instruction format): To PSW 2 bits 00 and 01 (ILC). Refer to "PSW Registers" in this Chapter.

Bits 00 to 07 (op code): To the op decode circuitry, including invalid op detection.

Bits 08 to 11 (Ra register): To the 'branch on condition' circuitry and to hardware register address decode (floating-point and/or accelerator feature).

Bits 08 to 15 (Ra and Rb registers): To the register address specification circuitry for floating-point operations and multiply and divide operations (double-word operands), to the ABC funnel (CLI operation), to the 'test under mask' and to the FPR address generation circuitry.

Bits 08 to 19 (Ra, Rb and Rc registers): To the storage address generation logic or to the GPR address decode (accelerator feature).

Bits 12 to 19 (Rb and Rc registers): To the zero detection circuits (I-fetch use).

Controls

The instruction registers are set and reset during an I-cycle.

PSW REGISTER

Introduction

- Used in program execution.
- PSW bits 0 to 31 are referred to as "PSW 1 bits 0 to 31."
- PSW bits 32 to 63 are referred to as "PSW 2 bits 0 to 31."
- The PSW register consists of several separate registers and unlatched logic.
- The entire PSW, or in certain circumstances only part of it, can be stored or loaded.
- PSW 1 is displayed indirectly via the SDR.
- PSW 2 is displayed indirectly by the SDR via the HW funnel, the ABC funnel and the B register.

The information held in the PSW register is used for correct program execution. Refer to IBM System/360 Principles of Operation, Form A22-6821, for further details.

NOTE: In terms of System/360 architecture the PSW is considered to be 64 bits long. To facilitate the numbering system, however, and since PSW handling in the 2044 always necessitates two complete cycles, PSW bits 0 to 31 are named "PSW 1 bits 0 to 31" and PSW bits 32 to 63 are named "PSW 2 bits 0 to 31."

As shown in Figure 2-16, PSW 1 is loaded from the corresponding bit positions of the B register (0 to 7 and 12 to 15) and stored through the SDR (0 to 7, 12 to 31). Only bits 2 to 7 of PSW 2 are loaded from the B register. Bit positions 8 to 31 are loaded from the corresponding bit positions of the BX register (instruction address). PSW 2 is stored via the HW funnel, the ABC funnel and the B register to the SDR.

The PSW registers consist of several separate sections that are physically located throughout the

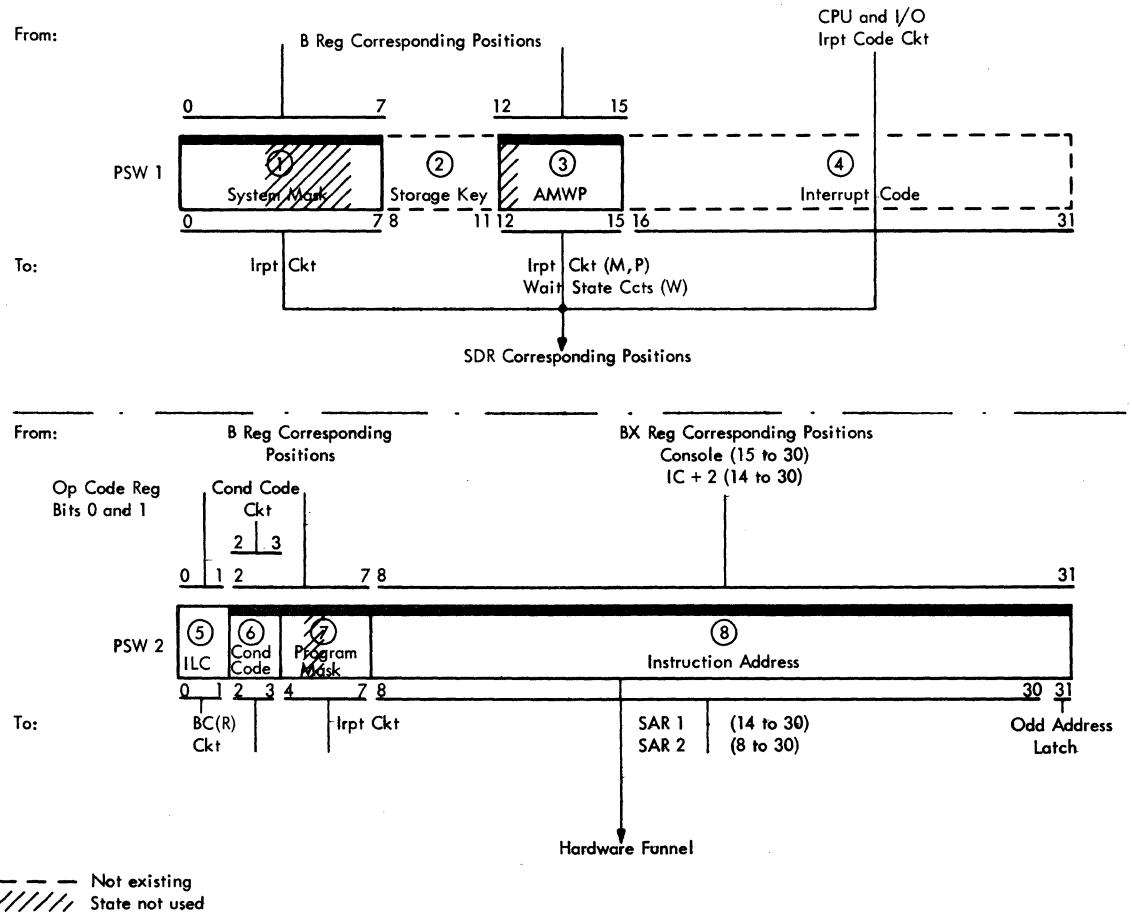


Figure 2-16. Model 44 Program Status Word

main frame. For clarity, the two registers are shown as a whole on the CPU data flow.

To enable the program to change the state of the CPU, fields of the current PSW can be changed during normal processing without preserving the previous conditions. These changes can be made as follows:

The 'system mask' can be changed by the 'set system mask' operation.

The Instruction Length Code (ILC) is set according to the current instruction.

The condition code is set according to the type and result of the operation performed.

The condition code and the 'program mask' can be changed by the 'set program mask' operation.

The Instruction Counter (IC) is updated during normal processing but can be changed by branch operations (or bits 15 to 30 from the console in the 'manual stop state').

The remaining fields are changed only by loading the entire PSW as in the load-PSW operation, or from the console.

However, to allow the status of the CPU to be inspected at certain times, the PSW must be stored (then becoming the "Old PSW"). This is achieved as follows:

PSW 2 is stored by branch and link operations. At the same time, the IC can be changed if required; the remainder of PSW 2 and PSW 1 is not changed.

The IC is stored in the BX register by the 'system reset' routine (from the console).

The entire PSW 1 and PSW 2 is stored and loaded by the 'supervisor call' (SVC) operation and by any interrupt (Irpt). The loaded PSW may or may not differ from the former PSW.

PSW 1 can be displayed in the 'manual stop state' by pressing the display pushbutton while the storage select rotary switch is set to PSW 0-31. Operation of the pushbutton causes the contents of the PSW 1 register (bits 0 to 7 and 12 to 15) to be set into the SDR; the remaining bits are made zero as the 2044 is not provided with a storage key and an interrupt code is generated only by an interrupt. The SDR then displays the contents of the PSW 1 register.

PSW 2 is displayed in the 'manual stop state' by pressing the display pushbutton while the storage select rotary switch is set to PSW 32-63. Operation of the pushbutton gates the entire PSW 2 through the HW funnel and the ABC funnel to the B (and C) register. The contents of the B register are subsequently gated to the SDR, which then displays the contents of the PSW 2 register.

System Mask

- Eight positions wide.
- Four positions only (PSW 1 bits 0, 1, 2 and 7) of the eight positions are used.
- Loaded via the B register and stored via the SDR.
- Each position is a medium-speed conventional latch.

PSW 1 bits 0 to 7 are associated with the I/O channel and external signals to allow an interrupt to occur if the relevant bit is a one or to hold the interrupt pending if the relevant bit is a zero.

Provision is made to accommodate all eight bits of the 'system mask' but the state of positions 3 to 6 does not concern the program. The significance of the four positions used is as follows:

Bit 0: Multiplexor channel 0
Bit 1: High speed multiplexor channel 1
Bit 2: High speed multiplexor channel 2
Bit 7: Timer, interrupt key and external signals

The system mask is changed (through the B register only) by certain operations. These are 'load PSW', 'set system mask', 'supervisor call', any interrupt, and console operations (with the storage select rotary switch set to PSW 0-31) in the 'manual stop state' only.

Each position is implemented in a conventional medium-speed latch, an example of which is given in Figure 2-17.

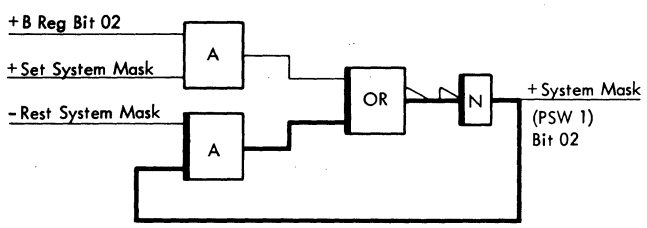


Figure 2-17. Typical PSW Latch (as used in Fields 1, 3 and 7)

Protection Key

- Positions 8 to 11 of PSW 1.
- The protection key field is checked for all zeros in B register during the loading of the new PSW 1, on CPU's with protection feature not fitted.

On CPU's where the protection feature is not fitted, no provision is made to accommodate positions 8 to 11 of PSW 1. The protection key of the new PSW 1 is checked for zeros in the B register when PSW 1 is loaded and sets the 'protect key not zero' latch when it is not zero.

Machine State

- PSW 1 positions 12 to 15.
- Each position is a conventional latch.
- Bit 12: 1 = ASCII mode (not used by the program)
  0 = EBCDIC mode.
- Bit 13: 1 = Machine check interrupt masked.
- Bit 14: 1 = CPU in wait state
  0 = Running state.
- Bit 15: 1 = CPU in problem state
  0 = Supervisor state.
- Loaded from the B register with the entire PSW 1 and PSW 2.

The machine state latches consist of four bit positions (PSW 1 bits 12 to 15), each position being a medium-speed conventional latch as shown in Figure 2-17. These mask bits can be changed only when the entire PSW 1 and 2 is loaded ('load PSW', SVC and any interrupt).

Position 12 (ASCII)

This position controls the coding generation for decimal data. Since decimal data is not used in the 2044 the state of this position is ignored by the program and the machine is therefore always in EBCDIC mode.

Position 13 (Machine Check Mask)

When on (a 1 bit), detected machine malfunctions are allowed to cause an interrupt. When off (zero), machine checks are ignored and the machine will attempt to complete the current instruction followed by the next sequential instruction. Refer to the section "Checking" in this Chapter.

Position 14 (Wait State)

When set to zero, this position allows the CPU to run. When set to one, the position forces the CPU into the wait state, and the wait lamp on the operator's console is lit.

Position 15 (Problem State)

When set to one, the privileged instructions are not allowed and cause a program interrupt to occur. When set to zero (supervisor state) all instructions are valid.

Interrupt Code

- PSW 1 bits 16 to 31.
- Used to identify the cause of an interrupt.
- Implemented in unlatched logic.
- Gated to the SDR only on an interrupt.

The cause of an interrupt can be found by analyzing the interrupt code of the old PSW. An interrupt code is generated only while storing the PSW during the interrupt handling. Therefore, no provision is made to accommodate this field in the current PSW.

A table of the interrupt codes applicable to the 2044 appears in the "Interrupts" section of Principles of Operation - Processing Unit, Form Y33-0002.

Instruction Length Code

- PSW 2 bits 0 and 1.
- Formed from the Op register bits 0 and 1.
- The old PSW gives the ILC in halfwords (program and SVC interrupts only).
- Changes when Op register bits 0 and 1 change.

PSW 2 bits 0 and 1 indicate the length, in halfwords, of the last interpreted instruction when a 'program' interrupt or SVC interrupt occurs. The ILC is unpredictable (and useless) for I/O, external, or machine check interrupts; in other cases, it helps to locate the instruction which was being handled when the interrupt occurred and is set according to the table shown in Figure 2-18. These positions are not latches but are derived directly from the Op register bits 0 and 1 as shown in Figure 2-19.

NOTE: An instruction length code of three (11) is set in an SS format operation, since an invalid op code is detected from the Op register after the

ILC	PSW 2 Bits 0 and 1	Op Code Bits 0 and 1	Instruction Length
0	00	Impossible without ILC circuit malfunction	
1	01	00	1 Halfword (RR)
2	10	01 or 10	2 Halfwords (RX, RS or SI)
3	11	11	3 Halfwords (SS)

Figure 2-18. Instruction Length Code Table

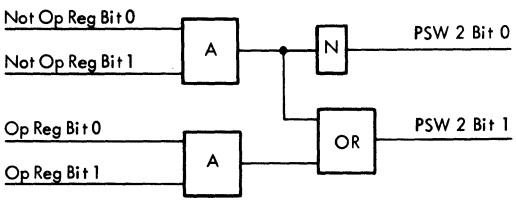


Figure 2-19. PSW 2 Bits 0 and 1 (Instruction Length Code)

register has been set, and because the invalid op code gives a program interrupt in the 2044.

Condition Code

- PSW 2 bits 2 and 3 are also known as the condition code bits 0 and 1.
- Set according to the final result in most operations.
- Meaning depends on the operation performed.
- Also loaded with a new PSW or 'set program mask' operation.

Figure 2-20 shows a typical section of the logic associated with setting the condition code. The logic shown is associated with the add, subtract and compare operations.

The setting of the condition code needs two signals. The first signal is produced by combining the result obtained with the op decode output (refer to the lines 'data condition code Bit 0' and 'data condition code Bit 1' in Figure 2-20). The other signal is the 'set' timing. The op decode output again selects a suitable timing pulse (refer to 'set condition code' line shown). If the data condition code bit line is not active, the reset is performed by the set line by deconditioning the OR block marked with an asterisk (*).

The condition and timing which bring up these two signals vary with the operation performed in order to match the operation end. This is shown in the ALD's. The condition codes for all instructions are summarized in the section "Branching" in IBM System/360 Principles of Operation, Form A22-6821.

Program Mask

- PSW 2 bits 4 to 7 (Bit 5 not used).
- Each position is a conventional latch.
- Allows the masking of three kinds of program interrupt:
  - Bit 4: Fixed-point overflow.
  - Bit 6: Exponent underflow.
  - Bit 7: Significance.
- Can be loaded from the B register with 'set program mask', 'load PSW' and any interrupt.
- Is stored via the B register with 'branch and link' instructions and any interrupt.

The program mask positions consist of four medium-speed latches of the type shown in Figure 2-17.

Each bit is associated with a program exception. When the corresponding mask bit is a one, an interrupt is allowed to occur. When the mask bit is zero, no interrupt occurs and the exception is ignored. Details of program exceptions are given in the "Interrupts" section of Principles of Operation - Processing Unit, Form Y33-0002.

The program mask may be replaced by information (PSW) from the corresponding positions of the B register during any interrupt cycles (including SVC), 'load PSW' and 'set program mask' operations.

The program mask can be stored as a part of the old PSW during any interrupt cycle or it can be stored during the 'branch and link' instruction (in a GPR).

The functions associated with each bit are as follows:

Bit 4 (Fixed-point Overflow)

When bit 4 is zero, a 'fixed-point overflow' exception does not cause an interrupt. The condition code is set regardless of the program mask bits.

Bit 6 (Exponent Underflow)

This position is used when the floating-point feature is installed. When it is zero, no interrupt occurs on an 'exponent underflow' exception.

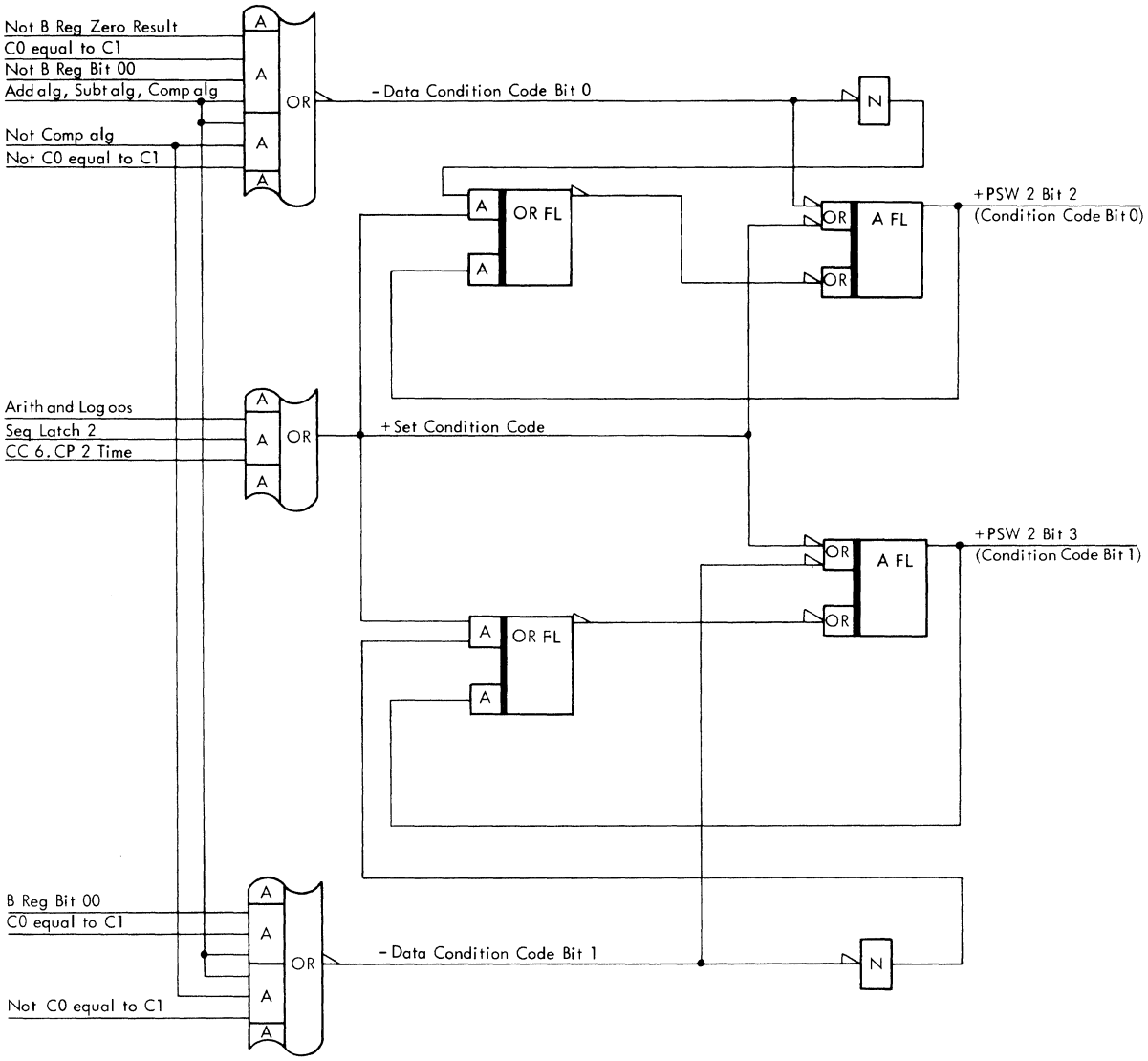


Figure 2-20. Typical Setting of the Condition Code Latches

Bit 7 (Significance)

This position masks a 'significance' exception (fraction result zero in FP add or subtract operations). The state of the position is used when the floating-point feature is installed. The result is made zero in the case of an exception, regardless of the state of the program mask bit.

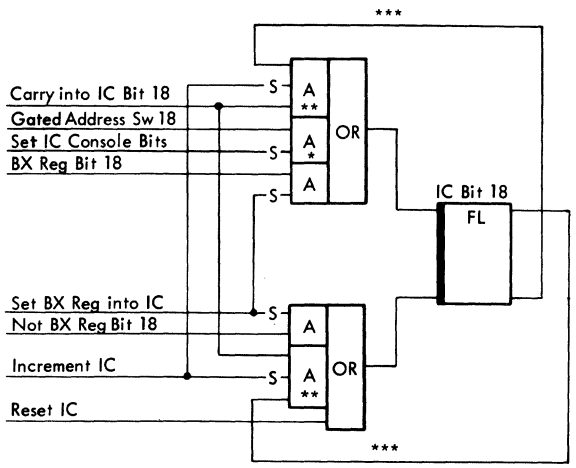
Instruction Address

- PSW 2 bits 8 to 31.
- Known as the instruction counter (IC) in the ALD's.
- Specifies the leftmost byte position of the next instruction.
- Each position is a multi-input trigger.
- Bits 15 to 31 are displayed on the console.
- Loaded from the BX register or from the console; the console loads bits 15 to 30 only.
- Positions 14 to 30 are updated with the IC + 2 operation.
- Used to generate extension storage addresses during the 'system reset' process.

The instruction counter is formed by bits 8 to 31 of PSW 2 and specifies the address of the first byte of the next instruction. Except for 'system reset', the 24-bit address is gated to the SAR's during an I-cycle to address an instruction. The IC is then updated according to the instruction length in order to contain the next sequential instruction address. The principles of the incrementing (by IC + 2 operation) are given in detail in "Main Storage Addressing" in this chapter.

Figure 2-21 shows a typical IC position as used for bits 15 to 30. Each position is a multi-input trigger which is described in Appendix B2 of FEMM IBM System/360 Model 44, Form Y33-0007. The shift pulses shown are generated by a circuit of the type described in Appendix B1 of the same manual.

Because the maximum storage capacity is 131,072 bytes, and since an instruction address must be even, only positions 15 to 30 are needed. To detect program errors, however, the entire 24-bit address is loaded from the corresponding BX register positions (bits 8 to 31) with branching instructions or with the new PSW loading. Oversize storage addressing is detected by compressing the IC bits 8 to



S = Shift Pulse with the Appearance of the Condition
* = Delay for Bits 8 to 14 and 31
** = Delay for Bits 8 to 13 and 31
*** = Not for Position 14

Figure 2-21. Typical Instruction Counter Position

14 to SAR 2 bit 14. The presence of a bit in the IC bit 31 position sets the 'odd address' latch. If either SAR 2 bit 14 or the 'odd address' latch is on, an addressing exception is caused. See the section "Checking" in this Chapter.

Loading from the console address switches is to IC bits 15 to 30 only. However, loading PSW 2 from the console loads all 32 positions of the PSW 2 (24 IC bits).

Because of the addressing exception detection, IC + 2 affects bits 14 to 30 only.

During a 'system reset' routine, the normal content of the IC is stored in the BX register. The IC is then used to scan areas of the extension storage with the aid of IC + 2 (see "Console" in FEMM IBM System/360 Model 44, Form Y33-0007).

SHIFT COUNTER

- Specifies the number of positions to be shifted in the B and BX registers.
- Six-position register numbered 0 (leftmost) to 5.
- Maximum (decimal) capacity is 63.
- Each position is a multi-input trigger displayable on the console.
- Value is reduced by one for each shift cycle.
- It is set from the B register (shift operations) or by logic circuitry.

Use

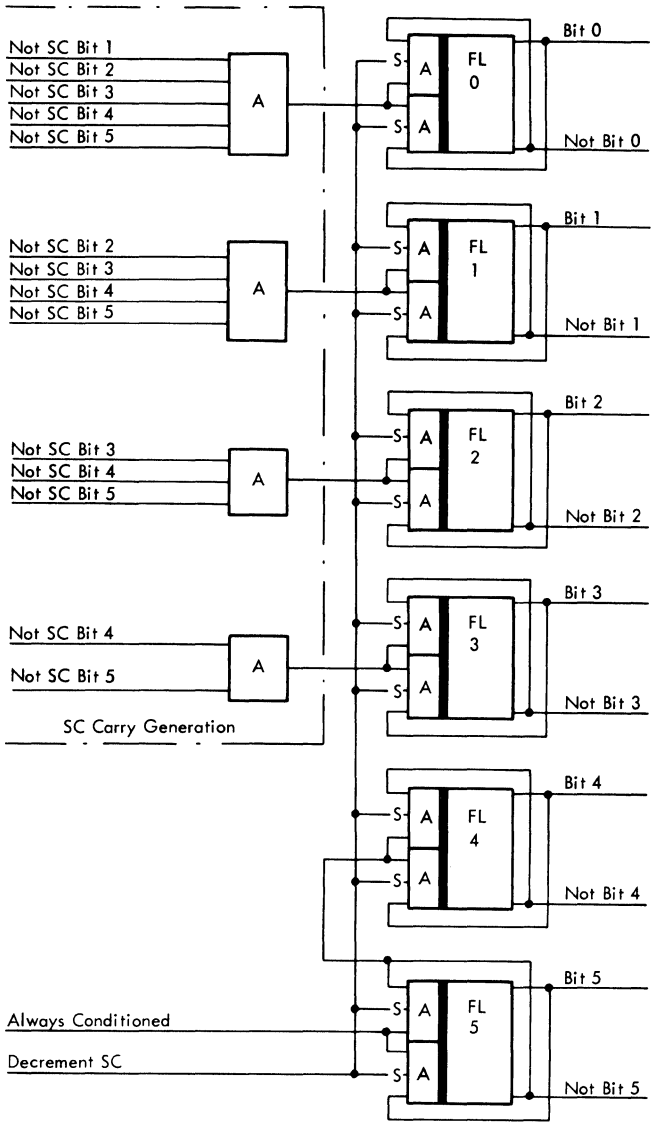
The Shift Counter (SC) is loaded with the number of single shifts to be performed in the B and BX registers. (These function as a single 64-bit register.) As there are six positions, the maximum number of shifts is 63, corresponding to shifting all bits but one out of the B and BX register pair in shift double operations.

Description

Each shift counter position is a multi-input trigger as described in Appendix B2 of FEMM IBM System/360 Model 44, Form Y33-0007 and the logic of a typical position is shown in Figure 2-22. The shift pulses are delivered by circuits of the type described in Appendix B1 of the same manual.

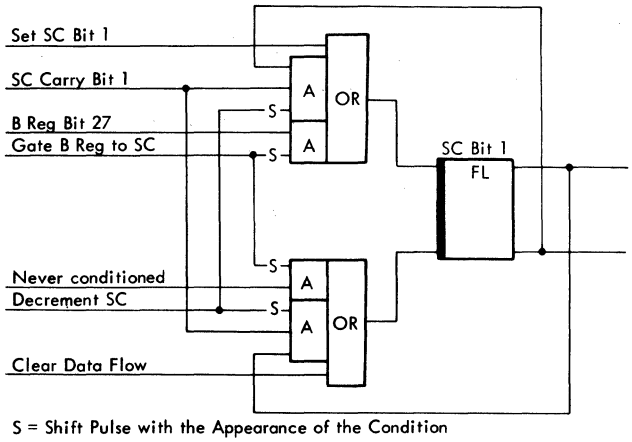
The entire register is displayable on the console with the CPU display roller in the appropriate setting. The value in the shift counter is reduced as shown in Figure 2-23. With each decrementing pulse the value of the counter content is reduced by one. The decrement is made with the aid of the shift counter carry circuits (represented inside the broken line area of Figure 2-23).

The decrementing pulse appears normally during every shift cycle, usually with the same timing as the shift pulse (shift operation) but, in certain cases, the value in the shift counter may be reduced without an accompanying shift; refer to the "Machine Instructions" section in Principles of Operation - Processing Unit, Form Y33-0002.



S = Shift Pulse with the Appearance of the Condition

Figure 2-23. Shift Counter Decrementing Circuits



S = Shift Pulse with the Appearance of the Condition

Figure 2-22. Typical Shift Counter Position

Input and Output

With any shift operation, the input to the shift counter is from the B register positions 26 to 31, B register bit position 26 corresponding to shift counter position 0, etc. Circuitry is also provided to set the shift counter automatically with the correct value for any other operations using shifts one or four (FPA); for example:

Divide operations force shift counter bits 0 and 5 to one (content equals 33);

Multiply operations force shift counter bit 0 to one (content equals 32);

Multiply halfword operation forces shift counter bit 1 to one (content equals 16).

Other shift counter values can be set in floating-point operations according to the type and progress of the operation and depending upon the 'FP precision' switch setting.

In addition to the carry generation and the detection of a shift counter zero value, the output decoding circuits also give several combinations that are used during multiply, divide and floating-point operations; these are indicated in the ALD's.

TIMING

The purpose and function of the various clocks and controls is to provide timed pulses for sequencing and controlling all operations within the 2044.

