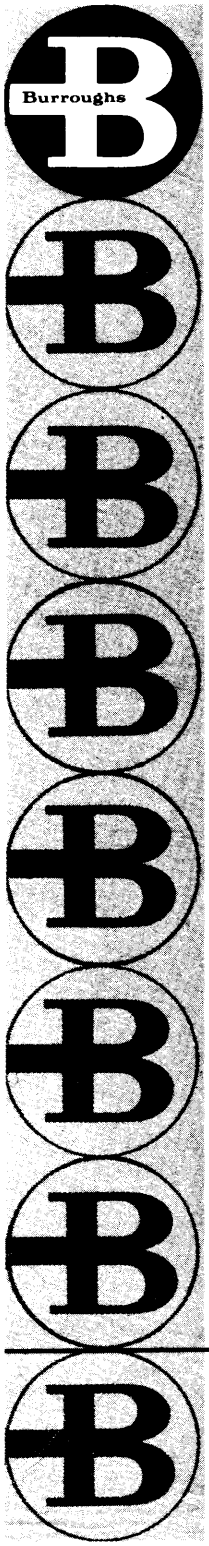
BJ-27

JUNE 1966
REVISED: OCTOBER 1966



INSTRUCTION OPERATIONS

for the

B8501
CENTRAL PROCESSOR MODULE

REFERENCE MANUAL

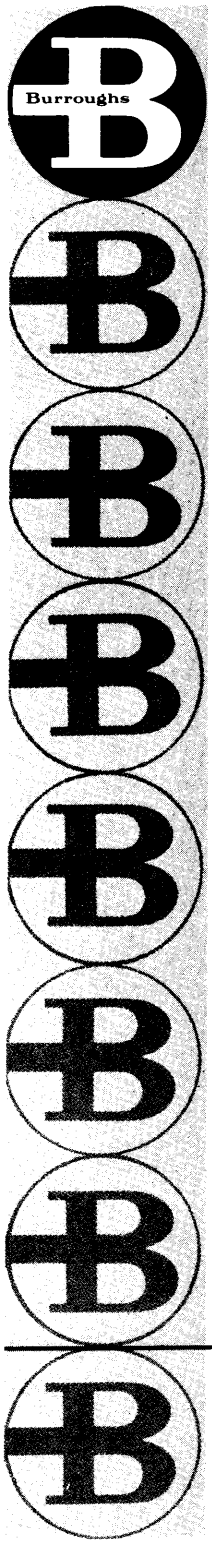
This publication replaces BJ-27 Manual dated July 1966.

Burroughs Corporation

Defense, Space & Special Systems Group

BJ-27

JUNE 1966
REVISED: OCTOBER 1966



INSTRUCTION OPERATIONS for the B8501 CENTRAL PROCESSOR MODULE

REFERENCE MANUAL

This publication replaces BJ-27 Manual dated July 1966.

Burroughs Corporation

Defense, Space & Special Systems Group

The information in this publication is subject to change. Revisions will be issued to advise of such changes and additions.

Original Issue: June 1966
First Revision: July 1966
Second Revision: October 1966

TABLE OF CONTENTS

TABLE OF CONTENTS

(CONTINUED)

TABLE OF CONTENTS

(CONTINUED)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1

INTRODUCTION

PURPOSE OF MANUAL

This manual describes the operation of the instruction set for the Central Processor Module B8501. Also included is a brief description of the characteristics, functions, and features of the Processor Module, the Memory Module B8505, and the I/O Control Module B8510. The Interrupt System, the jump sequences, I/O initiation and instruction set are described only in the detail required for a programmer to understand.

NOTE

The characteristics and functions of the B8500 Data Processing System are fully explained in the following manuals:

- System Reference Manual, BJ-8
- Executive and Scheduling Program Reference Manual, BJ-4
- I/O Control Module Reference Manual, BJ-25

GENERAL SYSTEM FUNCTIONS

In the B8500 Data Processing System, each function is processed by a type of module that is specifically designed for its task: the B8505 Memory Module provides high speed storage functions, the B8510 I/O Module controls the input/output functions, and the B8501 Central Processor Module executes the data processing functions. All modules of one type are identical. Thus, a standard interconnection network provides for freedom in the selection of a system configuration.

MEMORY MODULE

As required for a specific installation, there may be as few as 1 and as many as 16 B8505 Memory Modules in a B8500 System. Each Memory Module can operate as an independent storage unit, which allows simultaneous accessing of separate Memory Modules. All programs are written with relative addressing to ensure relocatability.

There may be as many as 16 communication busses connected to the Memory Modules, and these busses can serve the Console and any combination of Central Processor Modules and I/O Modules up to 14. (One buss is used for test facilities.)

I/O CONTROL MODULE

A feature of the B8510 I/O Control Module is its independence from the Central Processor Module. The I/O Module, except for sharing Memory Modules with the Central Processors, is a separate processor with its own local memory unit, logic, arithmetic functions, and communications capabilities, which minimizes Central Processor time expended for I/O functions. Because it provides channels for all data transfers between peripheral devices and the System's Memories, the I/O Module can control transfers from one peripheral device to another peripheral device independent of direct Central Processor control. The system operations of the I/O Module are programmed separately from those of the Central Processor Modules.

CENTRAL PROCESSOR MODULE

The primary functions of the B8501 Central Processor Module are to execute arithmetic calculations, special control functions, data transfer operations, and Interrupt services.

The Central Processor Module, which uses a fixed-word, variable-length instruction (See Section 2), consists of three functional stations:

- Advance Station (ADVAST).
- Final Station (FINST).
- Communications Unit (COMM).

The three stations are as operationally independent as possible. In general, address arithmetic is performed at ADVAST, data manipulation operations are performed at FINST, and interfacing, fetching, and storing functions are performed at COMM. A simplified drawing showing the principal paths for the flow of data and control information between the three stations, together with a listing of the functions, is given in Figure 1-1.

Advance Station

ADVAST is the pre-execution stage for all instructions. An Instruction Look-Ahead (ILA) buffer in ADVAST contains instructions that are fetched in advance of their use. A minimum of 24 instructions can be awaiting execution. With this amount of look-ahead, COMM keeps the ILA sufficiently ahead of actual ADVAST computations to mask the time taken in fetching program words. From the top of the ILA, ADVAST removes one instruction at a time and modifies addresses, if required.

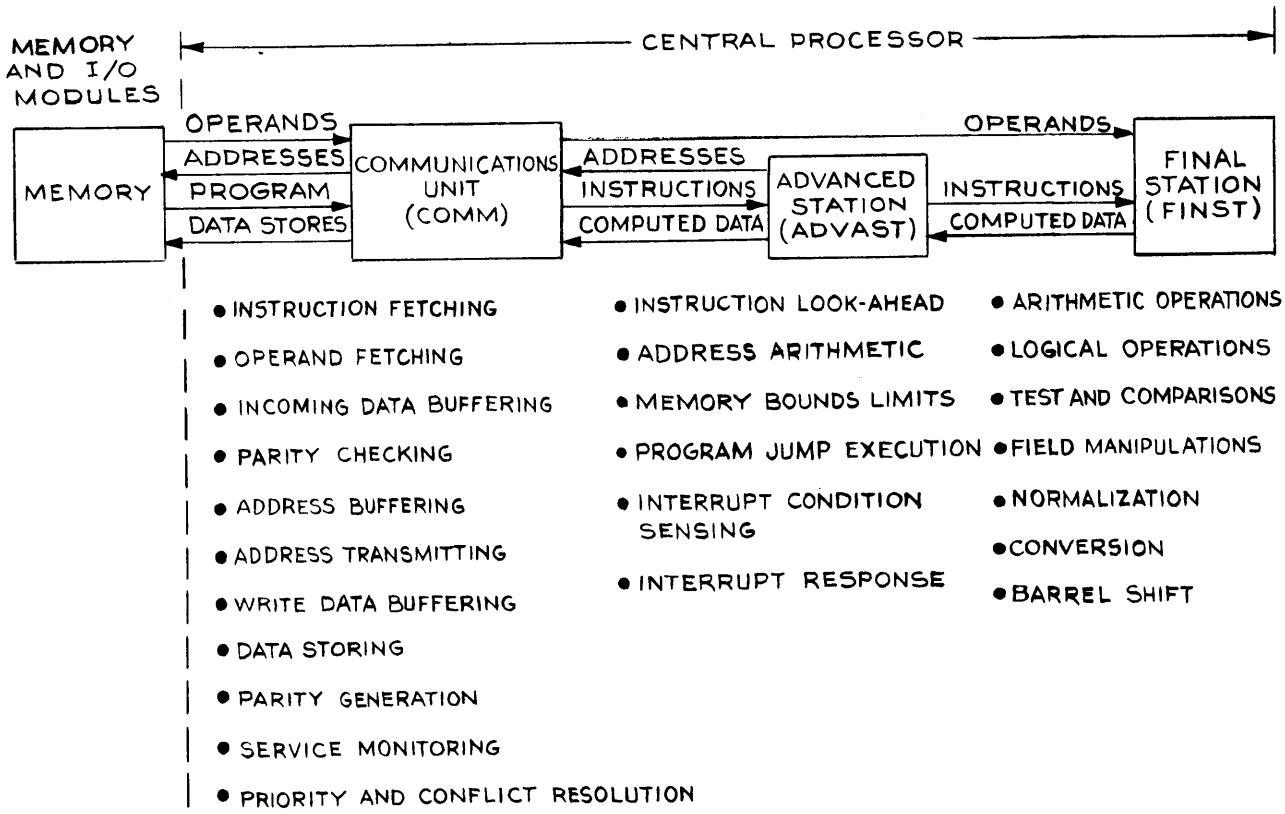


Figure 1-1. Functions of Central Processor Stations

The fetch and storage of required operands can be directed to different areas of memory with the use of one of the several base address registers available in ADVAST for operand address modification. This feature permits parts of programs to be located in different areas of memory. For example, data in one area of memory can be referred to by use of the Base Data Register, and instructions in another area of memory can be referred to by use of the Base Program Register.

By address modification, ADVAST develops an effective operand address and presents it to COMM for memory service. Meanwhile, the instruction in ADVAST is placed at the bottom of the Final Station waiting queue (FINQ). When the operands arrive from memory through COMM, they are placed in TEMPQ adjoining their associated instruction, in Final Station (FINST).

ADVAST also performs bounds checks on address references to memory, and if any limits are violated, an Interrupt Control Register (ICR) bit is set.

Final Station

Primarily, the Final Station (FINST) performs arithmetic, logical, Stack, and Stack test operations. During the time that ADVAST is partially executing instructions, FINST is completing the execution of other instructions removed from the top of the waiting list (FINQ).

There can be up to 14 operands in FINST at a given time. These operands, as a group, are contained in the Stack. The top four positions are the T Register, the S Register, the N Register, and the M Register (see Figure 1-2). These four locations are the data manipulation registers. A separate P Register is provided for operations that require an extension of the T Register.

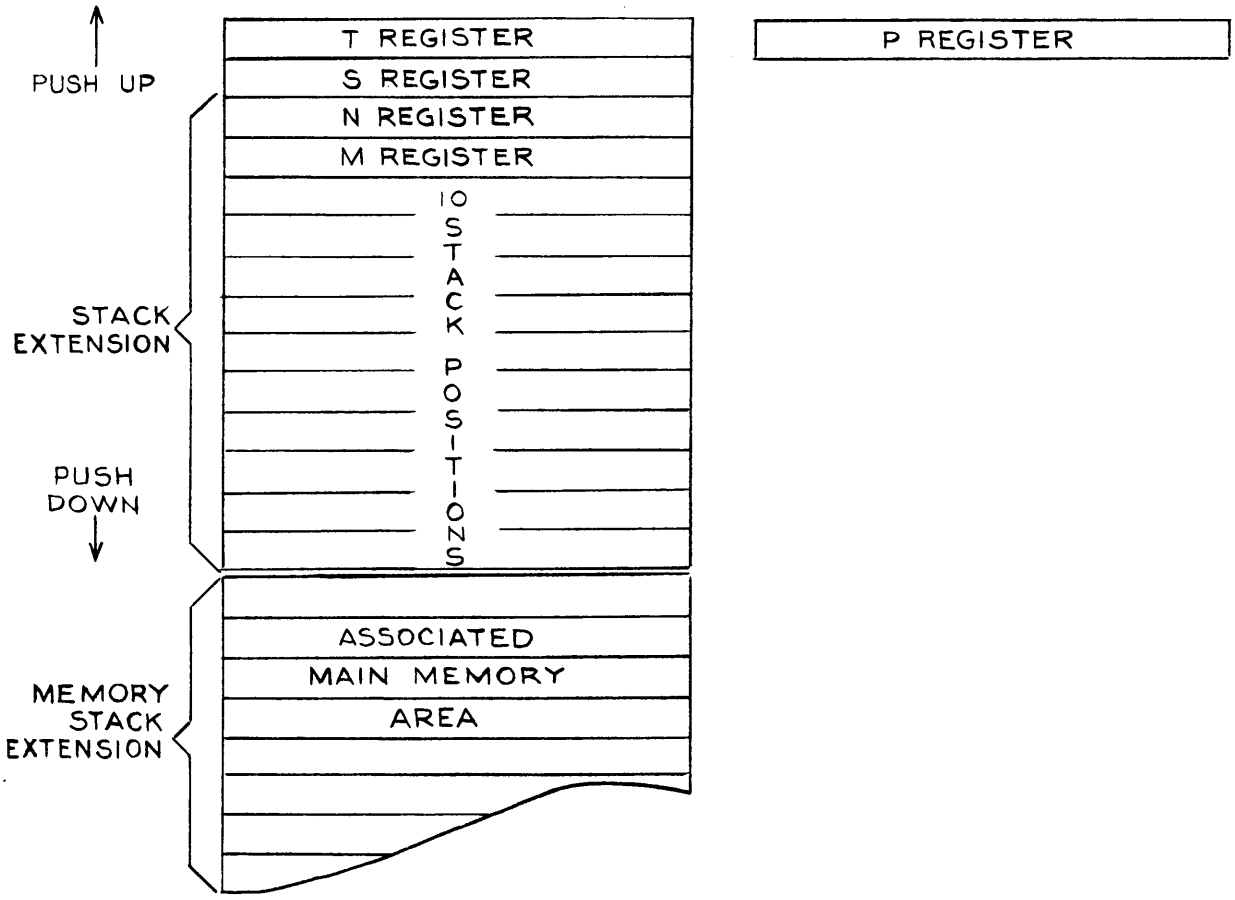


Figure 1-2. Stack in Final Station

The T Register is the most often used, and words that are operands in an instruction execution may come directly from memory to the T Register as the result of a fetch. Also, however, the 14 Stack positions are dynamic and the control logic of the Stack can effectively push up the operands toward the T Register, or it can push the operands down into memory (Stack Extension).

Pushing operands up or down is controlled by instructions (e.g., SSM, FMS), and causes the T Register to be emptied or loaded. In this way, Stack operands can be pushed into memory automatically. Data in the Stack Extension is fetched or stored in and out of the bottom of the Stack. If the Stack is pushed down, then operands at the bottom of the Stack are placed in the Stack Extension (memory) in sequential locations. The operands pushed up and out of the T Register are either lost or stored as specified by a particular instruction. This causes more operands to be placed into the bottom of the Stack from the Stack Extension.

Thus, operands are presented to the Stack at T, and are stored sequentially one on top of the other in the Stack. Likewise, as they are called for execution, the operands are presented sequentially, the most recent operand first.

Communications Unit

The Communications Unit (COMM) performs and coordinates all references to the Memory Module by ADVAST and FINST. Specifically, COMM performs the following functions:

1. Fetches instructions for the ADVAST waiting list.

2. Fetches operands for related instructions located in the FINST waiting list (FINQ).

3. Fetches and stores operands between the bottom of the Stack and memory.

4. Stores operands from the top of the Stack for a Store to Memory instruction.

Because of the simultaneous operation of ADVAST and FINST, it is possible that a conflict can occur within the Central Processor with regard to memory service. As a result, requests to COMM for memory operations are serviced in priority order.

COMM also generates parity on stores to memory, and checks parity on memory fetches.

JUMPS

Because of segmentation of programs in memory, the execution of jumps (instruction transfers of control) are of three different types:

- Direct jump to another instruction within the present program segment without changing the contents of any base register (associated data, program, or reference areas).
- Indirect jump requiring a new instruction location (PCR) in a new program area (BPR), and a new data area (BXR), and may require the development of a return control word in order to effect a return jump.
- A jump that executes all the steps of an indirect jump, plus setting up a new Program Reference Table (PRT).

When a jump is executed, most of the above operations are performed by machine functions. More details about actual register contents and procedures are explained in Section 4 under the Jump Instructions (page 4-14).

REGISTERS

Table 1-1 contains a list of the Central Processor registers, and a brief description of their functions. These registers are contained in the Central Processor and can be read, set, or reset under program control.

Table 1-1. Central Processor Registers

Mnemonic	Name	Function
AAR	ADVAST Address Register	An address-modified register that can be set with a value, usually by an index instruction, and used to modify relative operand addresses of instructions following the index. AAR may also modify some variant field values. If an instruction resets the AAR, then ICR bit 65 is set.
AMAR	ADVAST Memory Address Register	Contains the absolute 18-bit address of the last reference to memory initiated by ADVAST.
ABL	Alternate Bounds Lower Register	Overrides lower normal bounds if set with an address. Bit 37 in ICR is set if alternate bounds violation.*
ABU	Alternate Bounds Upper Register	Overrides upper normal bounds if set with an address. Bit 37 in ICR is set if alternate bounds violation. *

*Alternate Bounds Registers, when set, will remain set until AAR is reset.

