SEL 32

REFERENCE MANUAL

SEL 32

REFERENCE MANUAL

August 1975

SYSTEMS
ENGINEERING LABORATORIES

- Specifications Subject to Change Without Notice

- Publication Number 324-322000-001
- Printed in USA
- Price $20.00

LIST OF EFFECTIVE PAGES

The total number of pages in this manual is 286, consisting of the following:

Page	Issue
Title	Original
A	Original
i through iv	Original
1-1 through 1-4	Original
2-1 through 2-12	Original
3-1 through 3-10	Original
4-1 through 4-4	Original
5-1 through 5-186	Original
6-1 through 6-28	Original
A-1 through A-8	Original
B-1 through B-6	Original
C-1 through C-4	Original
D-1 through D-8	Original
E-1 and E-2	Original
F-1 and F-2	Original
G-1 and G-2	Original
H-1 and H-2	Original

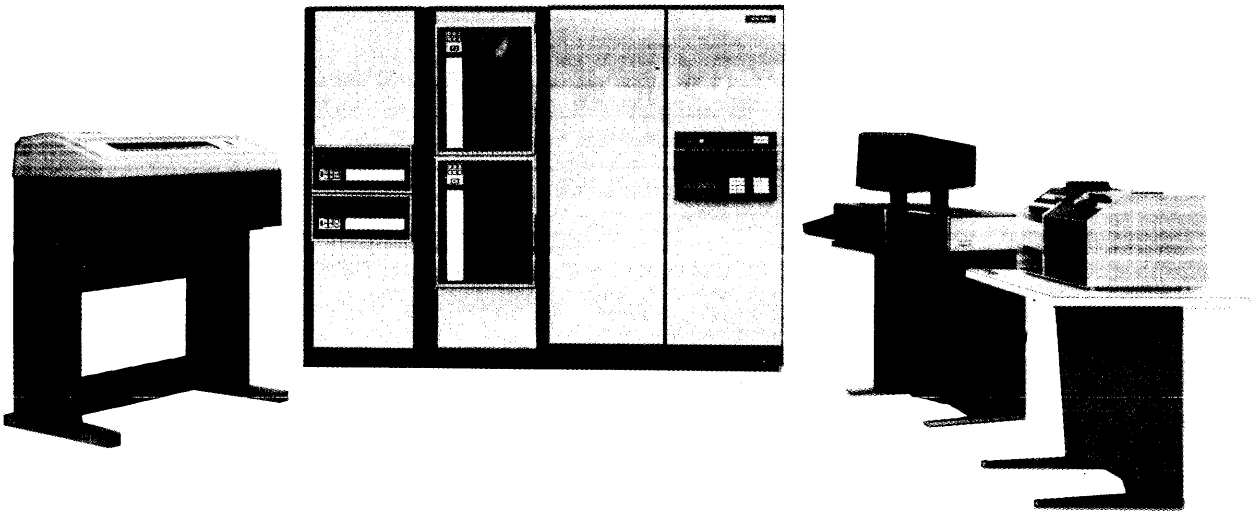
TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

LIST OF TABLES



SEL 32 System Configuration

SECTION I

INTRODUCTION

GENERAL DESCRIPTION

The SEL 32 Series is a hierarchy of true 32-bit machines providing full 32-bit computer performance. This performance is further enhanced by a high-speed, flexible input/output system and a floating-point processor which is standard on all models.

The SEL 32 architecture is designed around a synchronous time multiplexed SEL Bus having a transfer rate of 26.67 million bytes per second. The major system elements attached to the SEL Bus are the Central Processor Unit, Memory Bus Controllers (MBC), Real-Time Option Modules (RTOM), and Input/Output Microprogrammable Processors (IOM). A system block diagram is shown in Figure 1-1.

ADDRESSING

The SEL 32 capable of directly addressing any bit, byte, halfword, word (32-bits), or doubleword in up to 131,072 words (512K Bytes) of memory. Bits, bytes, halfwords, words and doublewords in extended memory (above 128KW) are addressed through the extended addressing mode.

FLOATING-POINT ARITHMETIC PROCESSORS

A firmware floating-point arithmetic processor is standard with all central processor units. This firmware floating-point arithmetic processor executes all floating-point instructions significantly faster than normal software floating-point routines.

GENERAL PURPOSE REGISTERS

The SEL 32 has eight general purpose registers, of which three can also be used for indexing, one for masking operations and another for linking.

REAL-TIME OPTION MODULE

The Real-Time Option Module provides 16 external priority interrupt levels, a real-time clock and an interval timer all on one card. Up to eight RTOM cards may be added to a system.

MEMORY SYSTEM

The Memory Bus Controller (MBC) is the interface between the SEL Bus and the memory modules. Each MBC handles up to 128K Words (512K bytes) of memory modules having the same cycle time. Memory is added to the system in increments of 8K Words (32K Bytes).

The SEL 32 Memory is organized into 32-bit data words and 4 parity bits (one for each byte). The Memory Bus Controller (MBC), communicates with the SEL Bus and controls the Memory Bus to which the Memory Modules are attached.

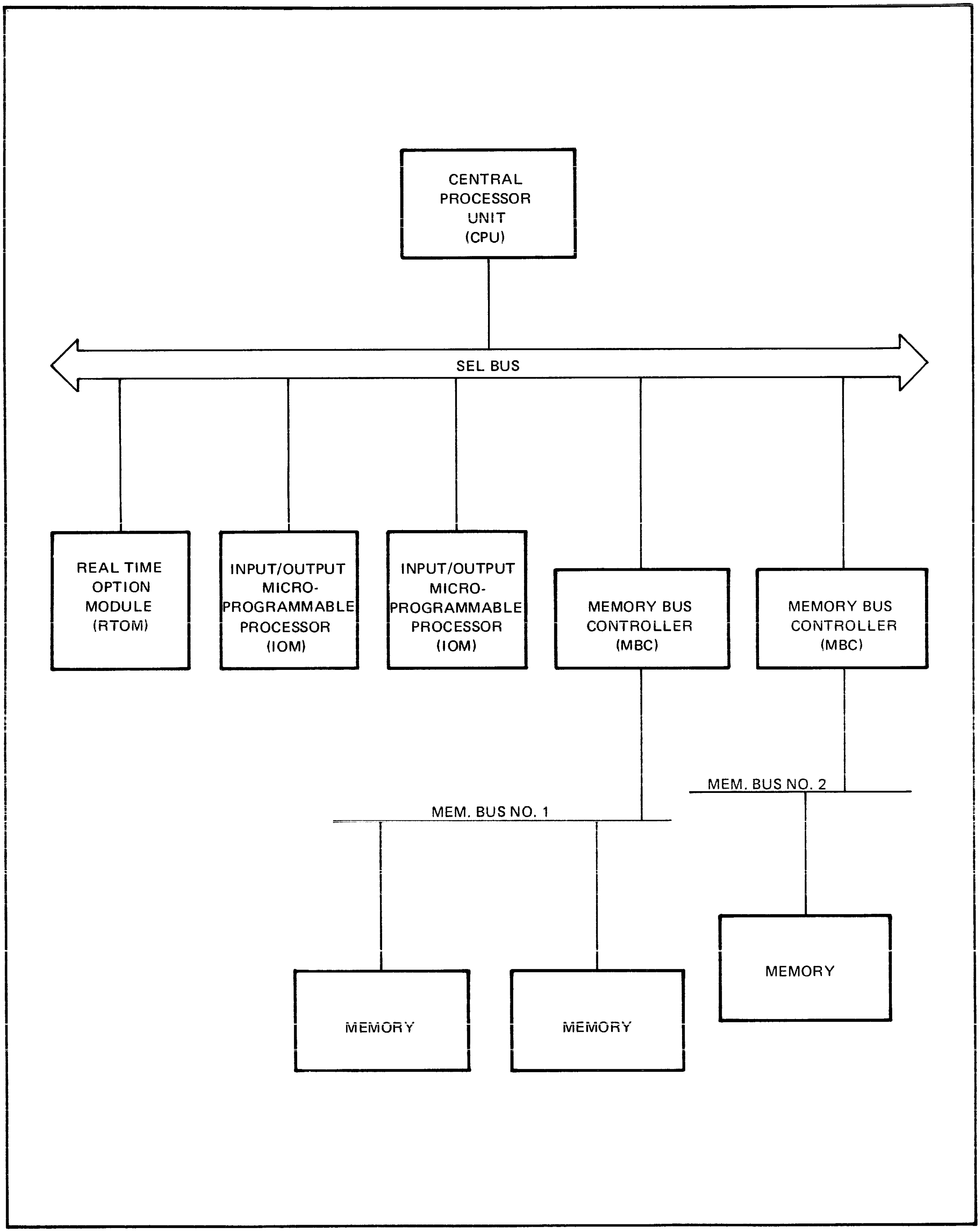

CENTRAL PROCESSOR UNIT (CPU)
SEL BUS
REAL TIME OPTION MODULE (RTOM)
INPUT/OUTPUT MICRO-PROGRAMMABLE PROCESSOR (IOM)
INPUT/OUTPUT MICRO-PROGRAMMABLE PROCESSOR (IOM)
MEMORY BUS CONTROLLER (MBC)
MEMORY BUS CONTROLLER (MBC)
MEM. BUS NO. 2
MEM. BUS NO. 1
MEMORY
MEMORY
MEMORY

Figure 1-1. SEL 32 Block Diagram

The MBC manages up to 16 overlapped Memory Modules. All modules under an MBC have the same cycle and access time. However, other MBC's may manage up to 16 fully overlapped Memory Modules with different speed characteristics.

Full Memory Module overlap allows MBC memory request to be initiated every 150 nanoseconds. In multiprocessor environments, Memory Modules are accessible between two SEL Buses. This is accomplished by attaching a second MBC to the Memory Bus to be shared, thus you can access a common memory data pool from two processors.

The MBC that contains the timing generator (clock), is referred to as the Master. The Master MBC (in Multiple MBC Configurations) generates the asynchronous timing signals; thereby, requiring the logic to synchronize with the clock.

The MBC receives the address from the SEL Bus and checks it against its own address high and low limits, if the address falls within these address limits, presents it to the addressed Memory Module at a time when the Memory Module is available, by way of the Memory Bus.

During a Write operation the MBC receives the data from the SEL Bus and generates odd parity then places it in the data buffers. The data is then applied to the Data In formatter and placed in the format dictated by the format bits as contained in the address word, then placed on the Memory Bus to be written into Memory.