Clocks

- All clocks are driven from one 8-megacycle (Mc) oscillator.
- After shaping and timing, the pulse cycle is 250 nanoseconds (ns).
- The clocks used in the 2044 are:
  - Basic clock
  - Read/write clock
  - Compute clock
  - I/O clock
  - Interface clock

Oscillator

An 8-Mc oscillator (OSC) provides pulses that are shaped and divided to produce square-wave 4-Mc pulses, of 250-ns cycle time, for clock driving; see Figure 2-24. The clock driving pulses (from which all clocks are derived) are distributed in the following forms:

- Clock special input/output (Clock special I/O)
- Clock special, compute clock or read/write (Clock special CC + RW)
- Gate A1
- For SAR Reset

Main Clock Pulses

The main clock pulses, CP 1 and CP 2, are formed from the clock driving pulses by an inverting amplifier circuit which produces two pulses, 180 degrees out of phase and delayed by 62.5 ns. Each clock pulse is on for 125 ns and off for 125 ns.

These pulses are used to define the outputs of the other clocks. CP 1 defines the 'odd' latch output and CP 2 the 'even' latch output (Figures 2-24 and 2-25).

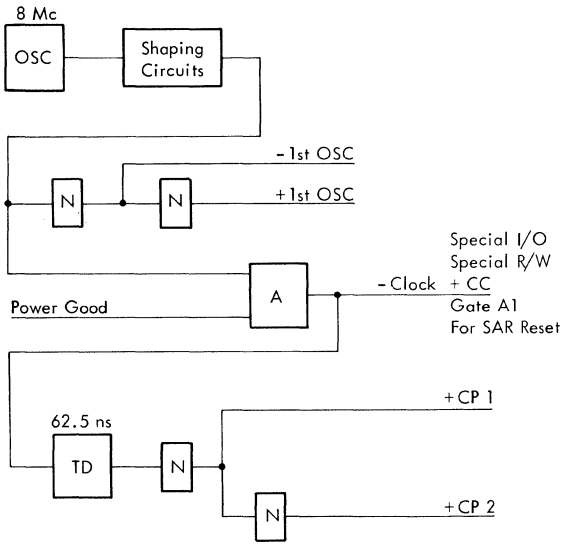


Figure 2-24. Oscillator Clock Shaping and Distribution

Early Clock Pulse

In some operations, for example multiply, the CP 1 pulse defining compute clock 1 (CC 1) does not occur early enough to meet the needs of shifting sequences; in floating-point operations an early CC 1 - CP 1 pulse is also needed.

To satisfy these requirements, a clock special pulse (for convenience, the I/O pulse) is delayed by 37 ns only (instead of the normal 62.5 ns) and is gated with CC 1. The resultant composite pulse (Special CC 1-CP 1 Early), of 125-ns duration is 25 ns in advance of the normal CP 1 pulse occurring at CC 1 time. Figure 2-26 shows the generation and timing of the early clock pulse.

Read/Write Clock

The read/write clock is used for main storage access. Figure 2-27 shows the logic associated with the generation of the read/write clock pulses.

The clock consists of eight output latches and a split cycle latch. These latches are turned on and off by the special compute clock (CC) and read/write clock (R/W) pulses, and delayed pulses derived from them, on a line 'special dly R/W' shown in Figure 2-27. Each output latch is on for 250 ns and overlaps the next output latch by 125 ns as indicated in Figure 2-25.

The four outputs of the read/write clock, Read Clock 1 to Read Clock 4 (RC 1 to RC 4), define the read portion of the storage cycle, while the four outputs, Write Clock 1 to Write Clock 4 (WC 1 to WC 4), define the write portion. A storage cycle consists of the pulses RC 1 to WC 4 and takes 1 μs.

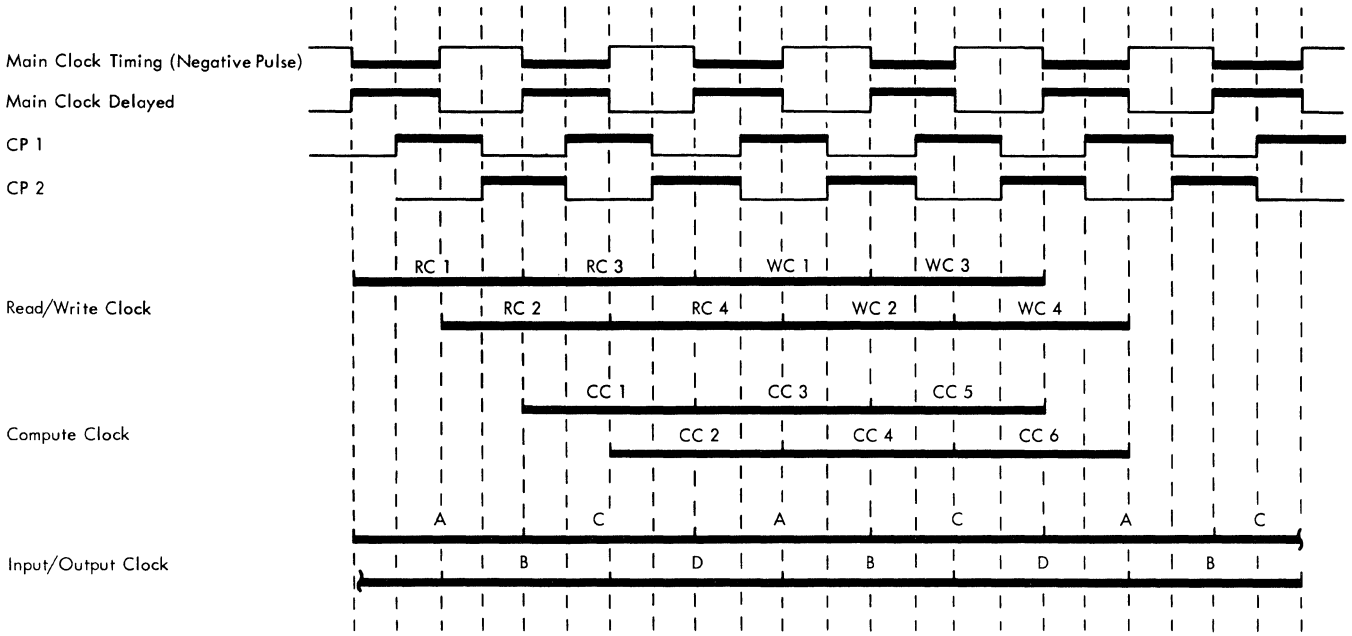


Figure 2-25. Relationship of Clock Pulses

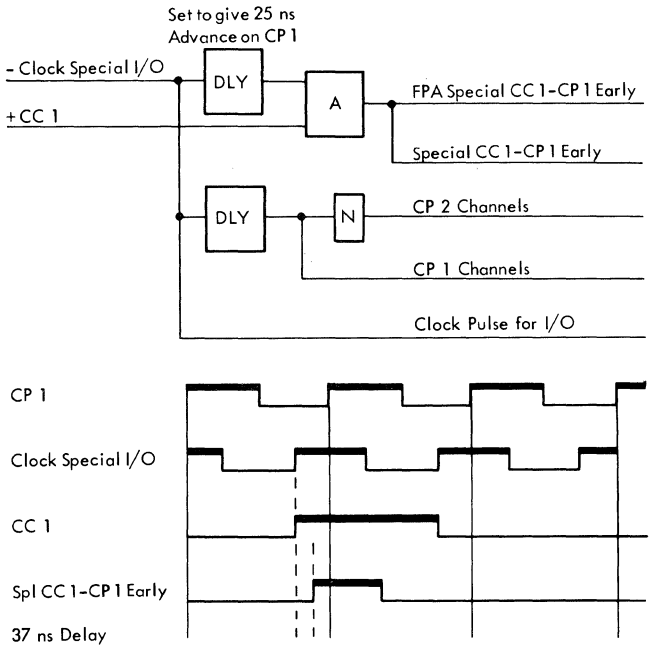


Figure 2-26. 'Special CC1 - CP1 Early' Pulse

Split Cycling

Split cycling is used when time can be saved by delaying the writing back of data into storage. If the result of an operation will be available within six cycles from read time, it is advantageous to delay the write cycle until the six cycles are concluded, then to write back the results of the completed operation. See Figure 2-28 for timing comparison.

Compute Clock

The compute clock is used for moving data from register to register throughout the CPU, except for data movement exclusively in the input/output channel.

The compute clock, which is formed and driven in a similar manner to the read/write clock, consists of six output latches. The compute clock is asynchronous with respect to the read/write clock, except that odd pulses of both clocks are defined by CP 1 pulses and even pulses by CP 2 pulses. Therefore, when both clocks are running together, odd pulses of one clock will always overlap odd pulses of the other. See Figures 2-25 and 2-29.

The compute clock may run in any of the following cycles depending on the operation being performed:

CC 1 to CC 2
CC 1 to CC 4
CC 1 to CC 6

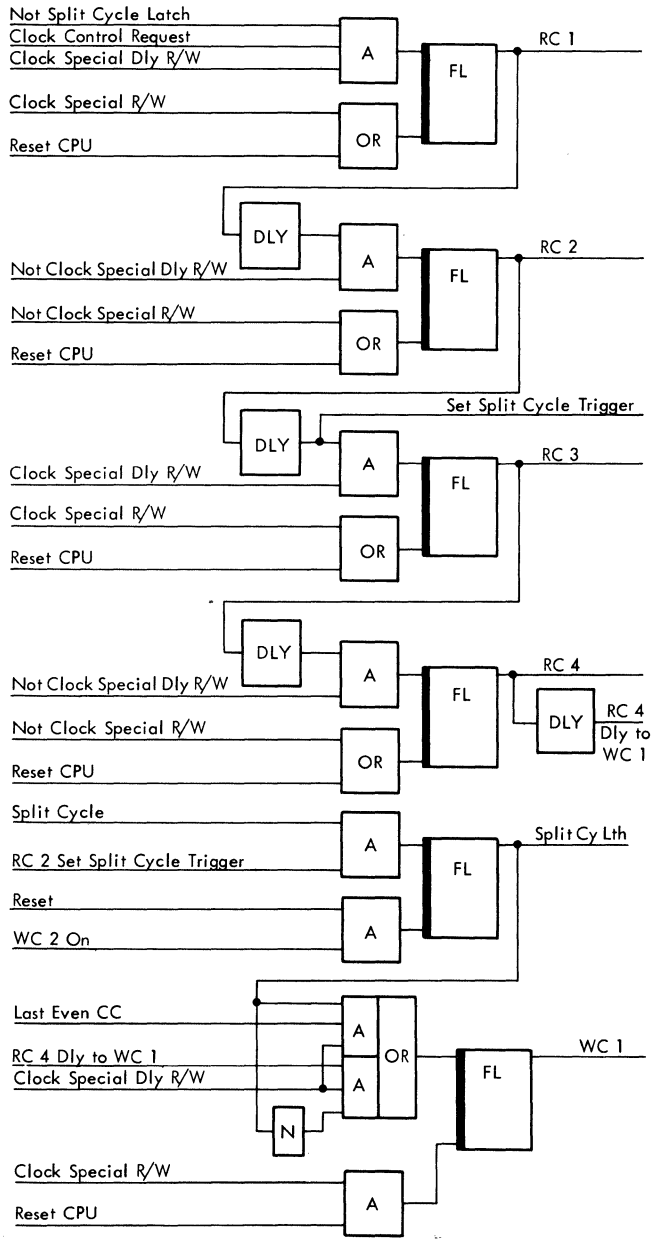


Figure 2-27. R/W Clock RC1 to RC4 and Continuation to WC1

One complete compute clock cycle may be followed immediately by another compute clock cycle as required by a particular operation (for example, multiply).

It is sometimes necessary for the compute clock to cycle CC 1-CC 2, CC 1-CC 2, and so on. The 'wrap compute clock at 250' latch is set on at the beginning of CC 2 to accomplish this. A signal to start the compute clock goes to the CC 1 latch at the same time as a 'stop' signal is sent to the CC 3 and CC 5 latches. Thus the compute clock is prevented from running on CC 1-CC 2-CC 3, etc. but is started again at CC 1.

If, during compute cycles, main storage is not being accessed (the read/write clock is not running), the I/O channels may have access to main storage by starting the read/write clock for an I/O cycle. Refer to "Cycle Controls" in this chapter for further explanation.

I/O Clock

The I/O clock is provided for channel-only functions and is physically similar to the previously-described clocks. Four output latches are driven by special I/O pulses from the main oscillator circuit. The I/O clock differs, however, from the other clocks in that the latches form a continuous ring so that, when the clock is started, it runs continuously while machine power is on (see Figure 2-25).

The four outputs are A, B, C and D and, as with the other clocks, the pulses are defined by the clock pulses CP 1 and CP 2, A and C pulses are defined by CP 1, and B and D are defined by CP 2.

Interface Clock

The interface clock is used only for multiplexor channel 0 operations. The clock is generated from 2-Mc pulses (clock special I-F) that are derived from the I/O clock pulse A delayed by 15 ns. Figure 2-30 shows the interface clock generation.

The generating pulses are continuously running, but the interface clock runs only when it is gated. It is started on any clock special I-F pulse. The interface clock comprises two sets of parallel pulses:

Overlapping pulses of 485-ns duration on four lines.

Non-overlapping pulses of approximately 235-ns duration on four lines.

An interface clock cycle lasts for approximately 1.25 μs.

Clock Distribution

The distribution of all pulses is designed to compensate as much as possible for the shortening of pulses caused by their transit through logic blocks. CP 1 is adjustable by 5-ns increments and is set to be mid-way in the odd-numbered pulses of clocks. CP 2 is out of phase with CP 1 and delayed by 8 ns.

The reference point of all clocks is the rise of the CP 1 pulse at board A2 on gate A.

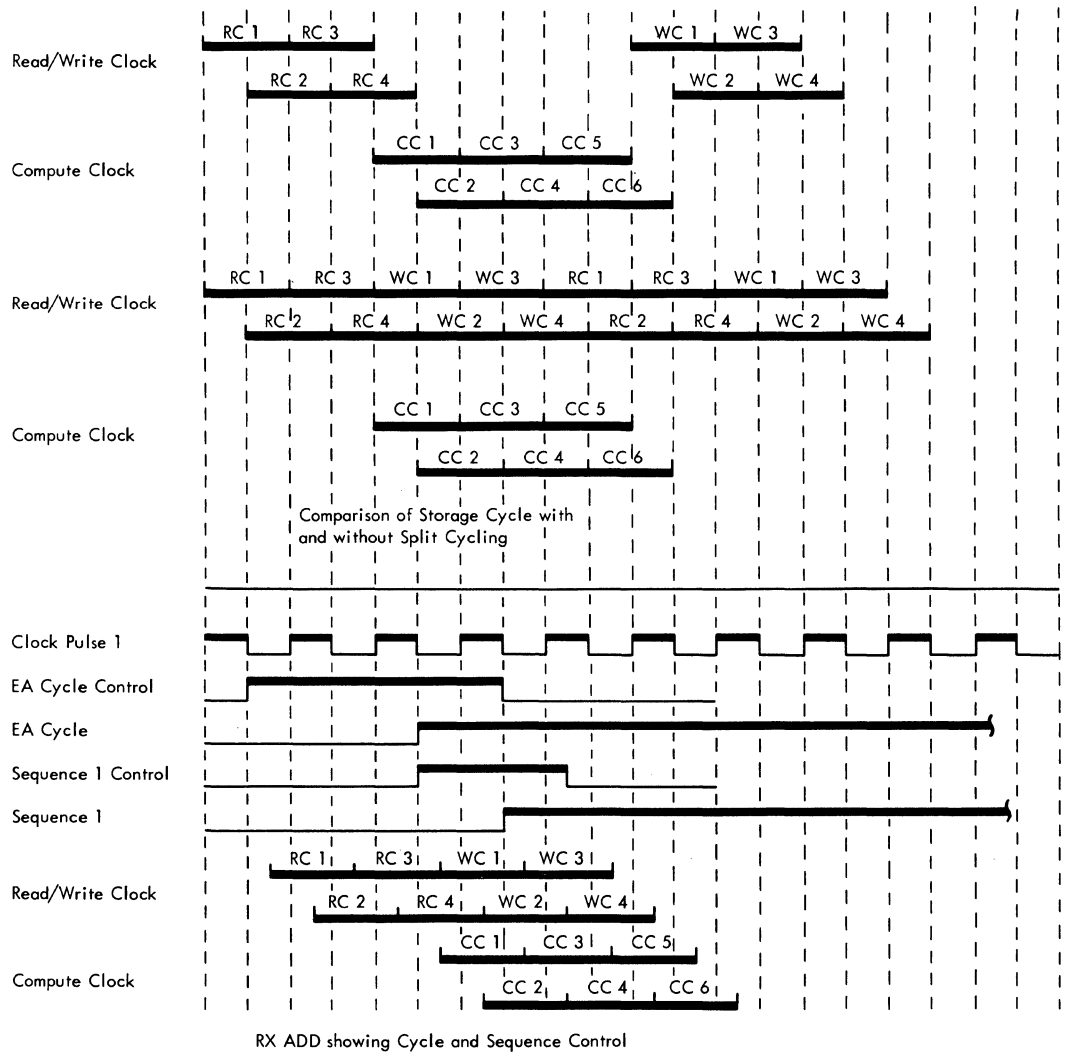


Figure 2-28. Timing Comparisons

Controls

Cycle Controls

- Ten types of storage-access cycles.
- Read/write cycles are defined.
- Only the cycles needed for a particular operation are initiated.
- Each of the ten types of access has a 'cycle control' latch and a cycle latch.
- Setting the 'cycle control' latch on also starts the read/write clock.

The normal sequence of the 2044 when obeying program instructions stacked in storage is: initial program load, I-fetch, execute, I-fetch, execute and so on, until the final instruction is executed.

The link between the previous instruction and the next I-fetch operation is the signal 'last execute' and 'no interrupt request' which sets the I-cycle control latch initiating the next I-fetch. The execute phase can overlap the next instruction phase by 250 ns.

During I-fetch, the first part of the instruction containing the operation code is read into the instruction registers and is decoded into signals which control the execution of the instruction. These control signals determine the order in which storage cycles and compute cycles are to be taken.

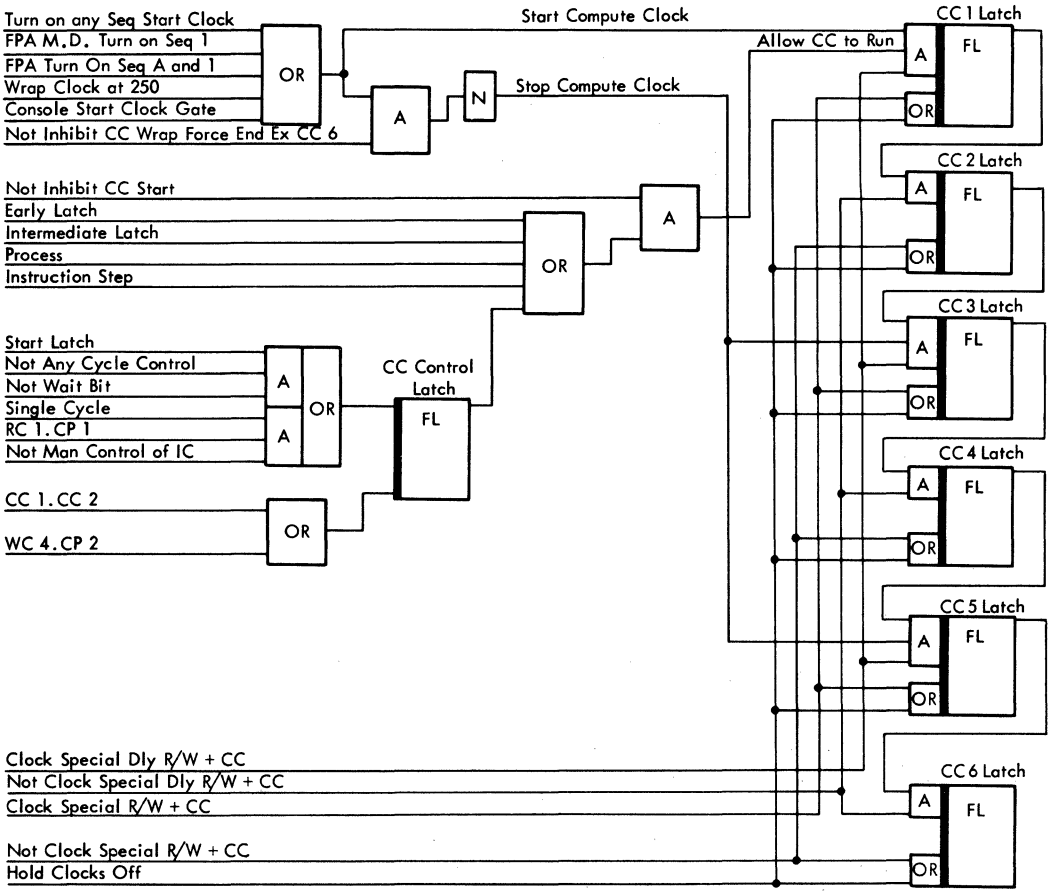


Figure 2-29. Compute Clock

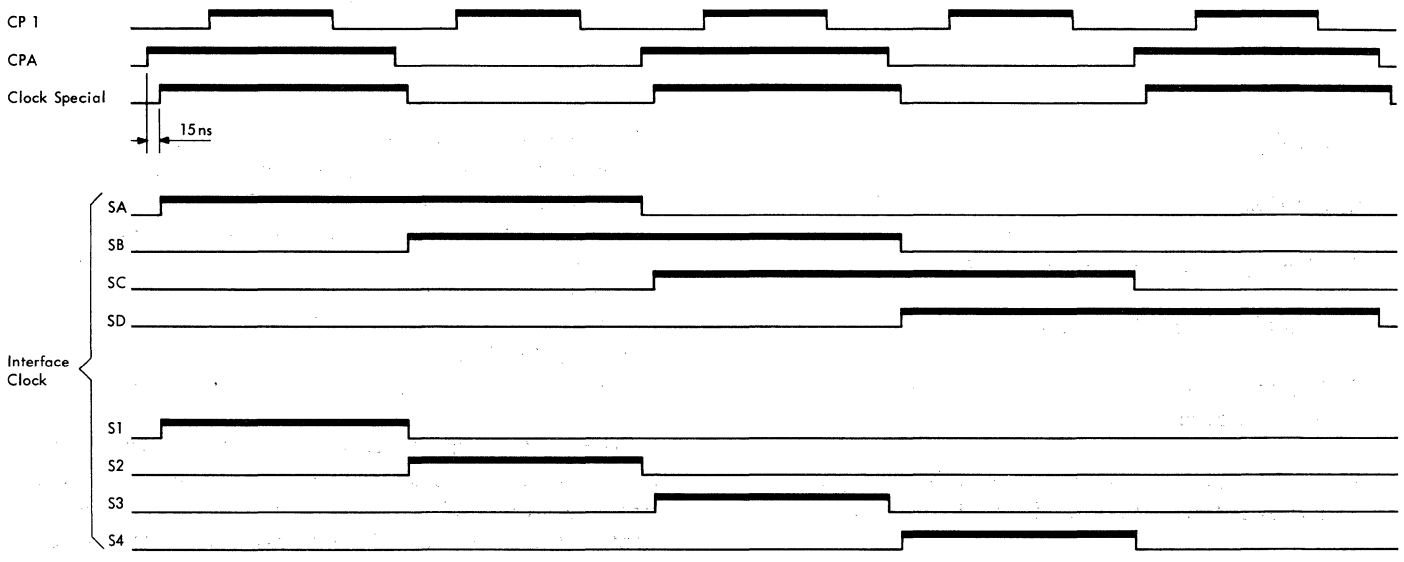


Figure 2-30. Multiplexor Channel 0 Interface Clock

The classification of the ten types of main-storage access is shown in tabular form in Figure 2-31.

Each group has a 'cycle control' latch and a cycle latch as shown in Figure 2-32. The read/write clock is started by a request to set a 'cycle control' latch on. Thus, every time the read/write clock is used, the type of cycle is defined as one of the ten groups shown in Figure 2-31.

Only those cycles needed for a particular operation are initiated (see Figure 2-28). The cycle controls are parity checked. If an even number are on a 'cycle control' check occurs.

Cycle control latches are set and reset as shown in Figure 2-17. Cycle latches are all set at RC 4, CP 2 time and are reset at RC 3, CP 1 time of the next read/write cycle. Thus, there should always be either a 'cycle control' latch on or a cycle latch on during normal operation.

To handle the PSW during interrupts, the interrupt cycles are further defined by two special sets of control latches. The PSW 1 control latch and PSW 1 latch are used to define the first and second interrupt cycles. The PSW new control period is used for the first and third cycles, and the PSW old control period is used for the second and last (fourth) cycles. Therefore, each cycle is clearly defined.

Cycle Priority

I/O operations always take precedence over CPU operations.

I/O data requests are sampled at the I/O cycle control latch one clock pulse before any other cycle requests are sampled. If an I/O request for storage access is found, the next CPU cycle control latch is allowed to be set but its effect (output) is inhibited or

	Purpose	Address Latch	Cycle Control Latch turned on by	Turn On	Turn Off	ALD Ref
I	Fetch Instructions	Instruction Counter of PSW 2	1. Console Leave Wait State 2. End Execute. NOT T Cycle Req. NOT Wait Bit. NOT INT Cycle Req. 3. System Reset. NOT PSW to be loaded	Untimed CP2 CP5	WC2-CP2 or Console Wait State	KC381
B	Fetch Core Base Reg Contents	Rc	B Cycle Request	WC4.CP2	WC2.CP2	KC371 KC501
X	Fetch Core Index Reg Contents	Ra	X Cycle Request	WC4.CP2	WC2.CP2	KC371 KC501
R1	Fetch Core R1 Operand	Ra	1. R1 Cycle Request 2. R1 Required (Compute Clock Running)	WC4.CP2 CP2	WC2.CP2	KC361 KC511 KC801
R2	Fetch Core R2 Operand	Rb	R2 Cycle Request	WC4.CP2	WC2.CP2	KC362 KC501
EA	Fetch Operand from Effective Add	B Reg	1. EA Cycle Request 2. Force EA Cycle and PSW 1 Latch (PSW Restart IPL)	WC4.CP2 CC6 I/O CP.WC2	WC2.CP2	KC363 KC521 KA121
I/O	Fetch or Store I/O Data or Fetch Control Information	Channel	1. I/O Request with any Cycle Control ON 2. I/O Request. NO Cycle Controls, RC1 to coincide with I/O clock A or D	WC3.CP1 Only CP1	WC2.CP2	KC352
INT	For Status Switching due to Interrupts	Forced Fixed Loc.	1. End Execute, Interrupt Cycle request 2. Interrupt Cycle Required (Int. Cycle NOT Old PSW 2 period NOT Wait for I/O because they are slow OR I/O Interrupt Latch. C Cycle. NOT C Cycle Req. Latch)	CP2 WC2.CP2	WC2.CP2	KK001 KK043
T	Update Interval Timer (Every 20 ms or 16.6 ms)	Forced Fixed Loc.	End Execute. Update Timer SS NOT Interrupt Cycle Request	CP2	WC2.CP2	KK101
C	Fetch Channel Address Word and change Channel Status Word	Forced Fixed Loc.	C Cycle Request. I/O Clock Pulse D (Dummy AND Position Available.)	CP2	WC2.CP2	FC351 FZ141

Figure 2-31. Cycle Control Chart

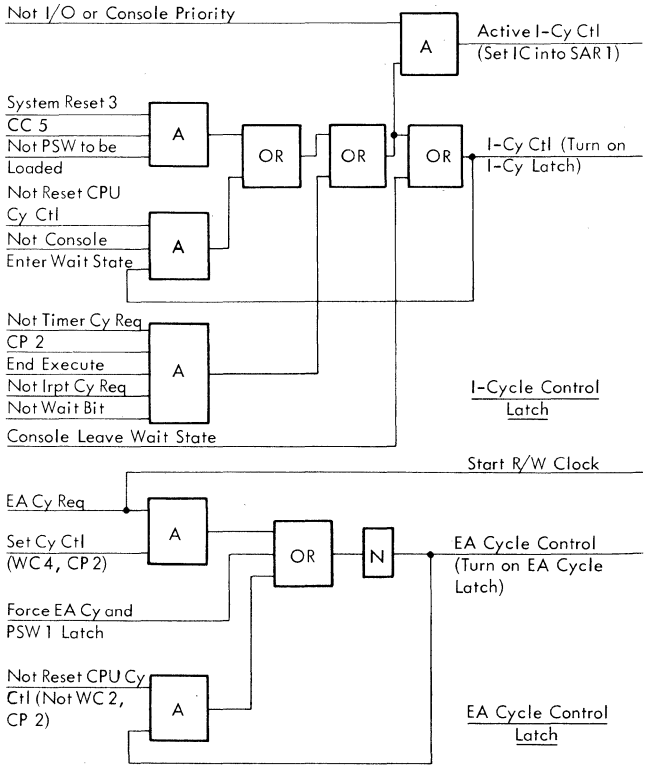


Figure 2-32. Examples of Cycle Control Latches

gated off until the I/O operation has finished with storage. Thus a means is provided of remembering the next CPU cycle to be resumed after the I/O operation has finished with main storage. See Figures 2-33 and 2-34.