Table 1-1. Central Processor Registers (continued)

Mnemonic	Name	Function
BSR	Barrel Shift Register	Contains the shift amount for a shift instruction. This register is not reset or stepped down as shifting takes place. The BSR is set by the BSR instruction.
BDR	Base Data Register	Contains the address of the first location of the data area for the active program.
BIAR1	Base Interrupt Address Register 1	Contains the starting address of the routine executed as a result of a Normal Mode Interrupt (see page 3-4).
BIAR2	Base Interrupt Address Register 2	Contains the starting address of the routine executed as a result of a Control Mode 1 Interrupt (see page 3-4).
BPR	Base Program Register	Contains the address of the first word of instructions in the active program segment.
BXR	Base Index Register	Contains the address of the first word of the data area for the active program segment. Note: The first location of the data area is reserved for the Return Control Word (RCW) to permit return to a previous program segment.
CNR	Computer Number Register	Contains respective processor number (binary).
ITC	Incremental Time Counter	Stepped at 50-nanosecond intervals, reaching a maximum count in 3.759 minutes, at which time ITC overflow, bit 40, is set in the ICR.
ICR	Interrupt Control Register	A 70-bit register that contains the status of Interrupt conditions.
IMR	Interrupt Mask Register	Controls the conditions that cause Interrupts.
JCR	Jump Control Register	Contents of the JCR control conditional program jumps. Control is dependent on whether the jump is direct or indirect (see page 4-14).
M	M Register	In Stack, the next data word position below N.
N	N Register	In Stack, the next data word position below S.

Table 1-1. Central Processor Registers (continued)

Mnemonic	Name	Function
NBL	Normal Bounds Lower Register	Contains lower limit to which memory can be addressed. If the limit is exceeded, bit 36 in ICR is set.
NBU	Normal Bounds Upper Register	Contains upper limit to which memory can be addressed. If limit is exceeded, bit 36 in ICR is set.
P	P Register	Used by RTS instruction, in arithmetic, and in shifting. This register should not be used for temporary storage of operands. The contents of the P Register must be saved if it contains meaningful results of the previously performed instruction.
PFR	Processor Fail Register	In the event of an Interrupt associated with a memory reference, the PFR indicates the particular unit within the processor that was involved in the error.
PCR	Program Counter Register	Contains the address of the next instruction to be executed by ADVAST.
PRT	Program Reference Table Register	Contains the address of the first location of the Program Reference Table for the active program unit. Note: This location is reserved for the PRT and PRTL value for the previous PRT.
PRTL	Program Reference Table Limit Register	Contains the address of the upper limit of the active PRT.
S	S Register	In the Stack, the next data word position below T.
SBL SBU	Stack Bounds Lower Stack Bounds Upper Registers	Limit registers that control the area of memory that can be used as the Stack Extension.
SEP	Stack Extension Address Register	Contains the address of the next four-word group in memory to be transferred to the Stack when it requires more operands. SEP+4 is the address of the area in memory which will contain the next four words pushed down in the Stack Extension if the Stack needs more space.
T	T Register	Top Stack Position. Results of most instructions are placed in T. All memory, register, fetch, and store instructions involve the T Register.

SECTION 2

CHARACTER AND WORD FORMATS

INTRODUCTION

This section contains a description of the bit configurations for the B8500 character set, data word formats, and control word formats.

B8500 CHARACTER SET

Table 2-1 displays the bit configurations and card codes for the B8500 character set.

Table 2-1. B8500 Character Set

B8500			PUNCHED CARD COMBINATION
INTERNAL BA 8421	OCTAL	PRINTER GRAPHICS	
00 0000	00	blank	None
00 0001	01	A	12, 1
00 0010	02	B	12, 2
00 0011	03	C	12, 3
00 0100	04	D	12, 4
00 0101	05	E	12, 5
00 0110	06	F	12, 6
00 0111	07	G	12, 7
00 1000	10	H	12, 8
00 1001	11	I	12, 9
00 1010	12	&	12
00 1011	13	. (period)	12, 8, 3
00 1100	14	<	12, 8, 4
00 1101	15	(	12, 8, 5
00 1110	16	+	12, 8, 6
00 1111	17	← (tape mark) or | (vert. line) ✓	12, 8, 7
01 0000	20	x (multiply) or ° (degree) ✓	11, 0
01 0001	21	J	11, 1
01 0010	22	K	11, 2
01 0011	23	L	11, 3
01 0100	24	M	11, 4
01 0101	25	N	11, 5
01 0110	26	O	11, 6
01 0111	27	P	11, 7
01 1000	30	Q	11, 8
01 1001	31	R	11, 9
01 1010	32	- (minus)	11
01 1011	33	$	11, 8, 3
01 1100	34	*	11, 8, 4
01 1101	35	)	11, 8, 5
01 1110	36	; (semi-colon)	11, 8, 6
01 1111	37	≤	11, 8, 7
10 0000	40	≥	12, 0
10 0001	41	/	0, 1
10 0010	42	S	0, 2
10 0011	43	T	0, 3
10 0100	44	U	0, 4
10 0101	45	V	0, 5
10 0110	46	W	0, 6
10 0111	47	X	0, 7
10 1000	50	Y	0, 8
10 1001	51	Z	0, 9
10 1010	52	≠	12, 11
10 1011	53	, (comma)	0, 8, 3
10 1100	54	%	0, 8, 4
10 1101	55	]	0, 8, 5
10 1110	56	>	0, 8, 6
10 1111	57	? (See Note 1.) or ' (apostrophe) ✓	0, 8, 7
11 0000	60	0	0
11 0001	61	1	1
11 0010	62	2	2
11 0011	63	3	3
11 0100	64	4	4
11 0101	65	5	5
11 0110	66	6	6
11 0111	67	7	7
11 1000	70	8	8
11 1001	71	9	9
11 1010	72	: (colon)	8, 2
11 1011	73	#	8, 3
11 1100	74	@	8, 4
11 1101	75	[	8, 5
11 1110	76	=	8, 6
11 1111	77	" (quotes)	8, 7

LOW ↑ COLLATING SEQUENCE ↓ HIGH

Tape Mark - Used on B8500 as EOF record.

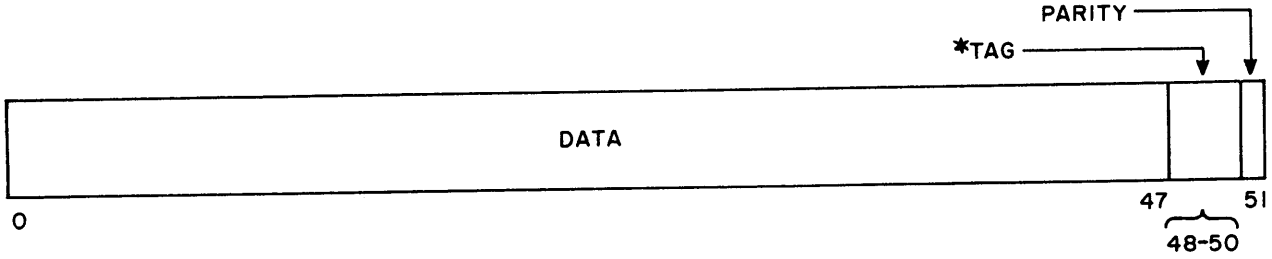
✓These symbols are available on special order (CER 1420).

NOTE 1. All other punched card combinations are "mapped" into this character.

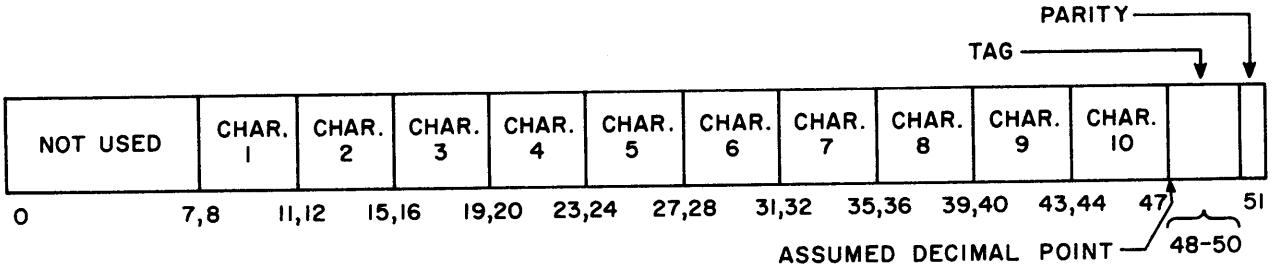
WORD FORMATS

DATA WORDS

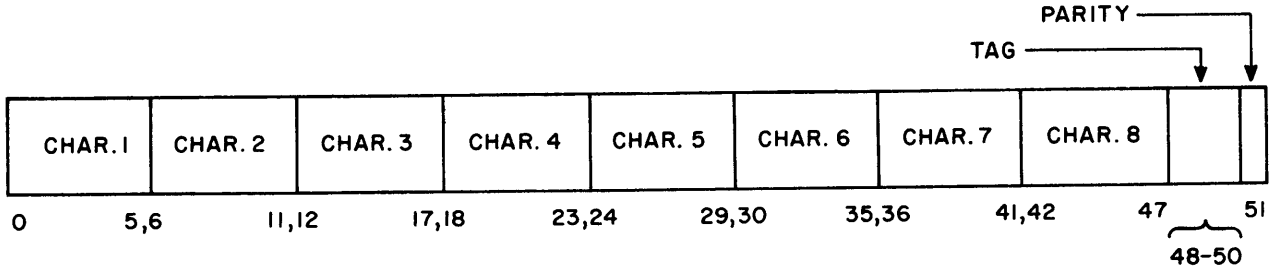
The following format is typical of all data words in memory. Different field configurations exist within the data as noted in the other data word formats.



Four-Bit BCD Word (Unsigned Numeric Characters)



Six-Bit BCD Word (Unsigned Alphameric Characters)

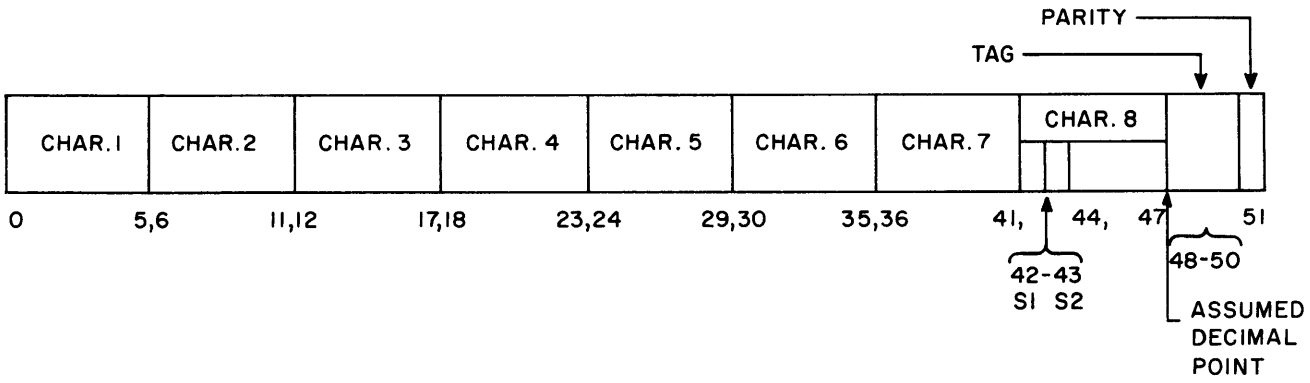


---

*TAG BITS

48, 49	See page 4-12, 4-13.
50	Interrupt tag bit. If set on examination of word fetched from memory, then an interrupt may occur depending on the processor mode and bit 59 in ICR/IMR. See page 3-1.

Six-Bit Word (Signed Numeric Characters)



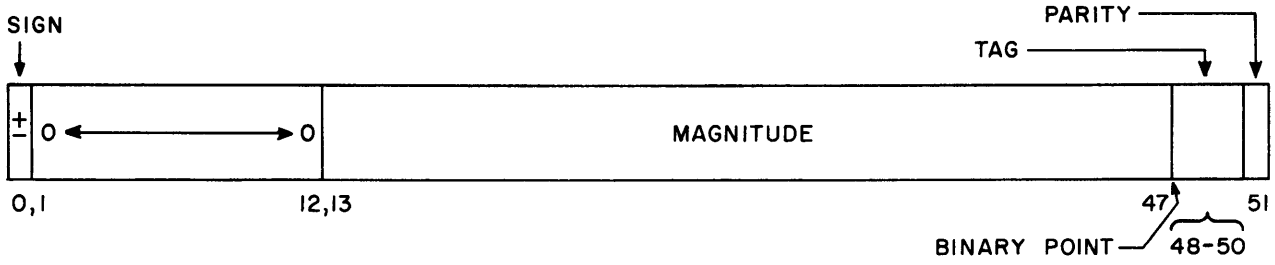
NOTE

The two most significant bits of characters 1 through 7 are set to 1's on an output word.

S1 S2 = The two most significant bits of character 8.

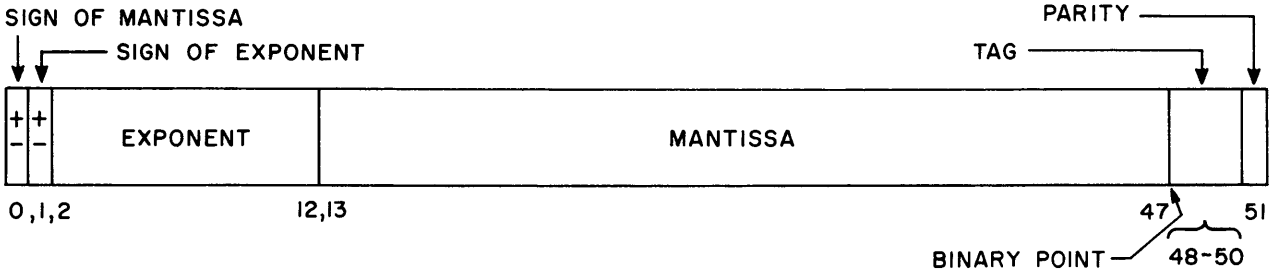
0 0 = Positive sign on an INPUT word.
0 1 = Negative sign on an INPUT/OUTPUT word.
1 0 = Positive sign on an INPUT word.
1 1 = Positive sign on an INPUT/OUTPUT word.

Integer Binary Word



Floating Point Words

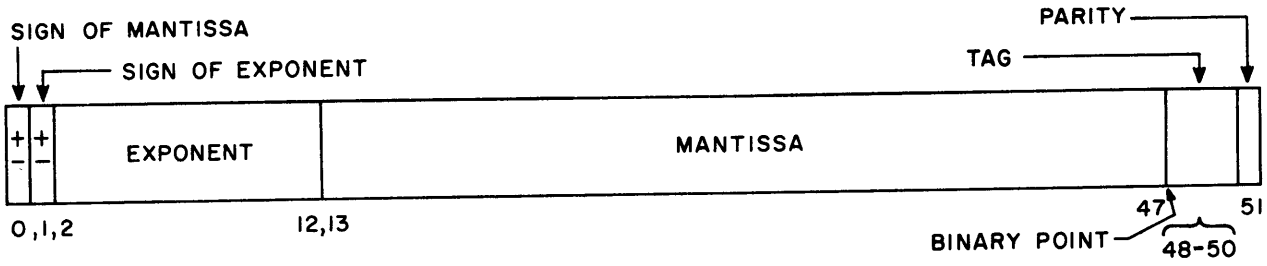
SINGLE PRECISION



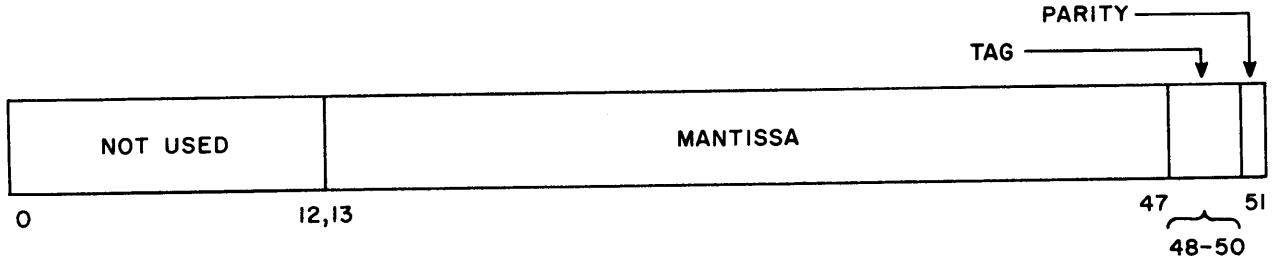
SIGNS: { 1 = Negative
0 = Positive

DOUBLE PRECISION

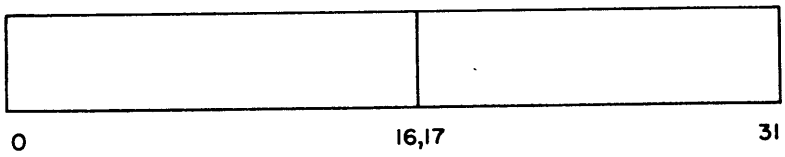
Word n



Word n+1



INCREMENTAL TIME CLOCK WORD

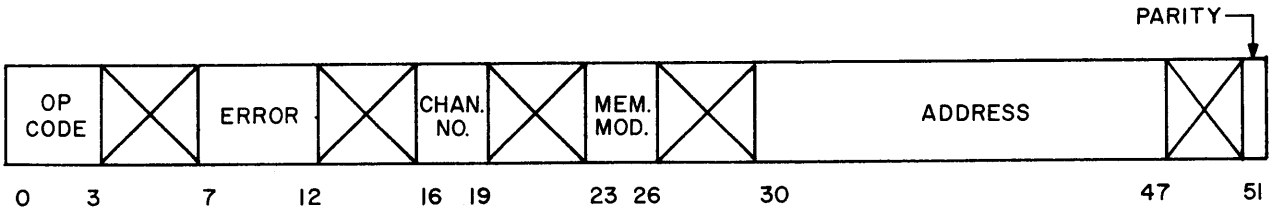


The ITC Register is stepped at 50-nanosecond intervals. The clock will reach a maximum count in 3.759 minutes, setting ITC overflow (bit 40) in the ICR. Bits 0 through 16 may be preset using the SSR instruction (loaded from positions 16 through 33 of the T Register). Bits 0 through 31 may be fetched by the FRS instruction. The clock word will be placed in the T Register right justified. After a fetch, bits 0 through 31 will be reset* if the processor is in the Control Mode.