During a Read operation the MBC receives the data specified by the Memory Address from the Memory Bus and places it in the format specified by the format bits by the Data Out formatter. The formatter checks the Data for correct parity and places it on the SEL Bus if no error is found.

MEMORY PROTECT

The Memory Protect system provides Write Protect for individual memory pages. A memory page consists of 512 words. Up to 256 pages (128K words) are protected at any single point in time. The memory protect register is changed only through the execution of privileged instructions.

INPUT/OUTPUT SYSTEM

Each Input/Output Channel has an I/O Controller which consists of an Input/Output Microprogrammable Processor (IOM) and Device Dependent Interface logic. The IOM is comprised of a SEL Bus Interface and a Microprogrammable Processor (MP). The maximum throughput of a standard Input/Output Channel is 1.2 million bytes per second.

BASIC INTERRUPTS AND TRAPS

The first RTOM in a system provides ten basic interrupts and traps for system operation and six external interrupts. These basic interrupts and traps include: Power Fail-Safe/Auto-Start, System Override, Memory Parity, Attention, Nonpresent Memory, undefined instruction, privilege violation, Call Monitor, Real-Time Clock and Arithmetic Exception. Three of the six external interrupts are used by the Real-Time Monitor software and the other three are available for user applications.

INSTRUCTION SET

The SEL 32 instruction set consists of 152 instructions of which 105 occupy a full memory word and 47 occupy only 16 bits in memory. A summary of these instructions is as follows:

Class	Number
Fixed Point Arithmetic	30
Floating-Point Arithmetic	8
Boolean	17
Load/Store	30
Bit Manipulation	8
Shift	13
Interrupt	5
Compare	11
Branch	10
Register Transfer	11
Input/Output	2
Control	7
Total	152

SECTION II

CENTRAL PROCESSOR

INTRODUCTION

This section describes the SEL 32 Central Processor, Memory, Interrupt System, Input/Output System and all associated instructions. The types of functions performed by the SEL 32 instructions include:

	Function	Instruction Group
1.	Move data between memory and the operating registers.	Load/Store
2.	Alter program sequence, with or without a test on a word value or flag.	Branch
3.	Comparison of the contents of a memory location with a register.	Compare
4.	Test, set or zero a bit in memory or a register.	Bit Manipulation
5.	Perform control functions within the processor.	Control, Interrupt
6.	Perform arithmetic or logical operations.	Arithmetic and Logic
7.	Transfer data to or from peripheral equipment.	I/O

Also included in this section are programming examples which demonstrate how some of these instructions can be used or how they function.

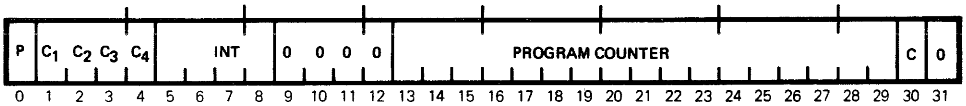
CENTRAL PROCESSOR

The central processing unit consists of a general purpose microprogrammed architecture which includes a register file, scratchpad memory, and proprietary parsing logic.

PROGRAM STATUS WORD

The Program Status Word (PSW) contains all machine conditions that must be preserved prior to context switching. The format of the PSW is shown in figure 2-1.

The Transfer Register to Program Status Word (TRSW) instruction allows a program to transfer the contents of a General Purpose Register to the PSW.



BIT 0 DESIGNATES THE PRIVILEGED STATE BIT.

BITS 1-4 DESIGNATE THE CURRENT CONDITION CODE.

BIT 6 DEFINES THE POSITION OF THE LAST INSTRUCTION EXECUTED.
BIT 6=0 LEFT HALFWORD OR FULLWORD
BIT 6=1 RIGHT HALFWORD

BITS 5, 7, 8 DESIGNATE THE INFORMATION NECESSARY TO ENABLE THE EXECUTION OF LONG INSTRUCTIONS TO BE CONTINUED AFTER AN INTERRUPT' AND MUST NOT BE DISTURBED BY SOFTWARE.

BITS 9-12 UNASSIGNED.

BITS 13-29 CONTAIN THE WORD ADDRESS (PC) OF THE INSTRUCTION CURRENTLY BEING EXECUTED.

BIT 30 DEFINES THE POSITION OF THE CURRENT INSTRUCTION (LEFT OR RIGHT INSTRUCTION).

BIT 30=0 LEFT HALFWORD

BIT 30=1 RIGHT HALFWORD

Figure 2-1. PSW Format

Execution of any Branch or Branch-and-Link instruction replaces the contents of bits 13 through 30 of the PSW with the effective address specified by the instruction. In addition, if the Branch Instruction specifies an indirect branch operation, the contents of bits 1 through 8 of the PSW are replaced by the contents of the corresponding bit positions in the indirect address location. The content of bit 30, in the PSW, is placed in bit 6 at the beginning of every instruction execution.

CONDITION CODES

A four-bit condition code is stored in the PSW on completion of the execution of most instructions. These conditions may be tested to determine the status of the results obtained.

CC1:	Arithmetic Exception
CC2:	Result is greater than zero
CC3:	Result is less than zero
CC4:	Result is equal to zero

The Branch Condition True (BCT), Branch Condition False (BCF) and Branch Function True (BFT) allow testing and branching on the condition codes.

INSTRUCTION MNEMONICS

The SEL 32 instruction mnemonics follow a very simple format. The basic types are:

L	load	or	LM	load masked
ST	store	or	STM	store masked
AD	add			
ADM	add memory to register			
ARM	add register to memory			
SU	subtract			
SUM	subtract memory from register			
MP	multiply			
DV	divide			
ADF SUF MPF DVF	floating-point arithmetic			
B	branch			
AN	AND			
OR	logical OR			
EO	exclusive OR			
C	compare			

These basic mnemonics are then augmented to define the operand data type. The five basic data types are:

B	byte	(8-bits)
H	halfword	(16-bits)
W	word	(32-bits)
D	doubleword	(64-bits)
I	immediate	(16-bits)

(A special set of instructions are provided for bit manipulation.)

Therefore, the resulting instruction mnemonics have the form:

LB	load byte
LMH	load masked halfword
STMW	store masked word
ADI	add immediate to register
SUMD	subtract memory doubleword

No instruction mnemonic is greater than four characters. This simple mnemonics format makes the SEL 32 instruction set easy to learn and use.

A complete summary of the SEL 32 instructions is presented in the appendix of this manual.

ASSEMBLER CODING CONVENTIONS

The basic assembler coding format for memory reference instructions is:

$$\text{XXXX} \quad \begin{Bmatrix} s \\ d \end{Bmatrix} \quad , \quad {}^*m, \quad x$$

which translated to

XXXX	Instruction mnemonic
$\begin{Bmatrix} s \\ d \end{Bmatrix}$	Source or destination General Purpose Register
*	Indirectly (optional)
m	Memory operand
x	Indexed by register number x

Non-memory reference instruction coding is similar to the memory reference format. Table 2-1 lists all codes used in defining the Assembler coding formats.

INSTRUCTION DEFINITION FORMAT

Each instruction definition includes the following information.

Instruction Name The full name of the instruction.

Op Code The four most significant hexadecimal digits of the instruction word are listed. Additional bits in the op code are set when the instruction is coded to address a General Purpose register, indirect addressing, or byte addressing.

Table 2-1. Assembler Coding Symbols

Code	Description
Captial Letters	Instruction Mnemonic
b	Bit number (0 through 31) in a General Purpose Register
c	Bit number (0 through 7) within a Byte
d	Destination General Purpose Register number (0 through 7)
f	Function
m	Operand Memory Address
n	Device Address
s	Source General Purpose Register number (0 through 7)
v	Value for Immediate Operands, number of Shifts, etc.
x	Index Register number 1, 2 or 3. Optional
*	Indirect Addressing. Optional
,	Assembler Syntax

Assembler Coding Format The coding format used by the SEL 32 Macro Assembler. Table 2-1 includes all the abbreviations and symbols used in the operand coding format.

Instruction Definition A definition of the operation performed by executing the instruction.

Summary Expression A symbolic or graphic description of the operation performed by the instruction. Summary expressions use the same abbreviations used in the assembler coding format, table 2-1. In addition, table 2-2 lists the codes and symbols used in the summary expressions.

Condition Codes The coding codes are set based on the results obtained by executing an instruction. The circumstances in which these condition codes are set (i.e. equal to 1) are noted with each instruction.

MEMORY

The basic Memory Module for the SEL 32 is 8,192 words (32,768 bytes). The module is organized as 8,192 words by 36 bits. Each word contains 32 data bits and four parity bits (one parity bit per byte).

From an addressing point of view, the entire memory is a set of contiguous locations whose addresses range from zero to a maximum dependent on the configuration of the particular system.

Memory reference instructions which address a byte in memory do not alter the other three bytes in the memory word containing the specified byte. Memory reference instructions which address a halfword also do not alter the other halfword of the memory location. The exception to this rule is the Add Bit in Memory instruction since the Add operation may propogate a carry to the most significant end of the word containing the specified bit.

Each full word instruction (32-bit) must be stored in memory on a word boundary (i.e., bits 30 and 31 equal to zero). Halfword instructions are stored two to a word. When a halfword is followed by a word instruction, the Assembler positions the instruction in the left half of the word and stores a No Operation (NOP) instruction in the right half of the word. This maintains the word boundary discipline. Memory Boundaries are illustrated in figure 2-2.

Word operands must be stored in memory on a word boundary. The most significant word of a doubleword operand must be stored in a memory location having an even word address with

Table 2-2. Summary Expression Symbols

Code	Interpretation
( )	"The contents of" as in (d) the contents of General Purpose Register number d.
→	Replaces; meaning the data to the left of the symbol replaces the data to the right. For example, (s) → (d), means the contents of GPR number s replaces the contents of GPR number d.
+	Plus indicates Addition.
-	Minus indicates Subtraction.
/	Division
x	Multiplication
v	Logical OR function.
&	Logical AND function.
⊕	Exclusive OR function.
+1	The register number or memory address is incremented by one register number or one memory word.
SE	Sign Extended, see Operand formats.
—	Not function. For example, $(\bar{s})$ is the one's complement of the contents of GPR number s.
Subscripts	Indicates specific bit positions in a 0 through 31 bit word. Also used to indicate a specific Condition Code number as in CC_V means the Condition Code specified by v.

Summary expression examples.

$$(s_{24-31}) \rightarrow (d_{24-31})$$

The contents of bits 24 through 31 of GPR d are replaced with the contents of bits 24 through 31 of GPR s.

$$[\text{zeros}_{0-23}, \text{byte operand}] \rightarrow (d)$$

The byte operand is appended with zeros in positions 0 through 23 and the resulting word replaces the contents of GPR d.