An interrupt request initiates an interrupt cycle at 'end execute' and holds off the normal I-cycle. Similarly, a timer cycle is initiated at 'end execute' if there is an output from the update timer single-shot and 'interrupt request' is not up. C cycles occur during an I/O operation or during an I/O interrupt sequence. In an I/O operation, they are controlled by the operation being performed and occur as required. C cycles are used for fetching the Channel Address Word (CAW) and for manipulating the Channel Status Word (CSW).

All other cycles occur as required after I-cycles except that EA cycles can be forced on at the end of Initial Program Load (IPL) and 'force PSW restart' (see Figure 2-32).

Interrupts

Interrupts are taken between the end of the E-phase and the start of the next I-phase.

I/O data requests are serviced between storage cycles on a "cycle-stealing" basis. If the CPU is occupied by a succession of compute cycles, but main storage is not being used, an I/O data service accesses storage without interfering with the progress of the instructions. I/O data cycles require only the use of SAR and Storage Data Register (SDR) so that the rest of the data flow registers are undisturbed when I/O breaks in.

There are a few cases (PSW and CSW handling) where the CPU must take two consecutive main-storage cycles (double cycle) without interference. I/O operations are held off by a 'double cycle' latch which makes the two storage cycles appear as one cycle to the channels.

Sequence Controls

- Five sequence controls.
- Compute clock cycles are defined by sequence controls.
- The number and order of sequence depend on the operation being performed.

In some operations, more than one complete compute clock cycle of the full six pulses CC 1 to CC 6 may be needed. Sequence controls are used so that each pulse of an operation may be exactly defined.

Each compute clock is defined by one of five sequence controls numbered 1 to 5. In some operations, the same set of events may need to be repeated, in which case the same sequence control is repeated.

Single Cycle Mode

The 2044 can be operated for diagnostic purposes in 'single cycle' mode. This mode permits instructions to be stepped through manually, one cycle at a time, from the console. While the rate switch is in the single cycle position, the machine is in the hardstop state.

Single cycle operations are as follows:

1. If a cycle control latch is on and no 'sequence control' latch is on, the machine takes a full read/write clock cycle. If, during this cycle (prior to WC 4, CP 2) the compute clock is started, the compute clock cycle is completed as well as the read/write cycle.
2. If a 'sequence control' latch is on and no 'cycle control' latch is on, the machine takes a compute clock cycle and stops at the end of that cycle (CC 2, CC 4 or CC 6).
3. If a 'sequence control' latch is on and a cycle control latch is on, the machine takes the compute clock cycle and stops with the cycle control latch still on. The next depression of the start push-button produces the read/write clock cycle.

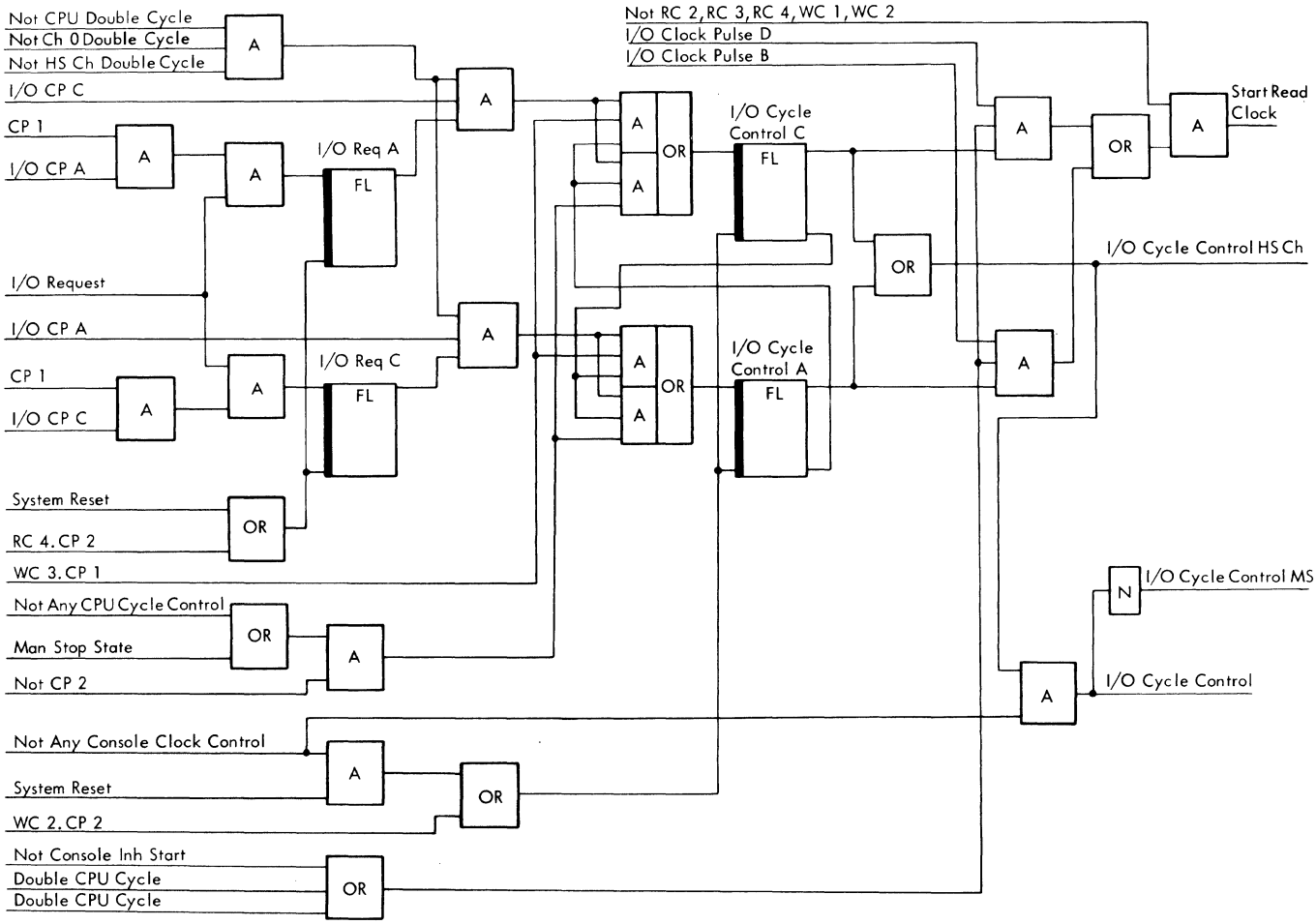


Figure 2-33. I/O Cycle Control Latches

4. If a split cycle is initiated, the machine completes the split cycle and stops at the end of WC 4.

5. If a double cycle is initiated, the machine takes both read/write clock cycles before stopping.

6. When single cycling channel-to-CPU operations, the machine stops at the end of each main storage cycle.

7. When single cycling channel-to-device operations, the machine stops at each outbound tag request, that is, when the in tag is received and the out latch is set, but immediately prior to setting the 'out tag' latch.

NOTE: Although sequencing is stopped, the setting of cycle controls and sequence controls is not inhibited. This ensures that the CPU will restart at the correct read/write or compute cycle when the start pushbutton is operated.

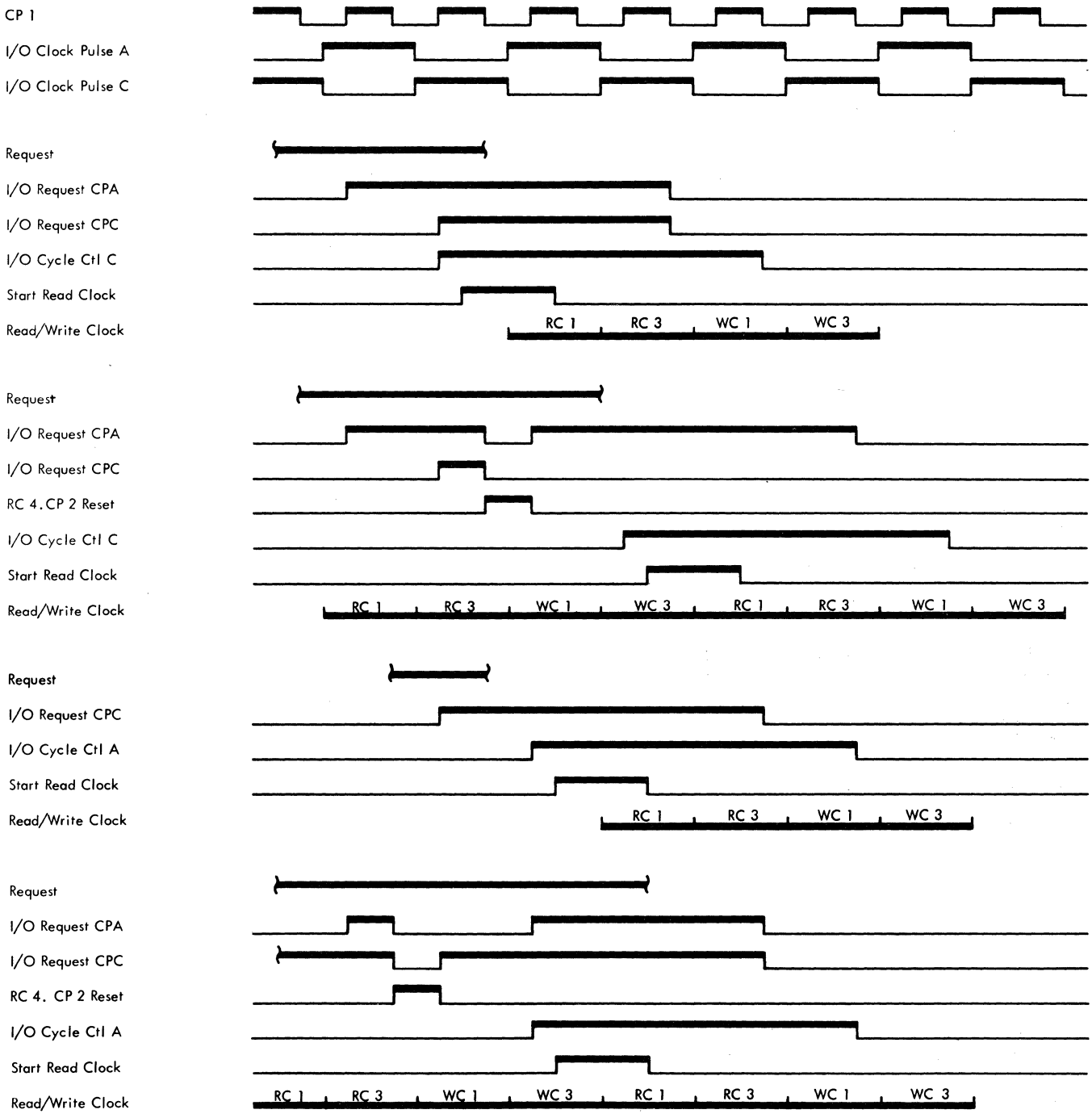


Figure 2-34 I/O Cycle Control Timing Chart

MAIN STORAGE ADDRESSING

INTRODUCTION

- The IBM 2044 CPU has a split main storage.
- Main storage output is 32 bits (4 bytes) wide.

In order to address a halfword or word in main storage an address register is required. In the 2044 CPU, two Storage Address Registers (SAR 1 and SAR 2) are used. A register, called the Storage Data Register, is also required to read data into or out of main storage.

The 2044 has a split main storage with an output one halfword wide from each half. The main storage consists of two or more basic operating memories, each having a capacity of 16K or 32K bytes (9 bits), dependent on the model. A storage capacity table of the models and further details of storage capacity and locations are contained in the "Main Storage" section of this chapter.

As the main data flow in the 2044 is 32 bits wide, and data must often be operated on in halfwords only, a facility for interchanging each 16 bits is required. The interchange function is performed by the true/criss-cross which is located on the data flow between the SDR and ABC funnel.

Storage Address Registers

- Two registers are used: SAR 1 and SAR 2.

Two storage address registers are needed for addressing locations on halfword boundaries. Each SAR has a 17-bit capacity (numbered 14 to 30) and the contents can be displayed on the console. The least-significant bit in the instruction counter (bit 31) does not form part of the address in the SAR.

In order to obtain any full word instruction located on a halfword boundary in main storage, the address in the two SAR's must be different. This difference is accomplished by the same address being placed in both registers and the value of SAR 1 being increased by two, thus changing the address by one halfword. Figure 2-35 shows a typical position of the SAR.

Storage Data Register

The storage data register has a capacity of four bytes (32 bits), plus four parity bits. As bad parity must not be stored in main storage, the SDR checks the parity of its contents and if necessary, generates the required parity bit. For parity checking information refer to "Checking" in this chapter.

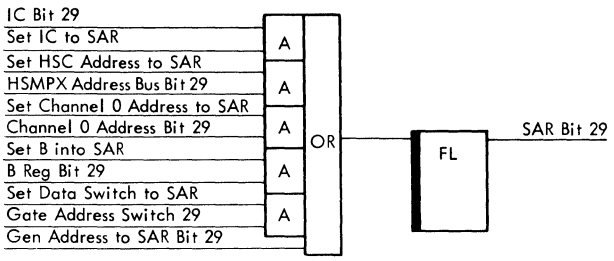


Figure 2-35. Sample Position of SAR

This register is used to place data into or receive data from main storage, sending data to the funnel or the channels and receiving data from the B register and the channels.

The contents of the SDR can be displayed by the console lights and can be changed by manual switching; refer to "Console" in FEMM IBM System/360 Model 44, Form Y33-0007. Figure 2-36 shows a typical position of the SDR.

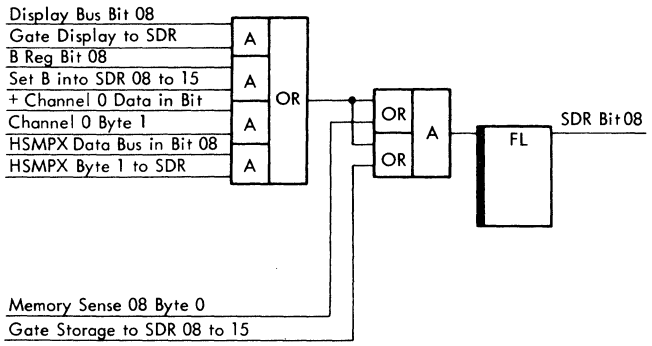


Figure 2-36. Sample Position of SDR

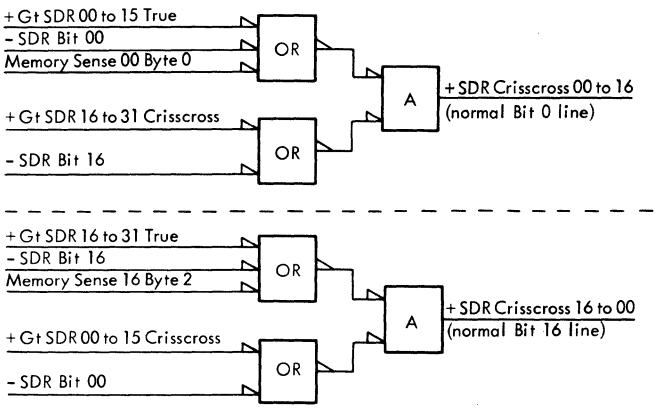
True/Criss-Cross Function

- Used for 16-bit interchange.
- Capable of producing all ones.

The true/criss-cross function is required to enable the interchange of the two halfwords contained in the SDR, should this be required by the operation. This function, which is logically contained in the output of the SDR, is performed as follows. Each bit position of the SDR output is controlled by one of four gates:

Gate SDR bits 0 to 15 true.
Gate SDR bits 16 to 31 true.
Gate SDR bits 0 to 15 criss-cross.
Gate SDR bits 16 to 31 criss-cross.

These gates control which of two bits from equivalent positions in the two halfwords (for example: bits 0 and 16) shall be placed on any given output line. The four gates and the way in which they are used are shown in Figure 2-37.



Gate True	Gate Crisscross	Output to Funnel
+	+	Impossible Condition
-	-	All One's
+	-	True Bit
-	+	Crisscross Bit

Figure 2-37. True/Criss-cross Output (Two Positions)

The control over which facility (true or criss-cross) is used is exercised mainly by SAR 2 bit 30. To have a criss-cross output from SDR, the conditions of this bit are opposed for instruction and data addresses, as follows:

Instruction addressing when SAR 2 bit 30 = 1.

Data addressing when SAR 2 bit 30 = 0.

These controls are explained more fully under the heading "Addressing" in this section.

An indirect feature of the true/criss-cross function is the ability to produce all ones at the input to the ABC funnel. The all-ones output is achieved by raising the four previously-mentioned gates and is used by the arithmetic registers under the control of a gating pulse at the ABC funnel; the resultant action of the four possible gate conditions is shown in Figure 2-37.

Instruction Counter and IC + 2 Carry Generator

- The instruction counter is contained in bits 8 to 31 of PSW 2.
- Updating the instruction counter is performed by the IC + 2 carry generator.

The instruction counter (IC) of the 2044 is located in the PSW 2 register, bit positions 8 to 31. Of the existing bit positions, only 15 to 31 are used to accommodate the instruction address. Bits 8 to 14 are compressed to check for a one bit (invalid address) which, if found, signals an 'address exception'. Bit 31 of the IC does not form part of the address in the SAR and, if set during an I-cycle, will cause a 'specification exception' to occur. For the handling of both these conditions, refer to "Checking," in this chapter. Bit 31 of the IC can however be set, for example, in store or insert character operations and I/O control. In these instances, it is used to set the 'odd address' latch which is subsequently used in the particular operation.

The instruction counter is updated by the IC + 2 carry generator. It is incremented by two or by four, by the IC + 2 function being performed either once or twice depending on whether the instruction is on halfword or word boundaries. The IC + 2 carry generator also modifies the instruction counter output to SAR 1 (see Figure 1001 in the FEMD, Form Y33-0008) by changing the address by plus two; the change is accomplished by placing the instruction counter contents in SAR 1, changing the state of bit 30, and propagating any carry.

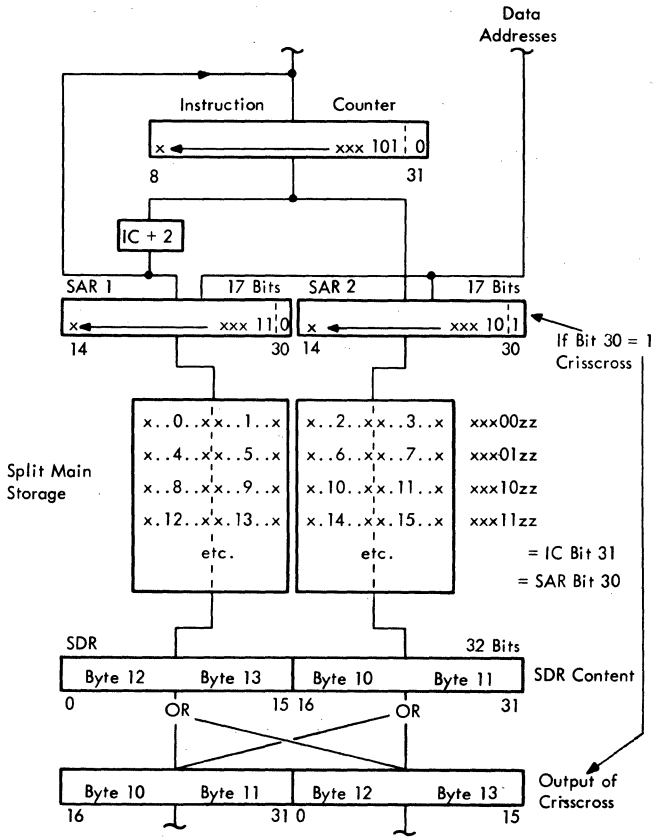


Figure 2-38. Instruction Address with Criss-cross

ADDRESSING

- There are two types of addressing: for instructions or for data.
- The contents of SAR 1 and SAR 2 are the same only during data addressing.

Main storage can be addressed for either instructions or data. When instructions are addressed, the SAR contents differ as SAR 1 is incremented by 2; whereas when data is addressed, the address on the data bus is placed directly into the SAR's.

Instruction Address

The address contained in the instruction counter consists of 17 bits, but, when this address is placed into the two SAR's, the least-significant bit in each SAR is ignored as it does not form part of the required address. The SAR's address their respective halves of storage for one halfword from each, these halfwords being placed in the SDR.

SAR 2 bit 30 is used to control the true/criss-cross operation. In instruction addressing, when SAR 2 bit 30 equals 1 a criss-cross will be performed on the data contained in SDR when it is gated to the ABC funnel. If SAR 2 bit 30 equals zero, no criss-cross operation is performed as the SDR contents are in the required format.

Figure 2-38 shows an instruction address with criss-cross.

Data Address

Of the 17 bits in the data address, only 16 are placed on the data address bus to the SAR's.

As with instruction addressing, the least-significant bit of each SAR (bit 30) is not used for addressing purposes and, similarly, bit 30 of SAR 2 controls the true/criss-cross. However, in this instance, the need to criss-cross is governed by the following conditions:

The operation is a data fetch
SAR 2 bit 30 = 0
Operation is on halfword operands

If any of these conditions is not satisfied the data in SDR will be gated direct on to the data flow.

Figure 2-39 shows data addressing with a resultant true output.

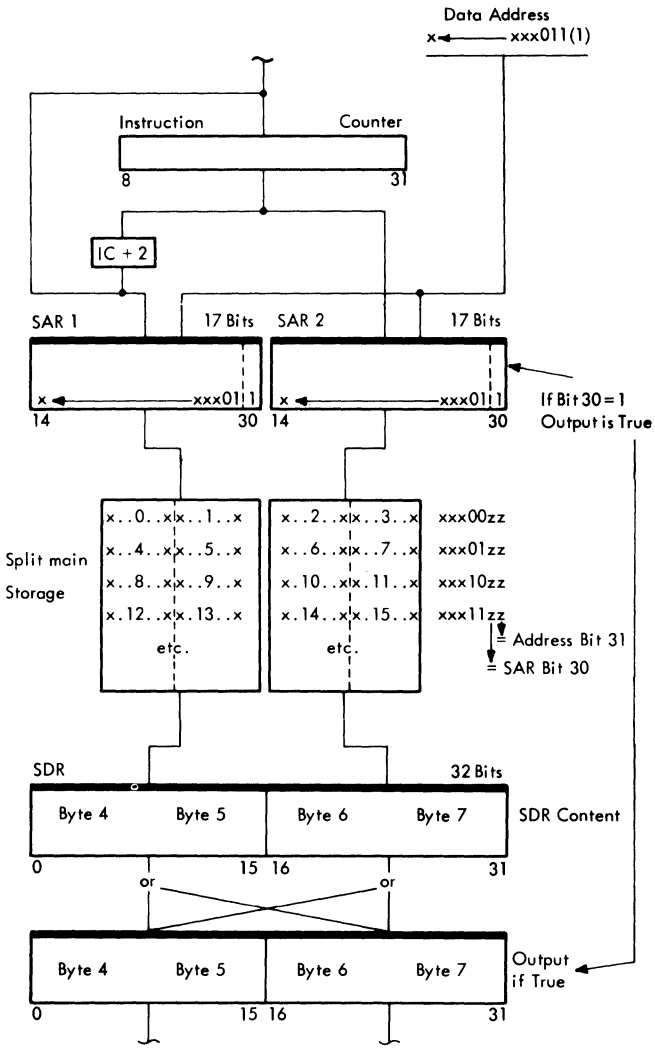


Figure 2-39. Data Address with True Output

MAIN STORAGE

INTRODUCTION

- The main storage is controlled by the CPU.
- 1.0 microsecond read/write cycle.
- Full word (four bytes) operation is provided.
- The main storage uses SLT circuitry and packaging.

The main storage for the IBM 2044 CPU is a magnetic core storage. The functions of the storage unit are to accept and retain data input from the CPU or from input/output units, and to deliver data to the CPU or the I/O units. The storage unit handles all input and output as data, making no distinction between program instructions, control words or data. On each access, the storage either writes or reads one full word (four bytes of eight bits with parity for each byte).

Capacity

- Three sizes of main storage are available: 32K, 64K or 128K bytes, dependent on model.
- The extension storage is additional, its size depending on model.

Main storage consists of two sections:
1. The main storage section
2. The extension storage section.

The extension storage is not addressable by the programmer, except by using the diagnose instruction. Extension storage is a reserved area of main storage for channel Unit Control Words (UCW's), basic GPR's, and FPR's.

16K or 32K byte modules of storage known as Basic Operating Memories (BOM's), are assembled as gates to make up the required storage sizes of the IBM 2044. The BOM has data positions that are only 18 bits wide; to provide the simultaneous, full word, 36-bit (32 bits of data plus four parity bits) access that is required, two BOM's are addressed at the same time. The two 18-bit output words from the two BOM's fill each half of the 36-bit wide storage data register.

Main storage is available in three sizes while the size of the extension storage section increases with the size of main storage to a maximum of 8K bytes.

The storage sizes of the various models are as follows:

Model	2044E	2044F	2044G
Main Storage Size (Bytes + Parity Bits)	32K (32,768)	64K (65,536)	128K (131,072)
Extension Storage Size (Bytes + Parity Bits)	2K (2,048)	4K (4,096)	8K (8,192)
BOM Array Size (Bytes)	16K	32K	32K
Quantity of BOM's	2	2	4

The terms 32K, 64K and 128K (number of bytes of main storage and extension storage shown in the table) are used to denote a specific storage size, and are employed in the remainder of this description of main storage.

In order to provide the maximum main storage size of 128K bytes, four BOM's, each of 32K bytes, are used.

In both read and write cycles four bytes are handled. Two storage units are addressed simultaneously, each unit handling one halfword.

Speed

- 1.0 microsecond read/write cycle.
- Split-cycle operation with a CPU write cycle always following a CPU read cycle.
- Storage access time is approximately 450 nanoseconds.

The cycle time is defined as the time required by the CPU for reading out or writing a full word.

In a CPU read cycle a full word is read from main storage, one CPU read cycle taking 0.5 microseconds (μs). In a CPU write cycle, a full word is written into main storage, one CPU write cycle taking 0.5 μs. There may be a gap of indefinite duration following the CPU read cycle (split-cycle operation), but a CPU write cycle must always follow a CPU read cycle. The CPU read cycle develops a signal called 'read call' which is sent to the storage to initiate a storage read cycle. The maximum period between the time when the read call pulse from the CPU reaches the main storage logic, and when the data (read from the storage) is available to the CPU, is known as the "access" time; this time is of approximately 450 ns duration.