---

* Reset in this text means "set to zero".

MEMORY FAIL WORD

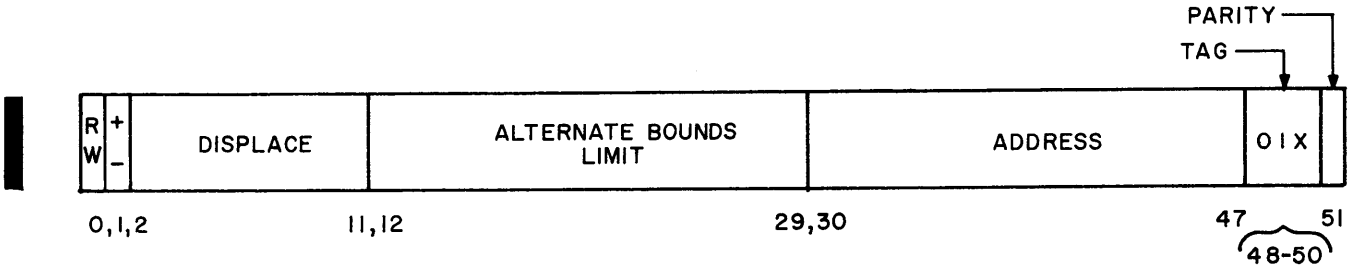


Op Code - A memory operation

Error - Bit 7 = Parity Error of the Control Word (to memory).
Bit 8 = Parity Error on Input Data (to memory).
Bit 9 = Wrong Memory Address (to memory).
Bit 10 = Parity Error on Output Data (from memory).
Bit 11 = Illegal Memory Op Code (to memory).
Bit 12 = Spare.

CONTROL WORDS

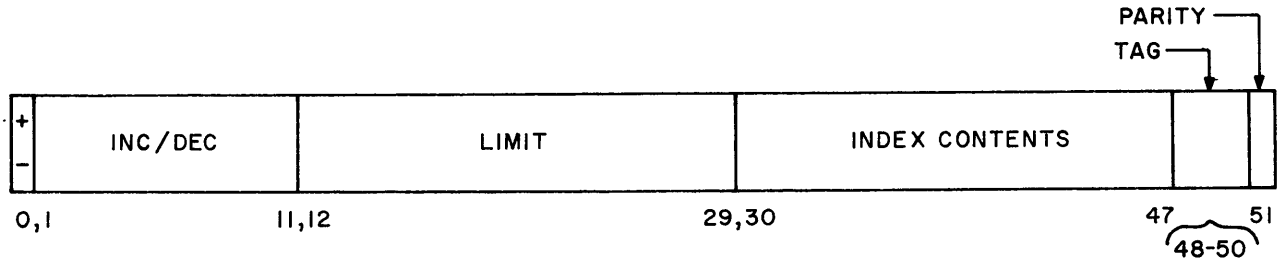
Alternate Bounds Descriptor Word



The descriptor is fetched from memory by the FMA or FMC instruction and placed in an Alternate Bounds Register. The area is defined by the address and the alternate bounds limit. It is a read-only area if bit 0 is reset.

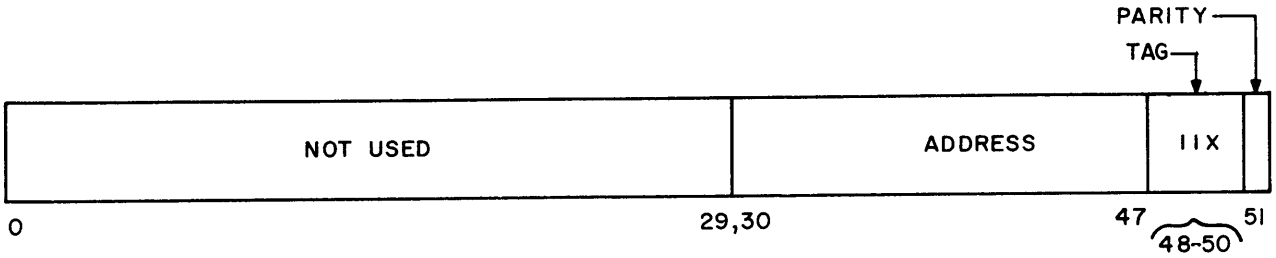
The displace field modifies the address loaded into the AAR by an FMA instruction (see page 4-12).

Index Word



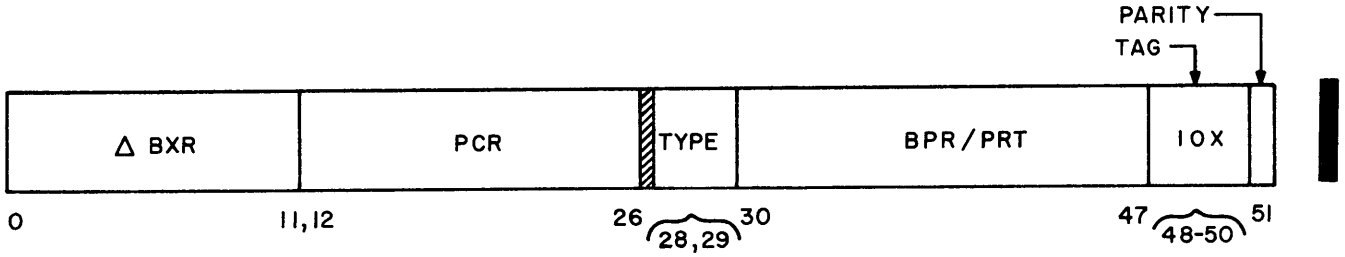
The word is fetched from memory to perform address modification. Index contents may be incremented or decremented, and may also be compared against a limit value or the top of the Stack, depending on the type of index instruction.

Indirect Address Word

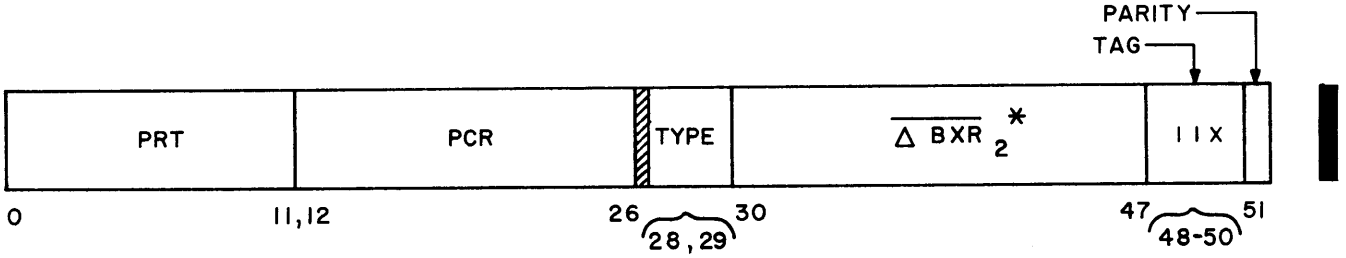


The word is fetched from memory by an FMA or FMC instruction. (See FMA and FMC instructions.) If the tag equals 11, another memory access is performed and the tag of the word fetched is examined again. This sequence is repeated until tag ≠ 11.

Jump Control Word (JCW)



Return Control Word (RCW)



INSTRUCTION WORDS

See Section 4 for the format of instruction words.

*$\overline{\Delta BXR}_2$ means "the two's complement of ΔBXR".

SECTION 3

INTERRUPTS

INTERRUPT CONDITIONS

Table 3-1 is a list of the Interrupt conditions treated in the B8500 System. The first column indicates the bit position of the Interrupt Condition Register (ICR) and the Interrupt Mask Register (IMR) for a particular interrupt. The second column indicates the bit position in the T or S Registers of the (ICR) or (IMR) when transferred to FINST as a result of an FRS instruction (Figure 3-1).

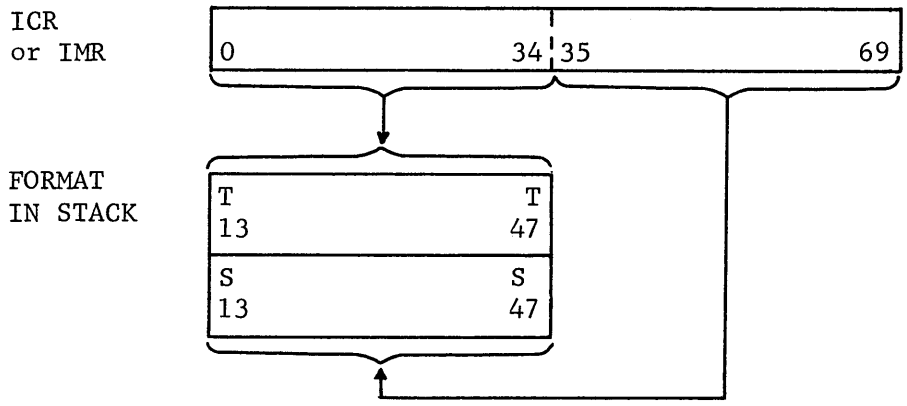


Figure 3-1. Format of ICR or IMR when Transferred to FINST

This is also the format necessary before an SSR is executed to load the Interrupt Mask Register. Using SSR to load the IMR and ICR is restricted by the fact that it is only possible if the processor is in a Control Mode. Sampling the ICR is possible and legal, if properly masked, in both Control and Normal Modes. Sampling the ICR in Control Mode results in the register being reset, but an FRS of the ICR in Normal Mode does not.

Table 3-1. Interrupt Conditions, B8500 System

Bit Position in ICR/IMR*	Bit Position in the Stack	Cause of Interrupt	
0	T13	Memory 1 Error	For specific type of error, see Memory Fail Register (Page 2-6)
1	T14	Memory 2 Error	
2	T15	Memory 3 Error	
3	T16	Memory 4 Error	
4	T17	Memory 5 Error	
5	T18	Memory 6 Error	
6	T19	Memory 7 Error	
7	T20	Memory 8 Error	
8	T21	Memory 9 Error	
9	T22	Memory 10 Error	
10	T23	Memory 11 Error	
11	T24	Memory 12 Error	
12	T25	Memory 13 Error	
13	T26	Memory 14 Error	
14	T27	Memory 15 Error	
15	T28	Memory 16 Error	
16	T29	I/O 1 Complete	External interrupts from I/O and C.P. Modules
17	T30	I/O 1 Error	
18	T31	Interrupt by Processor 1	
19	T32	I/O 2 Complete	
20	T33	I/O 2 Error	
21	T34	Interrupt by Processor 2	
22	T35		External Spares for Future System Expansion
23	T36		
24	T37		
25	T38		
26	T39		
27	T40		
28	T41		
29	T42		
30	T43		
31	T44		
32	T45		
33	T46		
34	T47	- Spare -	
35	S13	PRT	Bounds Violations
36	S14	Normal	
37	S15	Alternate	
38	S16	Stack	
39	S17	- Spare -	
40	S18	Incremental timer overflow	
41	S19	- Spare -	
42	S20	Exponent underflow	Arithmetic Errors
43	S21	Exponent overflow	

*Interrupt Condition Register/Interrupt Mask Register

Table 3-1. Interrupt Conditions, B8500 System (continued)

Bit Position in ICR/IMR		Bit Position in the Stack	Cause of Interrupt	
44		S22	Mantissa underflow	Arithmetic Errors
45		S23	Mantissa overflow	
46		S24	Non-Normalized Operand	
47		S25	Divide by Zero	
48		S26	- Spare -	
49		S27	- Spare -	
50		S28	- Spare -	
51		S29	SSM/SSMA to "READ ONLY" Alternate Bounds Area	
52		S30	SSM PRT Relative	Illegal Instruction and Illegal Variant
53		S31	FRS/SSR to Class A Registers (Table 3-2)	
54		S32	FRS to Class B Registers (Table 3-2)	
55		S33	SSR to Class B Registers (Table 3-2)	
56		S34	FRS to Class C Registers (Table 3-2)	
57		S35	- Spare -	
58		S36	Illegal Tags	
59		S37	Interrupt Tag Bit Set	
60		S38	No Access to Memory	
61		S39	Parity Error from Memory	
62		S40	STOP Instruction	
63		S41	Nonexistant op code or variant	
64		S42	Illegal Instruction - ICN, FMT, IOP or ITB	
65		S43	AAR was reset	
*66	CM1	S44	SSR Class C is illegal in Normal Mode - Mask is CM1 or CM2	
*67	CM2	S45	IRR Illegal in Normal Mode - Mask is CM1 or CM2	
**68	- STORQ MASK	S46	Force tags to zero on data from T to STORQ	
**69	- STACK MASK	S47	Force tags to zero on data to and from bottom of Stack	

* There are actually no mask bits for Interrupt conditions 66 and 67. The Control Mode Flags act as masks for these interrupts.

** There are no actual interrupt conditions associated with mask bits 68 and 69, STORQ mask and STACK mask. These bits are loaded with the remainder of IMR, as are the Control Mode bits, CM1 and CM2.

Table 3-2. Illegal Variants

The following configurations of the register variant (R) [see page 4-3] are illegal if not properly masked. The specific mask bits are as follows: Class A registers, IMR 53 for FRS or SSR; Class B registers, IMR 54 for FRS and IMR 55 for SSR; Class C registers, IMR 56 for FRS and C.M. 1 or C.M. 2 for SSR.

Class A*		Class B*		Class C***	
Register	Code	Register	Code	Register	Code
BSR	40	SEP	20	BIAR1	61
PCR	41	BXR	21	BIAR2	62
JCR	42	BPR	22	PFR	63
		BDR	23	AMAR	64
		NBLR	24	ITC	65
		NBUR	25	IMR	76
		ABLR	26	ICR	77
		ABUR	27		
		PCS	30*		
		SLBR	31		
		SUBR	32		
		PRT	33		
		PRTL	34		
		CNR	37*		
		* Nonexistent Variant for SSR.			

* Class A Registers; General Program use.
** Class B Registers; Utilized by Executive and Scheduling Program
***Class C Registers; Utilized in Interrupt Processing.

NORMAL MODE INTERRUPTS

In Normal Mode, a condition that sets a bit in the ICR, when a corresponding bit is set in the IMR, results in an interrupt jump procedure. If a mask bit is not set, then no interrupt will result. Exceptions: 1) If ICR bit 62 is set, a STOP in Normal Mode, with no corresponding mask bit set, results in an unconditional halt. If the mask bit is set, then an interrupt jump is executed. 2) When ICR bits 66 and 67 are set, regardless of mask, an interrupt jump results.

CONTROL MODE 1 INTERRUPTS

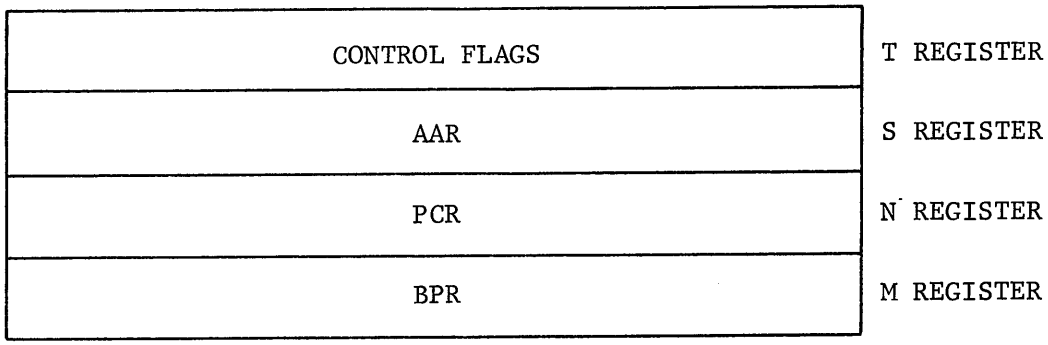
The Mask Register is not used during Control Modes 1 or 2. Most bits set in ICR while the processor is in Control Mode 1 are ignored. The exceptions are that setting bit 60, 61, 62 or 63 will result in an unconditinal interrupt jump.

CONTROL MODE 2 INTERRUPTS

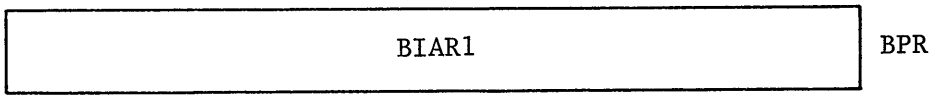
Interrupt bits set during Control Mode 2 are generally ignored, except that setting bit 60, 61, 62 or 63 result in an unconditional halt. Bits 58 and 59 will cause a halt if they are set by instructions FMA, FMC (first fetch only), SRJ, SRR, and STOP. However, if bits 58 and 59 are set by instructions FMS, FMSA, FMT, X, XM, FMC (after first fetch), and JXMT, then no action is taken.