(m), (m+1) is a doubleword effective memory address.

(d), (d+1) is a doubleword even-odd GPR pair.

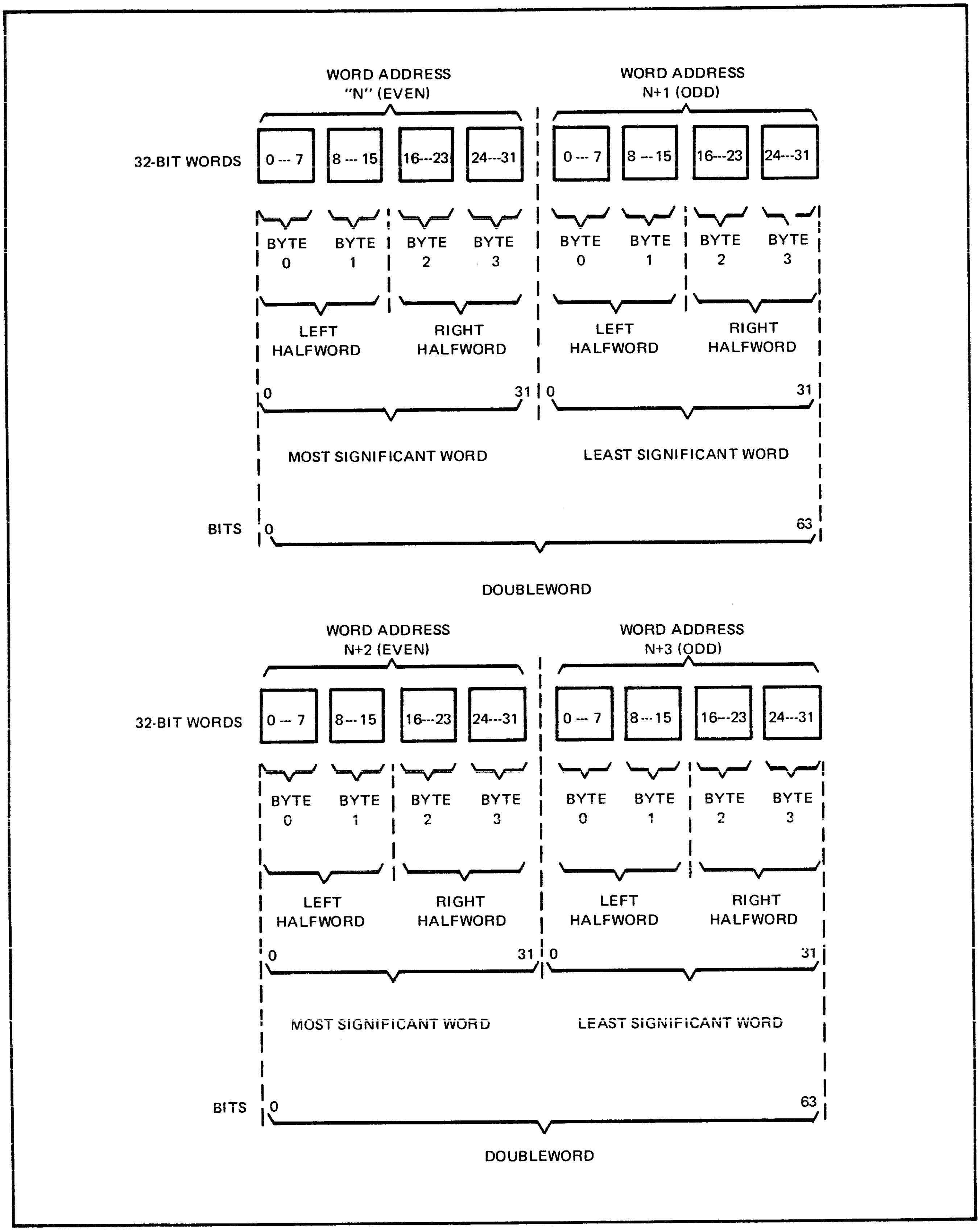


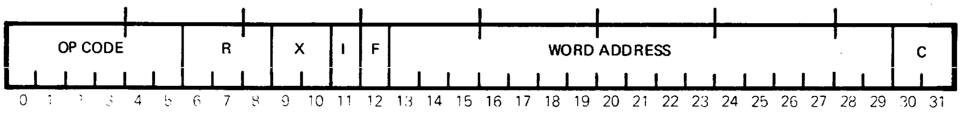
Figure 2-2 Information Boundaries in Memory

the least significant word stored in the next sequentially higher (i.e., odd word) location. Some examples of memory addresses are as follows:

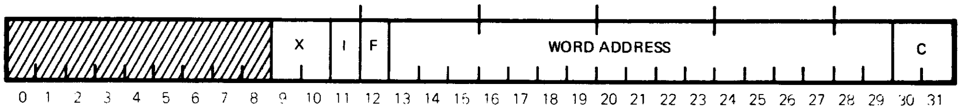
Byte	Halfword	Word	Doubleword
00000	00000	00000	00000
00001			
00002	00002		
00003			
00004	00004	00004	
00005			
00006	00006		
00007			
00008	00008	00008	00008
00009			
0000A	0000A		
0000B			
0000C	0000C	0000C	
0000D			
0000E	0000E		
0000F			
00010	00010	00010	00010

MEMORY REFERENCE INSTRUCTIONS

Bits 9 through 31 have the same format as every memory reference instruction whether the effective address is used for storage or retrieval of an operand, as an indirect address operand, or to alter program flow. The Memory Reference Instruction format is shown below:



Bits 9 and 10 specify the General Purpose Register to be used as an index register, bit 11 is the indirect bit, bits 12 through 31 define the word address and data type. The effective address of the instruction depends on the values of I, X, and bits 12 through 31. If I and X are both zero, bits 12 through 31 address the data type defined by bits 13 through 29.



The format of the F and C bits have been selected so that any selected data type byte, 16-bit halfword, 32-bit fullword, or 64-bit doubleword, can be conveniently indexed by that data type. The possible combinations of F and C bits are as follows:

F	C	Data Type
0	00	32-bit fullword
0	01	16-bit left halfword (bits 0-15)
0	10	64-bit doubleword
0	11	16-bit right halfword (bits 16-31)
1	00	byte 0 (bits 0-7)
1	01	byte 1 (bits 8-15)
1	10	byte 2 (bits 16-23)
1	11	byte 3 (bits 23-31)

Direct Addressing

When X is equal to zero (no indexing) and I is equal to zero (no indirect), the effective memory address is taken directly from bits 13 through 29 of the memory reference instruction. The Store Word instruction is coded:

```
STW      0,0
```

and is assembled as hexadecimal D4000000. When executed, this instruction stores the contents of General Purpose Register 0 directly into memory location zero.

The Store Byte instruction is coded:

```
STB,     0,1
```

and is assembled as hexadecimal D4080001. Note that the F and C fields of the instruction have been altered. When executed, this instruction stores the contents of General Purpose Register 0 directly into memory byte location 1.

Indexed Addressing

Any data type may be indexed by adding a bit at the bit position corresponding to the displacement value for each data type. These are as follows:

Data Type	Bit Position
byte	31
halfword	30
word	29
doubleword	28

If X is non-zero, (specifying indexing) bits 13 through 31 are used to produce a memory address by adding it to the contents of the General Purpose Register specified by X. General Purpose Registers 1, 2 and 3 function as Index registers.

For selective or indexed addressing, the displacement is a two's complement integer contained within one of the General Purpose Registers used for indexing. For word indexing, bit 29 of the Index register is the least significant bit of the address. If bit 29 of General Purpose Register 3 is set to a 1 state, to provide a displacement of 1 word, the Indexed Store instruction is coded:

```
     STW      0,0,3
```

This now stores the contents of GPR 0 in memory indexed by the contents of GPR 3. The instruction would assemble as D4600000. The calculated effective word operand address (after indexing) would be 00004. Therefore, the contents of GPR 0 will be stored in memory location 00004.

Indirect Addressing

If I is equal to zero, addressing is direct, and the address already determined from X and bits 12 through 31 is the effective address used in the execution of the instruction.

If I is equal to one, addressing is indirect, and the processor retrieves another address specified by the operand address. In this new address, bits 9 and 10 select the index register and bit 11 is the indirect bit; bits 12 through 31 specify the effective address as in the memory reference instructions. In order to use the indirect addressing capability in the SEL 32, the instruction would be coded:

```
     STW      0,*0
```

which causes bit 12, the indirect bit, to be set to a one's state. When executed, this instruction stores the contents of GPR 0 in the memory location whose address is stored in memory location zero.

Multi-level indirect addressing can be performed when each new address taken from memory has the indirect bit (bit 11) set to a one. The process of fetching indirect addresses continues until an address has bit 11 equal to zero. This address then is the effective operand address.

Indirect and Indexed Addressing

Indirect addressing can be combined with indexing at any indirect level. An example of indirect addressing with indexing is shown as follows:

```
Location  Machine      Byte
Counter   Instruction  Address   Label   Operation   Operand

                                         PROGRAM
P00000                                   REL
P00000    C9800004               STRT    LI          3,4
P00004    AC90000C     P0000C            LW          1,*LOC1
P00008    3055                           CALM        X'55'
P0000A        0002
P0000C    00100010     P00010    LOC1    ACW         *LOC2
P00010    00700014     P00014    LOC2    ACW         *LOC3,3
P00014    00000000               LOC3    DATAW       0
P00018    0000001C     P0001C            ACW         LOC4
P0001C    0000FFFF               LOC4    DATAW     X'000'0FFFF'
P00020                 P00000            END         STRT
```

The first executable instruction is a Load Immediate (LI) to load a value of 4 into index register number 3. The next instruction to be executed is the Load Word (LW). This instruction directs the machine to load General Purpose Register 1 indirectly using the contents of LOC1 as the operand address. The address in LOC1, however, has the indirect bit on so the machine uses this address to fetch the contents of LOC2. The contents of LOC2 has an indirect bit on, but also points to GPR 3 for indexing. The machine then takes the address contents of LOC2 and adds to it the contents of GPR 3 (which increases the address by four bytes). The resulting address points to LOC4. The address stored in LOC4 has the indirect bit off. The machine then uses the address P0001C stored in LOC4 as the final operand address and loads GPR 1 with the hexadecimal value 0000FFFF.