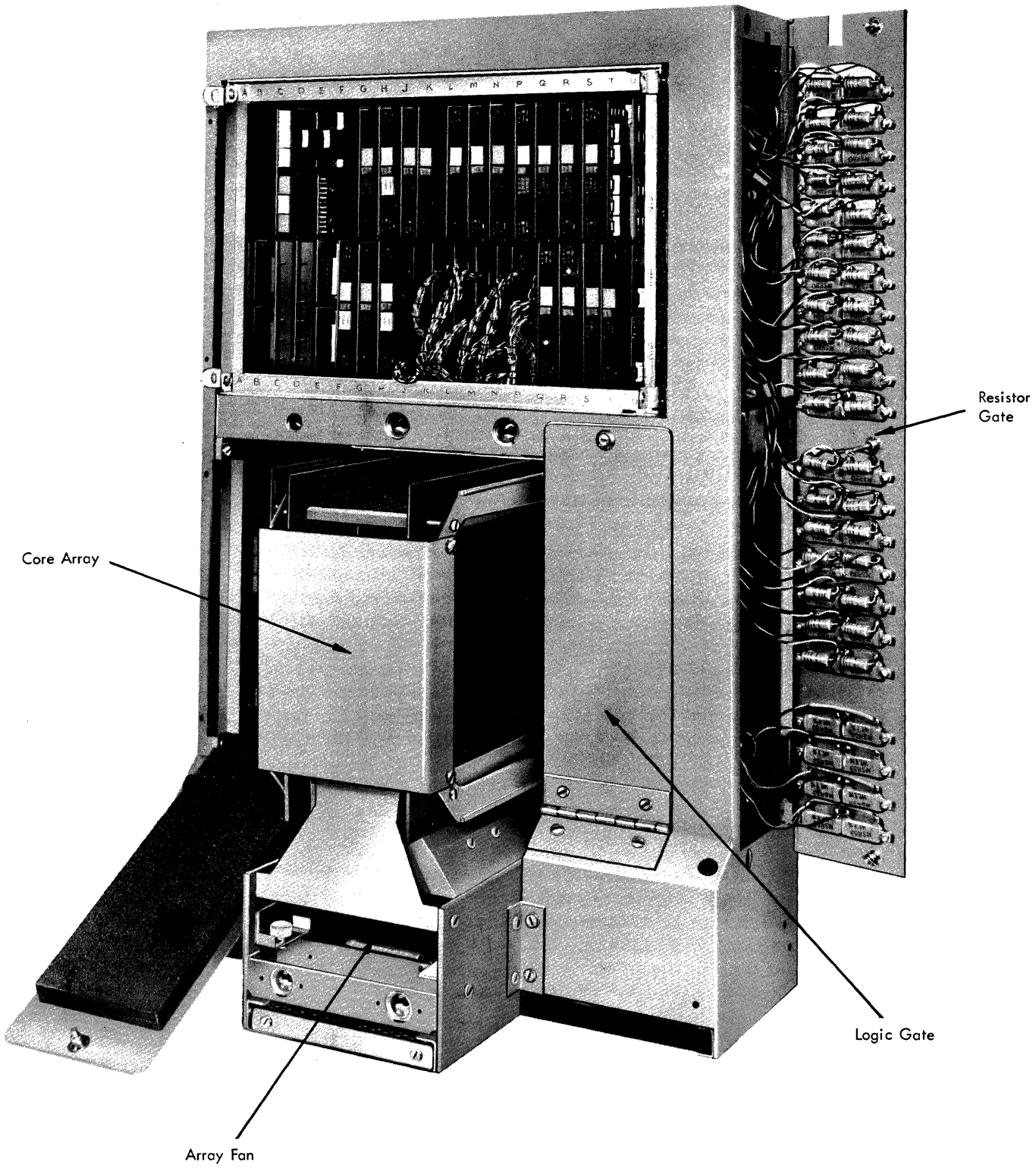


Figure 2-40. 2044 Storage Unit (BOM)

Storage Cycles

- The system specifies read or write operation.
- The cycles have asynchronous operation.
- The system provides the start signal.
- The system provides the address.
- The storage data register provides data to be stored and accepts data read-out.

The main storage integral circuits control the reading or writing of data. The CPU sends to the main storage logic circuitry the basic commands 'read call' or 'write call'. The storage cycle then proceeds under control of the storage unit clock, independently of the CPU timing.

Because the storage units are entirely independent of the 2044, communication between the two takes place over a number of signal lines which form collectively the storage/CPU interface (refer to "Core Storage Unit Interface" in this section). Essentially, this interface transfers address information, input data, output data, and timing signals. The basic data flow is as follows:

1. The CPU places the address into SAR 1 and SAR 2.
2. SAR 1 provides the address for storage units (BOM's) 1 and 3. SAR 1 bit 15 selects either storage unit 1 or 3 (when SAR 1 bit 15 = 0, storage unit 1 is selected and when bit 15 = 1, storage unit 3 is selected).
3. SAR 2 provides the address for storage units 2 and 4. SAR 2 bit 15 selects either storage unit 2 or 4 (when SAR 2 bit 15 = 0 storage unit 2 is selected and when bit 15 = 1, storage unit 4 is selected).

These controls are summarized in the following table (applicable to 128K storage only):

SAR	Bit 15	Storage Unit Selected
1	0	Storage Unit 1
1	1	Storage Unit 3
2	0	Storage Unit 2
2	1	Storage Unit 4

At the appropriate time, the storage unit is signalled by the CPU to begin a storage read cycle (read call). The halfwords located at the addresses specified by SAR 1 and SAR 2 are read out of the storage and placed on the data line to the SDR. When the storage read cycle is complete, the storage stops and awaits a write cycle.

The full word of data read out may be placed back into storage (the two halfwords to the locations specified by SAR 1 and SAR 2) or a different full word may be placed in these addressed locations on the following CPU write cycle.

The full word to be placed into main storage is first set into the SDR. The CPU then signals the storage to begin a storage write cycle (write call). The storage timing circuits start and the same positions are again addressed. This time, however, a write operation is performed and the full word in the SDR is placed into storage. One halfword, bytes 0 and 1, is placed in the location addressed by SAR 1; the other halfword, bytes 2 and 3, is placed in the location addressed by SAR 2.

FUNCTIONAL UNITS

Basic Operating Memory

The basic operating memory is shown in Figure 2-40.

The BOM is available with two different sizes of storage array. The smaller of these is a nine-plane array containing 16K bytes of main storage and 1K byte of extension storage, while the other is an 18-plane array which contains 32K bytes of main storage and 2K bytes of extension storage. The BOM also contains the array heater and fan, two logic gates and their fans, and the resistor gates. The three storage sizes of the 2044 are made up from these basic modules as shown in the preceding table in "Capacity" and in Figure 2-41.

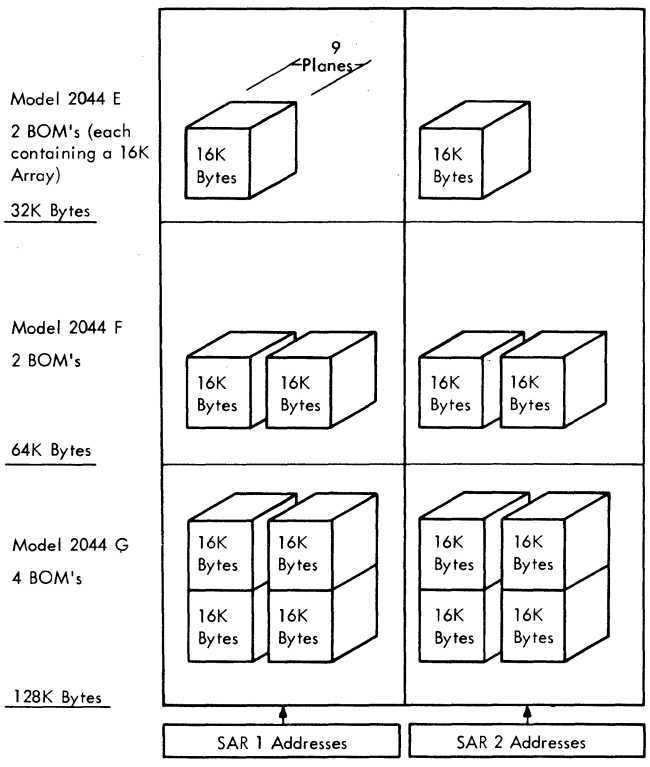


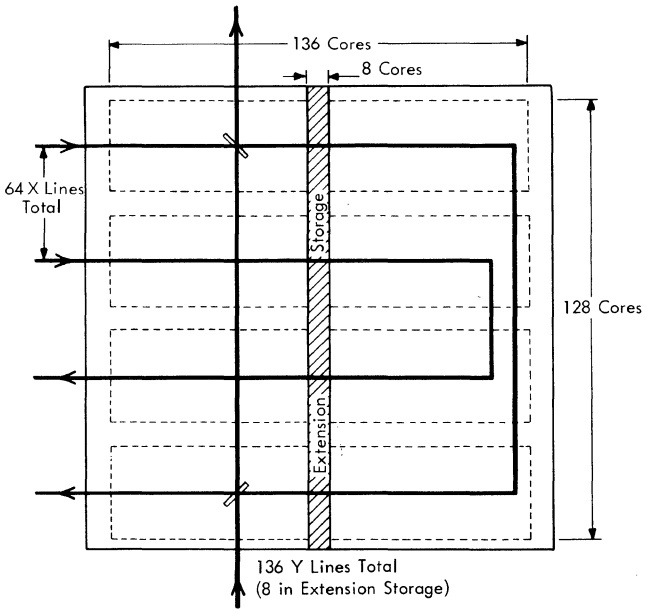
Figure 2-41. Main Storage Capacity

Core Array

- The core array consists of a number of discrete core planes.
- Three wires pass through each core.
- The horizontal drive lines are called X lines.
- The vertical drive lines are called Y lines.
- A combined sense/inhibit line is used.

The arrangement of the 32K storage array is shown in Figure 2-44. An array is built from a number of core planes. Each plane is constructed of a plastic frame (approximately 6-1/2 inches square) that is crossed horizontally by 128 X lines and vertically by 136 Y lines and fitted at each intersection of X and Y lines with a ferrite core. Of the 17,408 cores in each plane, 16,384 are used for main storage and 1,024 for extension storage.

The core plane consists of four sections. Each X line passes through two of these sections (Figure 2-42) and, therefore, each X line crosses any Y line at two points; thus two bit positions can be addressed in each plane by the selection of a single pair of X and Y lines.



Total: 17,408 Cores
(16,384 Cores in Main Storage and
1,024 Cores in Extension Storage)

Figure 2-42. Core Plane

A combined sense/inhibit line in each section is wound parallel to the X lines and also passes through each core as shown in Figure 2-43.

The 16K storage array consists of nine planes, eight data bits and one parity bit. Two of these arrays are used to form the 32K array as shown in Figure 2-44. The two arrays share a common set of Y lines, each of which passes through all 18 planes. Two sets of X lines are used in the 32K array, one set in the 0 to 8K halfword addressable section, and one set in the 8 to 16K halfword addressable section.

In each section the X lines are divided into two groups. Thirty-two of these lines are selected by SAR bit 29 and pass through sections A1 and A2, while the remaining 32 are selected by Not SAR bit 29 and pass through sections B1 and B2. A core in section A1 or B1 is a byte 0 bit, while a core in section A2 or B2 is a byte 1 bit. As each X line selects two bit positions, bits 0 and 8, 1 and 9, 2 and 10, etc. are addressed simultaneously.

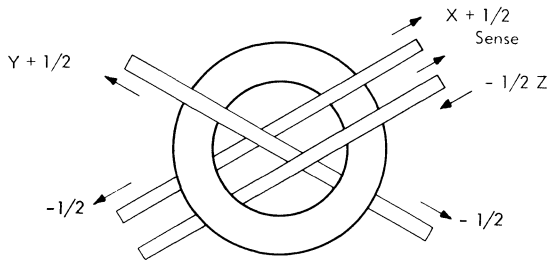


Figure 2-43. Core Wires (Three-Wire System)

Common Sense/Inhibit

- Sense/inhibit lines run parallel to X lines.
- Both legs of a sense/inhibit line are connected together at one end.
- Open ends feed sense amplifiers and inhibit current terminators.
- Looped end is connected to the inhibit driver.

The 32K storage array contains eight sections each having nine planes, therefore providing a total of 72 combined sense/inhibit lines for each array. The sense/inhibit logic is shown in Figure 2-45. Eighteen sense amplifier/inhibit driver cards are used in each storage unit, each card containing four sense amplifiers (SA's) and four inhibit drivers. The four sense/inhibit lines associated with each data bit in the array are connected to one card.

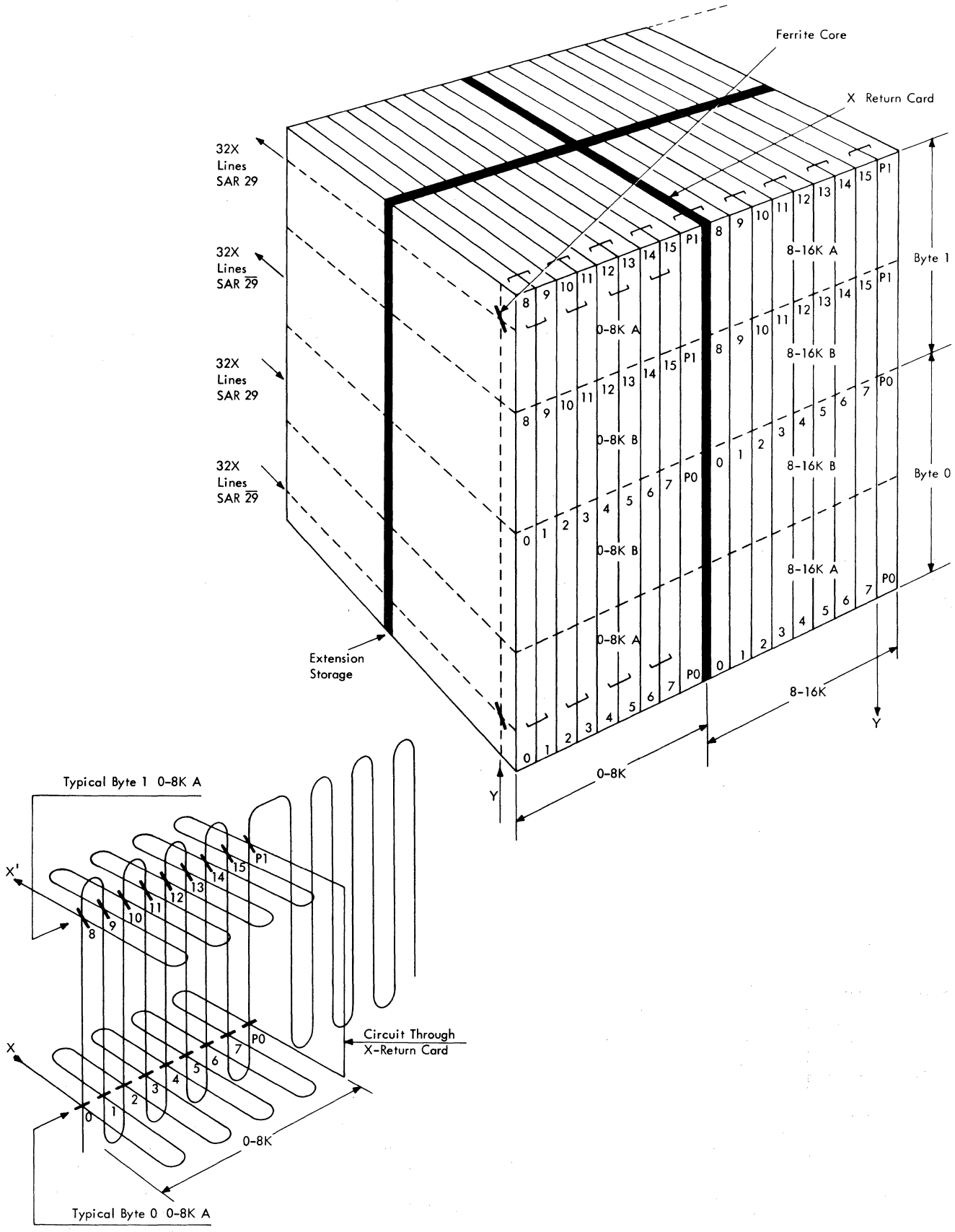


Figure 2-44. Storage Array (32K)

S A and Inhibit Gates Select Chart

SAR Bits			S A and
15	16	29	Inhibit Gates
Unit Select	0	1	0-8K A1 A2
	0	0	0-8K B1 B2
	1	1	8-16K A1 A2
	1	0	8-16K B1 B2

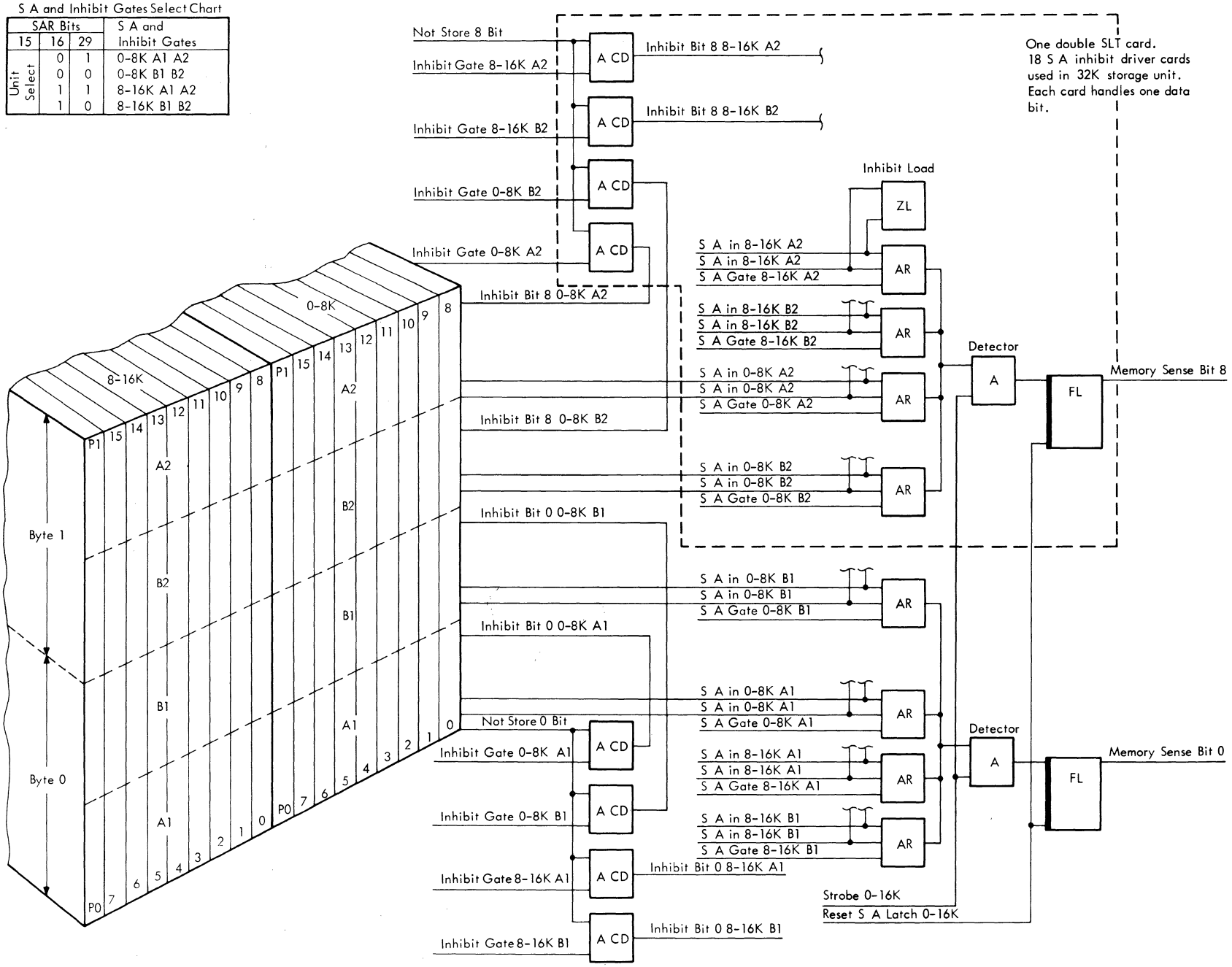


Figure 2-45. Common Sense/Inhibit Logic

As only one of the four locations associated with each bit can be active at one time, the sense amplifiers and inhibit drivers are gated, depending on SAR bits 16 and 29 as shown in the following table.

SAR Bits		8K Section	Section for which SA and Inhibit Drivers are active	
16	29		Byte 0	Byte 1
0	1	0 to 8K	A1	A2
0	0	0 to 8K	B1	B2
1	1	8 to 16K	A1	A2
1	0	8 to 16K	B1	B2

Sense

- The X and Y current attempts to flip all cores to zeros during the read cycle.
- Cores that were at one produce a sense pulse.

The logic of the sense circuit is shown in Figure 2-46.

During the read cycle the coincidence of an X and Y current will attempt to flip all cores to a zero state. Any core that contains a one bit will flip causing a pulse on the sense line, while any core that was at zero will remain at zero and therefore produce no pulse on the sense line.

The open ends of the sense line are connected to a sense amplifier which is gated by the contents of bits 16 and 29 of SAR. The output of the four sense amplifiers associated with each bit are dot OR'ed and fed to a detector which is gated by a strobe pulse derived from the core clock. The detector, in turn, drives a sense latch the output of which is fed to the SDR.

Inhibit

- The current in X and Y wires during the write cycle would write all ones.
- The inhibit current opposes the X current and prevents core flipping.

The inhibit logic is shown in Figure 2-46.

During the read cycle all cores in the addressed location are flipped to zero. During the write cycle the half-write current flowing in both the X- and Y-drive lines would flip the cores to ones. If the bit to be stored is a zero, inhibit current is raised in the sense/inhibit line in the opposite direction to the current flowing in the X line, thereby cancelling the half-write current in the X line and preventing the core from flipping.

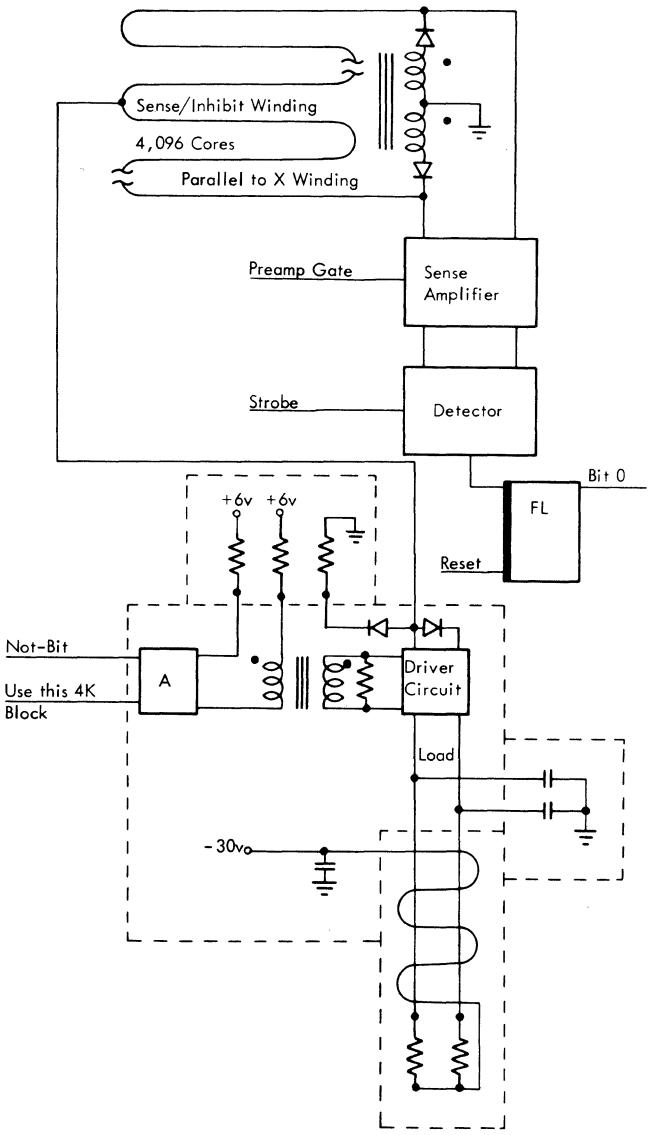


Figure 2-46. Sense and Inhibit Logic

Figure 2-46 shows a diagram of the inhibit driver arrangement for one core plane of an array. The outputs of the corresponding SDR bit positions are fed to all four inhibit drivers for that bit. The correct driver is then selected, depending upon the contents of SAR bits 16 and 29.

Core Storage Unit Interface

- The Core Storage Unit (CSU) used in the 2044 is a standard unit.
- The CPU communicates with the core storage by means of the CSU interface.

The core storage unit of the 2044 is a standard unit used also in other System/360 models. Therefore it is necessary to provide a standard set of signal lines (CSU interface) for communication between the CPU and core storage. All addresses, data and control signals are transmitted over this interface. The CSU interface is shown in Figure 2-47.

Storage Addressing

- SAR 1 is used to address storage units 1 and 3.
- SAR 2 is used to address storage units 2 and 4.

Fifteen lines are used to carry addresses from the SAR's to the storage units. The storage units always access a full word, therefore SAR bit 30 is not transferred from SAR to the storage units. If SAR bit 14 is on, it denotes an invalid storage address. It is therefore necessary to transfer only SAR bits 15 to 29 across the CSU interface.

On machines having 128K of main storage, SAR bit 15 is used to select either storage units 1 and 2, or 3 and 4, as shown in the preceding table under the heading "Storage Cycles."

The addresses are set into the SAR's at the beginning of the read cycle (RC 1, CP 1) and, at the same time, the signal 'read call' is generated. Thus, at read call time, the SAR's are not valid. It is therefore necessary to start the read cycle in all storage units irrespective of the required address. Later in the read cycle, SAR bit 15 is used to gate the X and Y current in the correct storage units.

If extension storage is to be addressed, the 'extension storage' latch is set, blocking one end of the normal Y lines and activating the extension storage Y lines.

Storage Data-Bit Lines

- Thirty-six lines transfer data from the SDR to main storage.

Thirty-six data-bit lines are used to connect the SDR to the storage unit inhibit drivers. Eighteen of these lines connect SDR bytes 0 and 1 to storage unit 1; the remaining 18 lines connect SDR bytes 2 and 3 to storage unit 2. In a 128K main storage, these lines are carried by inter-unit cables from storage unit 1 to storage unit 3 and from storage unit 2 to storage unit 4.

Storage Sense-Bit Lines

- Thirty-six lines carry the sense data from main storage to the SDR.

Thirty-six storage sense-bit lines connect the 'sense amplifier detector' latches to the SDR. Eighteen of these lines carry bytes 0 and 1 from storage unit 1 to the SDR while the other 18 lines carry bytes 2 and 3 from storage unit 2 to the SDR. As with the storage data-bit lines inter-unit cables are used to connect storage units 1 and 3, and 2 and 4.

Storage Clocks

- Each storage unit has a separate clock.
- The clock consists of delay lines and timing latches.
- The clock is started by either read call or write call.

The storage clock and timing can be seen in Figures 2-48 (read clock) and 2-49 (write clock).

Each storage unit of the 2044 contains its own clock. This allows the storage to operate independently with respect to the CPU. The clock cycles are started by one of two signals from the CPU; if a read cycle is required, the CPU generates 'read call', whereas, if a write cycle is required, 'write call' is generated. Both of these lines start the 'storage delay' line that consists of two 250-ns delay lines connected together to form a 500-ns delay line. The delay line is tapped at 25-ns intervals, and the outputs are used to control a set of latches which provide the timing pulses required by the storage.

PRINCIPLES OF OPERATION

Read Call to Storage

The 'read call' to storage is generated whenever the CPU requires a storage read cycle. The read clock in the CPU is started and at RC 1, CP 1 time, 'read call' is sent to all storage units via the CSU interface. The read cycle timing is shown in Figure 2-48; the timings depicted do not take circuit delays into account.

Purpose of Latches used during Read Cycle

Read Set Latch: The 'read set' latch controls the set and reset of the other timing latches during the read cycle.

Pulse Width Latch: This latch produces a 100-ns pulse to the delay line input.

Read 1 Latch: This latch provides timing to the X-control drivers and the 'strobe' latch.

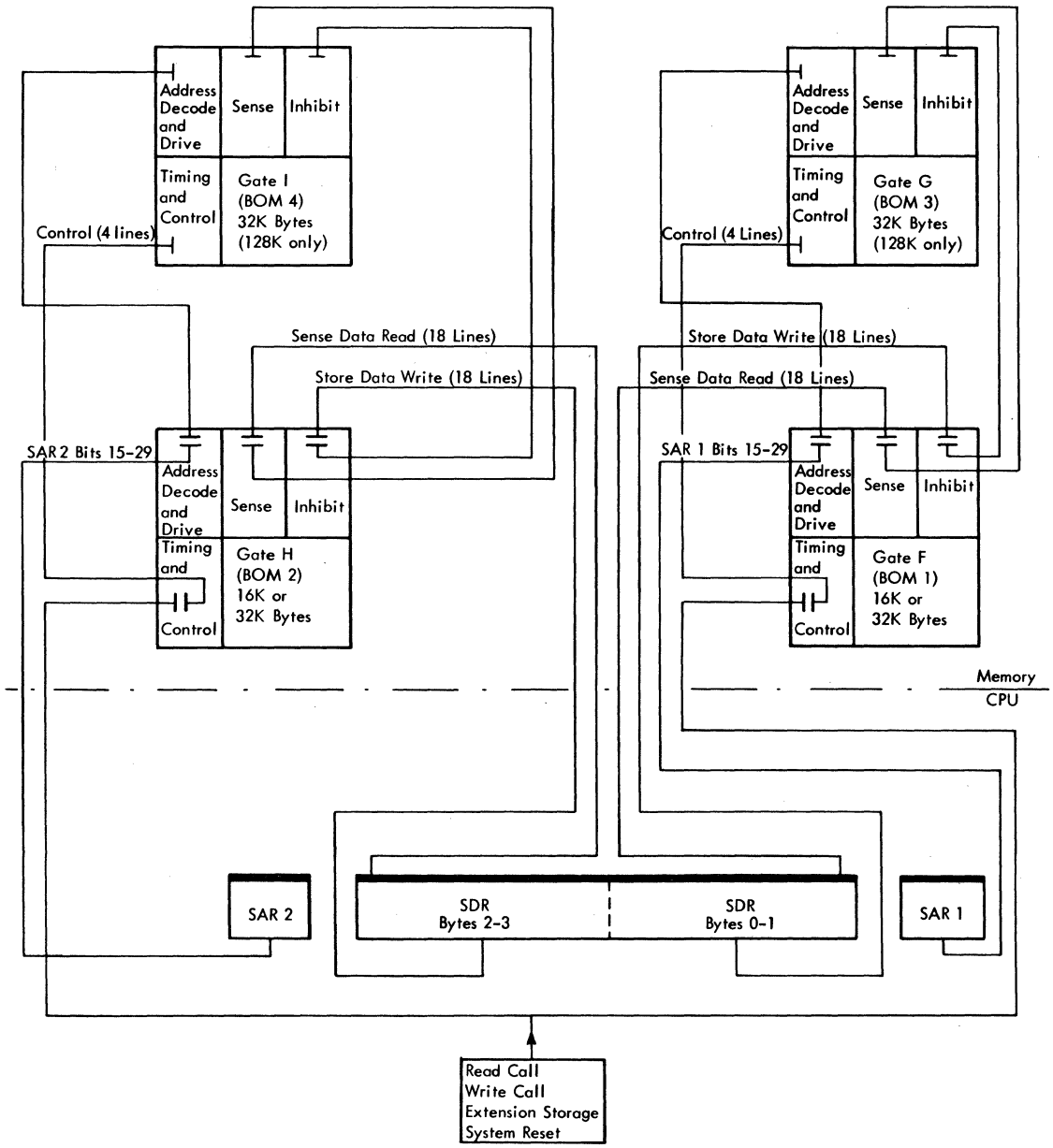


Figure 2-47. Core Storage Unit Interface (64K)

Read 2 Latch: This latch provides timing to the Y-control drivers.