INTERRUPT JUMP PROCEDURE

During the execution of an interrupt jump, the Stack is stepped down once, and the contents of BPR are placed in the T Register. Then the Stack is pushed down again and the contents of PCR are placed in T. PCR is reset, the Stack is further pushed down, and the contents of AAR are placed in T. The Stack is pushed down one more time, and control flags are placed in T. (see page 3-7). If the processor is in Normal Mode, the contents of the BIAR1 Register are placed in BPR, and the processor is set to Control Mode 1. If the processor is in Control Mode 1 when an interrupt occurs, then the above registers are saved, the contents of BIAR2 are placed in BPR, and the processor is set to Control Mode 2. The new contents of BPR and PCR define the address of the next instruction to be executed (Figure 3-2).

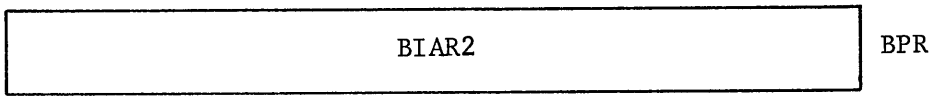


In Normal Mode, upon interrupt:



then enter CM1.

In CM1, upon interrupt:



then enter CM2.

Figure 3-2. Contents of Stack and BPR upon Completion of an Interrupt Jump Instruction

INTERRUPT RETURN

Performed by the Interrupt Routine Return instruction (IRR) page 4-64.

TAG CONTROL

Bit positions 68 and 69 in the Interrupt Mask Register are used only for tag control. Setting bit 68 will reset the tags and interrupt bit to 0 upon the transfer of data from T to memory. Setting bit 69 will reset the tags and interrupt bit to 0 upon the transfer of words from the bottom of the Stack to memory.

PROCESSOR FAIL REGISTER

Figure 3-3 illustrates the format of the Processor Fail Register (PFR). The PFR is loaded only in the event of an interrupt associated with a memory reference, which causes the initiation of IRJ.

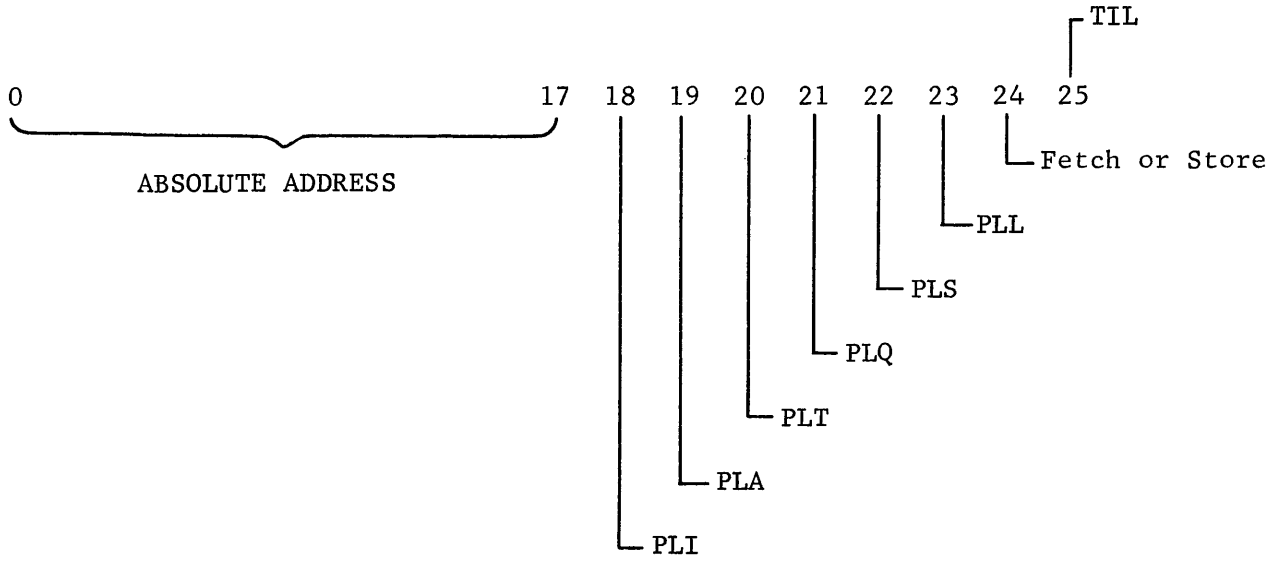


Figure 3-3. Format of Processor Fail Register

Bits 0 through 17 contain the Absolute Address of the word involved in the interrupt.

Bit 24 if set, indicates that the reference was a Store; reset indicates a Fetch.

Bits 18 through 23, respectively, indicate what particular unit within the processor was involved in the error, as follows:

PLI - Index/PRT Q - Single word Store only

PLA - ADVAST - Single word Fetch or Store

PLT - Temp Q - Single word Fetch only

PLQ - Storage Queue - Single word Store only

PLS - Stack - four word Fetch or four word Store

PLL - Instruction Look Ahead - four word Fetch only

Bit 25 indicates detection of an illegal tag or an interrupt tag on an ILA Fetch.

PROCESSOR CONTROL FLAGS

Figure 3-4 illustrates the format of the control flags as they are placed in the Stack during IRJ.

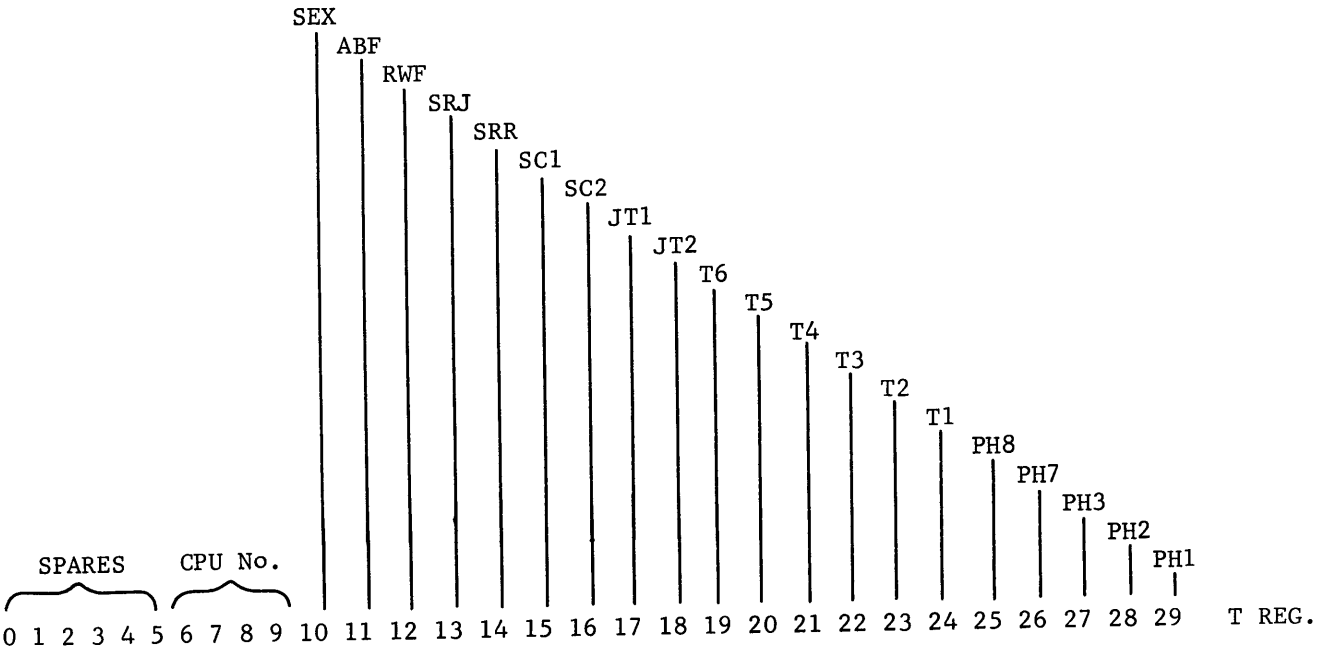


Figure 3-4. Format of Interrupt Control Flags in Stack

Bit 10 indicates whether the Stack was in extension mode at the time of the interrupt.

Bit 11 indicates if Alternate Bounds were in effect at the time of interrupt (ABF).

Bit 12 (RWF) indicates whether the Alternate Bounds object was a read or write area.

Bits 13 and 14 indicate if either a subroutine jump or subroutine return sequence was in process at the time of interrupt.

Bits 15 and 16 indicate the number of syllables involved in the last instruction executed before processing IRJ. When the Central Processor is returned to Normal Mode through IRR, and if it is desired to repeat the last instruction, this amount of syllables must be subtracted from the syllable count stored away on IRJ.

JT1	JT2	Type
0	1	Segment Nonreturn
1	1	Segment Return
1	0	Intra-Segment Return
0	0	Procedure (New PRT)

The least-significant 11 bits (bit positions 19-29) are useful only when interrupted during the execution of SRJ or SRR. They aid in defining the particular type and cause of interrupt.

Only ABF, RWF, and SRJ are restored in an IRR instruction.

SECTION 4

B8501 CENTRAL PROCESSOR INSTRUCTION SET

INTRODUCTION

This section describes the B8501 Central Processor Instruction Set. The instructions are presented within their logical groups by the expected frequency of use. The groups are arranged in the following order:

- Fetch and Store Instructions
- Jump Instructions
- Logical Instructions
- Field Manipulating Instructions
- Shift Instructions
- Indexing Instructions
- Arithmetic Instructions
- I/O and Control Instructions

NOTE

Appendix A is a working index of instructions, which can be used as a quick reference.

INSTRUCTION FORMAT

Instructions are divided into syllables of six bits each (fee Figure 4-1). All instructions contain at least one syllable, which is the op code shown in octal. Instructions that contain only an op code and an address field use three syllables and can designate a relative address from 0 to 4095 decimal locations. A variant field may also be included which may occupy one, two, or three syllables.

NOTE

Program code appears in memory as a continuous string of instructions, i.e., with 2 to 8 instructions per memory word, depending on the number of syllables per instructions.

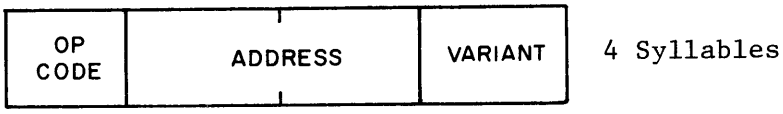


Figure 4-1. Example of an Instruction Format

In all instructions except SLIT or X, if AAR is used and not reset, then it will contain the contents of A1A2 syllable, plus the previous AAR value, plus a base if specified.

If required, the variant field (hereafter called "Variant") can be one, two, or three syllables. The variant field begins at either bit position 6 or 18, relative to the start of the instruction depending on whether or not an address field is required. The variant symbols are defined in Table 4-1.

The operation of each instruction is described, followed by a description of the instruction's variant field control.

If applicable, the final contents of the T Register and Stack are indicated. Also given are the Interrupts that may be generated as a direct result of the instruction.

TIMING CRITERIA

Timing information is estimated and can vary greatly because of memory access times, Interrupts, and COMM unit availability.

There are three execution times specified for each instruction: one for the Advanced Station (ADVAST), one for the Communication Station (COMM), and one for the Final Station (FINST). The execution time for a sequence of instructions can be approximated by separately summing the times for each station and then choosing the largest of these three sums.

The time required for COMM to fetch instruction words is not charged as time involved in each instruction since COMM is capable of fetching instruction words in advance of their use in ADVAST. These instruction words are fetched to the Instruction Look-Ahead queue during periods that the other stations in the processor are not using COMM. "FINQ to empty time" is the sum of FINST times for each instruction waiting in FINST plus 0.1 microsecond.

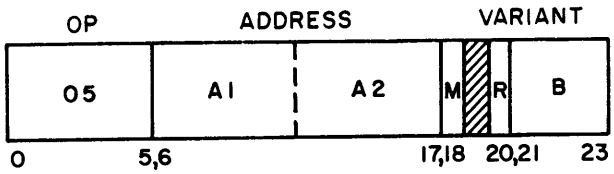
Table 4-1. Definition of Variant Symbols

Symbol	Definition
A	Specifies the shift amount added to the AAR which is to be placed in the shift amount register by the Barrel Shift Register instruction, BSR.
B	Specifies the base register to be used in address arithmetic involved in the following instructions: FMA, FMC, FMS, FMT, and SSM. With an SHF instruction, specifies the type of shift to be performed.
C	Specifies the type of conversion to be done by CBB instruction, the type of jump with SJ instruction and type of conditional test with JSTL, JSTA, JXMT instructions.
D	Specifies which one of four double-length arithmetic operations are to be done by ARIT instruction and direct or indirect jump specifier in SJ instruction.
F	Tag control specifier in FMT instruction.
I	The I syllable is used in the ESP instruction as a relative address which is added to the BIAR1.
L	Specifies the length of the field for all field instructions except INSD and EXTD.
L1, L2	Field length specifier in EXTD and INSD. Literal value for FAS and SLIT instructions.
M	Specifies type of modification in the JXMT instruction.
N	Specifies the Processor in the ICN and IOP instructions.
R	Specifies the register to be addressed by either the FRS instruction or the SSR instruction, and type of compare with JSTA, JSTL, JFT instructions and controls reset of AAR with SSM, FMS and FMC instructions.
S	Specifies the starting position of fields for all field instructions, the various operations of the RTS instruction, and syllable location in the SJ instruction.
T	Specifies the I/O Module to be addressed by the IOP instruction, and the type of tests to be performed by the JFT, JSTA, or JSTL instructions. Also specifies Stack manipulations at the conclusion of the instruction.

FETCH AND STORE INSTRUCTIONS

The Fetch and Store instructions transfer data between the Central Processor registers and the T Register, or between Main Memory and the T Register. The one exception is the FMA instruction, which can load the bounds registers or the AAR directly from memory.

FMS
Fetch Memory to Stack



4 Syllables

DESCRIPTION

The contents of memory, as addressed by the second and third syllables, plus the AAR plus a base register (if specified), are placed in the T Register. If a base PRT is used, then word tag bits 48 and 49 will be reset when the operand is placed in T.

VARIANT

M - 1 = Fetch Memory Fail Register (most significant four bits of address developed by AAR and base will specify Memory Module).

R - 1 = Inhibit AAR reset

0 = Reset AAR

B - 000 = No base

001 = BPR - No bounds check

010 = BXR
011 = BDR
} Alternate or normal bounds check

100 = PRT - PRT bounds check

STACK

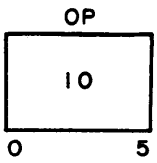
Stepped down once before loading T.

INTERRUPTS

Bit 36 in ICR set if normal bound violation.
Bit 37 in ICR set if alternate bounds violation.
Bit 35 in ICR set if PRT bounds violation.
Bit 59 in ICR set if word fetched has interrupt tag set.
Bit 63 in ICR set if illegal base variant specified.
Bit 65 in ICR set if AAR is reset.

TIMING

ADVAST 0.3 usec. COMM 0.6 usec. FINST 0.1 usec.



1 Syllable

FMSA
Fetch Memory to Stack Absolute

DESCRIPTION

The contents of memory, as addressed by AAR, will be placed in the top of the Stack. The AAR final contents are reset.

VARIANT

None.

STACK

Stepped down once before loading T.

INTERRUPTS

Bit 59 in ICR set if word fetched has interrupt tag bit set.

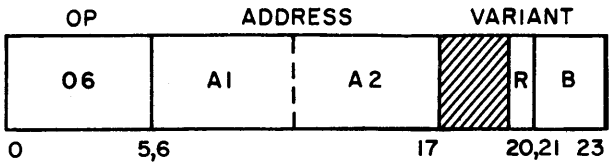
Bit 36 in ICR set if normal bounds violation.

Bit 37 in ICR set if alternate bounds violation.
Bit 65 in ICR set unconditionally.

TIMING

ADVAST 0.2 usec. COMM 0.6 usec. FINST 0.1 usec.

SSM
Store Stack
to Memory



4 Syllables

DESCRIPTION

The contents of the T Register are stored into memory, as addressed by the second and third syllables, plus the AAR, plus a base register (if specified). Tag bits are reset on the word stored if IMR bit 68 is set.

VARIANT

R - 0 = Reset AAR
1 = Inhibit AAR Reset

B - 000 = No base
001 = BPR
010 = BXR
011 = BDR
} Normal or alternate bounds check
100 = PRT-PRT bounds check (note interrupts below).

STACK

Stepped up once after storing T.

INTERRUPTS

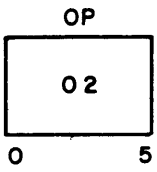
Bit 36 in ICR set if normal bounds violation.
Bit 37 in ICR set if alternate bounds violation.
Bit 35 in ICR set if PRT bounds violation.
Bit 65 in ICR set if AAR reset.
Bit 51 in ICR set if store to alternate bounds (read only area).
Bit 52 in ICR set if base PRT specified.

TIMING

ADVAST 0.3 usec. COMM 0.4 usec. FINST 0.1 usec.

Access to memory required.

SSMA
Store Stack to Memory Absolute



1 Syllable

DESCRIPTION

The contents of the T Register are stored in memory as addressed by the AAR. Adder Register. The tag bits are reset on the word stored if IMR bit 68 is set. The AAR final contents are reset.