The ACW statement is a Macro Assembler directive used to generate an address constant. The DATAW is also a Macro Assembler directive and the CALM X'55' is a call to the monitor exit service.

SECTION III

INPUT OUTPUT SYSTEM

INPUT/OUTPUT ORGANIZATION

Input/Output (I/O) Operations consists of transferring blocks of bytes, halfwords, or words, between core memory and peripheral devices. Transfers are possible at rates up to 1.2 million bytes per second, and are performed in an automatic manner which requires minimum Central Processor involvement.

All system components which particpate in the execution of an I/O operation are illustrated in figure 3-1. The peripheral device/s shown may be either data processing - type devices such as disc files, magnetic tape units, line printers, card readers, and card punches; or they may be real-time system devices such as data acquisition subsystems, communications control units, or system control units. Each Input/Output channel has an I/O controller which consists of an Input/Output Microprogrammable Processor (IOM) and Device Dependent Interface logic. The IOM is further defined as consisting of a SEL Bus Interface and a Microprogrammable Processor (MP). Each Input/Output channel performs fully buffered transfers of information concurrently with transfer operations of other Input/Output channels. There may be a total of 16 Input/Output channels, all of which can perform block transfers concurrently at combined rates of up to 26,666,667 bytes per second.

An individual Input/Output controller has the capability of handling up to eight device addresses. Depending upon the characteristics of an Input/Output controller, it may interface to either multiple devices of the same type, or several different type devices. Each peripheral unit attached to the system has a device address. This address includes a 4-bit channel number and a 3-bit unit number. When an Input/Output operation is initiated, the device address is sent from the CPU to alert the proper Input/Output controller and then in turn the proper device that a transfer is desired. Two types of I/O instructions (Command Device and Test Device) are executable.

Transfer of a block of information is initiated by execution of a Command Device instruction in the Central Processor. This instruction, illustrated in figure 3-2, specifies the device, the direction of transfer, and other control parameters required to condition the device to generate or accept data. The control parameters are defined in figure 3-3. The Input/Output controller consisting of an IOM and Device Dependent logic accepts the Command Device from the Central Processor, routes the device control parameters to the device

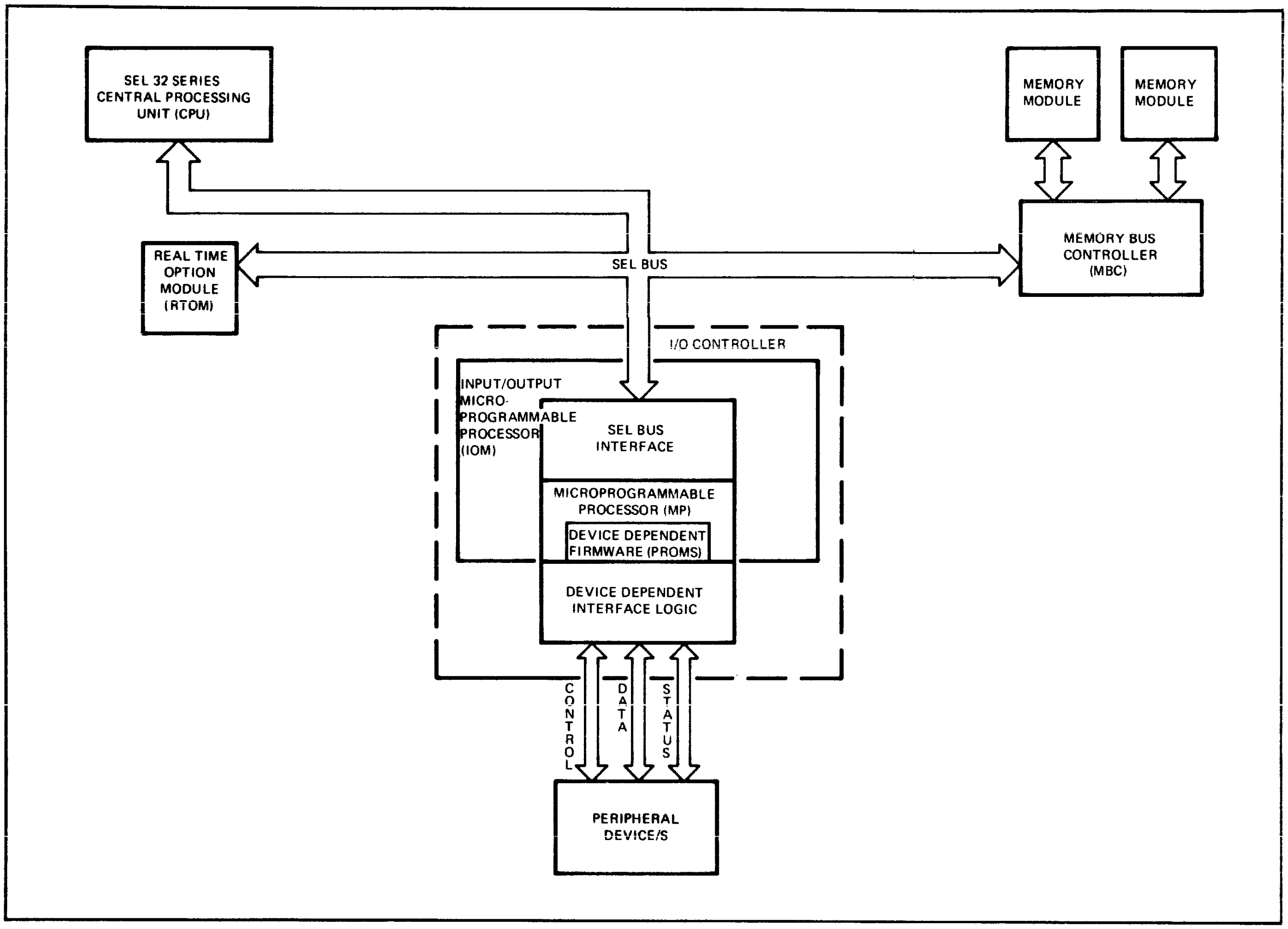
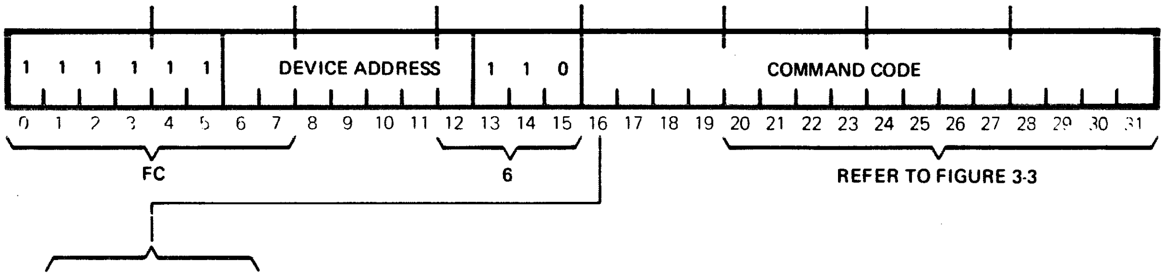

SEL 32 SERIES CENTRAL PROCESSING UNIT (CPU)
MEMORY MODULE
MEMORY MODULE
REAL TIME OPTION MODULE (RTOM)
SEL BUS
MEMORY BUS CONTROLLER (MBC)
I/O CONTROLLER
INPUT/OUTPUT MICRO-PROGRAMMABLE PROCESSOR (IOM)
SEL BUS INTERFACE
MICROPROGRAMMABLE PROCESSOR (MP)
DEVICE DEPENDENT FIRMWARE (PROMS)
DEVICE DEPENDENT INTERFACE LOGIC
CONTROL
DATA
STATUS
PERIPHERAL DEVICE/S

Figure 3-1 SEL 32 Input/Output Organization



BIT 16 = 0   BIT POSITIONS 20 THROUGH 31 OF THE FUNCTION CODE ARE UNIQUE TO THE DEVICE
BIT POSITIONS 18 AND 19 PROVIDE THE FOLLOWING INFORMATION:

BIT 18 = 1 TRANSFER CURRENT WORD ADDRESS
BIT 19 = 1 TERMINATE

BIT 16 = 1   A TRANSFER IS TO BE INITIALIZED AND BITS 18 AND 19 OF THE FUNCTION CODE WILL PROVIDE THE FOLLOWING INFORMATION:

BIT 18 = 1 HARDWARE ADDRESS SOURCE
BIT 19 = 0 OUTPUT TRANSFER
BIT 19 = 1 INPUT TRANSFER

Figure 3-2 Command Device Instruction Format

BIT				CD FUNCTION BIT FORMAT FOR STANDARD DEVICES									
DEVICE		20	21	22	23	24	25	26	27	28	29	30	31
FIXED HEAD DISC	FORM 0	SELECT TRK		SELECT TRK	TRK ADDR 256	TRK ADDR 128	TRK ADDR 64	TRK ADDR 32	TRK ADDR 16	TRK ADDR 8	TRK ADDR 4	TRK ADDR 2	TRK ADDR 1
	FORM 1	SEQ TRK0 HD0	SEQ 1 REV	1 REV	DISC SURF 4	DISC SURF 2	DISC SURF 1	SCT 32	SCT 16	SCT 8	SCT 4	SCT 2	SCT 1
MOVING HEAD DISC	FORM 0	RESTR. SEEK	0	SEEK	0	TRK ADDR 128	TRK ADDR 64	TRK ADDR 32	TRK ADDR 16	TRK ADDR 8	TRK ADDR 4	TRK ADDR 2	TRK ADDR 1
	FORM 1		SEQ 1 REV	1 REV	HD ADDR 16	HD ADDR 8	HD ADDR 4	HD ADDR 2	HD ADDR 1	SCT 8	SCT 4	SCT 2	SCT 1
MAGNETIC TAPE	FORM 0	RWD BKSP REC. BKSP EOF	RWD ERS 4,WRT EOF BKSP EOF	RWD. ADV EOF WRT EOF	0	0	556=1 800=0	0	0	NOTE: ADV REC=ALL ZEROS			
	FORM 1	0	0	0	INTCHG=1 PACK=0	EVEN=1 ODD=0	556=1 800=0	0	0	0	0	0	0
CR/P	FORM 0	HLF ASC=0 BIN=1	HLF ASC=0 BIN=1	0	0	0	0	0	0	NOTE: OFFSET CARD=ALL ZEROS			
	FORM 1	BIN=1	AUTO BIN=1	0	0	0	0	0	0	0	0	0	0
LINE PRINTER	FORM 0	ADV FORM	FRMT 4	FRMT 2	ADV LINE FRMT 1					NOTE: TOF=FRMT 0			
	FORM 1	ADV FORM	FRMT 4	FRMT 2	ADV LINE FRMT 1					NOTE: TOF=FRMT 0			
CARD READER	FORM 0	NOTE: OFFSET CARD=ALL ZEROS (1000 CR ONLY)											
	FORM 1	MODE BINARY	AUTO MODE		NOTE: TRANSLATE=ALL ZEROS								
PTS	FORM 1	IGNORE LDR											
TELE	FORM 1	KEYBRD ECHO											

Figure 3-3 Command Device Function Bit Format For Peripheral Devices

specified in the instruction, and initializes the transfer of a block of data. The Transfer Control Word contains the starting memory address and the number of transfers to be made and is contained in a memory location dedicated to each I/O channel.