X-Source Read Latch: This latch provides the timing pulse to the X-source read driver.

Y-Source Read Latch: This latch provides the timing pulse for the Y-source read driver.

Strobe Latch: The output of this latch is fed to a further delay line which provides the strobe pulse for the sense amplifier circuits.

Write Call to Storage

The write cycle timing is shown in Figure 2-49.

The 'write call' to storage is generated when the CPU requires a storage write cycle. If the cycle is not a split cycle, the write clock is started at RC 4. During split-cycle operation, the write clock will be started as soon as the data to be stored is available. The 'write call' is generated at WC 1, CP 1 of the write clock cycle, and starts the storage clock in all storage units. The 'write set' latch for each storage unit to be used is set by 'write call' under the control of SAR bit 15.

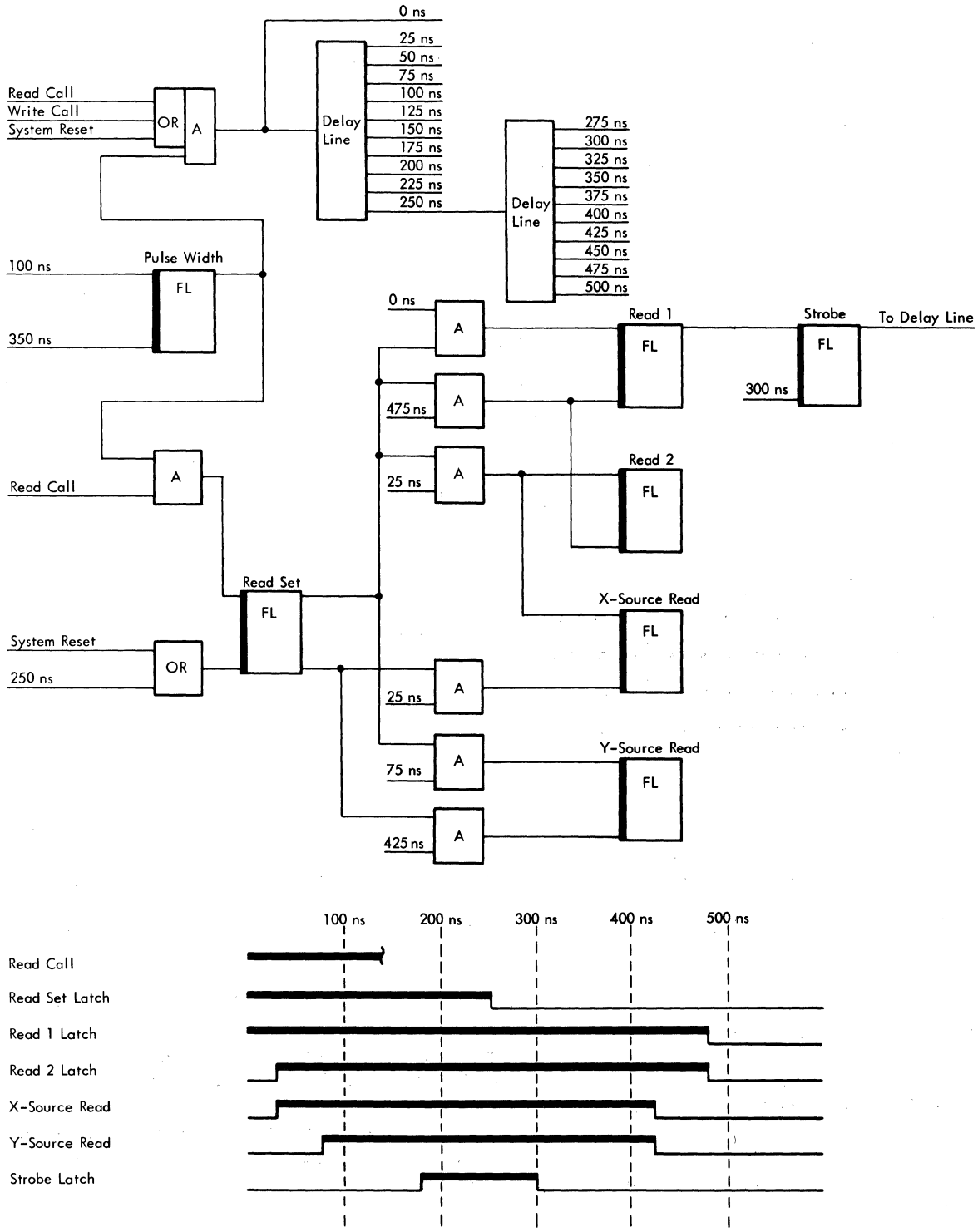


Figure 2-48. Read Cycle

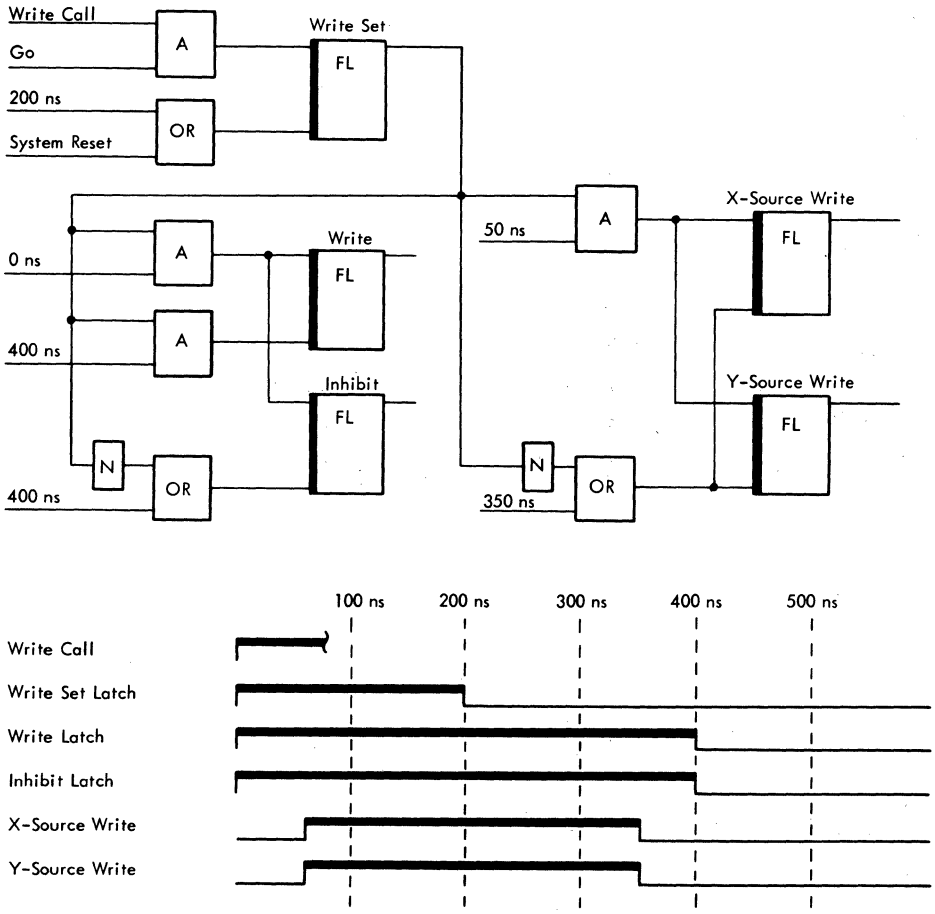


Figure 2-49. Write Cycle

Purpose of Latches used during Write Cycle

Write Set Latch: This latch controls the set and reset of the write cycle timing latches.

Write Latch: This latch provides the timing to the X- and Y-control drivers during a write cycle.

X-Source Write Latch: The 'X-source write' latch provides timing for the X-source driver during a write cycle.

Y-Source Write Latch: This latch provides timing for the Y-source driver during a write cycle.

Inhibit Latch: The 'inhibit' latch provides the timing to the inhibit driver.

System Reset: The effect of 'system reset' on the storage clock circuits is that the 'read set' latch and the 'write set' latch are both reset. However, the delay line drive pulse is brought up and the delay line takes a cycle, in order to provide a reset to the remaining clock latches.

X- and Y-Drive Current Sources

- These current sources supply drive current to the X and Y windings.
- Four sources only are necessary for each BOM.
- The sources are divided into two groups: X-read and X-write, and Y-read and Y-write.
- The sources are common to all array addresses within a storage unit and depend on gates to select a specific address.

Figure 2-50 shows the circuit of a current source.

Each current source consists of a transformer secondary winding. The primary windings of each transformer are driven by a transistor circuit. During a storage read cycle, the 'Y-source read' latch provides the timing pulse to the Y-read current source. If a storage unit is selected, the Y-source circuit is turned on causing current to flow in the Y-source read transformer primary winding. This current, in turn, causes the transformer secondary

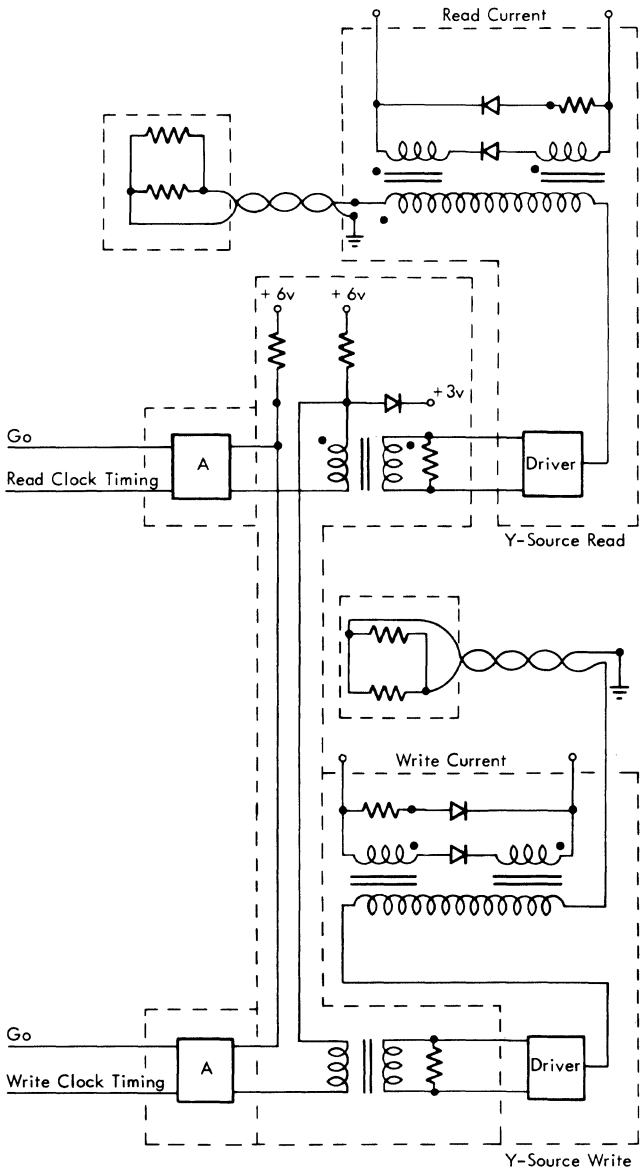


Figure 2-50. X-and Y-Drive Current Sources

current to flow. By this time, the selection circuitry has coupled the source transformer to a 'Y-drive' line, and current flows through the drive line. During a storage write cycle, 'Y-source write' is turned on which, in turn, causes current flow in the opposite direction in the same 'Y-drive' line.

The same action occurs in the X-source read and X-source write circuits.

X- and Y-Drive Gate and Selection Circuits

- The gate and selection system directs drive current to a single X line and a single Y line.
- The gate and selection logic consists of control drivers, address decodes and gates.

The principle of the gating and selection system is shown in Figure 2-51.

The purpose of the gate and selection system is to route drive current from a current source to a single X line and a single Y line. The gate and selection system acts as a switch at each end of the drive lines. Thus the current source supplies the operating current and the gate and selection circuitry routes the current to the appropriate drive lines. The X and Y drivers are shown schematically in Figure 2-52.

Two 8K addressable sections are used in the storage unit. In both 8K sections, eight read/write decode drivers are distributed at both ends of the 64 X-drive lines. Each decode drive consists of a read decode drive and a write decode drive.

X-Read Decode Drive

Three basic inputs to the read decode drive are as follows:

1. The AND'ed input of SAR bits 24, 25 and 26 at one end and SAR bits 27, 28 and 29 at the other end. These inputs select one of eight read decode drivers at one end and one of eight read decode drivers at the other end.
2. The timing from the X-read control driver. Two X-read control drivers are used in each 8K section, SAR bit 16 controlling the setting of these drivers. When SAR bit 16 equals 0 and the storage clock is on, the X-read control drivers in the 0 to 8K section are active; when SAR bit 16 equals 1 and the storage clock is on, the X-read control drivers in the 8 to 16K section are active.
3. The currents supplied by the X-read current source.

To select and drive one X-drive line during a read operation, the following action takes place:

1. One of the eight read decode drivers on one side of the array is active.
2. The eight lines from this decode drive are each routed to a separate read decode drive on the other side of the array.
3. One of these separate read decode drivers is also active, allowing current to flow through one only of the eight lines.

X-Write Decode Drive

Three basic inputs to the write decode drivers are as follows:

1. The AND'ed input of SAR bits 24, 25 and 26 on one side of the array and SAR bits 27, 28 and 29 on the other side; these signals are common with read decode drive.

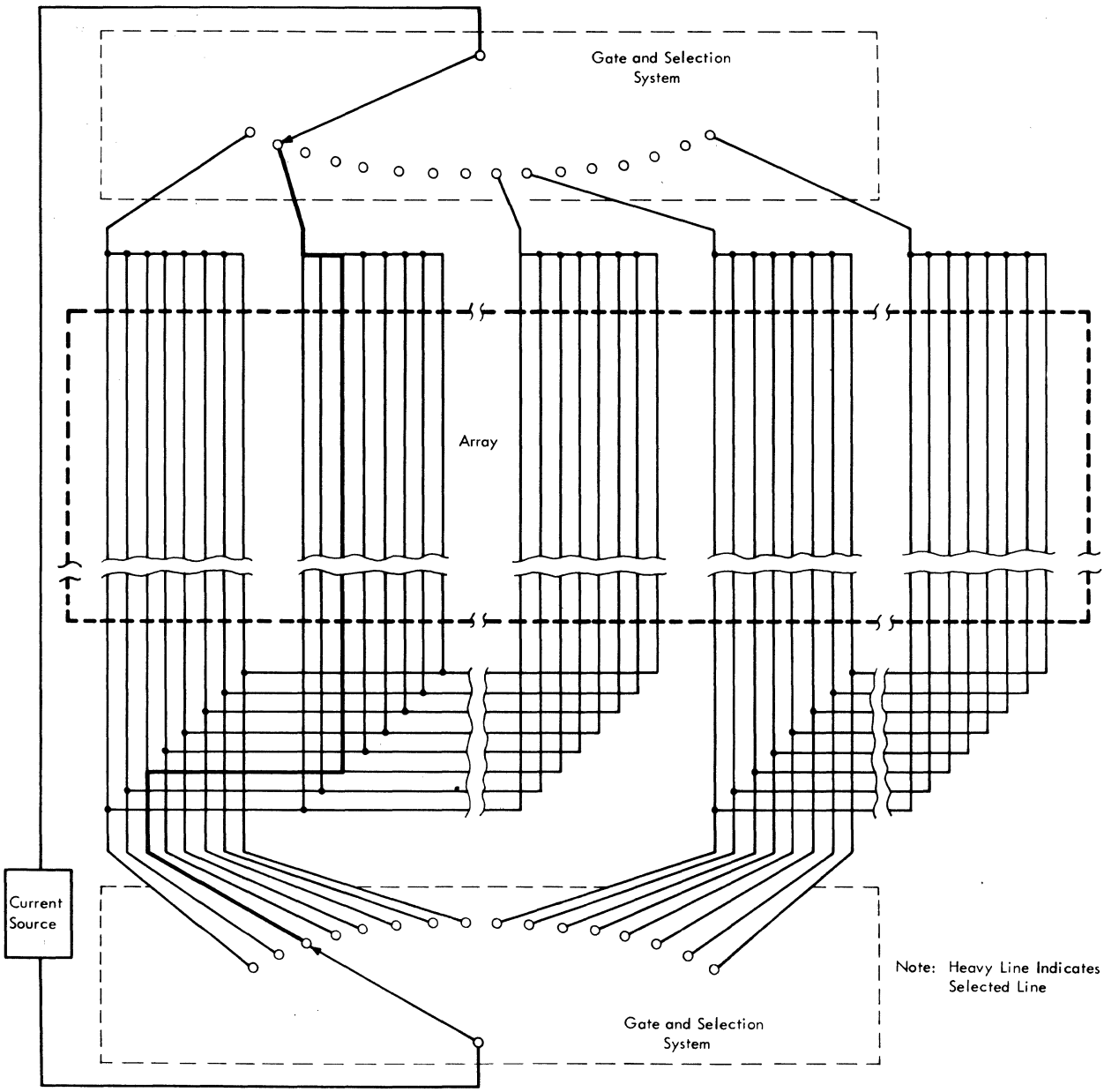


Figure 2-51. Gate and Selection System

2. Timing from the X-write control driver. Two X-write control drivers are used in each 8K storage section, SAR bit 16 controlling the setting of these drivers. If SAR bit 16 equals 0 and the storage clock is present (output and X-write latch), the X-write control drivers in the 0 to 8K section are turned on; if SAR bit 16 equals 1 and the storage clock is present, the X-write control drivers in the 8 to 16K section are turned on.

3. The current supplied by the X-write current source.

To select and drive one X-drive line during a write operation, the following conditions must be met:

1. One of the eight write decode drivers on one side of the array must be active.
2. One of the eight separate write decode drivers on the other side of the array also must be active, allowing current to flow through one only of the eight lines.

The address decode is common to both the read and write decode drives. Thus the same X lines are driven in both read and write operations although the write current is in the opposite direction to the read current.

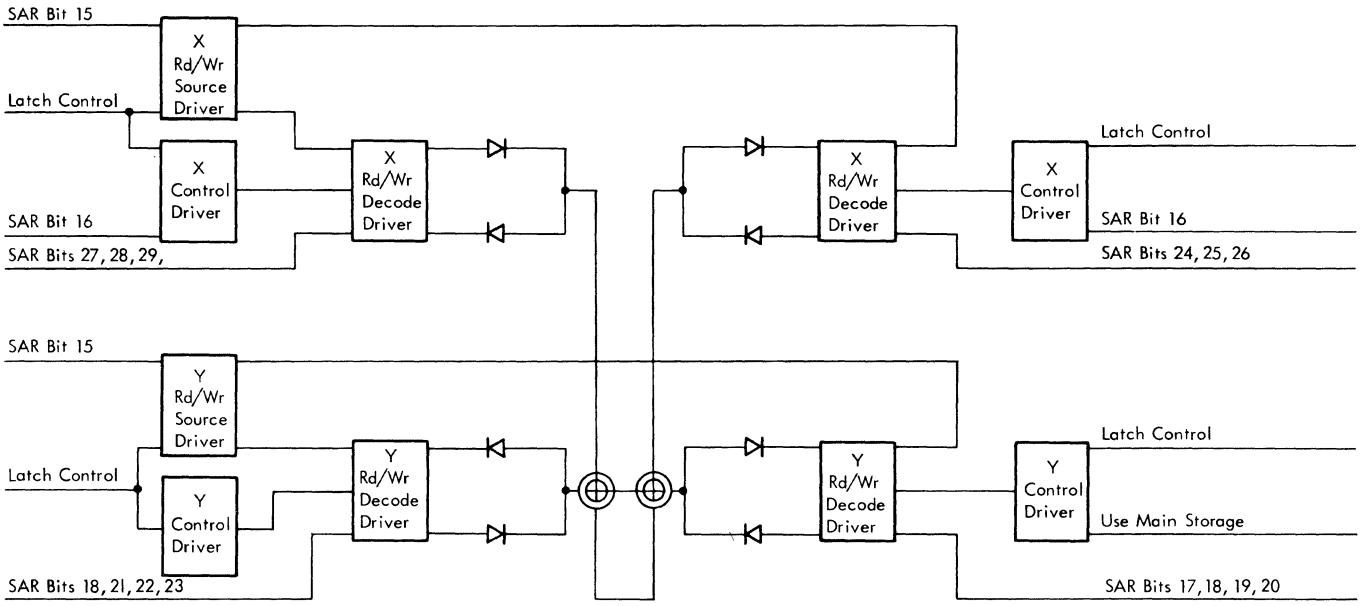


Figure 2-52. Main Storage X and Y Line Drivers

Y-Line Drivers

The Y-line driver arrangements are similar to those for X-line drive. Because there are 128 Y lines in the 32K storage array, 16 read and write decode drivers are needed at each end of the array. SAR bits 17, 18, 19 and 20 are AND'ed at one end of the array and SAR bits 18, 21, 22 and 23 are AND'ed at the other end of the array. The output from this AND'ing selects one of the 16 read/write decode drivers at each end of the array.

To select and drive one Y-drive line, the following conditions must be met:

1. One of the 16 read/write decode drivers on one side of the array must be active. The eight Y lines from this decode drive are each routed to the separate read/write decode drivers on the other side of the array.
2. One of these other decode drivers also must be active, allowing current to flow through one of the eight lines.

SAR bit 18 is used to select a read/write decode drive at each end of the array. The eight read/write decode drives at one end of the array are selected when SAR bit 18 equals 0 and those at the other end when SAR bit 18 equals 1.

X- and Y-Decode Numbering Scheme

The numbering scheme used to identify the X- and Y-decode drivers makes it possible to identify the active circuit by knowing which bits are present in the SAR. See Figure 2-53.

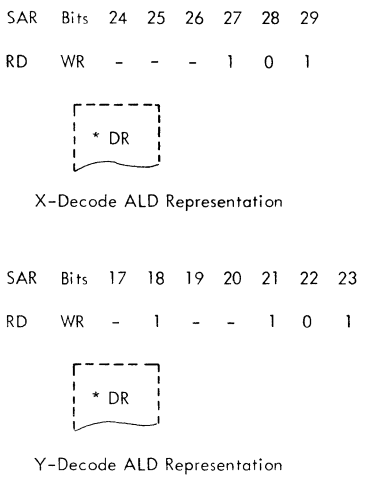


Figure 2-53. X-and Y-Decode Drivers

X-Decode: The X-decode drivers use SAR bits 24 to 29. The following three symbols are used in the ALD's to show the conditions necessary to activate a decode driver:

- - : this bit is not used to control this driver.
- 1 : this bit must be present in order to activate this driver.
- 0 : this bit must not be present in order to activate this driver.

Therefore the X-decode driver specified in Figure 2-53 is activated when bit 27 of the SAR is on, bit 28 of the SAR is off and bit 29 of the SAR is on. The status of bits 24, 25 and 26 will not affect this driver.

Y-Decode: The Y-decode drivers use SAR bits 17 to 23 and are shown in the ALD's in the same way as the X-decode drivers. Therefore, the Y-decode driver specified in Figure 2-53 is activated when SAR bits 18, 21 and 23 are on, and bit 22 is off. Bits 17, 19 and 20 will not affect this driver.

Example of Storage Read/Write

Assume that the following binary address is in the storage address register:

0	0111	0111	0111	1	0	1	0
15					29	30	31

The sequence of operation is that the CPU calls first for a read cycle, then for a write cycle. The byte read out is either regenerated (placed back into the addressed location) on the write cycle or new information is set to the SDR by the CPU to be stored on the write cycle. The operational sequence is as follows:

1. Start the storage clock.
   Read call
   Not pulse width control
2. Turn on the 'read set' latch to enable a read cycle.
   Read call
   Not pulse width control
3. Set the 'use main storage' line to define the area of storage to be addressed:
   + Use main storage = Main storage addressed
   - Use main storage = Extension storage addressed.
4. Select and drive one Y line with read current. This operation requires the turning on of two read control drivers (one for each end of the Y line), two address decode switches (one for each end of the Y line), two read gates (one for each decode switch), and the Y-read current source.
   a. Turn on the Y-control drivers RD 0 to 16KB and RD 0 to 16KA.
      These drivers provide the signals, RD 2 control 0 to 16KA and RD 2 control 0 to 16KB.
      Use main storage
      Read 2 (from storage clock)
   b. Turn on the Y-read current source.
      Y-source read (from storage clock)
      Go (Not SAR bit 15)
   c. Turn on RD - 1 - - 1 1 1. This is a Y-decode switch for the source side of the Y line. The gates are on the same logic page and feed the decode switches directly.
      Y-read current source
      Read 2 control 0 to 16KA
      SAR bit 22
      SAR bit 23
      SAR bit 21 control
      SAR bit 18 control
   d. Turn on RD 1 1 1 0 - - -. This is the Y-decode switch for the sink side of the Y line.
      RD 2 control 0 to 16KB address
      RD current sink
      SAR bit 17 and bit 18
      Not SAR bit 20
      SAR bit 19

5. Select and drive one X line with read current. This operation involves the turning on of two read control drivers (one for each end of the X line), two address decode switches (one for each end of the X line), two read gates (one for each address decode switch), and the X-read current source.
   a. Turn on the X-control drivers RD 0 to 8KA and RD 0 to 8KB.
      RD 1 (from clock)
      Not SAR bit 16
   b. Turn on the X-read current source.
      X-source read
      Go (Not SAR bit 15)
   c. Turn on RD - - -1 1 0. This is the X-decode switch for the source side of the X line.
      X-read source
      RD 1 control 0 to 8KA address
      SAR bit 27
      Not SAR bit 28
      SAR bit 29
   d. Turn on RD 0 1 1 - - -. This is the sink side of the X line.
      X-RD sink
      RD 1 control 0 to 8KB address
      Not SAR bit 24
      SAR bit 25
      SAR bit 26

6. Condition the sense amplifier gates so that the appropriate sense windings are gated to their respective sense amplifiers. The gates for this address are 0 to 8K B1 (to byte-0 sense amplifiers) and 0 to 8K B2 (to byte-1 sense amplifiers).
   Not SAR bit 16
   Not SAR bit 29
   Read 2 (from clock)
   Go (Not SAR bit 15)
7. Amplify and gate the sense pulses to the sense amplifier detector latches.
   SA gate 0 to 8K B1 (byte-0 sense amplifiers)
   SA gate 0 to 8K B2 (byte-1 sense amplifiers)
   SA input, any bit, 0 to 8K
   Strobe 0 to 16K (from clock)
8. The sense amplifier detector latches are set into the SDR.

9. Without changing address in the SAR's, the CPU requests a storage write cycle and starts the storage clock.

Write call
Not pulse width latch.