VARIANT

None.

STACK

Stepped up once after storing T.

INTERRUPTS

Bit 36 in ICR set if normal bounds violation.

Bit 37 in ICR set if alternate bounds violation.

Bit 51 in ICR set if stored to alternate bounds (read only area).

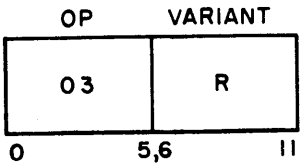
Bit 65 in ICR is set unconditionally.

TIMING

ADVAST 0.2 usec. COMM. 0.4 usec. FINST 0.1 usec.

Access to memory required.

FRS
Fetch Register to Stack



2 Syllables

DESCRIPTION

The contents of the register specified by R and the AAR are placed in the T Register, and the S Register also if ICR or IMR is specified. The final AAR contents are reset.

VARIANT

R -

ABL	= 26	BXR	= 21	PCR	= 41
ABU	= 27	CNR	= 37	*PCS	= 30
AMAR	= 64	ICR	= 77	PFR	= 63
BDR	= 23	IMR	= 76	PRT	= 33
BIAR1	= 61	ITC	= 65	PRTL	= 34
BIAR2	= 62	JCR	= 42	SBL	= 31
BPR	= 22	NBL	= 24	SBU	= 32
BSR	= 40	NBU	= 25	SEPR	= 20

STACK

Stepped down once before loading T (stepped down twice if ICR or IMR specified).

INTERRUPTS

Bit 53 in ICR set if fetch from Class A Register.

Bit 54 in ICR set if fetch from Class B Register.

Bit 56 in ICR set if fetch from Class C Register.

Bit 65 in ICR set unconditionally.

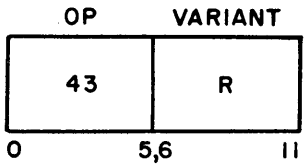
TIMING

ADVAST 0.2 usec. COMM 0 usec. FINST 0.1 usec.

---

*Purge Stack; contents of Stack are pushed down into Main Memory. Final SEP placed in T [30, 16].

SSR
Store Stack
to Register



2 Syllables

DESCRIPTION

The contents of the T Register are transferred to the register specified by R and AAR. Note: If ICR or IMR is specified, then transfer is from both the T and S Registers. The final AAR contents are reset.

VARIANT

R -	ABL	=	26	BXR	=	21	PCR	= 41
	ABU	=	27	ICR	=	77	PFR	= 63
	AMAR	=	64	IMR	=	76	PRT	= 33
	BDR	=	23	ITC	=	65	PRTL	= 34
	BIAR1	=	61	JCR	=	42	SBL	= 31
	BIAR2	=	62	NBL	=	24	SBU	= 32
	BPR	=	22	NBU	=	25	* SEPR	= 20
	BSR	=	40					

STACK

Stepped up once after transfer (twice if ICR or IMR specified).

INTERRUPTS

Bit 53 in ICR set if store to Class A Register.

Bit 55 in ICR set if store to Class B Register.

Bit 65 in ICR set unconditionally.

Bit 66 in ICR set if store to Class C Register.

TIMING

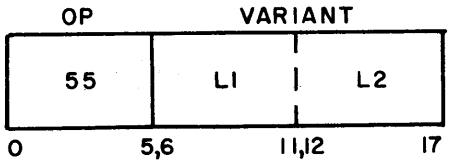
ADVAST 0.2 usec. COMM 0 usec. FINST 0.1 sec.

Plus time for FINQ to empty.

---

* Load Stack. Contents of T Register [30, 16] are placed in SEP. Then, operands are brought into the stack so that the resultant operand in T was at location addressed by original T [30,16].

FAS
Fetch Address Register to Stack



3 Syllables

DESCRIPTION

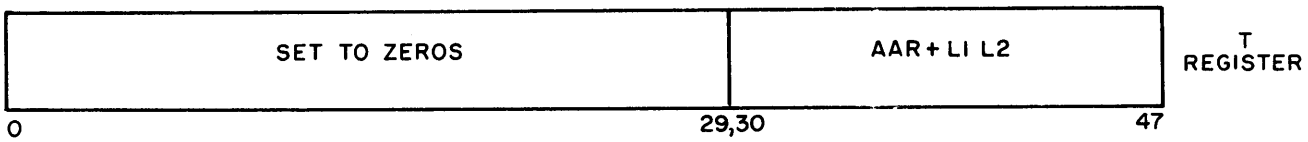
The sum of the contents of the AAR (if any) and L1 L2 are placed in the T Register, right justified. AAR final contents are reset.

VARIANT

L1 L2 - Actual value added to AAR.

STACK

Stepped down once before loading T. After loading T, bit positions 0-29 are zeroed.



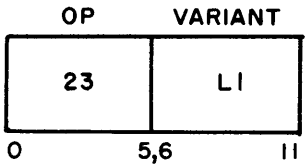
INTERRUPTS

Bit 65 in ICR set unconditionally.

TIMING

ADVAST 0.3 usec. COMM 0 usec. FINST 0.1 usec.

SLIT
Short Literal



2 Syllables

DESCRIPTION

The value of L1 is placed in the T Register, right justified.

VARIANT

L1 - Actual value of L1 is placed in T Register.

STACK

Stepped down once before loading T. Positions 0 thru 41 of T are reset.

INTERRUPTS

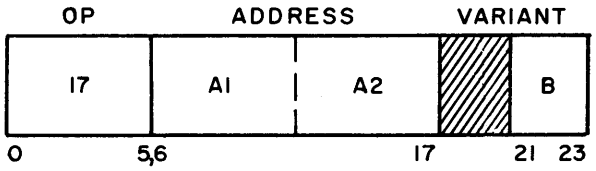
None.

TIMING

ADVAST 0.1 usec. COMM 0 usec. FINST 0.1 usec.

FMA

Fetch Memory to Address Register



4 Syllables

DESCRIPTION

The contents of memory, as addressed by the AAR A1 A2 and a base PRT, or location fetched in an alternate bounds area, are checked. One of four following sequences will result depending on the tag bit configuration of the word fetched (see Bounds/Base Table.) If word is not PRT relative, or not within an alternate bounds area, then a NOP tag condition will be effected.

B/B TABLE REF.	SEQUENCE IF PRT RELATIVE OR IN AB AREA

A - INDIRECT ADDRESS memory contents, as addressed by address field of word, are examined again for tag bit configuration.

B - TAG JUMP condition (see page 4-14). If AB, then error Interrupt.

C - ALTERNATE BOUNDS WORD. Lower and upper bounds field of word fetched is placed in respective Alternate Bounds Registers. Lower bounds address is also placed in AAR and modified by displace value, and the read/write bit is saved.

D - NOP. Address field of word fetched is placed in AAR.

E - Illegal.

VARIANT

B - 000 = No base
001 = BPR
010 = BXR
011 = BDR
100 = PRT

BOUNDS/BASE TABLE

Tag 48, 49	Normal	Alternate	PRT
00	D	D	D
01	D	C	C
10	D	E	B
11	D	A	A

STACK

Unchanged.

INTERRUPTS

Bit 35 in ICR set if PRT Bounds violation.

Bit 36 in ICR set if Normal Bounds violation.

Bit 37 in ICR set if Alternate Bounds violation.

Bit 58 in ICR set if tag of word fetched in Alternate Bounds area = 10.

Bit 59 in ICR set if interrupt tag bit set.

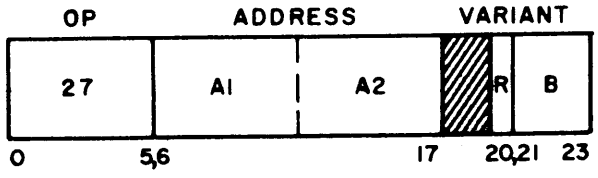
TIMING

ADVAST 0.3 usec. COMM 0 usec. FINST 0 usec.

Timing values based on FMA being PRT relative and all operands located in local associative memory.

Note: Bounds check on first PRT, AB or NB fetch only.

FMC
Fetch Memory Conditionally



4 Syllables

DESCRIPTION

The contents of memory, as addressed by the AAR, A1 A2, and a base (if specified), are checked. One of the four following sequences will result depending on the tag bit configuration of the word fetched per Bounds/Base Table below.

B/B TABLE REF. — SEQUENCE IF FIRST WORD FETCHED IS PRT RELATIVE OR IN AB AREA

- A - INDIRECT ADDRESS memory contents, as addressed by address field of word, are examined again for tag bit configuration.
- B - TAG JUMP condition (see page 4-14). If AB, then an error Interrupt.
- C - ALTERNATE BOUNDS WORD. Lower and upper bounds field of word fetched is placed in respective Alternate Bounds Registers. Lower bounds address is modified by displace value and placed in AAR.
- D - NOP. Address field of word fetched is placed in AAR.
- E - Illegal.

In addition, the contents of memory as addressed by the AAR are placed in the T Register.

VARIANT

B - 000 = No base
    001 = BPR
    010 = BXR
    011 = BDR
    100 = PRT
R - 1 = Inhibit reset AAR
    0 = Reset AAR

BOUNDS/BASE TABLE

Tags	Normal	Alternate	PRT
00	D	D	D
01	D	C	C
10	D	E	B
11	D	A	A

STACK

Stepped down once before loading T.

INTERRUPTS

Bit 35 in ICR set if PRT Bounds violation.

Bit 36 in ICR set if Normal Bounds violation.

Bit 37 in ICR set if Alternate Bounds violation.

Bit 58 in ICR set if tag or word fetched in Alternate Bounds area = 10.

Bit 65 in ICR set if AAR reset.

TIMING

ADVAST 1.35 usec. COMM 0.6 usec. FINST 0.1 usec.

Timing values based on FMA address in Main Memory

*Note: Bounds check made on first PRT, AB, or NB reference, and on address of the operand placed in T Register.

JUMP INSTRUCTIONS

Jumps can be initiated under three different conditions:

1. The result of a Set Up Jump instruction with variant that specifies unconditional jump.

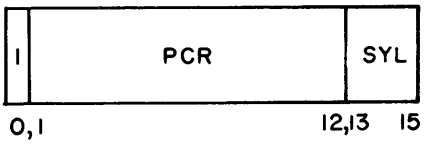
2. A conditional test instruction which, if true, jumps as specified by the previously set Jump Control Register (JCR).

3. Tag jump condition in which a word fetched PRT relative has a tag specifying "Jump" as examined during execution of an SJ, FMA, or FMC instruction.

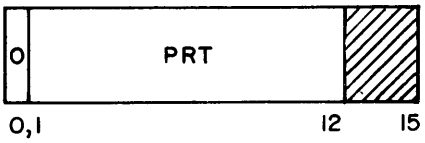
While unconditional jumps can be initiated and executed by a Set Up Jump instruction, conditional jumps are initialized by a Set Up Jump instruction that presets the JCR for a later conditional test instruction.

The format of the Jump Control Register are:

If Direct Jump



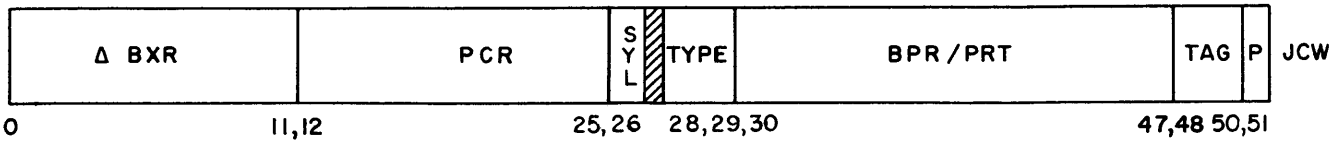
If Indirect Jump



JUMPS INITIATED UNDER CONTROL OF JCR

If a conditional jump is direct, then the PCR field of the JCR if placed in the PCR, and the next instruction, as specified by the new PCR, is executed. If a conditional jump is indirect, then the contents of memory, as addressed by the PRT field of JCR and PRT, is placed in the Jump Control Word (a special local register).

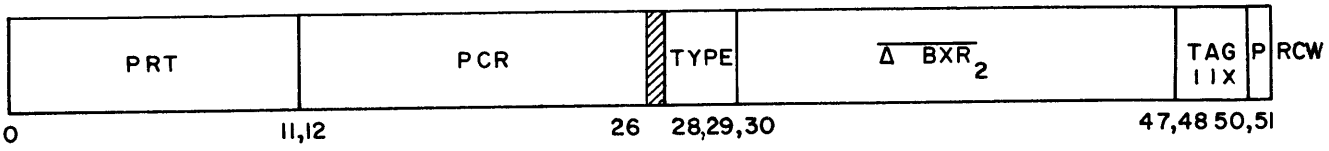
Bit 58 in the ICR is set if the JCW tag (bits 48 and 49) does not equal 10.



The following operations depend on the type of JCW.

If Type

- 01 = A segment jump without return control. Requires steps 3, 4, & 5.
- 11 = A segment jump with return control. Steps 1, 3, 4, and 5.
- 10 = An intra-segment jump. Steps 1, 3, and 5 on following pages.
- 00 = A procedure type jump. Steps 1, 2, 4, and 5 on following pages.



Note: Return Control Word is stored in memory in the first location of BXR area. (Address = BXR).

Step 1

1. New BXR value is developed by adding the present BXR value with the Δ BXR field of the word (JCW of previous segment jump) located in PRT at the location specified by the PRT Register, plus PRT field of the word at the location specified by the present BXR contents.

2. The present PCR contents are placed in the PCR field of the RCW.

3. The 2's complement of the ΔBXR field of the previous JCW is placed in the $\overline{\Delta BXR}_2$ field of the RCW.

4. The TYPE field of JCW is transferred to the TYPE field of RCW.

5. The RCW is then placed in memory at a location specified by new BXR value. ICR bit 36 is set if there is a Normal Bounds violation.

Step 2 (See Program Reference Table Below.)

1. The contents of the PRT Register and the PRTL Register are placed in register SAVE.

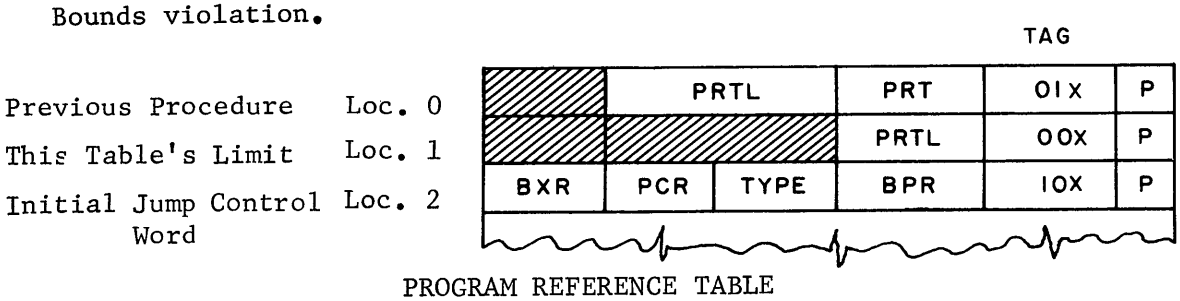
2. The JCW PRT field is placed in PRT Register.

3. The contents of the location specified by PRT Register + 1 (Loc 1) are placed in PRTL Register.

4. The contents of register SAVE are placed into location specified by the PRT Register. ICR bit 35 is set if there is a PRT Bounds violation.

5. The contents of the memory location specified by PRT +2 (Loc 2) are fetched. This is JCW for Steps 4 and 5.

6. The literal value "2" is placed into the PRT address field of the location (RCW) specified by the BXR Register. ICR bit 36 is set if there is a Normal Bounds violation.



PROGRAM REFERENCE TABLE

Step 3

*The PRT address field of JCR is placed in the PRT address field of the location in memory specified by the BXR Register. ICR bit 36 is set if there is a Normal Bounds violation.

Step 4

The BPR field of JCW is placed in BPR.

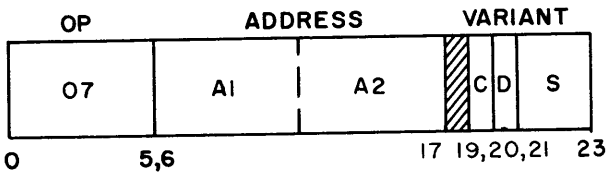
Step 5

The PCR field of JCW is placed in PCR.

---

*If jump was initiated under condition 3 on page 4-14 (tag jump), then the above step 3 is skipped and the word fetched with the jump tag is placed into JCW, and the PRT relative address at which it was located is placed in the RCW (address 0 relative to the current BXR).

SJ
Set up Jump



4 Syllables

DESCRIPTION

CONDITIONAL DIRECT OR INDIRECT JUMP. The contents of A1A2 and AAR are placed in the Jump Control Register address field. Contents of D and S are placed in the Jump Control Register D and S fields. The next instruction in sequence is executed.

UNCONDITIONAL DIRECT JUMP. The contents of A1A2 plus the AAR are placed in PCR word address field. S is placed in PCR syllable field. The next instruction, as specified by the new PCR, is executed.

UNCONDITIONAL INDIRECT JUMP. The contents of A1A2, the AAR plus the base PRT, are examined, and if tag field equals 10, then the tag jump procedure will follow (see page 4-14, item 3). If tag does not equal 10, then bit 58 in the ICR is set.

VARIANT

C - 1 = Conditional Jump.
0 = Unconditional Jump.

D - 1 = Direct Jump.
0 = Indirect Jump.