TRANSFER CONTROL WORD

The Transfer Control Word contains a 20-bit address which defines the memory location for each transfer. It also contains a positive 12-bit binary Transfer Count (TC). The Transfer Count plus the Format Code (FC) permit transfers of blocks of information having any number of bytes, halfwords, or words up to 4,096. The format of the Transfer Control Word is shown in figure 3-4.

The presence of the Format Code in the Transfer Control Word permits transfers of bytes, halfwords, or words. The Format Code is designed such that when F is equal to one in a given Transfer Control Word, the address is incremented in bit position 31 each time a transfer occurs. Therefore, each transfer is stored in or read from a consecutive byte in memory in the order:

```
           Word N                         Word N+1

---Byte 0,Byte 1,Byte 2,Byte 3   Byte 0,Byte 1, Byte 2,Byte 3---
```

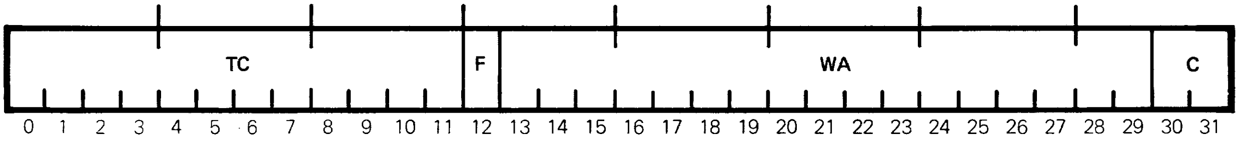
The proper binary value of Format Code for accessing consecutive halfwords in memory is F equal to 0, C equal to Y1, where Y equal to zero designates left halfword and Y equal to one designates right halfword. With this value of Format Code, the address is incremented in bit position 30 each time a transfer is made. This results in the desired accessing of consecutive halfwords.

The proper value of Format Code for consecutive word accessing is FC equal to 000. When this value is present in a given Transfer Control Word, the I/O controller increments the Transfer Control Word in bit position 29 each time a transfer occurs.

The Format Code values discussed above are summarized in Table 3-1.

Each time the address is incremented, the transfer count is decremented. Therefore, the block length is always defined by the number of memory accesses and not by the number of words transferred.

The Test Device (TD) instruction is used to acquire status information from the Input/Output controller and the associated device(s). Three levels of the TD instruction (8000, 4000, and 2000) may be used to acquire this information. The status information is in the form of four condition code



BITS 0-11 DESIGNATE THE NUMBER OF TRANSFERS TO BE MADE BETWEEN MEMORY AND THE DEVICE CONTROLLER CHANNEL.

BITS 12,30,31 SPECIFY THE FORMAT CODE FOR EACH TRANSFER (SEE TABLE 3-1).

BITS 13-29 DESIGNATE THE MEMORY LOCATION FOR EACH TRANSFER.

Figure 3-4. Transfer Control Word Format

Table 3-1. Transfer Control Word Format Code

Information Format	FC
Byte	1XX
Halfword	0Y1
Word	000

XX = Byte number
Y = 0 designates left halfword
Y = 1 designates right halfword

bits for each level of test. The TD instruction does not initiate any action in the device. The TD 8000 instruction is used by the CPU to test the general status of the addressed device and associated I/O controller. The TD 4000 instruction is used by the CPU to allow further definition of the errors indicated in the TD 8000. The TD 2000 instruction is used by the CPU to obtain 16-bits of status information from the device/controller. This instruction causes the addressed I/O controller to transfer a 16-bit Status Word to the memory address specified. The 16-bit Status Word may be placed in memory in either the right or left half word position depending on bits 30 and 31 of the address. Figure 3-5 provides a breakdown of the Test Device instruction format. Figure 3-6 provides the status information returned on executing TD 2000 instruction to the standard peripheral devices.

INPUT/OUTPUT CONTROLLER

Each Input/Output controller consists of an Input/Output Microprogrammable Processor (IOM) and Device Dependent Interface logic. The Microprogrammable Processor (MP) and the Device Dependent Interface logic is customized for each device. The firmware (PROM's) for a given Input/Output controller is a set of PROM's that plug into the controller board. The information contained within the PROM's is device dependent.

This design technique provides extreme flexibility for custom designed interfaces since the basic MP and SEL Bus interface is also available as a General Purpose I/O controller (GPIO). All that is needed to convert the GPIO controller into a special purpose I/O controller is the device dependent interface logic and the firmware microprogram.

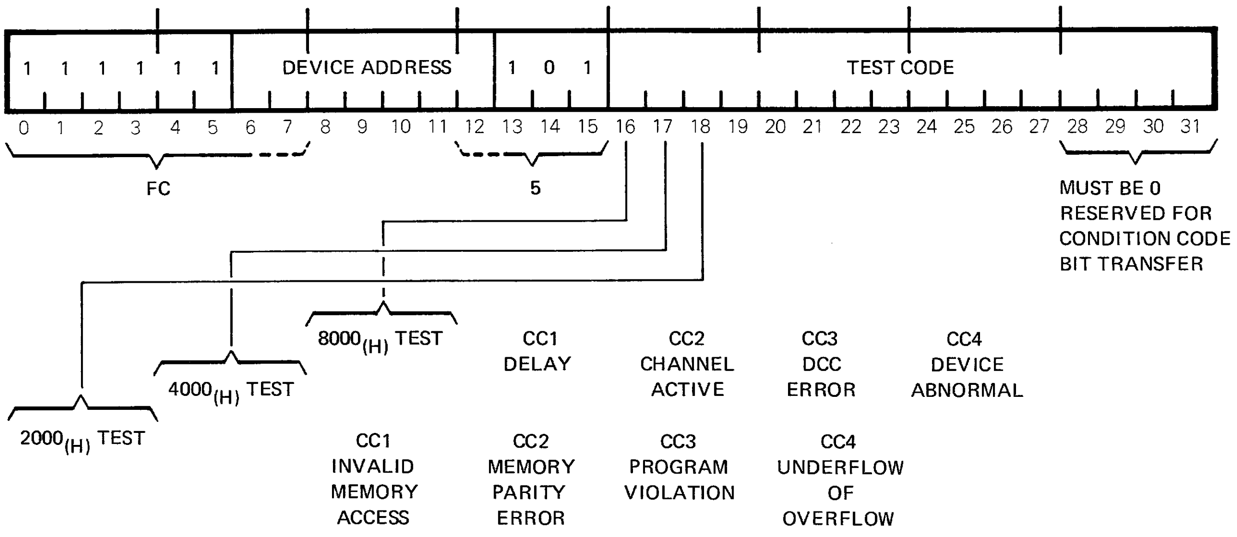
The maximum throughput of an Input/Output controller is 1.2 million bytes per second.

There are two types of Input/Output controllers:

1. Multiple Device Controller (MBC)

2. Multiple Controller Controller (MCC)

The MDC controls like devices, such as four magnetic tapes. The MCC emulates multiple controllers such as the TLC Input/Output controller that controls a teletype, card reader and a printer. MCC Input/Output controller's are multiplexed controllers handling more than one device simultaneously directly to memory. The Asynchronous Data Set interface (ADS) is an example of a Multiplexed Controller. The ADS handles four half or full duplex lines directly to memory on a message basis. Four memory input buffers and four output buffers can be active at one time.



CAUSES A TRANSFER TO THE MEMORY LOCATION, SPECIFIED BY THE NORMAL TRANSFER INTERRUPT DEDICATED LOCATION; OF 16 BITS OF CONTROLLER STATUS INFORMATION. THESE 16 BITS OF STATUS INFORMATION ARE UNIQUE TO EACH DEVICE WHICH USES THE $2000_{(H)}$ LEVEL TEST.

CC2 = 0 STATUS TRANSFER WAS PERFORMED
CC2 = 1 STATUS TRANSFER WAS NOT PERFORMED
CC4 = 1 CONTROLLER IS ABSENT OR POWERED OFF

Figure 3-5. Test Device Instruction Format

UPPER HW	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LOWER HW	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LINE PRINTER	0	PROG VIO	DEV INOP	0	0	0	0	DATA LOST	0	BOF	0	0	0	DEV BUSY	0	0
MAG TAPE	0	PROG VIO	DEV INOP	VRC	0	REW IN PROG	CRC LRC	DATA LOST	0	EOT	BOT	EOF	0	DEV BUSY	FILE PROT VIO	ODD REC LGT
MOVING HEAD DISC	0	PROG VIO	DEV INOP	CKSM ERR	0	FILE UN-SAFE	SEEK IN PROG	DATA LOST	0	SECTOR BIT 8	SECTOR BIT 4	SECTOR BIT 2	0	SECTOR BIT 1	FILE PROT VIO	TRK ADDR ERR
FIXED HEAD DISC	0	PROG VIO	DEV INOP	CKSM ERR	0	0 SECTOR BIT 32	TRK SLCT IN PROG SECTOR BIT 16	DATA LOST	0	SECTOR BIT 8	SECTOR BIT 4	SECTOR BIT 2	0	SECTOR BIT 1	FILE PROT VIO	0
CARD READER/ PUNCH	0	PROG VIO	DEV INOP	ECHO CK.	0	PHOTO DIO ERR	0	DATA LOST	0	0	0	0	0	0	0	0

Figure 3-6. Test Device 2000 Status Information

SEL BUS INTERFACE

The Input/Output Controller SEL Bus interface contains the registers and SEL Bus drivers for a full 32-bit data transfer. The main function of this logic is to receive and drive communications on the SEL Bus. All the interface control logic, including controller address recognition, interrupt polling, and data transfer to and from the SEL Bus are included in the interface.