10. Set up the storage clock for a write cycle by turning on the write set latch.

Write call
Go (Not SAR bit 15)

11. For the write cycle it will be necessary to select and drive the same X- and Y-drive lines as were driven on a read cycle. On the write cycle, however, the current flow must be in the opposite direction. For the Y line, it is necessary to turn on two control drivers (one for each end of the Y line), two address decode switches (one for each end of the Y line), two address gates (one for each decode switch), and the Y-write current source.

a. Turn on the Y-control drivers, WR control 0 to 16KB and WR control 0 to 16KA.
   Write A (from clock)
   Use main storage
b. Turn on the Y-write current source driver.
   Y-source write (from clock)
   Go (Not SAR bit 15)
c. Turn on the WR - 1 - -1 1 1 write address decode driver. This includes the address gate and is on the sink end of the Y line.
   SAR bit 18 control
   SAR bit 21 control
   SAR bit 23
   SAR bit 22
   Y-write current sink
   WR control 0 to 16KA address
d. Turn on the WR 1 1 1 0- - - write address decode driver. This includes the address gate and is on the source end of the Y line.
   SAR bits 17 and 18
   Not SAR bit 20
   SAR bit 19
   WR control 0 to 16KB address
   Y-write current source

12. Select and drive the same X line in the opposite direction. This requires two control drivers (one for each end of the X line), two address decode switches (one for each end of the X line), two address gates (one for each decode switch), and the X-write current source.

a. Turn on X-control drivers WR 0 to 8KA address, and WR 0 to 8KB address.
   Write B (from clock)
   Not SAR bit 16.
b. Turn on the X-write current source.
   X-source write
   Go (Not SAR bit 15)
c. Turn on the X-decode driver for the sink end of the X line. This is WR- - -1 1 0.
   SAR bit 27
   Not SAR bit 28
   SAR bit 29
   X-write sink
   WR control 0 to 8KA address
d. Turn on the X-decode driver for the source end of the X line. This is WR 0 1 1 - - -.
   SAR bit 25
   Not SAR bit 24
   SAR bit 26
   X-write source
   WR control 0 to 8KB address

13. The appropriate set of inhibit drivers must be gated so that only one set of these drivers turns on. For this address, inhibit 0 to 8K B1 (to byte-0 inhibit drivers) and 0 to 8K B2 (to byte-1 inhibit drivers) must be turned on.

Not SAR bit 16
Not SAR bit 29
Inhibit (from clock)

14. For those bits that are to be set on, the inhibit driver must be blocked from turning on. The store lines block their respective inhibit drivers.

15. For those bits that are to be blocked from setting, the appropriate inhibit drivers are turned on by the (Not) store lines. Inhibit current opposes the effect of the X-drive current and the core is not set.

Extension Storage Addressing

Figures 2-54 and 2-55 show the basic principle of extension storage addressing.

Extension storage shares with main storage the external control circuits, the internal control and timing circuit, the sense/inhibit system and X-line driving. A separate extension storage Y-drive scheme is used to select any of the eight extension storage Y-drive lines in the storage unit.

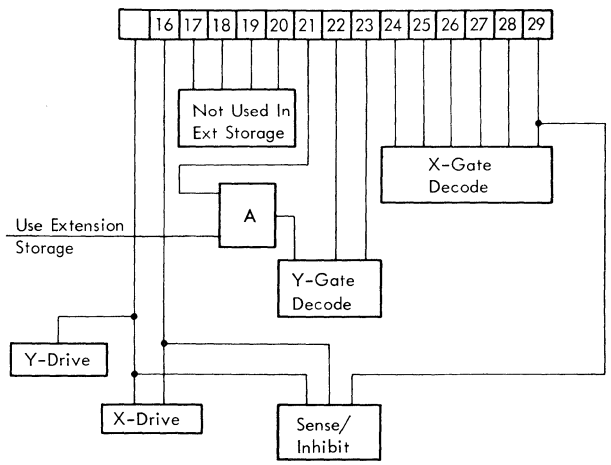
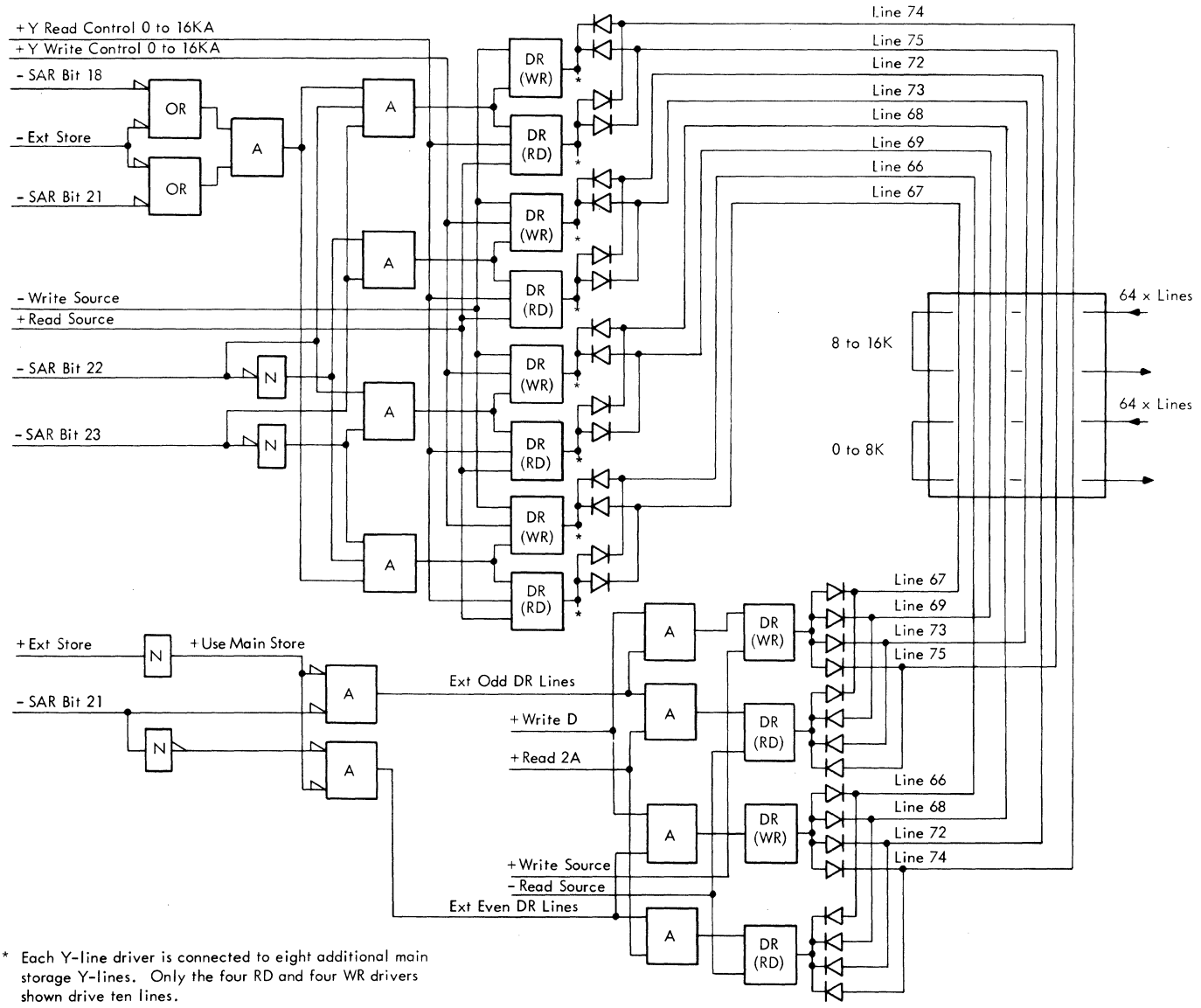


Figure 2-54. Extension Storage Addressing Scheme



* Each Y-line driver is connected to eight additional main storage Y-lines. Only the four RD and four WR drivers shown drive ten lines.

Figure 2-55. Extension Storage Y Read and Write Drive

The X-line selection, the X drive and Y drive are common with main storage. The selection of extension storage Y lines is implemented as follows.

Eight additional Y lines are used for extension storage in each storage unit (BOM). On one side of the array, these eight lines are arranged into four pairs and added to four of the normal read/write decode sets. These decode sets are:

SAR bits 18, 21, 22, 23
18, 21, 22, $\overline{23}$
18, 21, $\overline{22}$, 23
18, 21, $\overline{22}$, $\overline{23}$

Bits 22 and 23 are fed to the decode of the normal output of the SAR's. However, bits 18 and 21 are split by extension storage. On the other side of the array, these eight lines enter two pairs of gate decoders which are active only when there is a call for extension storage. SAR bit 21 is used to determine whether odd or even lines are active. These eight Y lines and normal selection of the X lines allow access to all extension storage locations up to the maximum extension storage size.

TEMPERATURE CONTROL AND POWER SUPPLIES

Storage Temperature Control

- The speed at which a core flips is dependent on temperature.
- For stable operation a constant core temperature is required.

The speed of a core storage unit will vary appreciably depending upon the temperature at which it is operating, and the magnitude of the current which is used to drive the cores. In order to stabilize the operation of the core it is necessary to either vary the current according to temperature, or to stabilize the temperature. The storage of the IBM 2044 is held at a constant temperature by a core heater and fan arrangement which is mounted at the bottom of the array. Each BOM is fitted with a heater box assembly.

The normal operating temperature of the array is 37°C (99°F). However, it is possible to use the core when its temperature has reached 34°C (93°F).

The core heater consists of 13 wire-wound resistors connected in parallel. The resistors provide a total heat dissipation of 450 watts when operated at 208v ac. Connected in series with these resistors are two Silicon Controlled Rectifiers (SCR's), the operation of which is controlled by a proportional control unit and a thermistor.

The proportional control unit consists essentially of a power supply, a transistorized differential amplifier and a triggering circuit, the output of which is fed to the SCR's. This is shown in Figure 2-56. By altering the point in the mains sine wave at which the SCR's are fired, it is possible to either raise or lower the output of the heater. This firing point is decided by the thermistor which is mounted in the storage array. Thus, if the core temperature is low, the thermistor allows the control unit to fire the SCR's early in the sine wave, but if the temperature is high, the firing point will occur later. Refer to Figure 2-57.

Two thermal switches are used to indicate to the CPU that the core temperature is above 34°C (93°F) and not more than 49°C (120°F). The 34°C (93°F) thermostat consists of a glass tube containing mercury. Two wires pass through this tube, one of which becomes the switch common contact. A connection is made through the mercury to the other wire whenever the temperature exceeds 34°C (93°F). The switch detail is shown in Figure 2-58.

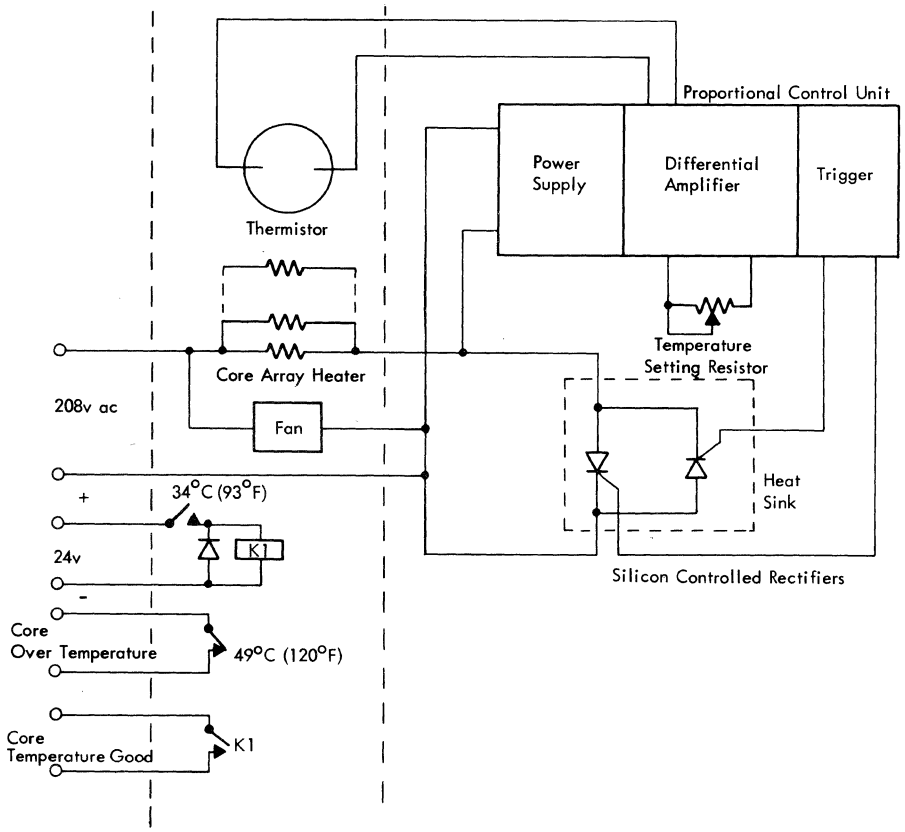


Figure 2-56. Storage Temperature Control

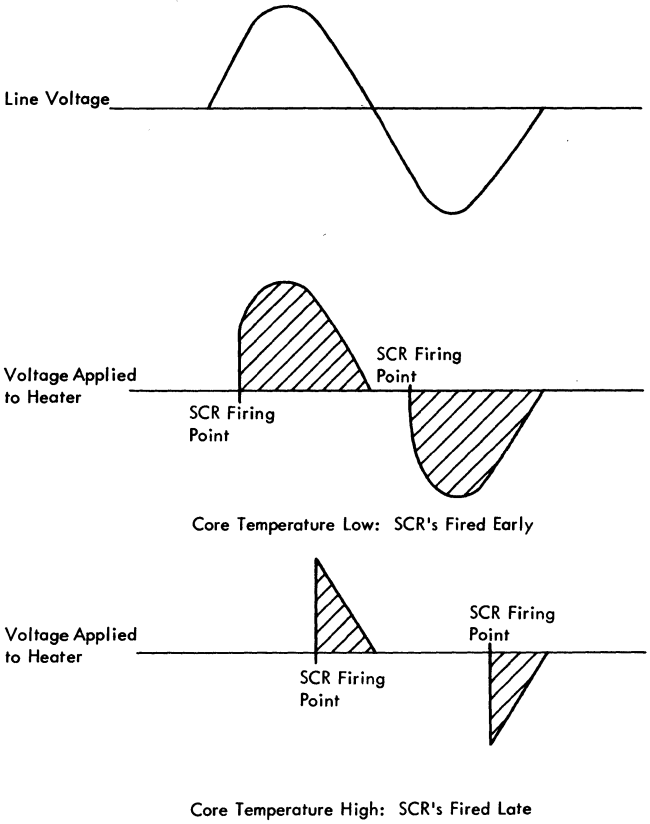


Figure 2-57. Core Heater Voltage

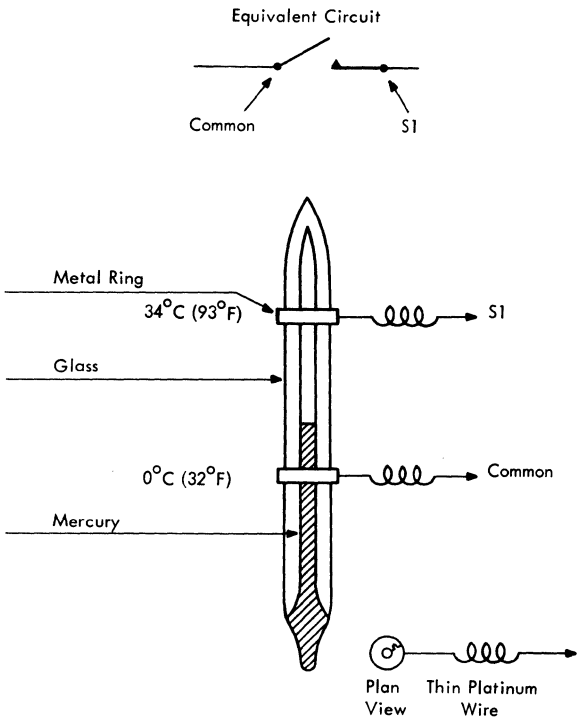


Figure 2-58. Thermostat Detail

Power Supplies to Main Storage

- Main storage power comes from the power supplies section of the main frame.
- An 18-volt supply is generated in the main storage frame, input coming from the CPU -30v supply.

Power for the storage frames comes from the Mid-Pac modules located in the main frame. The power supplies used by main storage are shown in the following table:

Power Supply	Voltage	Comments
PS 1	+3v	Standard
PS 2	-30v	Used to Produce -18v
PS 6	+6v	
PS 12	-3v	
24v	+24v dc	Used for Heater Control Relays

The -18v (±1v) supply is generated in the storage frame by means of two series regulator cards. The output of this power supply is used by the sense amplifiers. The voltage regulator is protected against short circuits. If the regulator output is shorted to ground, the power supply will automatically cut off and, when the short circuit is removed, will automatically turn on again.

CHECKING

MACHINE MALFUNCTION CHECKING

CPU Checking

- Failures are detected by:
  extensive software diagnostic programs, or
  limited hardware checking circuits.

- Parity checking can be made:
  in the storage area,
  in the high speed multiplexor channel, or
  across the standard interface.

- Control checking can be made on:
  active storage cycle controls,
  sequence control latches, or
  floating-point sequence-control latches.

- Provision is made to force a machine check (from the console for customer engineer use).

- A machine check:
  can be ignored,
  can give a hardstop, or
  can give a machine check interrupt.

Circuit malfunctions are detected and located mainly by diagnostic programs (for customer engineer use only). The descriptions and handling of these programs are given in the "Diagnostic Aids" section in FEMM, Form Y33-0007.

However, hardware checking circuits have been provided to allow limited detection of single machine errors. These circuits are associated with the following areas.

Redundancy Checking: Odd parity is generated and checked with information in the storage area, in the high speed multiplexor (HSMPX) channels, and across the standard interface. See Principles of Operation - Channels, Form Y33-0003, for the last two items.

Control Checking: This check is provided to ensure that only one active storage cycle control is on at one time, or that only one sequence control latch (including any floating-point sequence-control latch) is set. Note that a sequence-control cycle and a floating-point sequnce-control cycle may occur together and at the same time as an active storage cycle control.

Console Forced Machine Check: Although this is not a malfunction check, certain conditions and console settings allow a machine check to be generated from the console circuits in order to simulate an error. The simulated error in turn allows a machine check interrupt to occur. The console switch settings are the check control rotary switch set to FORCED RESTART or the MS address-compare rotary switch set to LOOP. This facility is essentially for customer engineer use.

The system is able to record intermittent parity or control failures detected by circuits. The system does not necessarily stop and intermittent failures have a minimum impact on customer use of the equipment, as the failures are stored in check latches. The outputs of the check latches are combined with several console and/or program conditions to give no reaction, a hardstop, or a machine check interrupt. The interrupt routine is usually programmed in order to perform diagnose instructions.

Parity Checking and Generation

- Odd parity is carried with each byte to and from storage or the HSMPX channels.

- Each parity checking circuit can set the SDR parity check latch.

- A parity check inhibit latch can prevent checking for each byte.

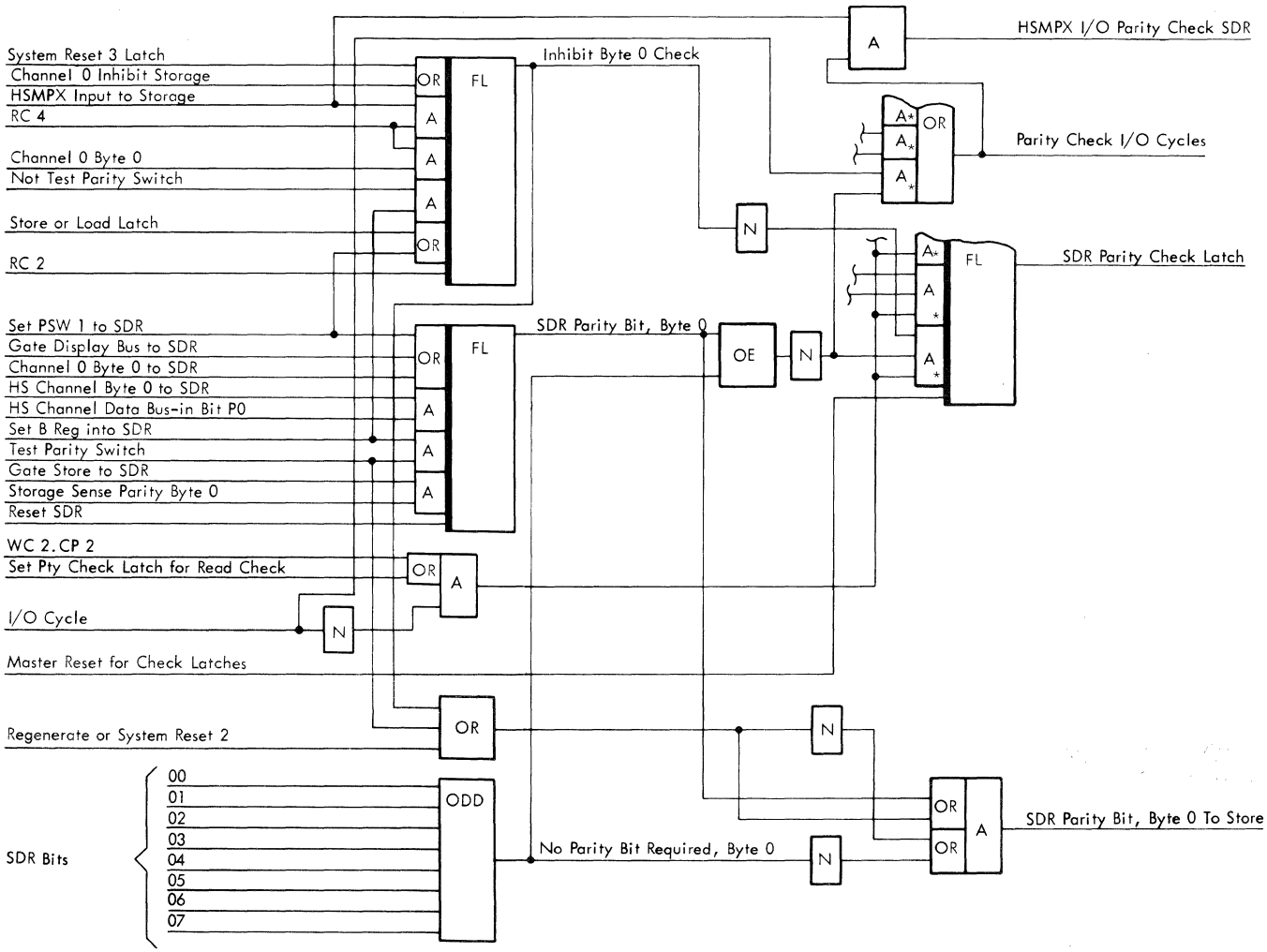
- A parity generation circuit is provided for each byte.

- An HSMPX channel parity error signal is sent back to the HSMPX channel.

- The SDR check lamp is lit when the SDR parity check latch is on.

Circuits are provided to check and to generate the odd parity bits for data associated with storage. These circuits are interleaved as shown in Figure 2-59.

Odd parity implies that when the number of bits within a byte is odd, no parity bit is required. The requisite analysis is made by the ODD block in Figure 2-59 (See Appendix B4 of the FEMM, Form Y33-0007, for a description of the ODD block.) The ODD block receives each SDR bit except the parity bit. The result of the analysis is EXOR'ed with the state of the SDR parity bit latch. Any error sets the SDR parity check latch at WC 2, CP 3 time or at the corresponding time during split cycles (CC 2, CP 2) when the data that has been read is used. The latch is reset with 'master reset for check latches' which is raised as shown in Figure



* Similar AND Block provided for each Byte

Figure 2-59. Example of Parity Checking (Byte 0)

2-60 on a system reset, by the console check reset switch, or when a 'machine check interrupt' is handled (reset CPU error latches).

The on state of the SDR check latch is indicated on the console by the SDR check lamp.

The parity bit of a byte sets the corresponding SDR parity bit latch in the same way as it sets other SDR bit positions. Parity is carried with a byte only if it originates in either main storage or an HSMPX channel. In all other cases, the inhibit byte check latch is set, which prevents the setting of the SDR parity check latch. This operation also conditions the parity generating circuits which produce correct parity for the byte before it enters main storage.

When an I/O channel is gated to or from the SDR, however, the SDR parity check latch is inhibited, but special high-speed circuitry detects an SDR parity check on an I/O cycle. See Figure 2-59. The HSMPX parity check signal is sent back to the appropriate HSMPX channel, where it is combined with the relevant subchannel to set a subchannel data check latch. (See Principles of Operation - Channels, Form Y33-0003.) The SDR parity check I/O cycle will generate the machine check interrupt code.

Fixed-Point and Floating-Point Sequence Control Checking

- The sequence control check latch is set with:
  more than one fixed-point sequence-control latch on, or
  more than one floating-point sequence-control latch on.

- One fixed-point sequence-control latch and one floating-point sequence-control latch may be on together and at the same time as an active storage cycle control latch.

- The sequence control latch is set at CC 1, CC 2 and reset with the master reset for check latches.

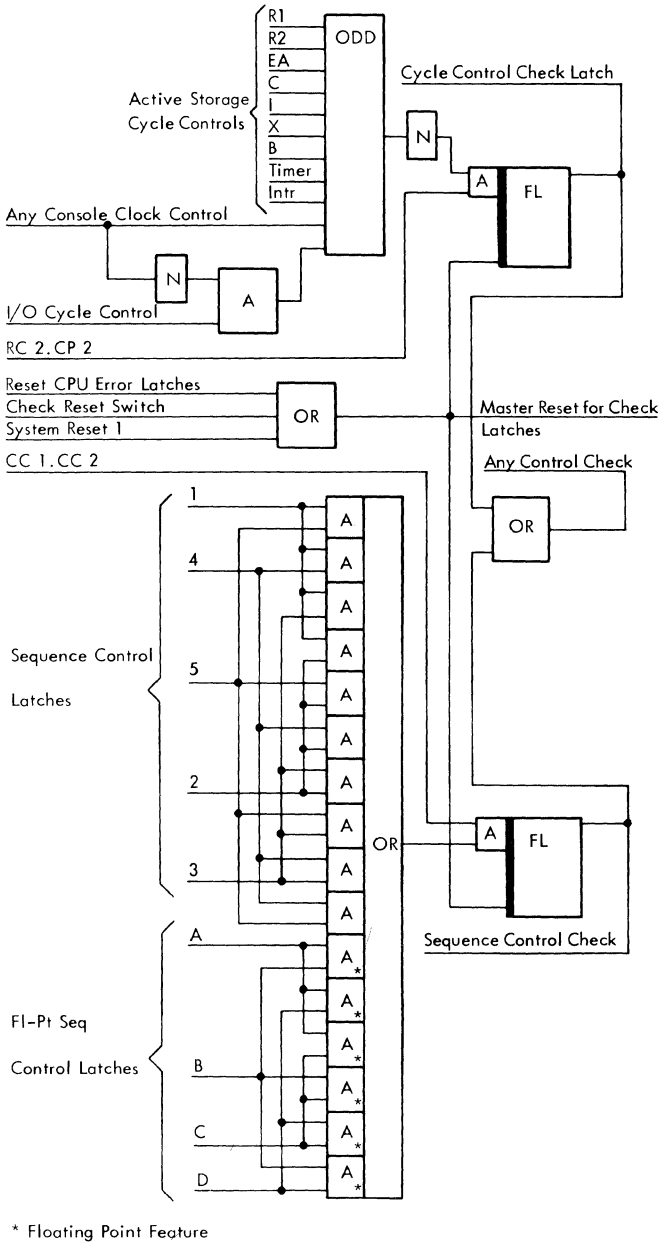


Figure 2-60. Control Checking

Figure 2-60 shows the circuitry which ensures that only one of the sequence controls (1, 2, 3, 4, 5) is on at one time. In the same way, the floating-point sequence controls (A, B, C or D) are also controlled, as shown, on the same OR block. Fixed-point and floating-point sequence controls are not combined; this allows one of each to be on at one time and, eventually, at the same time as an active storage-cycle control latch.

Any error condition is recorded by the sequence-control check latch which is set at CC 1, CC 2. This check latch is reset by the 'master reset for check latches' signal as explained previously. The on state of this latch is shown on the console by the control check lamp.

Cycle Control Checking

- An even number of active storage-cycle controls set the cycle control check latch.
- The cycle control check latch is set at RC 2, CP 2.

The cycle control check latch (Figure 2-60) is set whenever an even number of active storage-cycle controls are on at one time.

The function of the checking is to verify that only one active storage-cycle control is on at one time. Ideally, this would be achieved in the same way as the sequence control checking described in the preceding section. However, because of the improbability of having 3, 5, 7 or 9 active storage-cycle controls on at the same moment, and in order to save using 55 AND blocks, the associated OR blocks, and the corresponding response time, an ODD block has been used. The ODD block is explained in Appendix B4 of the FEMM, Form Y33-0007.

The lamp of the console displays the on state of the storage cycle control latches, even when the storage-cycle controls are inactive. The storage-cycle controls are active only if there is no I/O cycle request or console cycle request.