S = Designates instruction syllable within a word.

STACK

Unchanged.

INTERRUPTS

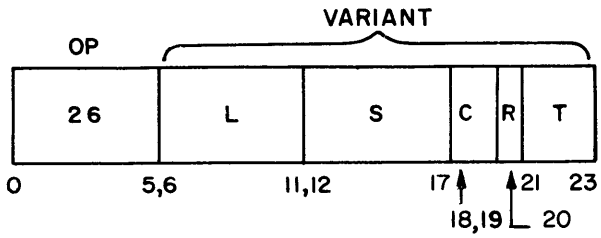
Bit 58 in ICR set if variant C = 0 and if JCW tag does not equal 10, or if RCW tag does not equal 11.

TIMING

ADVAST 0.3 usec COMM 0 usec. FINST 0 usec.

Memory access may be required.

JFT
Jump on
Field Test



4 Syllables

DESCRIPTION

The field defined by [S,L] plus AAR in the T Register is compared with the same specified field in the S Register, or zero. If the conditional test is true, then the jump will be executed as specified by the Jump Control Register, or a bit will be set in T[47, 1]. If the result is false, then the next instruction in sequence is executed. The AAR final contents are reset. The defined field tested is treated as a positive integer value.

VARIANT

L = Length of fields to be compared

S = Starting bit position of fields

NOTE

If S = 48, or L = 0, then T is unchanged. If S + L > 48, then the field of T is [S, 48-S].

R =
C =
T =
} Same as in JSTA, page 4-19.

STACK

The same as specified by the variant field R, C, T in the JSTA instruction.

INTERRUPTS

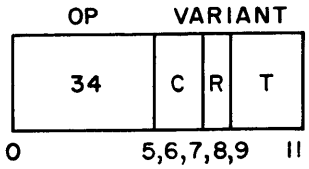
Bit 65 in ICR is unconditionally set.

TIMING

ADVAST 0.3 usec. COMM 0 usec. FINST 0.6 usec.

Plus time for FINQ to empty if C ≠ [10] or [11].

JSTA
Jump on Stack
Test Arithmetic



2 Syllables

DESCRIPTION

The arithmetic test as specified by C and T is performed. If conditional test is true, then the jump will be executed as specified by the Jump Control Register previously set by the SJ instruction, or bit in T [47,1] will be set. In either of above conditions, if the result is false, then the next instruction in sequence is executed. Operands are treated algebraically with signs and exponents considered.

NOTE

If required to execute a valid arithmetic comparison, operands will be normalized, compared, and then the results will be in a normalized, floating point form.

VARIANT

R - 0 = Compare T with 0.

1 = Compare T with S.

C - 00 = IF compare T with 0, then step Stack up once. IF compare T with S, then step Stack twice. If test TRUE, Jump.

01 = Hold Stack; if test TRUE, Jump.

10 = IF compare T with 0, then hold stack, reset T contents; if test was TRUE, set T [47,1]. IF compare T with S, then step Stack up once and reset T contents; if test was TRUE, set T [47,1].

11 = Step stack down once, reset T contents; if test was TRUE, set T [47,1].

T - 000 = Unconditionally false.

001 = T < S or T > Zero

010 = T = S or T = Zero

011 = T ≤ S or T ≥ Zero

100 = T > S or T < Zero

101 = T ≠ S or T ≠ Zero

110 = T ≥ S or T ≤ Zero

111 = Unconditionally true.

(continued)

STACK

As specified by variant field.

INTERRUPTS

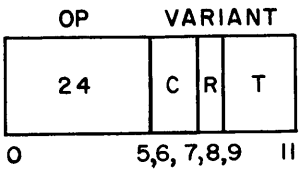
Bit 43 in ICR set if a floating point exponent underflow as a result of normalizing.
Bit 46 in ICR set if floating point operand not normalized.

TIMING

ADVAST 0.3 usec. COMM 0 usec. FINST 0.2 usec.

Plus time for FINQ to empty if C ≠ [10] or [11].

JSTL
Jump on Stack
Test Logical



2 Syllables

DESCRIPTION

The logical field test, as specified by the C and T variants, is performed. If the conditional test is true, then the jump will be executed as specified by the Jump Control Register that was previously set by the SJ instruction, or bit 47 in the T Register will be set and the next instruction in sequence will be executed. In either of above, if the test result is false, then the next instruction in sequence is executed. Operands are treated as 48-bit positive integer values.

VARIANT

Same as JSTA instruction, page 4-19.

STACK

As specified by the variant.

INTERRUPTS

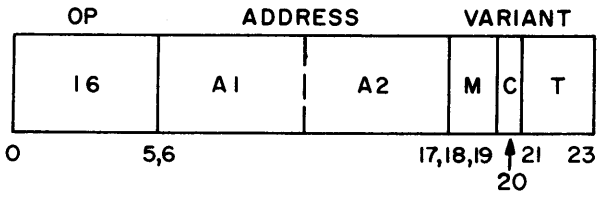
None.

TIMING

ADVAST 0.3 usec. COMM 0 usec. FINST 0.2 usec.

Plus time for FINQ to empty if C ≠ [10] or [11].

JXMT
Jump On Index
Modify and Test



4 Syllables

DESCRIPTION

The index word*, as addressed by AAR, A1, A2, and BXR, is placed in an index register, modified, and tested as specified by the variant field. After modification and test, the index word is returned to memory. If the result of the test is true, then the jump will be executed as specified by the JCR. If the result is false, then the next instruction in sequence is executed. The final contents of AAR are reset.

VARIANT

M - 00 = No modification.
01 = Add index increment to index.
10 = Add T Register [30, 18] to index.
11 = Subtract T register [30, 18] from index.

C - 0 = Compare index with T [30, 18] .
1 = Compare index with index limit.

T - 000 = Test is unconditionally false.
001 = Jump on index greater.
010 = Jump on index equal.
011 = Jump on index greater or equal.
100 = Jump on index less.
101 = Jump on index unequal.
110 = Jump on index less or equal.
111 = Test is unconditionally true.

STACK

Stepped up once if C = 0.

INTERRUPTS

Bit 36 in ICR set if normal bounds violation.

Bit 59 in ICR set if Interrupt tag bit set on word fetched.

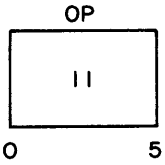
Bit 65 is set unconditionally.

TIMING

ADVAST 0.6 usec. COMM 0 usec. FINST 0 usec. Plus time for FINQ to empty if M = 10 or 11.

*See page 2-6 for index word format.

RET
Return from Subroutine



1 Syllable

DESCRIPTION

The contents of memory as addressed by BXR (RCW) are placed in a save register. If the RCW "type" field equals 00 (procedure jump), then the contents of memory, as addressed by the PRT Register, are placed in the PRT and the PRTL Registers. If tag bits 48 and 49 of this word (in PRT) do not equal 01, then an interrupt may occur. Also, the sequence given in A below is performed. If the type field does not equal 00, then only the sequence in A is performed.

NOTE

The word at the location specified by the PRT Register should always contain the contents of the PRT and PRTL Registers for the previous program segment. (See page 4-15.)

A- The $\overline{\Delta BXR}$ field of the RCW is added to the present BXR value, and the result is placed in the BXR. (This decrements the BXR back to the value it contained before the jump.) The PCR field RCW saved above is placed in the PCR. Then the contents of memory as specified by the BXR (after decrement), a Return Control Word, is again placed in the save register. The contents of memory as addressed* by the PRT field of RCW, and the PRT Base Register are placed in a JCW. This is the JCW that was used to jump into the segment to which control is now being returned. This use of the JCW is required, because the RCW does not contain the BPR of the calling segment. The BPR field of FCW is placed in the BPR, and the next instruction in sequence is specified by PCR.

VARIANT

None.

STACK

Unchanged.

INTERRUPTS

Bit 58 in ICR set if the tag of the RCW $\neq$ 11, or if the tag in fetch of PRT and PRTL $\neq$ 01.

*Bit 35 in ICR set if PRT bounds violation.

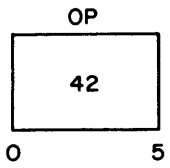
TIMING

ADVAST 1.6 usec. COMM 1.2 usec. FINST 0 usec.

LOGICAL INSTRUCTIONS

The logical instructions are used to manipulate the operands on the T Register and the S Register. All 48 bits of the operands are referenced. Tag fields of operand results will be reset.

AND
And



1 Syllable

DESCRIPTION

The contents of the T Register are ANDED with the contents of the S Register. The results are placed in the T Register.

Boolean Values		
S	T	RESULT
0	0	0
0	1	0
1	0	0
1	1	1

VARIANT

None.

STACK

The original contents of T and S are lost. Stack is stepped up once and results are placed in T.

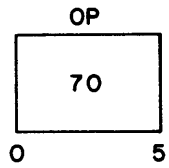
INTERRUPTS

None.

TIMING

ADVAST 0.1 usec. COMM 0 usec. FINST 0.1 usec.

COMP
Complement



1 Syllable

DESCRIPTION

The contents of the T Register are complemented and the result is placed in the T Register.

VARIANT

None.

STACK

Unchanged except for T.

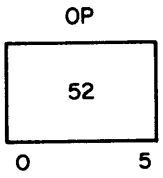
INTERRUPTS

None.

TIMING

ADVAST 0.1 usec. COMM 0 usec. FINST 0.2 usec.

IMP
Implication



1 Syllable

DESCRIPTION

The contents of the T Register are IMPLIED with the contents of the S Register. The result is placed in the T Register.

T	S	RESULT
0	1	0
1	0	1
1	1	1
0	0	1

VARIANT

None.

STACK

The original contents of T and S are lost. The Stack is stepped up once and the results are placed in T.

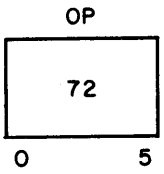
INTERRUPTS

None.

TIMING

ADVAST 0.1 usec. COMM 0 usec. FINST 0.1 usec.

OR
Or



1 Syllable

DESCRIPTION

The contents of the T Register are OR'd inclusively with the contents of the S Register. The result is placed in the T Register.

S	T	RESULT
0	0	0
0	1	1
1	0	1
1	1	1

VARIANT

None.

STACK

The original contents of T and S are lost. The Stack is stepped up once and the results are placed in T.

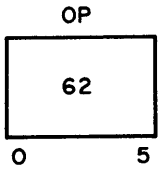
INTERRUPTS

None.

TIMING

ADVAST 0.1 usec. COMM 0 usec. FINST 0.1 usec.

ORX
Exclusive Or



1 Syllable

DESCRIPTION

The contents of the T register are OR'd exclusively with the contents of the S Register. The result is placed in the T register.

S	T	RESULT
0	0	0
0	1	1
1	0	1
1	1	0

VARIANT

None.

STACK

The original contents of T and S are lost. The Stack is stepped up once, and the results are placed in T.

INTERRUPTS

None.

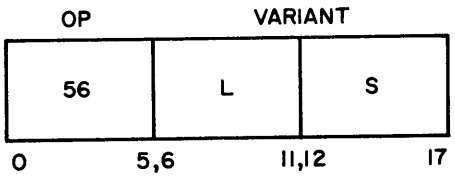
TIMING

ADVAST 0.1 usec. COMM 0 usec. FINST 0.1 usec.

FIELD MANIPULATING INSTRUCTIONS

The field manipulating instructions operate on the bit configurations in the T Register and the S Register. Some instructions also involve the P Register. Variant field Specifiers L, L2, S may be modified by the contents of AAR.

CLRF
Clear Field



3 Syllables

DESCRIPTION

The contents of the field in T, specified by the variant L and S, plus AAR, are set to zeros. The unspecified area of T is unchanged. The AAR final contents are reset. The tag field is reset.

VARIANT

L = length of field.

S = starting bit position of field.

NOTE

If L = 0 or S = 48, then T is unchanged. If (S+L) > 48, then field of T is [S, 48-S].

STACK

Unchanged except for T.

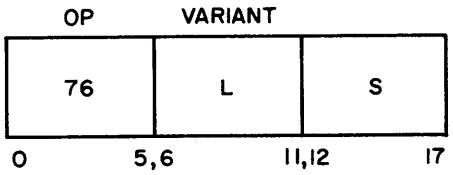
INTERRUPTS

Bit 65 in ICR is unconditionally set.

TIMING

ADVAST 0.3 usec COMM 0 usec. FINST 0.5 usec.

FILF
Fill Field



3 Syllables

DESCRIPTION

The contents of the field in the T Register, specified by L and S, plus AAR, are set to 1's. The unspecified area of T is unchanged. The AAR final contents are reset. The tag field is reset.

VARIANT

L = Length of field

S = Starting bit position of field

NOTE

If S=48 or L=0, then T is unchanged. If L + S > 48, then the field of T is [S, 48-S].

STACK

Unchanged except for T.

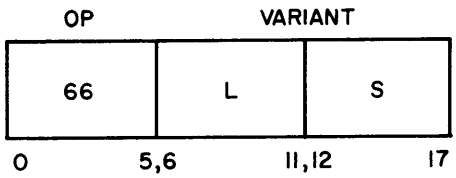
INTERRUPTS

Bit 65 in ICR is unconditionally set.

TIMING

ADVAST 0.3 usec. COMM 0 usec. FINST 0.5 usec.

COMF
Complement Field



3 Syllables

DESCRIPTION

The contents of the field in the T Register, specified by L and S, plus AAR, are complemented. The unspecified area of T is unchanged. The AAR final contents are reset. The tag field is reset.

VARIANT

L = Length of field.

S = Starting bit position of field.

NOTE

If S=48, or L=0, then T is unchanged. If S + L > 48, then the field of T is [S, 48-S].

STACK

Unchanged except for T.

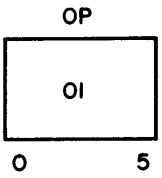
INTERRUPTS

Bit 65 in ICR is unconditionally set.

TIMING

ADVAST 0.3 usec. COMM 0 usec. FINST 0.5 usec.

DUP
Duplicate Top of Stack



1 Syllable

DESCRIPTION

The contents of the T Register are duplicated into the S Register. The original contents of the T Register are unchanged. The tag field is also duplicated.

VARIANT

None.

STACK

Stepped down once before T is duplicated into S.

INTERRUPTS.

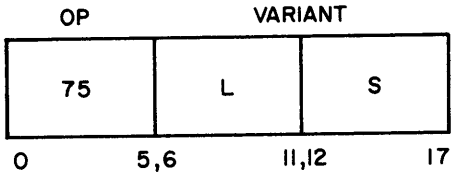
None.

TIMING

ADVAST 0.1 usec. COMM 0 usec. FINST 0.1 usec.

EXT

Extract Field



3 Syllables

DESCRIPTION

The contents of the field of the T Register, specified by L and S, plus AAR, are extracted and inserted right justified into the T Register. Positions not included in the specified field are set to zeros. The AAR final contents are reset. The tag field in T Register will be reset.

VARIANT

L = Length of field

S = Starting bit position of field

If L = 0, or S = 48, then the result in T = 0.

STACK

Unchanged except for T.

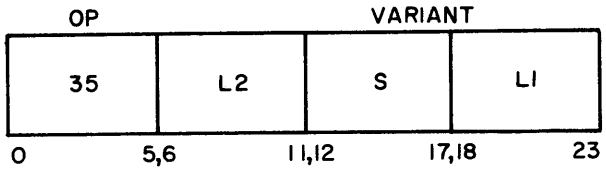
INTERRUPTS

Bit 65 in ICR is unconditionally set.

TIMING

ADVAST 0.3 usec. COMM 0 usec. FINST 0.4 usec.

EXTD
Extract Field Double



4 Syllables

DESCRIPTION

The field specified by variant L1L2 and S, within a double-length register formed by T and S, is right justified with the LSB of the field in the LSB position of the S Register. The unspecified resultant field in S and/or T Register is set to zeros. The tag fields of T and S results are reset.

Note: (L2 and S syllables may be modified by contents of AAR to derive actual field specifiers.) AAR final contents are reset.

VARIANT

S = Start of field (not greater than 47).
L1 L2 = Length of field.
} $S + L1\ L2 \leq 95$.

STACK

Unchanged (except T and S)

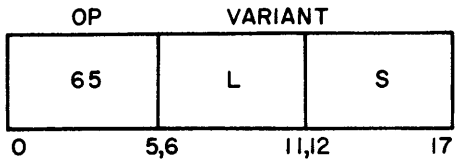
INTERRUPTS

Bit 65 in ICR is unconditionally set.

TIMING

ADVAST 0.3 usec. COMM 0 usec. FINST 0.6 usec.

INS
Insert Field



3 Syllables

DESCRIPTION

The contents of the right-justified field in the T Register, the length specified by L, are inserted into the S Register, with the starting bit of the field specified by S. After the field is inserted into S, the Stack is stepped up once with the tag field reset. The AAR final contents are reset.

NOTE

L and S may be modified by the AAR.

VARIANT

L = Length of field

S = Start bit position of field in S Register

(S=48 is the same as S=0)

(L=0, operand S is placed unchanged into T.)

STACK

Stepped up after inserting field into S Register, with final results in T.

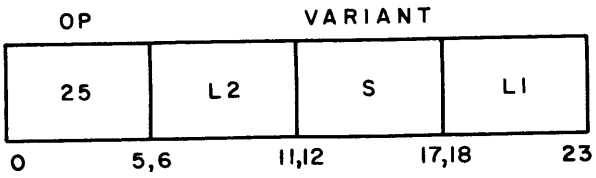
INTERRUPTS

Bit 65 in ICR is unconditionally set.