The Bus priority logic is controlled by the interface control logic. It polls for the SEL Bus, determines when it wins the poll, and then drives the transfer on the Bus. Priorities are set through physical switches in the Input/Output controller.

Responses to SEL Bus Transfers

An Input/Output controller will respond to all Bus transfers that it receives. It has three immediate responses:

1. Retry

2. Busy

3. Transfer Acknowledge

The sending Bus device, can determine the status of its transfer to the Input/Output controller by monitoring these lines. A Retry answer means that the Input/Output controller of the MCC type is temporarily busy. A Busy return means to set the busy Condition Code (CC) bit in the software instruction and proceed with the next instruction. An Input/Ouput controller of the MDC type would generate such a return. A Transfer Acknowledge indicates that the transfer was accepted and is being processed. If no answer is present in the Bus cycle following the transfer, a non-present Input/Output controller was addressed.

IOM DATA STRUCTURE

The IOM microcommands are 32-bits wide. Each microcommand allows parallel operation of data transfer and control. The IOM has the capability of addressing up to 4096 words of 32-bit control memory. Figure 3-7 provides a block diagram for the IOM. The IOM Data Structure provides for the transfer of data, arithmetic and logical manipulation of data, storing of device and controller status, decode of commands, and data buffering. It has a full 16-bit Arithmetic/Logic Unit (ALU).

Two 16 by 16-bit word register groups, RA and RB are available as working read/write memory. The output for each register pair is the input to the Arithmetic/Logic Unit.

The destination address and the most significant 16 bits of the SEL Data Bus are directed to the RA register group. The program counter and the ALU output are also directed to the RA register group. The least significant 16 bits of the SEL

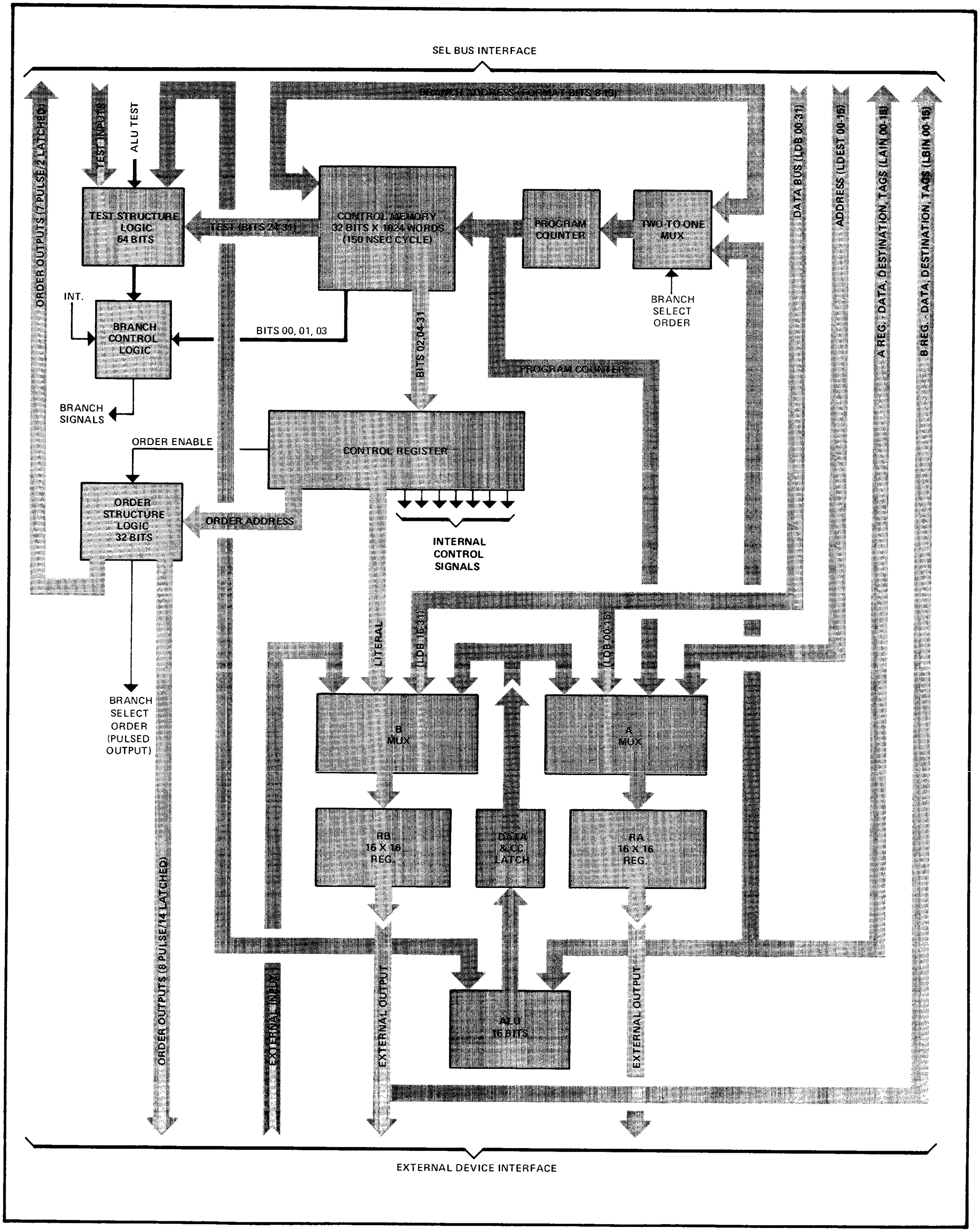


Figure 3-7. Block Diagram - I/O Microprogrammable Processor

data bus and 16 bits of data from the peripheral devices are directed to the RB register group. The ALU output and a 16-bit literal data from the control register are also input to the RB register group.

ARITHMETIC/ LOGIC UNIT

The Data Structure includes a full 16-bit Arithmetic/Logic unit with inputs from RA and RB. The ALU is equipped with a 3-bit status register which contains previous carry, all zeros condition, and the most significant bit.

DATA STRUCTURE CONTROL

A 32-bit by 1024 word microprogrammed control memory and a 48-bit test structure (32 implemented) control the flow of data and commands between the SEL Bus and peripheral devices.

TEST STRUCTURE

The IOM Test Structure is used with the Wait and Conditional Branch operations to control the sequencing and timing of instructions.

INTERRUPTS

The IOM has a single Master Interrupt line. For device controllers requiring more interrupts, the necessary Interrupt Mask register and Priority Decode logic is included in the Device Interface logic.

SECTION IV

PRIORITY INTERRUPTS

INTERRUPT SYSTEM

Interrupt requests are signals to the SEL 32 Central Processor that allow external events to alter the system's currently programmed course of action in such a manner that it may return to the programmed course as if no disturbance had occurred. Interrupt requests may be generated from a variety of sources, typical of which are:

a. An External Event
b. Completion of an Input/Output Block Transfer
c. Depressing the ATTENTION key on the Turnkey Panel

When an Interrupt Request is received, the SEL 32 internal logic examines the request and initiates a new course of action at the appropriate point in its cycle of operation. The current program status of the interrupted program is saved in the PSW (either undisturbed or restorable as explained below), and an indirect branch is forced via a memory location in memory assigned to the interrupt. The PSW is stored at this location, and the next sequential instruction in memory is executed. The programmed instruction sequence is then executed to fulfill the purpose for which the interrupt occurred. At the completion of this instruction sequence the interrupt handler transfers the PSW contents for the interrupted program back to the PSW register causing the interrupted program to continue as if nothing has happened.

PRIORITY INTERRUPT LEVELS

One hundred twenty eight levels of interrupts are available, with true priority nesting within the levels. Each level is assigned a unique memory location which contains the address of an interrupt service program, interrupt handler to service the device, or function initiating the interrupt. The highest priority interrupt has the lowest memory location (see table 4.1 "Priority Interrupt Dedicated Memory Locations").

The levels are fully nested since an interrupt of higher priority can interrupt a lower level but a lower level cannot interrupt a higher level.

Table 4-1. Priority Interrupt Dedicated Memory Locations

Priority Level (H)	Dedicated Address (H)	Function
00*	0F0	Power Fail Safe - Auto Start Interrupt
00*	0F4	Power Fail Safe - Auto Start Trap
01*	0F8	System Override Interrupt
01*	0FC	System Override Trap
02	100	Input/Output Channel 0 Transfer Interrupt
03	104	Input/Output Channel 4 Transfer Interrupt
04	108	Input/Output Channel 8 Transfer Interrupt
05	10C	Input/Output Channel C Transfer Interrupt
06	110	Input/Output Channel 10 Transfer Interrupt
07	114	Input/Output Channel 18 Transfer Interrupt
08	118	Input/Output Channel 20 Transfer Interrupt
09	11C	Input/Output Channel 30 Transfer Interrupt
0A	120	Input/Output Channel 40 Transfer Interrupt
0B	124	Input/Output Channel 50 Transfer Interrupt
0C	128	Input/Output Channel 60 Transfer Interrupt
0D	12C	Input/Output Channel 70 Transfer Interrupt
0E	130	Input/Output Channel 78 Transfer Interrupt
0F	134	Input/Output Channel 7A Transfer Interrupt
10	138	Input/Output Channel 7C Transfer Interrupt
11	13C	Input/Output Channel 7E Transfer Interrupt
12*	0E8	Memory Parity Trap
13*	0EC	Attention Interrupt
14	140	Input/Output Channel 0 Service Interrupt
15	144	Input/Output Channel 4 Service Interrupt
16	148	Input/Output Channel 8 Service Interrupt
17	14C	Input/Output Channel C Service Interrupt
18	150	Input/Output Channel 10 Service Interrupt
19	154	Input/Output Channel 18 Service Interrupt
1A	158	Input/Output Channel 20 Service Interrupt
1B	15C	Input/Output Channel 30 Service Interrupt
1C	160	Input/Output Channel 40 Service Interrupt
1D	164	Input/Output Channel 50 Service Interrupt
1E	168	Input/Output Channel 60 Service Interrupt
1F	16C	Input/Output Channel 70 Service Interrupt
20	170	Input/Output Channel 78 Service Interrupt
21	174	Input/Output Channel 7A Service Interrupt
22	178	Input/Output Channel 7C Service Interrupt
23	17C	Input/Output Channel 7E Service Interrupt (K/P)
24*	190	Nonpresent Memory Trap
25*	194	Undefined Instruction Trap
26*	198	Privilege Violation Trap
27*	19C	Call Monitor Interrupt
28*	1A0	Real-Time Clock Interrupt

*Present in first RTOM.