Circuitry is provided to inhibit the checking when 'any console clock control' and 'I/O cycle control' are active together, since the I/O cycle control is effective only in the absence of 'any console clock control'.

The on state of the cycle control check latch lights the console control check lamp (which is also used for sequence control checking).

Console Force Machine Check

- Simulates a machine check detection.
- Occurs with defined console settings:
  with FORCED RESTART switch position,
  with LOOP ON MS switch position.

In some cases the customer engineer may wish to initiate a machine check. Although this is not a machine malfunction check, the console force machine check is one of the conditions capable of giving a 'machine check interrupt' request.

Figure 2-61 shows the circuitry provided to allow this simulated machine check to occur. The significance of the lines shown is as follows.

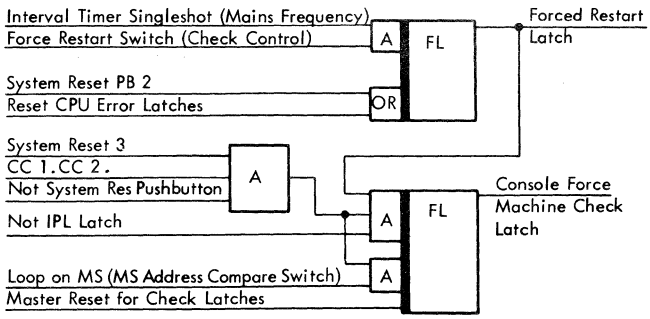


Figure 2-61. Console Force Machine Check

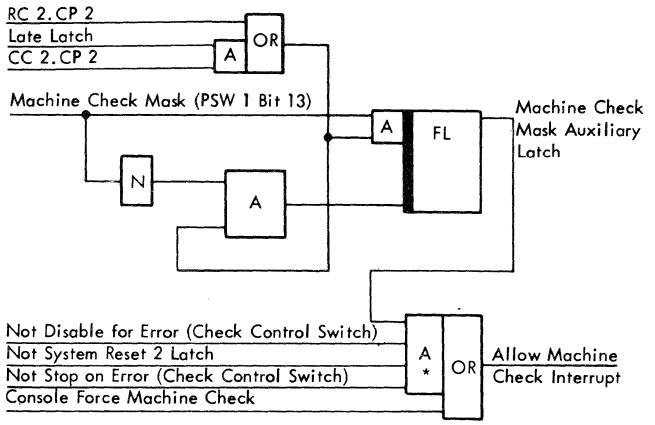


Figure 2-62. Machine Check Interrupt Enabling Circuitry

The check-control rotary switch in the FORCED RESTART position causes the forced-restart latch to be set each time the interval timer singleshot fires at the main-line frequency. (See "Update Interval Timer" in Principles of Operation - Processing Unit, Form Y33-0002.) Firing in turn sets the console force machine check latch at the end of the system reset which was initiated by the same conditions (namely, the check control switch position FORCED RESTART and the interval timer singleshot).

When the main storage address-compare rotary switch is at LOOP ON MS, the console force machine check latch is set every time the end of system reset approaches (sytem reset latch 3 and CC 1, CC 2). In this case, a system reset is initiated when a 'main storage address' comparison occurs.

The on state of the console force machine check latch is not recorded on the console.

Effects of Machine Check Detection

- Depending on certain program, console, or error conditions, a machine check:
  - can be ignored (disabled),
  - can cause a machine check interrupt,
  - can cause a hardstop, or
  - can be recorded in the CSW for channel malfunction.

Machine Check Disabled

- A machine check is disabled if PSW 1 bit 13 is zero.
- A machine check is disabled if the Console Check Control rotary switch is at DISABLE.
- A console machine check cannot be masked.

Figure 2-62 shows the logic circuitry provided to allow an interrupt to take place when a machine check is detected. A machine check interrupt cannot occur if any of the conditions on the AND block marked with an asterisk (*) are not met. The machine-check mask auxiliary latch reflects the state of the PSW 1 bit 13 (machine check mask). If this latch is off or if the check control rotary switch is on DISABLE (these two conditions override each other to suppress interrupts), the line 'allow machine check interrupt' can be active only on a console-forced machine check. In all other cases a machine check is ignored. If the check control rotary switch is in the stop position, an interrupt does not occur but a hardstop results.

Nevertheless, an ignored machine check remains pending while the check latches are not reset. Since the reset is achieved by the signal 'master reset for check latches' shown in Figure 2-60, a new PSW (with a PSW 1 bit 13 of one) can cause a machine check interrupt to occur. This is relevant for a fault detected under the control of the previous PSW. The signal 'master reset for check latches' is raised under the following conditions:

1. During a system reset.
2. By the console check-reset pushbutton.
3. With 'reset CPU error latches', which is active during a machine check interrupt, as described in Figure 2-64.

Condition 3 has no practical use, but a PSW 1 bit 13 of zero can be used, for example, to shorten an error loop. Further explanation is given in the Diagnostic Aids section of the FEMM, Form Y33-0007.

Machine Check Interrupt

- Can occur with 'allow machine check interrupt' active.
- Raised by 'machine check interrupt request'.
- 'End execute' is forced as soon as possible.

If none of the conditions arise that would disable the machine check interrupt (as described in the preceding section), the line 'allow machine check interrupt' in Figure 2-62 is active. Figure 2-63 shows the machine check interrupt circuitry. When the line 'allow machine check interrupt' is active with 'any CPU error' (which is activated when any CPU check latch is set), a 'machine check interrupt request' is raised. This condition causes 'interrupt cycle request' to be raised. The remaining condition that is required to set the machine-check interrupt latch is 'end execute'. This is raised as soon as possible. Therefore, as shown in Figure 2-63, the condition 'allow machine check interrupt' is used in conjunction with a sequence control check to inhibit the starting and the wrapping of the compute clock and, in conjunction with 'any CPU error', to force 'end execute' at WC 4.

However, a forced 'end execute' is prevented during load PSW operations or during interrupt cycles as it might cause only part of the PSW to be stored. The condition is prevented by the OR block marked with an asterisk (*) in Figure 2-63.

As soon as the compute and storage clocks are halted, 'end execute' is raised by 'machine check end execute'. 'Set interrupt latches' is then active with the first CP 2 and the machine check interrupt latch is set.

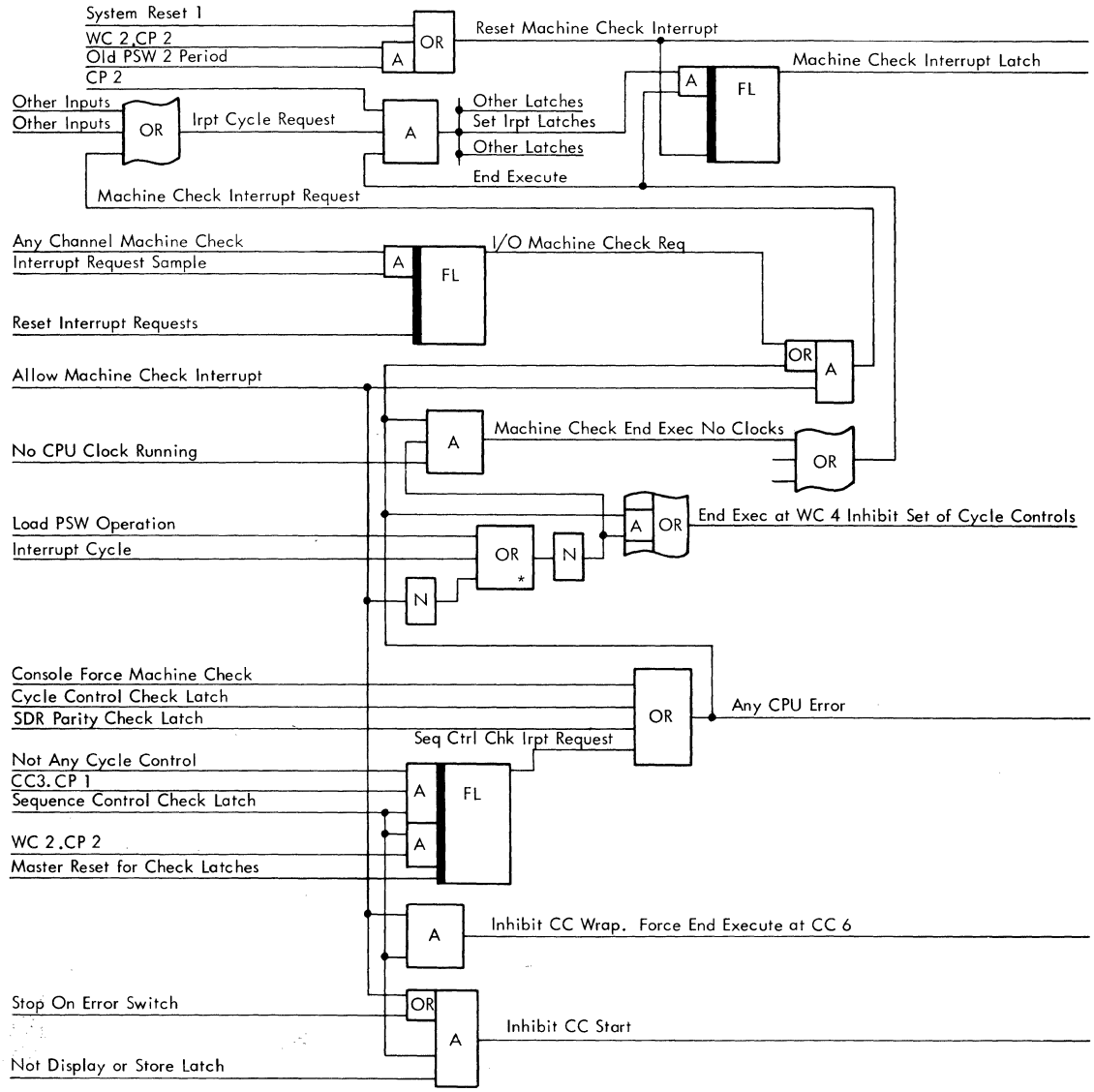


Figure 2-63. Machine Check Interrupt Circuitry

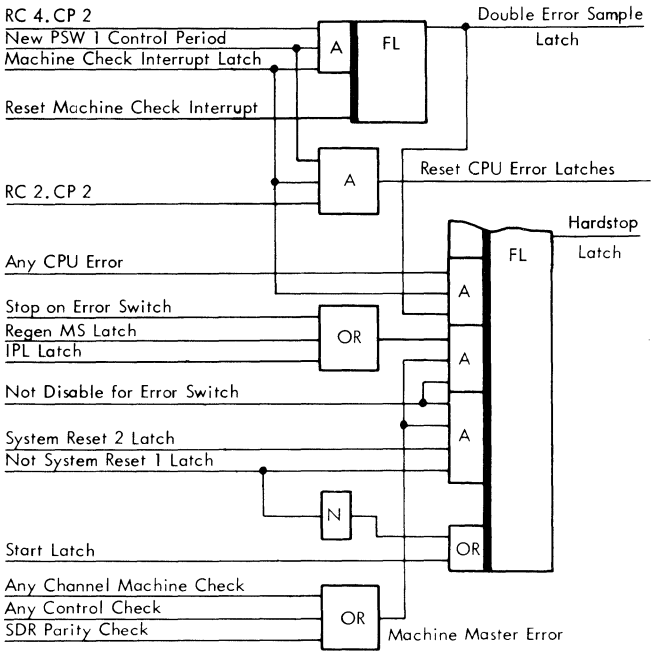


Figure 2-64. Machine Check Hardstop Circuitry

The machine-check interrupt latch remains on until the end of the interrupt cycles caused by it. Further details of the interrupt are given in the "Interrupts" section of Principles of Operation - Processing Unit, Form Y33-0002.

Machine Check Hardstop

- Occurs with a "double error" detection.
- Occurs with the check control rotary switch at the stop position.
- Takes place when a machine check occurs during an IPL, a system reset, or a 'regenerate main storage'.

Figure 2-64 shows the logic associated with a hardstop or a machine error detection.

The upper AND block shown as part of the hardstop latch detects any machine check occurring during a machine-check interrupt cycle (storing old PSW, fetching new PSW). Detection is made with the double-error sample latch which is on during the machine-check interrupt cycles. Note that the check latches are reset at RC 2 before the double-error sample latch is set (at RC 4).

The second AND block that can set the hardstop latch is activated by 'machine master error' (which is raised by any machine error detection as shown in Figure 2-64). 'Machine master error' is gated if the check-control rotary switch is not in the disable position, during an IPL or a regenerate-main-storage function, or if the switch is in the stop position.

The lower AND block can set the hardstop latch if the check-control rotary switch is not on DISABLE during that part of the system reset where the extension storage is validated (regeneration of the GPR's and cancellation of the UCW's). Any error detection (such as bad parity), sets the hardstop latch by raising the machine-error signal.

The hardstop latch is reset at the beginning of a system reset or when the start latch is set. Further details of the hardstop state are given in the "Console" section of FEMM, Form Y33-0007.

Channels and Device Checking

Channels and device checking is described in the sections dealing with the multiplexor channel 0 and the high speed multiplexor channel in Principles of Operation - Channels, Form Y33-0003.

Other Checking

Other checking includes all power supply checking and is separately described in the "Power Supplies" section of FEMM, Form Y33-0007.

PROGRAM EXCEPTION DETECTION

This section describes the circuitry used to generate a program interrupt, and covers the exception detection circuits and the setting of the program interrupt latch while initiating the interrupt cycle.

The handling of interrupt cycles is explained in the "Interrupts" section of Principles of Operation - Processing Unit, Form Y33-0002.

The ten following exceptions, introduced in "Interruptions" in System/360 Principles of Operation, Form A22-6821, are applicable to the 2044:

Operation exception
Privileged operation exception
Specification exception (items 1, 2, 3 and 7 in Form A22-6821)
Addressing exception
Fixed-point overflow exception
Fixed-point divide exception
Floating-point significance exception
Floating-point divide exception
Floating-point exponent overflow exception
Floating-point exponent underflow exception

The detection of any exception sets the corresponding exception latch which afterwards causes the program interrupt latch to be set as explained subsequently in "Program Exception Handling."

As the checking is done in such a way that the program fault is detected as soon as possible, the

operation is suppressed, terminated, or completed, depending of the type of exception, the point in the operating cycle at which the exception occurs and its impact on the result. Control of the operation is determined by whether or not 'end execute' is forced.

Operation Exception

- Is set by an invalid operation at approximately the end of an I-cycle.
- Is cancelled by a system reset or at the end of interrupt cycles.
- Decoding circuits detect any non-assigned or non-available operation code.

The logic circuitry for this exception is shown in Figure 2-65. It detects all possible op codes that do not correspond to an operation that is valid for the 2044.

As shown in Figure 2-65, the signal 'invalid operation' sets the operation exception latch towards the end of the instruction cycle in which the invalid op code is read. This latch remains on until the beginning of the last cycle of an interrupt or until the occurrence of a system reset. (The latch is turned off by reset interrupt requests.) If a specification interrupt occurs at the same time, the operation exception latch is reset to generate the correct interrupt code, since specification supersedes an operation exception.

The signal 'invalid operation' can be raised by nine separate AND blocks which perform together an invalid operation decode in the same way as the instruction decode is performed. The circuit operation (Figure 2-65) is as follows:

AND block 1 eliminates non-assigned RR format operation codes and Set or Insert Storage Key (SSK or ISK).

AND block 2 eliminates non-assigned RR and RX format operation codes, Convert to Binary (CVB) and Convert to Decimal (CVD).

AND block 3 detects non-assigned RX format operation codes.

AND block 4 eliminates the execute operation.

AND block 5 discards floating-point instructions if the floating point feature is not installed. Note that the switch is provided by a particular SLT card which is wired accordingly.

AND blocks 6 and 7 eliminate non-assigned operation codes in the floating-point range.

AND block 8 eliminates all SS format operation codes. Note that an exception created in this way causes an interrupt, under which circumstances, the old PSW 2 contains an instruction length code of 11 (see "PSW Registers" in the section "System Control Components" in this chapter).

AND block 9 detects all non-assigned and non-available RS and SI format operation codes.

Privileged Operation Exception

- Is set towards the end of the I-cycle when a privileged operation is attempted while the CPU is in the problem state.
- Is cancelled by reset interrupt requests.
- Feature instructions are privileged operations.

The privileged operations are:

Start I/O	Load PSW
Halt I/O	Set system mask
Test I/O	Diagnose
Test channel	Feature instructions

Figure 2-65 shows the conditions able to set the privileged operation exception latch if PSW 1 bit 15 is present, that is, when the CPU is in the problem state. The latch is set approximately at the end of the I-cycle in which the privileged operation code has been read.

The latch is reset by a reset interrupt request which is raised during a system reset or at the end of interrupt cycles.

The five feature instructions, consisting of read or write direct word, enable or disable priority mask, and exit priority interrupt are considered as privileged operations.

Specification Exception

- Detects an improper boundary specification for data or information held in core storage.
- Detects an improper GPR address.
- Detects invalid FPR addresses.
- Requests an interrupt when a PSW with a non-zero storage protection key is loaded.

Figure 2-66 shows the logic circuitry which is provided to detect addresses that specify incorrect boundaries for the particular unit of information. The detection applies to main storage, the GPR's and the FPR's. The actual addressing specifications for each instruction are given in System/360 Principles of Operation, Form A22-6821.

Circuitry is also provided to detect a non-zero storage protection key during a PSW loading as

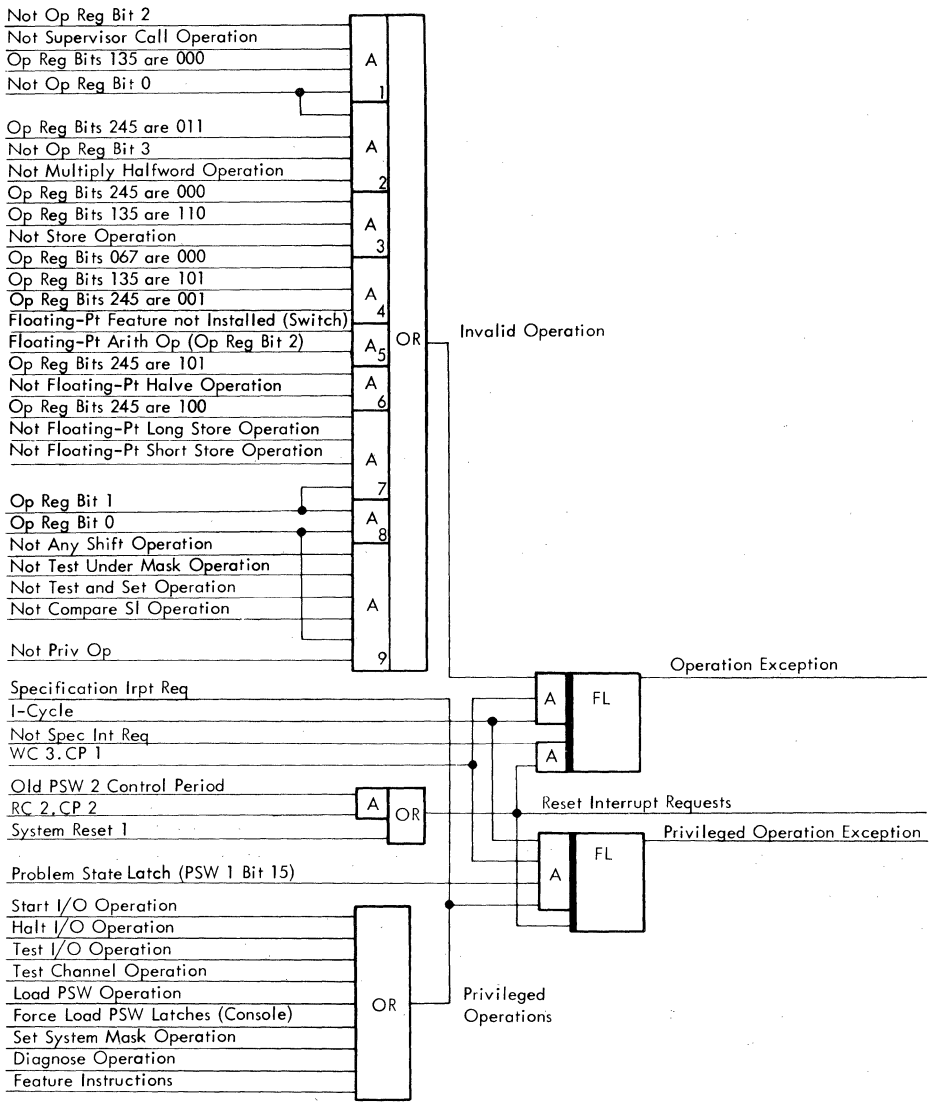


Figure 2-65. Operation and Privileged Operation Exceptions

referred to previously under the heading "PSW Register" in the section "System Control Components" in this chapter.

The specification interrupt request latch can be set by ten separate AND blocks. Each AND block is active at the time appropriate to the particular check, as checks are performed as soon as possible. The latch is reset by reset interrupt requests. The function of each AND block in Figure 2-66 is as follows:

AND block 1 prevents any attempt to address an odd-numbered GPR as operand 1 for fixed-point divide, multiply (except multiply halfword), and double-shift operations, since these involve a double-word operand 1 which implies that the addressed GPR must be even-numbered.

AND block 2 ensures that an instruction address defines a halfword boundary address in main storage (a multiple of 2). The block therefore rejects odd addresses which are on byte boundaries only.

AND blocks 3, 4, 5 and 6 are used with floating-point operations. Blocks 3 and 4 prevent an operand 1 from being addressed outside the correct FPR addresses which, in this case, are 0, 2, 4 and 6. Blocks 5 and 6 are used for the detection of incorrect operand 2 addresses in FPR's (RR format).

AND block 7 is used to determine if double-word information or data, such as PSW's or long-precision floating-point operands, is located in core storage on double-word boundaries (multiples of 8).

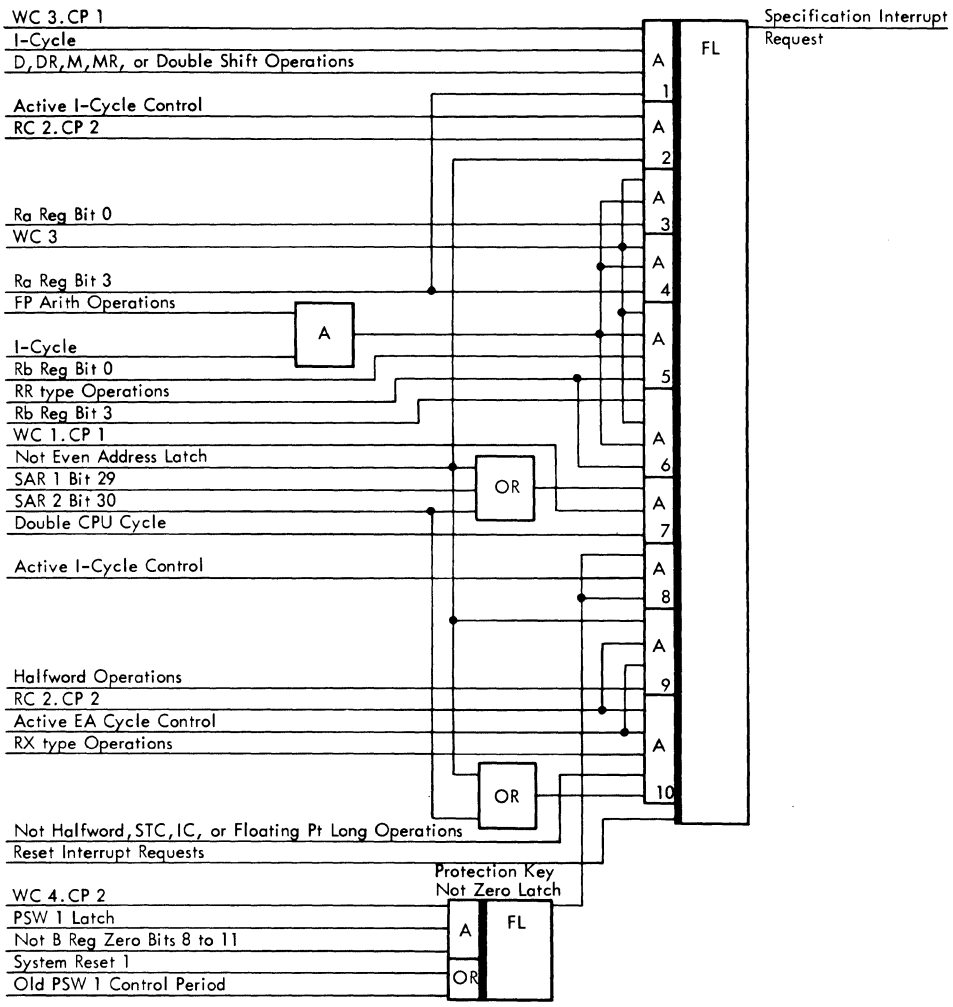


Figure 2-66. Specification Exceptions

AND block 8 sets the specification interrupt request latch during the old PSW 2 period (third status switching cycle) if the protection-key-not-zero latch is on. This latch is set during PSW 1 loading (second status switching cycle) if the storage protection key (PSW 1 bits 8 to 11) of the new PSW is not zero. The storage protection key is analyzed in the B register which is the normal path for loading the PSW. (See "PSW Registers" in the "System Control Components" section of this chapter.)

AND block 9 gives an exception if the address of a halfword operand is not on a halfword boundary in main storage (multiple of 2). This block also gives an exception if the address of a normal single-word operand 2 is not on a word boundary in main storage (multiple of 4).

Addressing Exception

- An exception is recognized when the main storage address is outside the available storage capacity.
- The exception latch is set towards the end of the cycle in which the information is read.

An addressing exception is recognized whenever an attempt is made to address data or information outside the available storage capacity of the particular model of the 2044. Figure 2-67 shows the logic circuitry associated with the addressing exception.

The address interrupt request latch differs from other exception latches in usage because of the way in which available SLT modules are connected to give the required latch function. The second logic

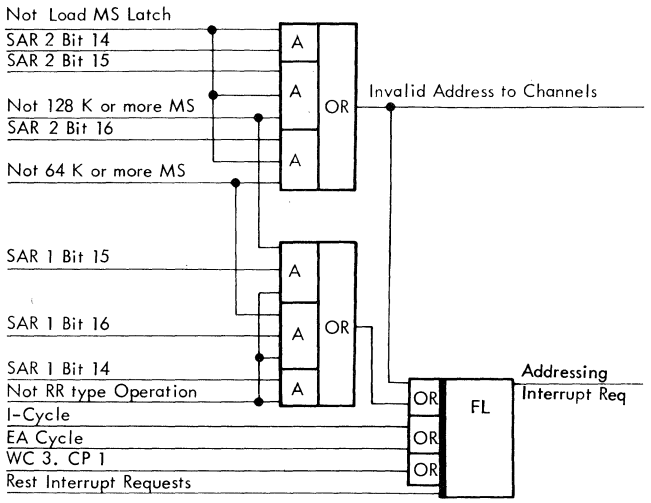


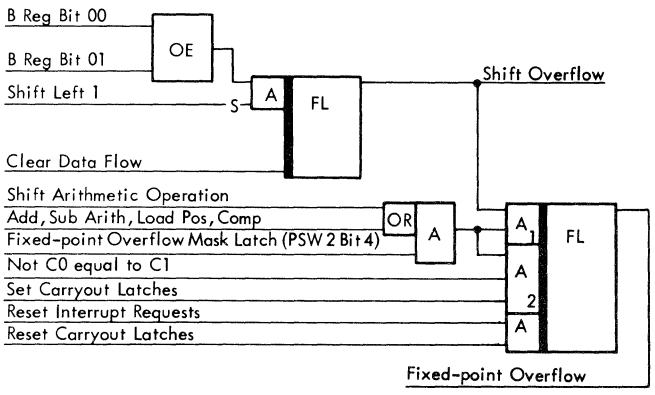
Figure 2-67. Addressing Exception

block allows the addressing of a halfword instruction situated on the last two locations of storage, since the address interrupt request latch is set at WC 3, CP 1 after the output from the instruction registers and decode circuits (for an eventual RR format operation) has been determined. The latch is reset by 'reset interrupt requests' (see Figure 2-65).

Fixed-Point Overflow Exception

- Can be masked by PSW 2 bit 4.
- Is set with add, subtract and shift arithmetic operations and with load positive or complement.
- Is reset by the reset interrupt requests signal.

Figure 2-68 shows the logic circuitry provided to generate a fixed-point overflow exception.



S = Shift pulse with the appearance of the condition

Figure 2-68. Fixed-Point Overflow Exception

Two conditions must be active to set AND block 1 which, in turn, sets the fixed-point overflow latch.