TIMING

ADVAST 0.3 usec. COMM 0 usec. FINST 0.8 usec.

INSD
Insert Field
Double



4 Syllables

DESCRIPTION

The right-justified field, length specified by L1 L2 (within double-length register formed by T and S), is inserted into N and M with the MSB of the field in the position specified by S. Then, the Stack is stepped up twice with the double length result in T and S. The tag fields are reset.

Note: The L2 and S syllables may be modified by the contents of AAR to derive actual field specifiers. AAR final contents are reset.

VARIANT

S = Start of field (not greater than 47).
L1 L2 = Length of field
} S + L1 L2 < 95.

STACK

Stepped up twice.

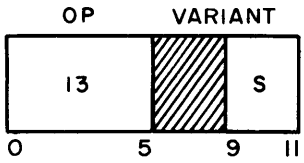
INTERRUPTS

Bit 65 in ICR is unconditionally set.

TIMING

ADVAST 0.3 usec. COMM 0 usec. FINST 1.2 usec.

RTS
Rearrange Top of Stack



2 Syllables

DESCRIPTION

The word positions, including the tag fields, in the top of the Stack, T, P, S, N, and M, are rearranged as specified by the variant field.

VARIANT OPTIONS

S -
- 000 = Exchange T and S.
- 001 = Replace P with T.
- 010 = Replace T with P.
- 011 = Swap T and P.
- 100 = Stack is stepped down twice. Contents of T are replaced by contents of N; Contents of S are replaced by contents of M.
- 101 = Step Stack down and reset T.
- 110 = Exchange registers N and M with T and S.
- 111 = Step Stack up.

STACK

As above.

INTERRUPTS

None.

TIMING

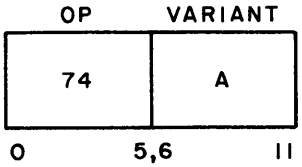
ADVAST 0.1 usec. COMM 0 usec. FINST { REVD 0.4 usec. / DUPD 0.2 usec. / ---- 0.1 usec. }

SHIFT INSTRUCTIONS

In the Shift instructions, the variant specifies the type of shift that is to be applied, normally, to the operand at the top of the stack. These variants specify right or left shifts, arithmetic or logical shifts, circular or end-off shifts, single or double shifts, and shift amount.

The shift amount specifier indicates whether the shift amount is to be taken from the BSR or the top of the Stack. If the shift amount is specified in the top of the Stack, the operand to be shifted is contained in the S Register.

BSR
Load Barrel
Shift Register



2 Syllables

DESCRIPTION

The amount A plus the contents of the AAR are placed in the Barrel Shift Register. The AAR is then reset.

VARIANT

A = 00 ——→ 77 (Shift amount)

STACK

Unchanged.

INTERRUPTS

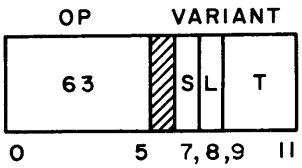
Bit 65 in ICR is unconditionally set.

TIMING

ADVAST 0.3 usec. COMM 0 usec. FINST 0.1 usec.

SHF

Shift



2 Syllables

DESCRIPTION

The contents of the (T or S) and P Registers are shifted as specified by the variant field. Arithmetic shifts greater than 35 are treated as 35. Logical shifts greater than 48 are treated as 48.

VARIANT

S - 0 = Shift T by the amount in BSR.
1 = Shift S by the amount in the T register.

L - 0 = Single length, shift T register.
1 = Double length, shift (T or S) and P registers.

T - 000 = Right, end-off, arithmetic.*
001 = Right, end-off, logical.**
010 = Right, end-around arithmetic.
011 = Right, end-around, logical.
100 = Left, end-off, arithmetic.
101 = Left, end-off, logical.
110 = Left, end-around, arithmetic.
111 = Left, end-around, logical.

*Arithmetic shift means field affected is in bit positions 13 thru 47.
**Logical shift means field affected is in bit positions 0 thru 47.

STACK

If the shift amount is specified by the T Register, then the S Register is shifted by the amount specified in the T Register and the Stack is stepped up once. If the shift amount is specified by BSR, then T is shifted and the Stack is unchanged.

INTERRUPTS

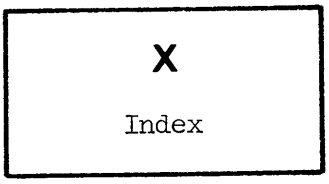
None.

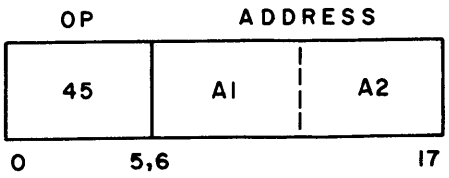
TIMING

ADVAST 0.1 usec. COMM 0 usec. FINST 0.2 usec.

INDEXING INSTRUCTIONS

Index words are in memory BXR relative, or in the T Register. In both cases, the resultant address is placed in the AAR.





3 Syllables

DESCRIPTION

The contents of memory, as addressed by BXR and A1 A2, are placed in index Q. The index Q contents field is added to the AAR, and the result is placed in the AAR. The address of the index word is unconditionally checked against normal bounds.

VARIANT

None.

STACK

Unchanged.

INTERRUPTS

Bit 36 in ICR is set if normal bounds violation.

Bit 59 in ICR is set if interrupt tag in word fetched is set.

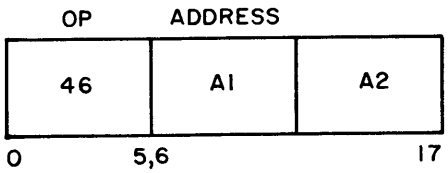
TIMING

ADVAST 0.5 usec. COMM 0 usec. FINST 0 usec.

Memory access may be required.

XM

Index and Modify Index



3 Syllables

DESCRIPTION

The contents of memory, as addressed by BXR and A1 A2, are placed in index Q. The contents of the AAR plus index Q address field, are placed in the AAR. The index Q increment or decrement then modifies the index Q contents field and the result is placed in the index contents field. Index is returned to memory. The address of the index word is unconditionally checked against normal bounds.

VARIANT

None.

STACK

Unchanged.

INTERRUPTS

Bit 36 in ICR is set if normal bounds violation.

Bit 59 in ICR is set if interrupt bit in word fetched is set.

TIMING

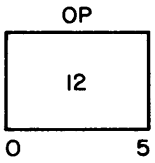
ADVAST 0.7 usec. COMM 0 usec. FINST 0 usec.

Memory access may be required.

*See page 2-6 for index word format.

XS

Index by Stack



1 Syllable

DESCRIPTION

The contents of the AAR plus the T Register (bits 30 thru 47) are placed in the AAR.

VARIANT

None.

STACK

Stepped up once after modifying AAR.

INTERRUPTS

None.

TIMING

ADVAST 0.3 usec.* COMM 0 usec. FINST 0.1 usec.

*Plus time to empty FINQ.

ARITHMETIC INSTRUCTIONS

There are three different word formats for arithmetic data:

1. Floating point.

2. Double precision floating point.

3. Integer.

Integer values, in effect, are represented in unnormalized floating point form with a zero exponent. Hence, almost all arithmetic instructions handle integer and floating point numbers.

Arithmetic operations with either operand in integer form, and a result that is too large to express in integer form, is in normalized floating point (except for instructions ADDM and SUBM).

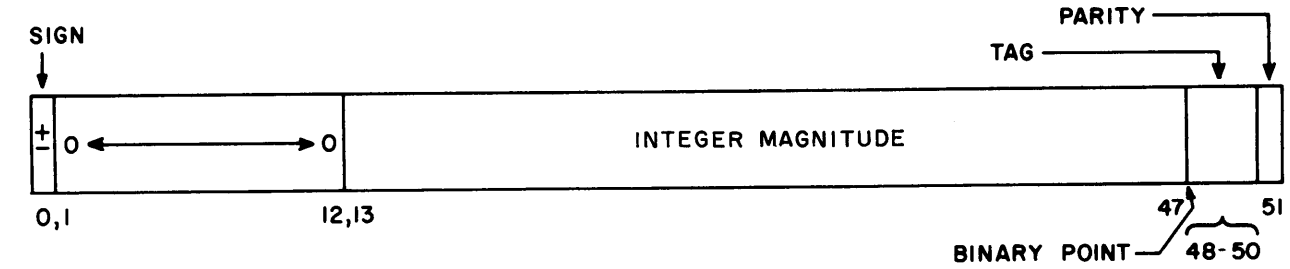
INT (Integerize) and DIVI (Integer Divide) instructions are available for obtaining integer results from floating point numbers.

Arithmetic instructions resulting in floating point numbers are normalized with the most significant bit (MSB) of the mantissa in bit position 13.

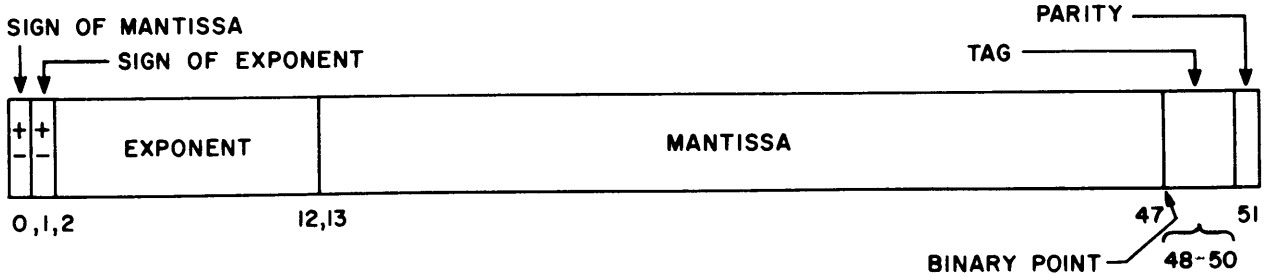
Tags of a one-operand instruction, e.g. NORMALIZE, INTEGERIZE etc., are reset. In a two-operand instruction, the tag of the S operand is placed in the tag of the result in T. In four-operand instructions (ARIT), the tag of N is placed in T.

WORD FORMATS

Integer Binary Word

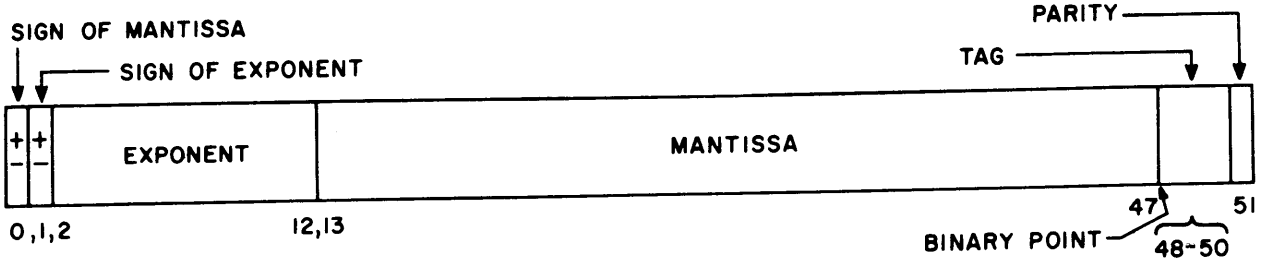


Floating Point - Single Precision

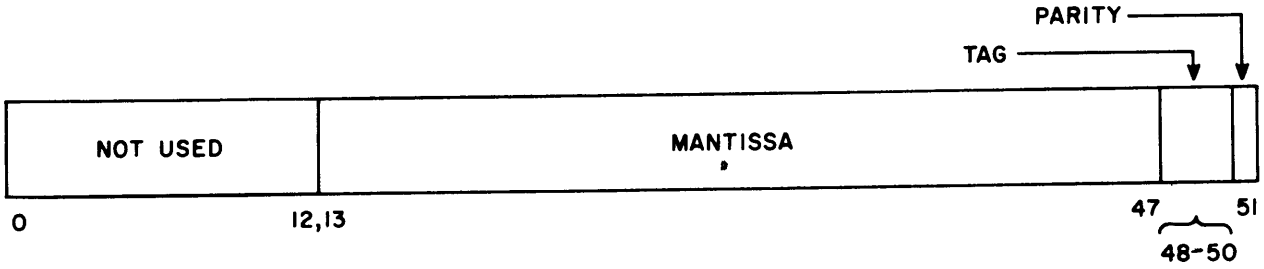


Floating Point - Double Precision

Word n

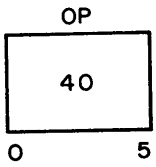


Word n+1



ADD

Add



1 Syllable

DESCRIPTION

The contents of the S Register are added to the T Register. The result is in the T register.

VARIANT

None.

STACK

The original contents of T and S are lost. The Stack is stepped up once, and the sum is placed into T.

INTERRUPTS

Bit 42 in ICR is set if exponent underflow, i.e., if the exponent result is < -2047.

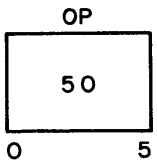
Bit 43 in ICR is set if exponent overflow, i.e., if the value of the sum exceeds the size of the exponent field.

TIMING

ADVAST	0.1 usec.	COMM	0 usec.	FINST	0.5 usec.

ADDM

Add Magnitude



1 Syllable

DESCRIPTION

The entire contents of the S Register are added to the entire contents of the T Register. The result is placed in the T Register.

VARIANT

None.

STACK

The original contents of T and S are lost. The Stack is stepped up once and the result is placed in T.

INTERRUPTS

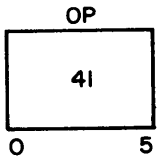
None.

TIMING

ADVAST	0.1 usec.	COMM	0 usec.	FINST	0.2 usec.

SUB

Subtract



1 Syllable

DESCRIPTION

The contents of the T Register are algebraically subtracted from the S Register and the result is placed in the T Register.

VARIANT

None.

STACK

The original contents of T and S are lost. The Stack is stepped up once and the result is placed into T.

INTERRUPTS

Bit 42 in ICR is set if exponent underflow, i.e., if the exponent result is < -2047.

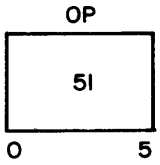
Bit 43 in ICR is set if exponent overflow.

TIMING

ADVAST	0.1 usec.	COMM	0 usec.	FINST	0.5 usec.

SUBM

Subtract
Magnitude



1 Syllable

DESCRIPTION

The entire contents of the T Register are subtracted from the entire contents of the S Register. The result is placed in the T Register. If T> S, then the result is in one's complement form.

VARIANT

None.

STACK

The original contents of T and S are lost. The Stack is stepped up once, and the result is placed in T.

INTERRUPTS

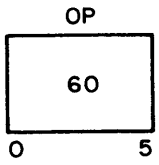
None.

TIMING

ADVAST 0.1 usec. COMM 0 usec. FINST 0.2 usec.

MUL

Multiply



1 Syllable

DESCRIPTION

The contents of the S Register are multiplied by the contents of the T Register and the results are placed in the T Register.

VARIANT

None.

STACK

The original contents of T and S are lost. The Stack is stepped up once, and the result is placed in T.

INTERRUPTS

Bit 42 set in ICR if exponent underflow.

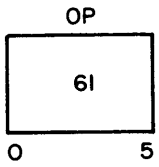
Bit 43 set in ICR if exponent overflow.

TIMING

ADVAST	0.1 usec.	COMM	0 usec.	FINST	1.0 usec.

DIV

DIVIDE



1 Syllable

DESCRIPTION

The contents of the S Register are divided by the contents of the T Register. The quotient, in normalized floating point, is placed in the T Register, and the remainder is placed in the P Register. If the mantissa of the S Register (dividend) = 0, then the T and P Registers are set to zero. If the mantissa of the T Register (divisor) = 0, the divide-by-zero bit is set in the ICR.

VARIANT

None.

STACK

The original contents of T and S are lost. The Stack is stepped up once, and the result is placed in T.

INTERRUPTS

Bit 42 in ICR is set if exponent underflow.

Bit 43 in ICR is set if exponent overflow.

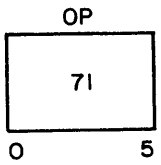
Bit 47 in ICR is set if divisor equals 0.

TIMING

ADVAST	0.1 usec.	COMM	0 usec.	FINST	4.1 usec.

DIVI

Divide for
Integer Quotient



1 Syllable

DESCRIPTION

The contents of the S Register are divided by the contents of the T Register. The integer part of the quotient is placed in the T Register. The integer remainder is placed in the P Register, right justified. (Dividend=(Integer quotient x divisor)+ remainder.) If the quotient is too large to express in integer form, then this instruction will function in the same way as the Divide instruction. If the value in the S Register is algebraically less than the value in the T Register (and neither = 0), then the T Register is set to zero and the S Register is transferred to the P Register.

VARIANT

None.

STACK

The original contents of T and S are lost. The Stack is stepped up once, and the result is placed in T and P.

INTERRUPTS

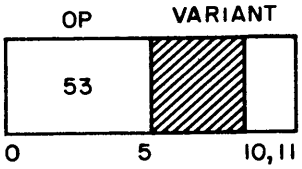
Bit 42 in ICR is set if exponent underflow.
Bit 43 in ICR is set if exponent overflow.
Bit 47 in ICR is set if divisor equals 0.

TIMING

ADVAST 0.1 usec. COMM 0 usec. FINST 4.5 usec.

ARIT

Double Precision



2 Syllables

DESCRIPTION

Double precision arithmetic is performed as specified by the variant field. One operand is in N and M, and the other operand is in T and S. Results are placed in the T and S Registers. The most significant operands are the N and T Registers.