Table 4-1. Priority Interrupt Dedicated Memory Locations (Cont'd)

Priority Level (H)	Dedicated Address (H)	Function
29*	1A4	Arithmetic Exception Interrupt
2A*	1A8	External Interrupt
2B*	1AC	External Interrupt
2C*	1B0	External Interrupt
2D*	1B4	External Interrupt
2E*	1B8	External Interrupt
2F*	1BC	External Interrupt (Last P.I. in Standard Module)
30	1C0	External Interrupt
↓	↓	↓ ↓
7F	2FC	External Interrupt

*Present in first RTOM.

SECTION V

COMPUTER INSTRUCTIONS

INTRODUCTION

This section contains the description for each of the computer instructions. The following paragraphs list the standard information given with each instruction.

Mnemonic

A two-to-four letter symbolic representation of the instruction name accepted by the assembler program.

Assembly Coding Conventions

A symbolic representation of the assembly coding format.

Instruction Name

A title that indicates the function performed by the instruction.

Operation Code

The Operation Code for each instruction is given in left justified hexadecimal format. This format is presented in a 16-bit skeleton form and takes into consideration the Augmenting Code and also the format bit used with byte-oriented instructions.

Format

A 16-bit or 32-bit machine language representation of the instruction. The operation code and all other fixed bits are given in their binary value.

Definition

The function performed by the instruction is described following the instruction format. All registers or memory locations which are modified are defined. Special considerations are treated as notes following the basic functional description.

Summary Expression

This expression supplements the verbal description of most instructions by showing symbolically the function performed by execution of the instruction. The symbols are defined in table 5-1.

Condition Code Results

An interpretation of the resulting 4-bit condition code contained in the Program Status Word register. This code defines the result of the operation.

Timing

The number of computer cycle times required to access the instruction and perform the execution. All instructions are executed in an integral number of cycle times.

Examples

Included in the examples with many of the instructions are Memory and Register contents before and after execution.

Table 5-1. Symbol Definitions

Symbol	Definition
>	Greater Than
<	Smaller Than
+	Algebraic Addition
-	Algebraic Subtraction
x	(or no symbol) Algebraic Multiplication
/	Algebraic Division
&	Logical AND
B_{m-n}	Bits m through n of a Computer Word
B_n	Bit n of a computer word where B_0 always refers to the most significant bit of a computer word. The letter n is also used to indicate scaling; e.g., 1_{15} indicates a one scaled at bit position 15.
CC_n	Condition Code bit n
:	Comparison Symbol
.	Concatenation Sign, e.g., R, R+1 indicates a double word consisting of (R) and (R+1), where R must be an even numbered register.
EA	Effective Address of an operand or instruction stored in memory.
EBA	Effective Byte Address.
EBL	Eight-Bit Location in memory specified by the EBA.
EDA	Effective Doubleword Address.
EDL	Sixty-four bit location in memory consisting of an even numbered word location and the next higher word location, specified by the EDA.
EHA	Effective Halfword Address.
EHL	Sixteen-bit location in memory specified by the EHA.
EWA	Effective Word Address.
EWL	Thirty-two bit location in memory specified by the EWA.
I	Indirect Address bit
IW	Instruction Word
( )	Contents of
⊕	Exclusive OR
PSWR	Program Status Word Register
R	General Register 0-7 (RO-R7)
R_{m-n}	Bits m through n of General Register R
R_n	Bit n of General Register R
SBL	Specified Bit Location with a byte. Used as a subscript to designate that the bit location is specified in the instruction word.

Table 5-1. Symbol Definitions (Cont'd)

Symbol	Definition	
SCC	Sets Condition Code bits	
SE	Used as a subscript to denote a sign extended halfword.	
v	Logical OR	
X	Index Register:	
	X Value	GP Register Used for Indexing
	00	None
	01	R1
	10	R2
	11	R3
-Y	Two's complement of Y	
Y	One's completion of Y, logical NOT function.	

LOAD/STORE INSTRUCTIONS

General Description

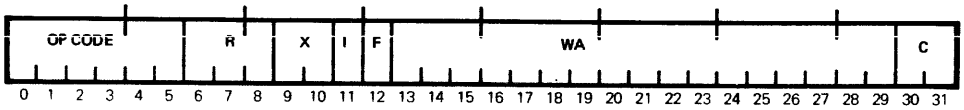
The Load/Store Instruction group is used to manipulate date between Memory and General Purpose Registers. In general, load instructions transfer operands from specified memory locations to General Purpose Registers; store instructions transfer data contained in General Purpose Registers to specified memory locations. Provisions have also been made to Mask or Clear the contents of General Purpose Registers, memory bytes, halfwords, words, or doublewords during instruction execution.

Instruction Formats

The Load/Store Instructions use the following three instruction formats:

Memory Reference

The format for most memory reference instructions is defined below. These instructions contain two addresses: a register number R, and a memory address having a 20-bit format.



Bits 0-5 define the Operation Code.

Bits 6-8 designate a General Purpose Register address (0 through 7).

Bits 9-10 designate one of three general purpose registers to be used as an index register.

X = 00 designates that no indexing operation is to be performed.
X = 01 designates the use of R1 for indexing.
X = 10 designates the use of R2 for indexing.
X = 11 designates the use of R3 for indexing.

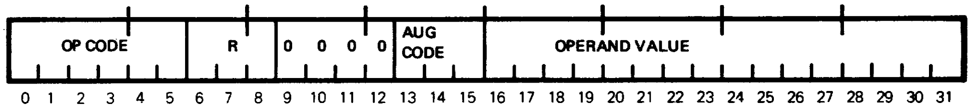
Bit 11 designates if an indirect addressing operation is to be performed.

I = 0 designates that no indirect addressing operation is to be performed.
I = 1 designates that an indirect addressing operation is to be performed.

Bits 12-31 specify the address of the operand when X and I fields are equal to zero.

Immediate

In immediate operand instructions, the right halfword of the instruction contains the 16-bit operand value. The format for these instructions is given below.



Bits 0-5 define the Operation Code.

Bits 6-8 designate a General Purpose Register address (0 through 7).

Bits 9-12 Unassigned.

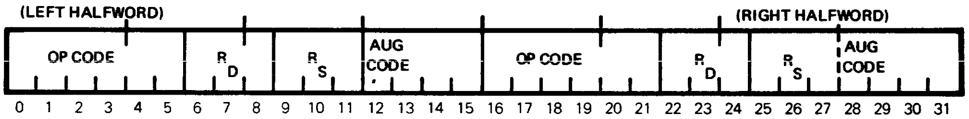
Bits 13-15 define Augmenting Operation Code.

Bits 16-31 contain the 16-bit operand value.

Arithmetic operands are assumed to be represented in two's complement format with the sign in bit 16.

Inter-Register

Inter-register instructions are halfowrd instructions and as such may be stored in either the left or right half of a memory word. The format for inter-register instructions is given below.



Left Halfword	Right Halfword	
Bits 0-5	16-21	define the Operation Code.
Bits 6-8	22-24	designate the register to contain the result of the operation.
Bits 9-11	25-27	designate the register which contains the source operand.
Bits 12-15	28-31	define the Augmenting Operation Code.

Condition Code Utilization

A Condition Code is set during most Load instructions to indicate if the operand being transferred was greater than, less than, or equal to zero. Arithmetic exceptions are also reflected by the Condition Code results. All Store instructions leave the Condition Code unchanged.

Memory To Register Transfers

Figure 5-1 depicts the SEL 32 positioning of information for transfer from memory to any General Purpose Register.

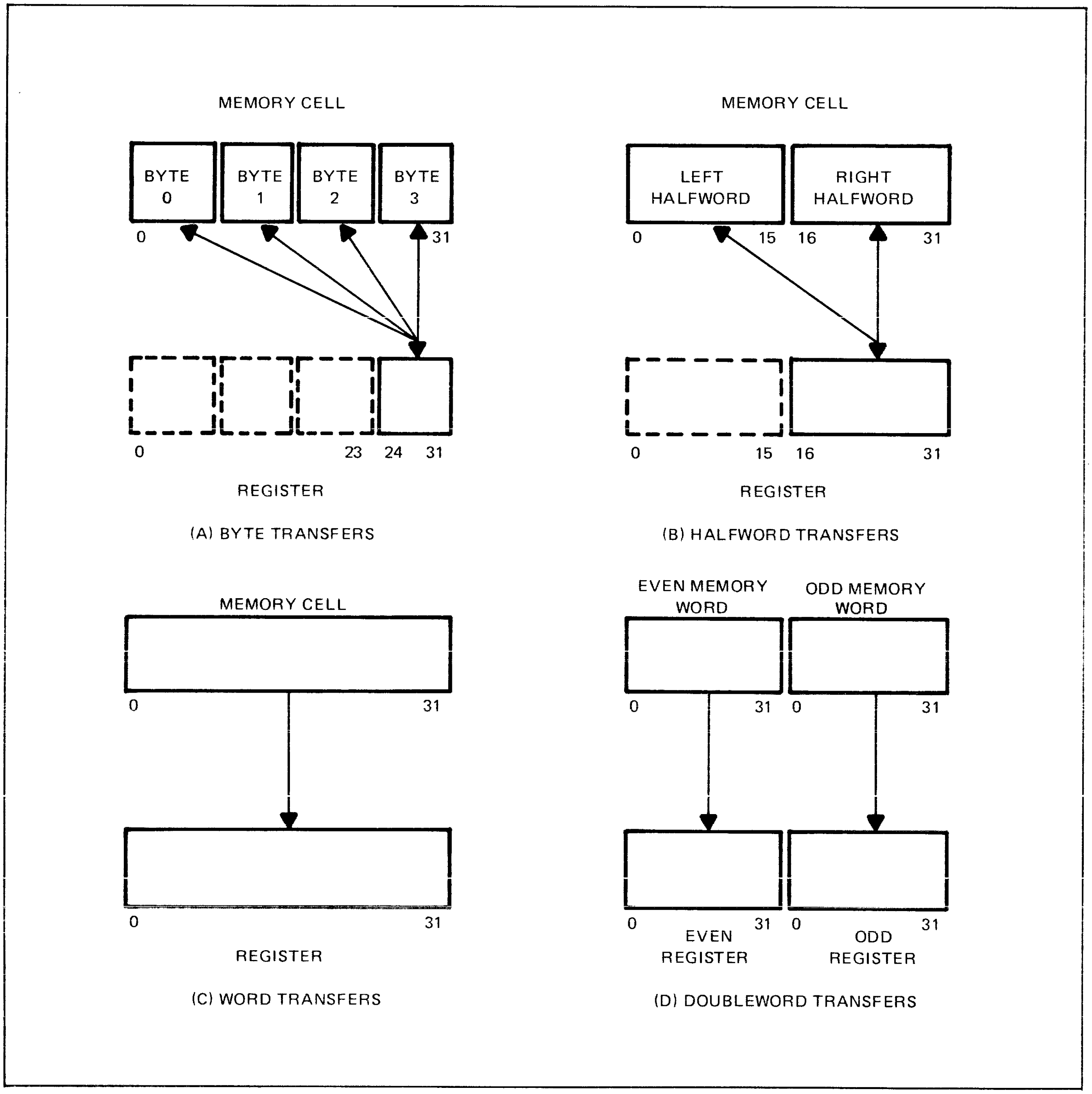
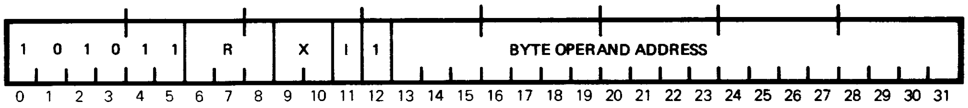


Figure 5-1. Positioning of Information Transferred Between Memory and Registers

LOAD BYTE

AC08



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and transferred to bit positions 24 through 31 of the General Purpose Register (GPR), specified by R. Bit positions 0 through 23 of the GPR specified by R are cleared to zeros.