1. The first condition occurs when a shift overflow has been detected during a shift operation. The shift overflow is stored in the FL block, which is a multi-input trigger as described in Appendix B2 of FEMM, Form Y33-0007. (Note that the shift pulse shown in Figure 2-68 is derived from a circuit of the type described in Appendix B1 of the same manual.) The effect of the output from the EXOR block is further described in the Fixed-Point Instruction section of Principles of Operation - Processing Unit, Form Y33-0002.
2. The second condition is derived from an AND block with shift arithmetic operations (logical shifts do not take note of lost bits) and with the fixed-point overflow mask latch (PSW 2 bit 4) on.

AND block 2 also needs the second condition of AND block 1, in conjunction with the condition of carry-out latches C0 and C1 not equal. (see "Add and Subtract Instructions" in Principles of Operation - Processing Unit, Form Y33-0002.)

The validity of operation of the latch with AND block 2 is clarified in Appendix B3 of FEMM, Form Y33-0007, since the peculiarities of setting the C0 and C1 latches apply equally to the fixed-point overflow latch.

The latch remains on until the reset interrupt requests signal is raised (see Figure 2-65), as the output from the reset carry-out latch is also fed to the latch to ensure correct logical operations.

Fixed-Point Divide Exception

- Recognized when the first divide cycle generates a quotient bit.
- Recognized towards the end of the operation with a quotient bit in the second cycle which is not the maximum negative number.
- Reset by 'reset interrupt requests'.

A fixed-point divide exception is generated when the quotient size (sign + integer) exceeds the quotient register size (BX register, 32 bits).

Two separate AND blocks are used to detect an oversized quotient (Figure 2-69). AND block 1 sets the exception latch if a quotient bit is developed during the first divide cycle, since 32 left shifts (corresponding to 32 divide cycles) are still to follow at this point. In this case, the signal 'end execute fixed-point divide' is active at the end of the compute cycle with CC 6 and stops the divide operation. A program interrupt then takes place immediately.

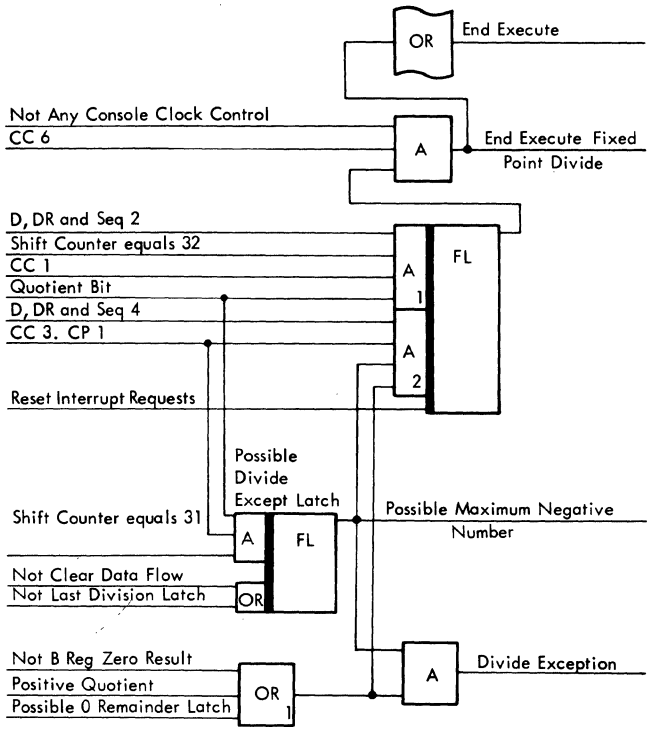


Figure 2-69. Fixed-Point Divide Exception

If a quotient bit is developed during a second divide cycle, it would normally be in position 00 at the end of the operation (31 left shifts from the least-significant position). Position 00 is the sign position, and the presence of a bit in this position at this point again indicates that the register capacity has been exceeded. If, however, the quotient is the maximum negative number and (since a quotient is developed in true form) one more position is needed to accommodate it, this number should be a one followed by 31 zeros. Therefore, when a quotient bit has resulted from the second divide cycle, it is stored by the possible divide exception latch which gives an output 'possible maximum negative number'. The quotient bit is not inserted in the BX register. It is used, with already existing circuits, to facilitate a test for zero (in the B register after interchange with the BX register), to make sure that it is the maximum negative number.

If it was anticipated that the quotient sign would be negative (from the previously set sign latches) and the possible 0 remainder latch is not set, an exception is not set and the B register bit 00 (sign) is gated inverted on completion of the operation. (See Figure 2-8 in this chapter.) If any of the conditions produce an output from OR block 1, a 'fixed-point divide exception' is recognized. The R1 storage cycle control produced to store the result is inhibited by divide exception and the result is not recorded.

Floating-Point Arithmetic Exceptions

- FP arithmetic exceptions are:
  - FP significance exception
  - FP divide exception
  - FP overflow exception
  - FP underflow exception
- The FP significance exception can be masked by PSW 2 bit 6.
- The FP underflow exception can be masked by PSW 2 bit 7.
- The logic is present only when the floating point feature is installed.

All the above FP arithmetic exceptions can cause a program interrupt to be requested. For the FP significance and the FP underflow exceptions, the corresponding mask bit in PSW 2 must be a one for the exception to be recognized.

For a full explanation of these types of exceptions refer to Floating Point Feature, Form Y33-0005.

PROGRAM EXCEPTION HANDLING

- An exception generates end execute if necessary.
- An exception generates a program interrupt if there is no machine check.
- An exception with program interrupt generates the appropriate program interrupt code.

The handling of a program exception consists normally of three main steps:

1. Termination of the operation (optional).
2. Generation of a program interrupt (if there is no machine check interrupt request).
3. Generation of an interrupt code (PSW 1 bits 28, 29, 30 and 31).

Termination of the Operation

According to the kind of exception and the time at which it is recognized, end execute (which is normally raised at the end of an operation) can be forced. Figure 2-70 shows the exceptions capable of forcing end execute.

The 2044 meets the specifications established by System/360 architecture, concerning the execution of the operation (completed, terminated or suppressed).

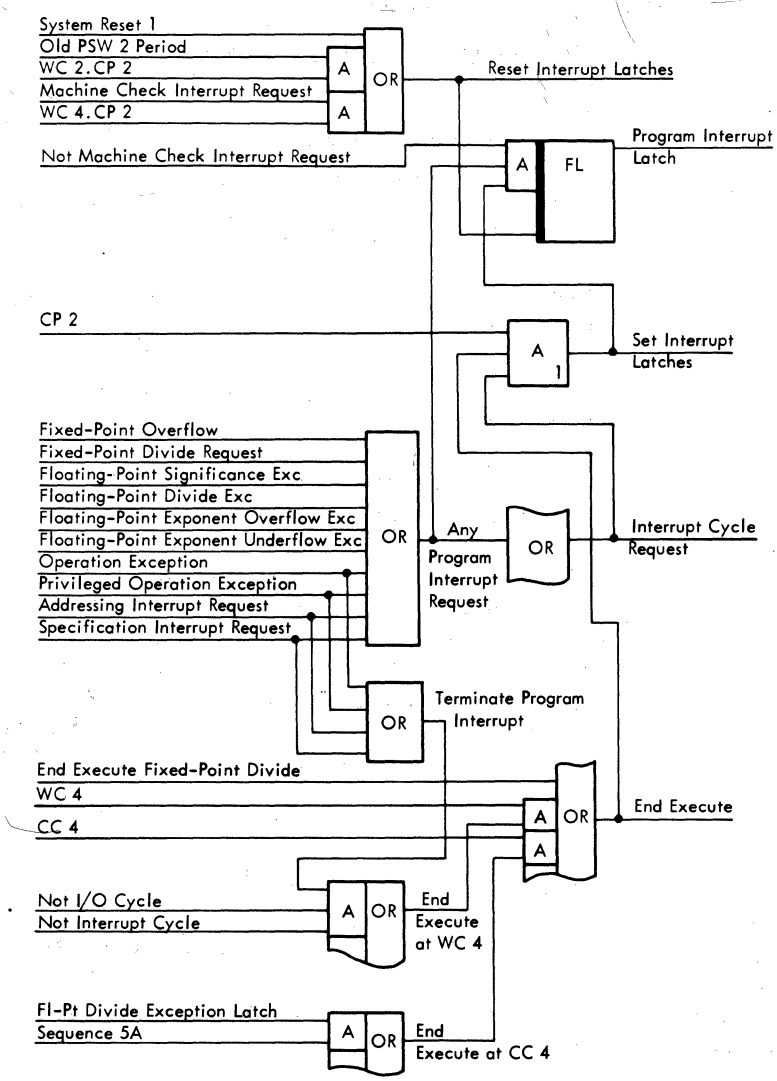


Figure 2-70. Program Exceptions Handling

Fixed-point overflow, exponent underflow, significance exceptions, and 2044 floating-point exponent overflow exceptions do not stop the operation; the interrupt is taken when end execute appears normally.

As shown in Figure 2-70, operation, privileged operation, addressing and specification exceptions force 'end execute' at WC 4.

A fixed-point divide exception forces 'end execute' at CC 6, as shown in Figure 2-69.

A floating-point divide exception forces 'end execute' at CC 4 (and sequence 5A).

The foregoing methods of ending an operation are summarized in tabular form at the end of this section.

Program Interrupt Generation

Figure 2-70 shows the logic circuitry provided to set the program interrupt latch. This is achieved with the output of AND block 1 which is active at end execute (see "Termination of the Operation") and CP 2 with any program interrupt request. The signal 'set interrupt latches' starts the interrupt cycles (old PSW 1 control cycle).

The latch is not set if there is a machine check interrupt request as this has a higher priority than program interrupt.

The latch is reset at the end of the interrupt cycles or during system reset, or at WC 4, CP 2

if there is a machine check interrupt request.

For complete details on the handling of program interrupts refer to "Interrupts" in Principles of Operation - Processing Unit, Form Y33-0002.

Interrupt Code Generation

As described in "PSW Registers" in the System Control Components section of this chapter, an interrupt code is generated at the time of storing the old PSW 1. Figure 2-71 shows the circuitry which generates the interrupt code as given in the following table according to the type of exception.

If a floating-point exception (exponent overflow, exponent underflow, significance or FP divide) occurs at the same time as either a specification or addressing exception, the FP exception latch is reset to prevent an FP interrupt code generation, in favor of a specification or addressing interrupt code.

In the same manner, a specification exception has priority over an operation exception. (See Figure 2-65.)

Program Interrupt Cause	Interrupt Code bits 28 to 31 (bits 16 to 27 are zeros)		Execution
Operation	0001		Suppressed
Privileged Operation	0010		Suppressed
Addressing	0101		Suppressed or Terminated
Specification	0110		Suppressed
Fixed-Point Overflow	1000	*	Completed
Fixed-Point Divide	1001		Suppressed
Exponent Overflow	1100		Completed
Exponent Underflow	1101	*	Completed
Significance	1110	*	Completed
Floating-Point Divide	1111		Suppressed

* Indicates that the interrupt is maskable by the program mask (PSW 2 bits 4 to 7)

NOTE: Operation "suppressed" means that the result is not recorded and that the operands are left unchanged although the execution may have progressed to a certain point.

Operation "terminated" means that the operation was not completed but might have changed the operands. The eventual result may not be used for further computation.

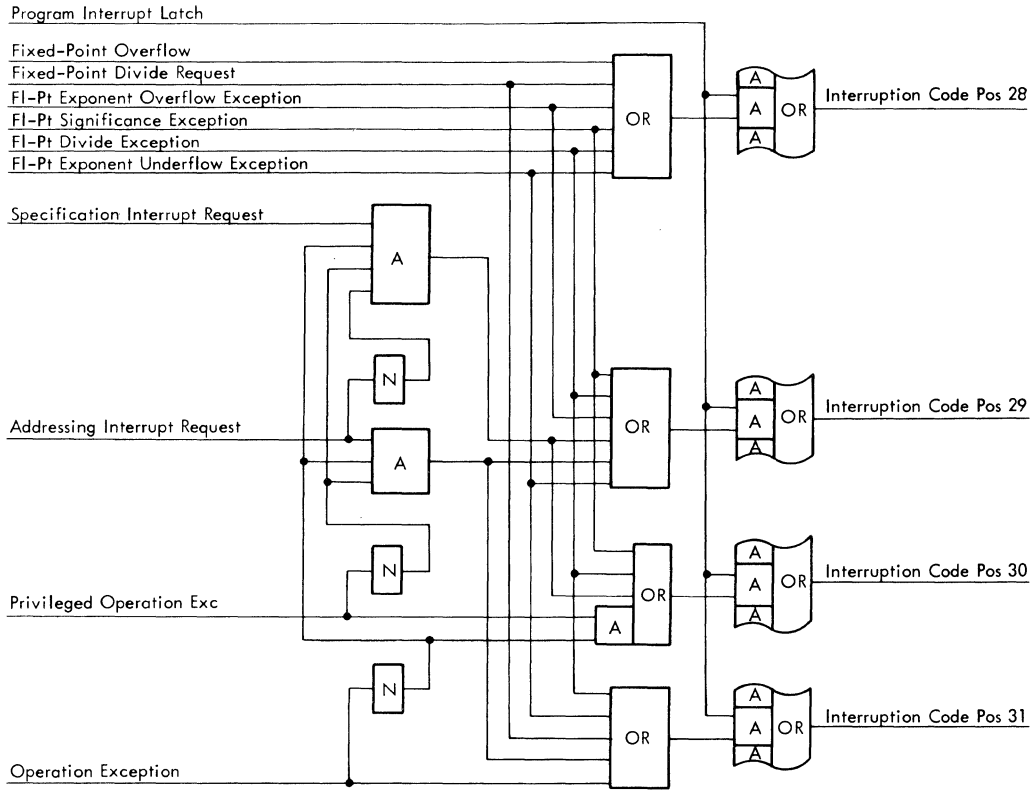


Figure 2-71. Program Interruption Code Generation

ACCELERATOR FEATURE

HARDWARE GENERAL-PURPOSE REGISTERS

MSLS Logic Modules

- Monolithic circuit using single silicon chip.
- Provides circuits for six bits per module.
- Packaging dictates that the six bits are used as three bits for each of two words.

The 16 hardware general-purpose registers for the accelerator feature consist of Medium-Speed Local Storage (MSLS) modules. Each module, manufactured as monolithic circuits on one silicon chip, is an integrated network of three bits for each of two words on a 16-pin SLT module package (Figure 2-72). It consists of the following circuits:

6 latches (EXOR L) : bit stores
2 write gates : drivers
6 read gates
1 unit cell inverter

The inputs and outputs of this unit are compatible with SLT circuit levels. No special drivers, sense amplifiers or output latches are required.

Each bit, or cell, has separate read and write lines, and common data-in and data-out lines. This allows a higher packaging density on a card, because card pins are the limiting factor.

MSLS Card

- MSLS card contains 16 MSLS modules.
- MSLS card provides 12 bits for each of eight words.
- Each card is divided into four MSLS arrays.
- MSLS array provides three bits for each of eight words.
- ALD representation shows MSLS arrays.

A MSLS card contains 16 MSLS modules (Figure 2-73). The data lines and the read/write lines of these modules are coupled in such a way that each

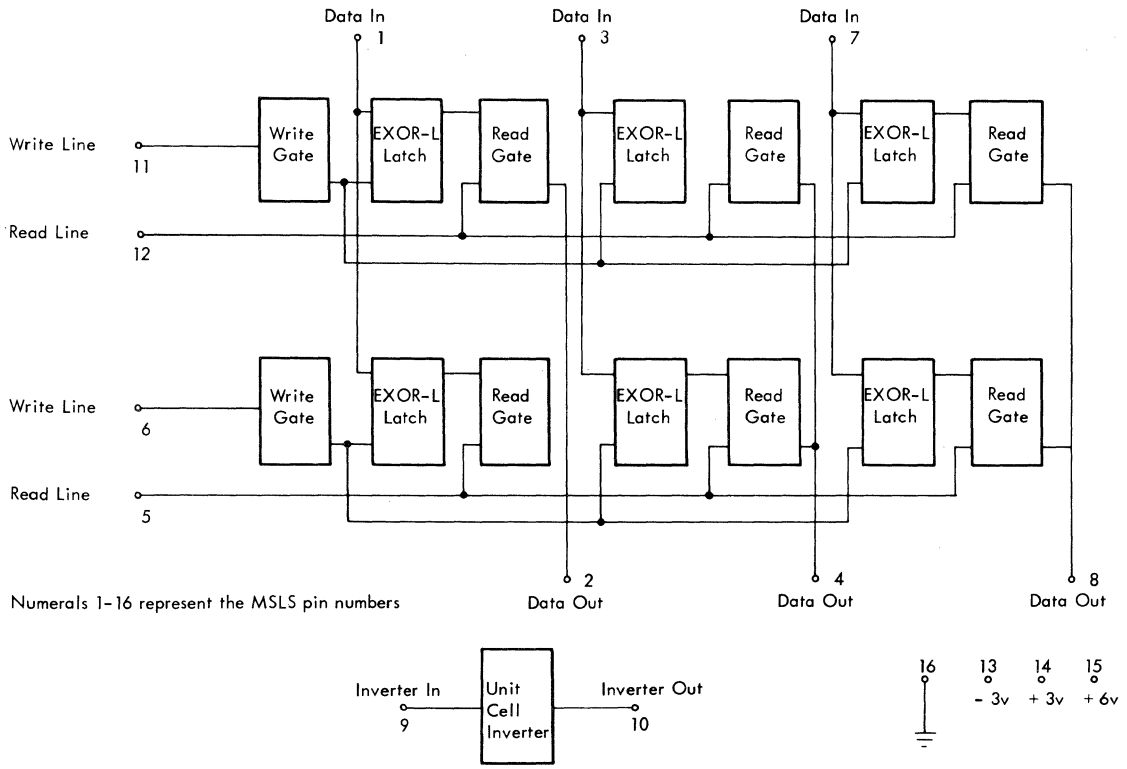


Figure 2-72. Six-Bit MSLS Logic Block

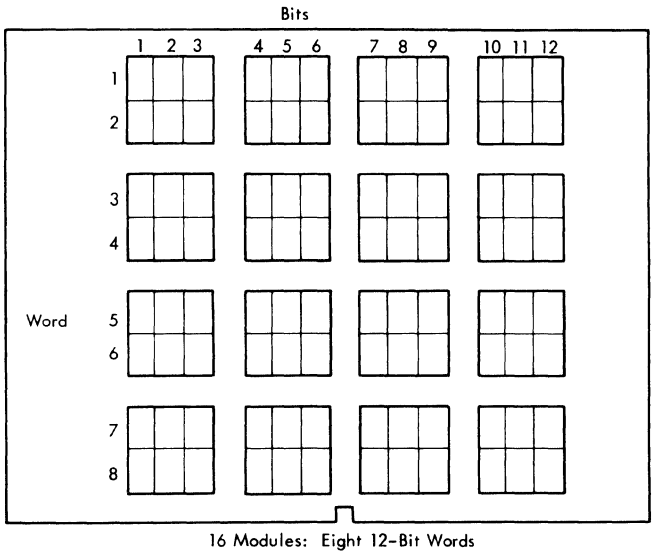


Figure 2-73. MSLS 96-Bit, 24-Pack Card

horizontal row of bit positions is in the same word, and each vertical column of bits represents the corresponding bit position of these words.

On the card shown in Figure 2-73, the horizontal rows represent words 1 to 8 and the vertical columns bits 1 to 12 of these words.

Each vertical column of MSLS modules within a card is termed an MSLS array for the purposes of ALD representation. Thus, each MSLS array represents three bits for each of eight words.

MSLS ALD Representation

- MSLS circuits are represented as MSLS arrays.
- MSLS array contains three bits for each of eight GPR's.
- Data-in lines are commoned to the corresponding bits of each of the eight words.
- Data-out lines common to corresponding bits of each of the eight words.

The ALD representation of the MSLS array (Figure 2-74) shows the data-in and data-out lines for each of the three bits, and the 'array drive write' and 'array drive read' lines for each of the eight words.

For GPR's 0 to 7, eleven arrays are required which provide 33 bits; the last bit positions of the arrays for each of the eight words are unused.

A further eleven MSLS arrays are required for GPR's 8 to 15.

MSLS Array Read/Write Controls

Read Out Operation

- Non-destructive read out.
- Read out with positive signal.

Data is read out of the appropriate three bits of the MSLS array when an 'array drive read' line is activated. This line causes the output data line to reflect the data in these bit positions, for as long as the read line remains active, without affecting the data in the cell, that is, the read out is non-destructive.

Write-In Operation

- 'Array drive write' is positive-going ac signal.
- Writes data in from corresponding data-in line.

The MSLS write drive line is normally positive. When this write line is made negative, the bits for that word are set to the state of the corresponding data-in line when the 'array drive write' line next rises to its positive condition.

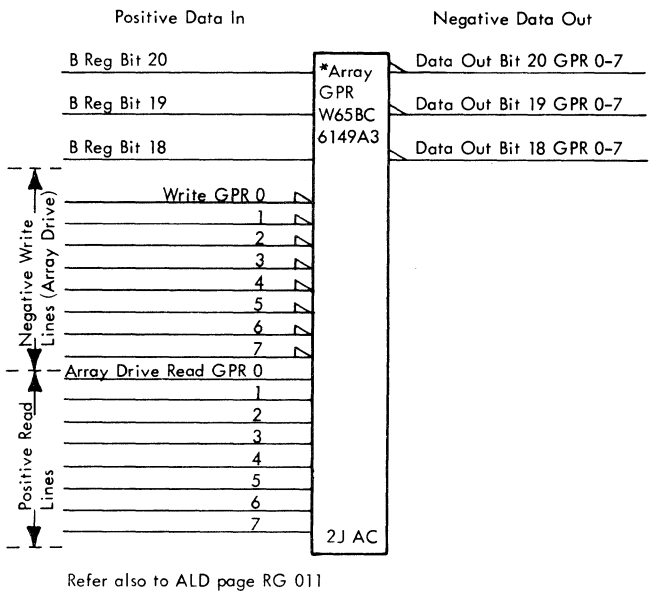


Figure 2-74. MSLS Array GPR's 0 to 7 Bits 18 to 20

MSLS Addressing Read Drive

- 'GPR read address' bits (GPR number) are read out of Ra, Rb, Rc or address switches, according to operation.

- Address bits are decoded in primary-read decode and secondary-read decode to provide 'GPR read drive' for the addressed GPR.

- Read drive is active only during the time the address is gated to provide 'GPR read address' bits.

The signals 'gate out Ra register', 'gate out Rb register', 'gate out Rc register' or 'display latch' cause the corresponding GPR address bits to be gated to the GPR address bit lines. The one exception is for a floating-point instruction, when the FPR address decode is made active and the GPR address decode inhibited.

The GPR address bit lines enter the primary-read decode, where the bit pattern of the GPR number is used to make active two of the following eight binary patterns:

```
0 - - 0   - 0 0 -      1 - - 0   - 1 0 -
0 - - 1   - 0 1 -      1 - - 1   - 1 1 -
```

The output of this primary-read decode then enters the secondary-read decode, where the bit patterns are combined to provide the appropriate read drive GPR line.

For example, if the GPR to be addressed is GPR 12, the 'GPR read address' lines are:

```
GPR read address bit 0 : 1
GPR read address bit 1 : 1
GPR read address bit 2 : 0
GPR read address bit 3 : 0
```

The active primary-read decode patterns are:

```
1 - - 0 and - 1 0 -
```

These two lines combine in the secondary-read decode to provide the 'read drive GPR 12' line.

MSLS Addressing Write Drive

- 'GPR write address' permanently gated from Ra register.

- Address bits decoded in primary-write decode to provide address patterns.

- Output of primary-write decode combines with write strobe in secondary-write decode to provide write drive for the addressed GPR.

The content of the Ra register is permanently available as an input to the GPR primary-write decode circuits, provided that a floating-point instruction is not being executed. For the latter case the FPR decode is used and the GPR decode is inhibited.

This primary-write decode produces two bit pattern lines in a similar manner to the primary-read decode.

The output of the primary-write decode network is made available to the secondary-write decode when the write strobe is generated for the MSLS.

This secondary-write decode then decodes the two bit patterns, in a similar manner to the secondary-read decode, to provide the write drive for the addressed GPR.

The write strobe controls the write-in to the GPR. When the strobe is active, the appropriate GPR is selected; when the strobe next drops to the inactive state, the GPR is loaded with the data on the corresponding data-in lines.

Data Flow

The 16 hardware GPR's are coupled into the main data flow (Figure 2-75) with an access time of 250 nanoseconds for a full 32-bit access to each register. This compares with the basic machine access time

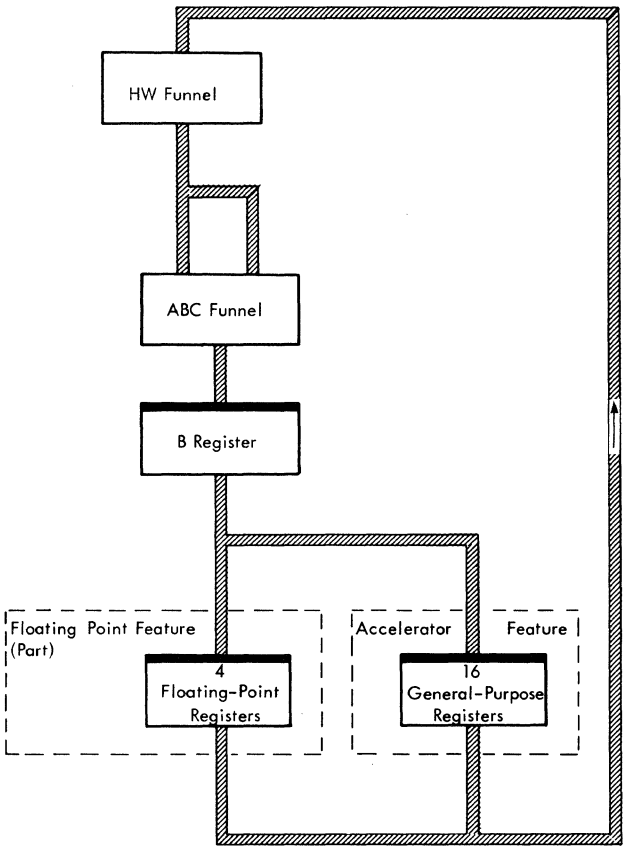


Figure 2-75. Accelerator Data Flow (Simplified)

of 1.0 microsecond (refer to "Speed" section in this chapter). The hardware GPR's have common data input lines from the B register output bus. Their output lines are also commoned and combine with the FPR's output on the hardware-register bus. This bus is taken to the HW funnel, whose output provides two separate inputs for the ABC funnel. These two inputs allow gating of full-word data to the ALS section of the CPU and allow for alignment of halfword operands (bits 16 to 31) or characters (bits 24 to 31) from the GPR's to either half of the main data flow. (See "ABC Funnel and HW Funnel" in this chapter.)

COMMENT SHEET

Introduction and Functional Units
Field Engineering Theory of Operation, Form Y33-0001

FROM

NAME _____ OFFICE/DEPT NO. _____

CITY/STATE _____ DATE _____

To make this manual more useful to you, we want your comments: what additional information should be included in the manual; what description or figure could be clarified; what subject requires more explanation; what presentation is particularly helpful to you; and so forth.

CUT ALONG LINE

How do you rate this manual: Excellent ____ Good ____ Fair ____ Poor ____

Suggestions from IBM Employees giving specific solutions intended for award considerations should be submitted through the IBM Suggestion Plan.

NO POSTAGE NECESSARY IF MAILED IN U.S.A.

FOLD ON TWO LINES (LOCATED ON REVERSE SIDE), STAPLE AND MAIL

fold fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

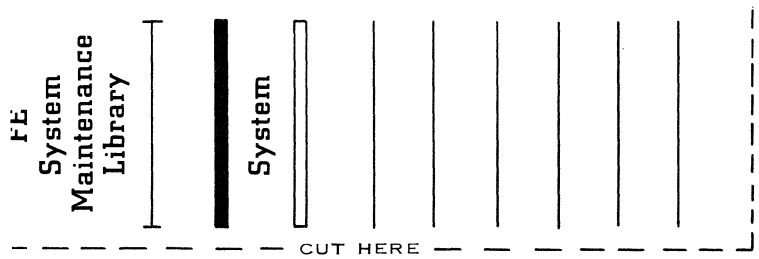
POSTAGE WILL BE PAID BY . . .

IBM CORPORATION

P.O. BOX 390
POUGHKEEPSIE, N.Y. 12602

ATTENTION: FE MANUALS, DEPT. B96,

fold fold



Y33-0001-0

Y33-0001-0 S/360 Model 44 Printed in U.S.A.

IBM

International Business Machines Corporation
Field Engineering Division
112 East Post Road, White Plains, N.Y. 10601