VARIANT

D - 00 = Add double

01 = Subtract double (subtrahend in T, and S)

10 = Multiply double

11 = Divide double (divisor in T, and S)

STACK

The original contents of T,S,N, and M are lost. The Stack is stepped up twice, and the result is placed in T and S.

INTERRUPTS

Bit 42 in ICR is set if exponent underflow.

Bit 43 in ICR is set if exponent overflow.

Bit 47 in ICR is set if divisor equals 0.

TIMING

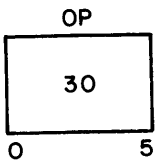
ADVAST 0.1 usec.

COMM 0 usec.

FINST { ADD 3.0 usec. / SUB 3.0 usec. / MUL 7.1 usec. / DIV 18.9 usec. }

NORM

Normalize



1 Syllable

DESCRIPTION

The mantissa contents of the T Register is shifted so that the leading 1-bit in the mantissa is positioned in the MSB position of the mantissa field (bit position 13 of T). The exponent field is algebraically decremented by the number of bit positions that the mantissa is shifted left to achieve normalization. If the entire mantissa field equals zero, then the entire T Register will be set to zeros.

VARIANT

None.

STACK

Unchanged.

INTERRUPTS

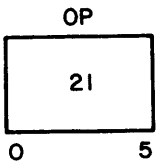
Bit 42 in ICR is set if exponent underflow.

TIMING

ADVAST	0.1 usec.	COMM	0 usec	FINST	0.2 usec.

RND

Round



1 Syllable

DESCRIPTION

The most significant bit of the mantissa in the P Register [13, 1] is added numerically to the mantissa in the T Register. The P Register is unchanged. The result is a normalized or integer number in the T Register. If the result in the T Register mantissa is zero, then the entire T Register is made zero.

VARIANT

None.

STACK

Unchanged.

INTERRUPTS

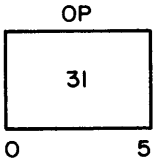
Bit 43 in ICR is set if exponent overflow.

TIMING

ADVAST	0.1 usec.	COMM	0 usec.	FINST	0.3 usec.

INT

Integerize



1 Syllable

DESCRIPTION

The floating point number in the T Register is converted to an integer value, right justified, in the T Register. If the integer value <1 (underflow), the contents of T are placed in P, and the entire T Register is reset. If the integer value $>(2^{35}-1)$, then mantissa overflow occurs, and the contents of the T Register are unchanged. The P Register is initially set to zero. If the instruction requires a right shift, bits are shifted from the T Register into the mantissa of the P Register, and, to allow further additions, the P Register is given a sign and corrected exponent from the T Register.

VARIANT

None.

STACK

Unchanged.

INTERRUPTS

Bit 44 in ICR is set if integer <1 (underflow).
Bit 45 in ICR set if integer $>(2^{35}-1)$ (overflow).

TIMING

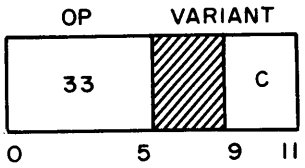
ADVAST	0.1 usec.	COMM	0 usec.	FINST	0.3 usec.

I/O, CONTROL, AND CONVERT INSTRUCTIONS

The I/O and Control instructions monitor certain functions within the Central Processor, Main Memory, I/O Module, and ESP. The convert instructions are for BDC/ binary numeric code conversion.

CBB

BCD Conversion



2 Syllables

DESCRIPTION

The contents of the T Register are converted as specified by the variant field. The result is placed in the T Register.

This instruction should only be used with operands that are known to be all numeric.

VARIANT

C - 000 = Four-bit BCD to binary (see page 2-3 for data format). The Binary result in the T Register is unsigned.

011 = Six-bit BCD to binary, signed. There are eight BCD characters in the T Register. If [42,2] in the T Register = 01, then the sign of the binary result in the T Register will be negative. Otherwise, the sign of T will be set positive.

010 = Six-bit BCD to binary, unsigned. The same as above except that the sign of the binary result in T will be positive.

100 = Binary to four-bit BCD. The result is in the T Register. The first eight positions of T will be zeros.

111 = Binary to six-bit BCD, signed. Eight BCD characters result in the T Register. If negative binary value, then set T [42,2] to 01. If positive, set [42,2] to 11. The two most significant bits of each character (except the rightmost) are set to 1's.

110 = Binary to six-bit BCD, unsigned. The same as the signed case above, except [42,2] of the T Register is unconditionally set to 11.

STACK

As above.

INTERRUPTS

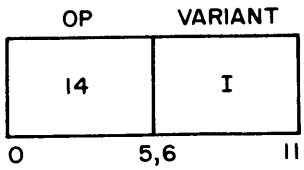
Bit 45 in ICR is set if the binary value is too large to convert into one word of BCD characters, i.e., > 2^{35} - 1(34359738367).

TIMING

ADVAST	0.1 usec.	COMM	0 usec.	FINST	4.2 usec. six-bit BCD Binary 2.9 usec. Binary six-bit BCD 5.2 usec. four-bit BCD Binary 4.0 usec. Binary four-bit BCD

ESP

Enter Executive & Scheduling Program



2 Syllables

DESCRIPTION

Upon execution of this instruction, the mode of the processor is reduced by one, i.e., from Normal Mode to Control Mode 1, or from Control Mode 1 to Control Mode 2. Registers are loaded into the Stack as noted below under STACK. An unconditional jump is initiated and the next instruction executed is in the location specified by BIAR1 (if the processor is in Normal Mode, and BIAR2 if the processor is in Control Mode 1) modified by the I variant. The AAR is reset.

VARIANT

I - Instruction word count is added to BIAR1 or BIAR2.

STACK

Stepped down four locations. The top four stack locations are as follows:

T = Control flip-flops.

S = AAR contents (before reset).

N = PCR contents.

M = BPR contents.

INTERRUPTS

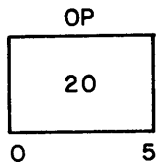
Unconditional.

TIMING

ADVAST 0.1 usec. + time to empty FINQ COMM 0 usec. FINST 0 usec.

ETB

Extract
Tag Bits



1 Syllable

DESCRIPTION

Within the T Register, the bits in the tag field (48-50) are inserted into the LSB positions of the data field (45-47). Bits 0 thru 44 are reset. Final tag contents are reset.

VARIANT

None.

STACK

Unchanged except for T.

INTERRUPTS

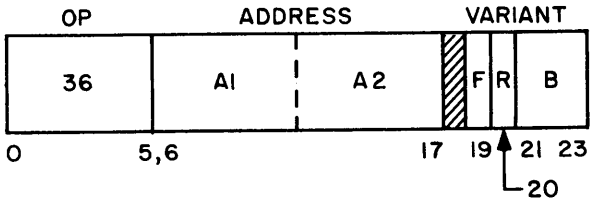
None.

TIMING

ADVAST 0.1 usec. COMM 0 usec. FINST 0.1 usec.

FMT

Fetch and Modify Tags



4 Syllables

DESCRIPTION

This instruction provides a means of controlling the use of specific areas of memory. The contents of memory, as addressed by the AAR, A1 A2, and a base (if specified), are unconditionally modified as specified by the F variant. This operation is performed in the Memory Module.

VARIANT

F - 1 = Set tag bits (48 and 49)
0 - Reset tag bits (48 and 49)

R - 1 = Inhibit reset of AAR
0 = Reset AAR

B - 000 = No base
001 = BPR
010 = BXR
110 = BDR
100 = PRT

STACK

The final contents of the T Register contain the word as it was in Main Memory before the tag was modified.

INTERRUPT

Bit 35 in ICR is set if PRT bounds violation.

Bit 36 in ICR is set if normal bounds violation.

Bit 37 in ICR is set if alternate bounds violation.

Bit 64 in ICR is unconditionally set by this instruction.

Bit 65 in ICR is set if AAR reset.

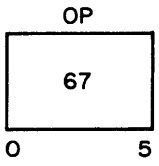
TIMING

ADVAST 0.3 usec. COMM 0.6 usec. FINST 0.1 usec.

Memory access required.

FINQ

Final Queue Halt



1 Syllable

DESCRIPTION

ADVAST places the FINQ instruction in the final Q, than halts and waits until FINST executes the instructions currently waiting in FINQ. When FINST empties, ADVAST is notified and continues with the next instruction in sequence.

VARIANT

None.

STACK

Unchanged.

INTERRUPTS

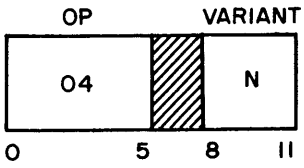
None.

TIMING

ADVAST 0.1 usec. COMM 0 usec. FINST 0.1 usec.

ICN

Interrupt
Computer N



2 Syllables

DESCRIPTION

This instruction sets a bit in the ICR of computer module N. If it is in Control Mode 1 and the HLT flag is set, then a restart of computer N will be executed.

VARIANT

N - The number of the computer module to be interrupted or started.

STACK

Unchanged.

INTERRUPTS

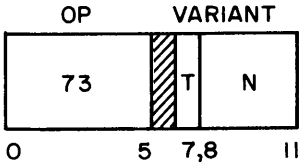
Bit 64 in ICR is unconditionally set by this instruction.

TIMING

ADVAST 0.3 usec. COMM 0 usec. FINST 0 usec.

IOP

Initiate Input/
Output Program



2 Syllables

DESCRIPTION

One of two operations occur depending on the variant T. If T is a 1, then the least significant 18 bits of the T Register are placed in the Job Stack Address Register of the I/O module specified by N. The Job Stack Flag bit in the I/O module is also set. If T is a zero, then only the Job Stack Flag bit in the specified I/O module is set.

VARIANT

T - 1 = Send new Job Stack Address and set Job Stack Flag.

0 = Set Job Stack Flag only.

N - I/O module number.

STACK

Unchanged.

INTERRUPTS

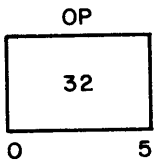
Bit 64 in ICR is unconditionally set by this instruction.

TIMING

ADVAST 0.3 usec. COMM 0 usec. FINST 0.1 usec.

IRR

Interrupt
Routine Return



1 Syllable

DESCRIPTION

This instruction is executed in Control Mode 1 or 2 to return to the next higher mode. The contents of T are sent to the respective control registers (see page 3-7/8). S is placed in AAR, N is placed in PCR, and M is placed in BPR. The processor is then set to Normal Mode if in CM1, or set to CM1 if in CM2. The address of the next instruction is specified by the PCR + BPR.

VARIANT

None.

STACK

Stepped up four locations.

INTERRUPTS

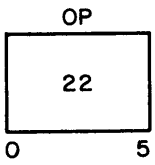
Bit 67 in ICR is set by this instruction, if the processor is in Normal Mode.

TIMING

ADVAST 0.4 usec. COMM 0 usec. FINST 0.4 usec.

ITB

Insert Tag Bits



1 Syllable

DESCRIPTION

The three LSB positions of the data field of the T Register (45-47) are inserted into the tag field of the S Register (48-50).

VARIANT

None.

STACK

Stepped up once after loading the tag in S.

INTERRUPTS

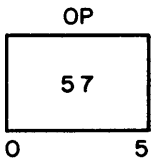
Bit 64 in ICR is unconditionally set by this instruction.

TIMING

ADVAST 0.1 usec. COMM 0 usec. FINST 0.1 usec.

NOP

No Operation



1 Syllable

DESCRIPTION

No operation. The next instruction in sequence is executed.

VARIANT

None.

STACK

Unchanged.

INTERRUPTS

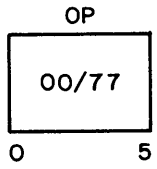
None.

TIMING

ADVAST 0.1 usec COMM 0 usec. FINST 0 usec.

STOP

Stop Processor



1 Syllable

DESCRIPTION

This instruction sets bit 62 in the ICR. If the processor is in Normal Mode or Control Mode 1 and the corresponding mask bit is set, then an interrupt jump is initiated. If the mask bit is not set, then the processor halts, and if the processor is in Control Mode 1, a restart may be initiated by an ICN instruction in another processor. If the processor is in Control Mode 2, the Stop instruction halts the processor. (Note: Either 00_8 or 77_8 can be the op code.)

VARIANT

None.

STACK

No change.

INTERRUPTS

Bit 62 in the ICR is set unconditionally.

TIMING

ADVAST 0.1 usec. COMM 0 usec. FINST 0 usec.

APPENDIX A

B8501 CENTRAL PROCESSOR INSTRUCTION SET

Instruction	Op Code (Octal)	Mnemonic	No. of Syllables	Variant	Page
FETCH AND STORE INSTRUCTIONS					
Fetch Address Register to Stack	55	FAS	3	L1,L2	4-10
Fetch Memory to Address Register	17	FMA	4	R,B	4-12
Fetch Memory Conditionally	27	FMC	4	R,B	4-13
Fetch Memory to Stack	05	FMS	4	R,B	4-4
Fetch Memory to Stack Absolute	10	FMSA	1	None	4-5
Fetch Register to Stack	03	FRS	2	R	4-8
Short Literal	23	SLIT	2	L	4-11
Store Stack to Memory	06	SSM	4	R,B	4-6
Store Stack to Memory Absolute	02	SSMA	1	None	4-7
Store Stack to Register	43	SSR	2	R	4-9
JUMP INSTRUCTIONS					
Jump on Field Test	26	JFT	4	L,S,R,C,T	4-18
Jump on Stack Test Arithmetic	34	JSTA	3	R,C,T	4-19
Jump on Stack Test Logical	24	JSTL	2	R,C,T	4-21
Jump on Index Modify and Test	16	JXMT	4	M,C,T	4-22
Return from Subroutine	11	RET	1	None	4-23
Set Up Jump	07	SJ	4	C,D,S	4-17
LOGICAL INSTRUCTIONS					
And	42	AND	1	None	4-24
Complement	70	COMP	1	None	4-25
Exclusive Or	62	ORX	1	None	4-28
Implication	52	IMP	1	None	4-26
Or	72	OR	1	None	4-27

APPENDIX A
(CONTINUED)

Instruction	Op Code (Octal)	Mnemonic	No. of Syllables	Variant	Page
FIELD MANIPULATING INSTRUCTIONS					
Clear Field	56	CLRF	3	L,S	4-29
Complement Field	66	COMF	3	L,S	4-31
Duplicate Top of Stack	01	DUP	1	None	4-32
Extract Field	75	EXT	3	L,S	4-33
Extract Field Double	35	EXTD	4	L2,S,L1	4-34
Fill Field	76	FILF	3	L,S	4-30
Insert Field	65	INS	3	L,S	4-35
Insert Field Double	25	INSD	4	L2,S,L1	4-36
Rearrange Top of Stack	13	RTS	2	S	4-37
SHIFT INSTRUCTIONS					
Load Barrel Shift Register	74	BSR	2	A	4-38
Shift	63	SHF	2	S,L,T	4-39
INDEXING INSTRUCTIONS					
Index	45	X	3	None	4-40
Index and Modify Index	46	XM	3	None	4-41
Index by Stack	12	XS	1	None	4-42
ARITHMETIC INSTRUCTIONS					
Add	40	ADD	1	None	4-45
Add Magnitude	50	ADDM	1	None	4-46
Double Precision	53	ARIT	2	D	4-52
Divide	61	DIV	1	None	4-50
Divide for Integer Quotient	71	DIVI	1	None	4-51
Integerize	31	INT	1	None	4-55
Multiply	60	MUL	1	None	4-49
Normalize	30	NORM	1	None	4-53
Round	21	RND	1	None	4-54
Subtract	41	SUB	1	None	4-47
Subtract Magnitude	51	SUBM	1	None	4-48

APPENDIX A
(CONTINUED)

Instruction	Op Code (Octal)	Mnemonic	No. of Syllables	Variant	Page
I/O, CONTROL, AND CONVERT INSTRUCTIONS					
BCD Conversion	33	CBB	2	C	4-56
Enter Executive and Scheduling Program	14	ESP	2	I	4-58
Extract Tag Bits	20	ETB	1	None	4-59
Fetch and Modify Tags	36	FMT	4	F,R,B	4-60
Final Queue Halt	67	FINQ	1	None	4-61
Initiate Input/Output Program	73	IOP	2	T,N	4-63
Insert Tag Bits	22	ITB	1	None	4-65
Interrupt Computer N	04	ICN	2	N	4-62
Interrupt Routine Return	32	IRR	1	None	4-64
No Operation	57	NOP	1	None	4-66
Stop Processor	00/77	STOP	1	None	4-67

B8501
CPM REFERENCE MANUAL
FORM BJ-27 OF OCTOBER 1966

CHECK TYPE OF SUGGESTION:

☐ ADDITION ☐ DELETION ☐ REVISION ☐ ERROR

AFFECTING:

PARAGRAPH _____ ON PAGE NO. _____

CUT ALONG THIS LINE

SUBMITTED BY
NAME _____ LOCATION _____

YOUR COMMENTS ON IMPROVING THIS PUBLICATION ARE GRATEFULLY ACKNOWLEDGED.

Burroughs Corporation

FOLD DOWN SECOND FOLD DOWN

Postage Will be Paid by Addressee

No Postage Stamp Necessary If Mailed in the United States

BUSINESS REPLY MAIL
First Class Permit No. 73, Paoli, Pa.

Burroughs Corporation
Defense, Space and Special Systems Group

Paoli, Pa., 19301

Attn: B8500 Program
Department 4419

FOLD UP FIRST FOLD UP

CUT ALONG THIS LINE