SUMMARY EXPRESSION

(EBL) → R_{24-31}

0 → R_{0-23}

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{0-31} is greater than zero
CC3: Always zero
CC4: R_{0-31} is equal to zero

EXAMPLE 1

Memory Location: 01000
Hex Instruction: AC 88 11 01 (R=1, X=1=0)

	PSWR	GPR1	Memory Byte 01101
Before Execution	00001000	517CD092	B6
After Execution	20001004	000000B6	B6

Note

The contents from memory byte 01101 are transferred to bits 24-31 of GPR1, Bits 0-23 of GPR1 are cleared. CC2 is set since the contents from GPR1 are greater than zero.

EXAMPLE 2

Memory Location: 01000
Hex Instruction: AD 28 14 00 (R=2, X=1, 1=0)

	PSWR	GPR1	GPR2	Memory Byte 01603
Before Execution	10001000	00000203	12345678	A1
After Execution	20001004	00000203	000000A1	A1

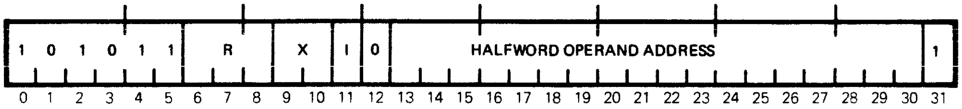
Note

The contents from memory byte 01603 are transferred to bits 24-31 of GPR2. Bits 0-23 of GPR2 are cleared, and CC2 are set.

LH
d,*m,x

LOAD HALFWORD

AC00



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed and the sign bit (bit 16) extended left 16 bit positions to form a word. This resulting word is transferred to the General Purpose Register (GPR) specified by R.

SUMMARY EXPRESSION

$(EHL)_{SE} \rightarrow R$

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 00408
Hex Instruction: AE 00 05 03 (R=4, X=1=0)

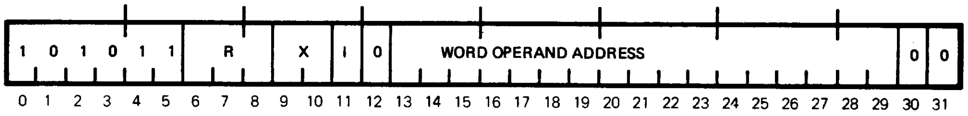
Before Execution	PSWR	GPR4	Memory Halfword 00502
	10000408	5C00D34A	930C
After Execution	PSWR	GPR4	Memory Halfword 00502
	1000040C	FFFF930C	930C

Note

The contents from memory halfword 00502 are transferred to bits 16-31 of GPR4. Bits 0-15 of GPR4 are set by the sign extension and CC3 is set.

LOAD WORD

ACOO



DEFINITION

The word in memory specified by the Effective Word Address (EWA) is accessed and transferred to the General Purpose Register (GPR) specified by R.

SUMMARY EXPRESSION

(EWL) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 02390
Hex Instruction: AF 80 27 A4 (R=7, X=1=0)

Before Execution	PSWR 00002390	GPR7 0056879A	Memory Word 027A4 4D61A28C
After Execution	PSWR 20002394	GPR7 4D61A28C	Memory Word 027A4 4D61A28C

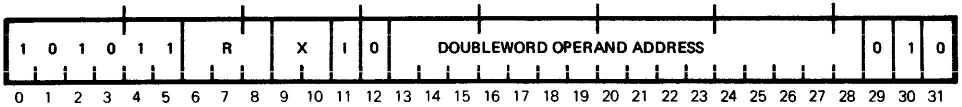
Note

The contents from memory word 027A4 are transferred to GPR7. CC2 is set since the contents of GPR7 is greater than zero.

LD
d,*m,x

LOAD DOUBLEWORD

ACOO



DEFINITION The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and transferred to the General Purpose Register (GPR) specified by R and R+1. R+1 is GPR one greater than specified by R. The least significant memory word is accessed first and transferred to the GPR specified by R+1. The most significant memory word is accessed last and transferred to the GPR specified by R.

NOTE The GPR specified by R must have an even address.

SUMMARY EXPRESSION (EWL+1) → R+1

(EWL) → R

CONDITION CODE RESULTS
CC1: Always zero
CC2: (R,R+1) is greater than zero
CC3: (R,R+1) is less than zero
CC4: (R,R+1) is equal to zero

EXAMPLE
Memory Location: 281C4
Hex Instruction: AF 02 8B 7A (R=6,X=I=0)

Before Execution

PSWR	GPR6	GPR7	Memory Word 28B78
400281C4	03F609C3	39BB510E	F05B169A

Memory Word 28B7C
137F8CA2

After Execution

PSWR	GPR6	GPR7	Memory Word 28B78
100281C8	F05B169A	137F8CA2	F05B169A

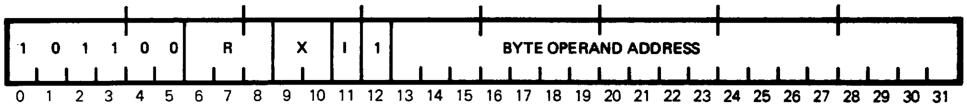
Memory Word 28B7C
137F8CA2

Note The contents from memory word 28B78 are transferred to GOR6 and the contents from memory word 28B7C to GPR7. CC3 is set.

LMB
d,*m,x

LOAD MASKED BYTE

B008



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and masked (Logical AND Function) with the least significant byte (bit 24 through bit 31) of the mask register R4. The result of the mask operation is transferred to bit positions 24 through 31 of the General Purpose Register (GPR) specified by R. Bit positions 0 through 23 of the GPR specified by R are cleared to zeros.

SUMMARY EXPRESSION

$(EBL)\&(R4_{24-31}) \rightarrow R_{24-31})$

$0 \rightarrow R_{0-23}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{0-31} is greater than zero
CC3: Always zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 00900
Hex Instruction: B0 88 00 A3 (R=1, X=I=0)

Before Execution	PSWR 00000900	GPR1 AA3689B0	GPR4 000000F0	Memory Byte 000A3 29
After Execution	PSWR 20000904	GPR1 00000020	GPR4 000000F0	Memory Byte 000A3 29

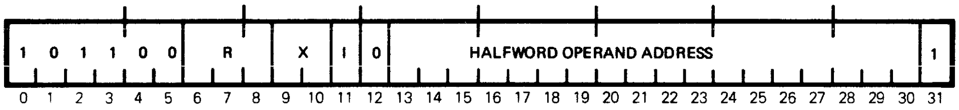
Note

The contents from memory byte 000A3 are logically ANDed with the right most byte of GPR4 and the result is transferred to bits 24-31 of GPR1. Bits 0-23 of GPR1 are cleared and CC2 is set.

LMH
d,*m,x

LOAD MASKED HALFWORD

B000



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed and the sign bit (bit 16) extended 16 bit positions to the left to form a word. This word is then masked (Logical AND Function) with the contents of the mask register R4. The result word is transferred to the General Purpose Register (GPR) specified by R.

SUMMARY EXPRESSION

$(EHL)_{se}\&(R4) \rightarrow R$

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 00300
Hex Instruction: B2 80 03 A1 (R=5, X=I=0)

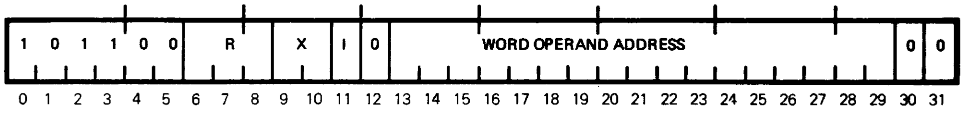
Before Execution	PSWR	GPR4	GPR5	Memory Halfword 003A0
	08000300	0FF00FF0	C427B319	A58D
After Execution	PSWR	GPR4	GPR5	Memory Halfword 003A0
	20000304	0FF00FF0	0FF00580	A58D

Note

The contents from memory halfword 003A0 are accessed, the sign is extended 16 bit positions, the result is logically ANDed with the contents of GPR4, and the final result is transferred to GPR5. CC2 is set, as the result is greater than zero.

LOAD MASKED WORD

B000



DEFINITION The word in memory specified by the Effective Word Address (EWA) is accessed and masked (Logical AND Function) with the contents of the mask register R4. The result word is transferred to ghe General Purpose Register (GPR) specified by R.

SUMMARY EXPRESSION (EWL)&(R4) → R

CONDITION CODE RESULTS
CC1: Always zero
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE
Memory Location: 00F00
Hex Instruction: B3 80 0F FC (R=7, X=I=0)

	PSWR	GPR4	GPR7	Memory Word 00FFC
Before Execution	00000F00	FF00007C	12345678	8923F8E8
	PSWR	GPR4	GPR7	Memory Word 00FFC
After Execution	10000F04	FF00007C	89000068	8923F8E8

Note The contents from memory word 00FFC are ANDed with the contents from GPR4. The result is transferred to GPR7, and CC3 is set.

LMD
d,*m,x

LOAD MASKED DOUBLEWORD

B000

1	0	1	1	0	0	R	X	I	0	DOUBLEWORD OPERAND ADDRESS	0	1	0
0	1	2	3	4	5	6 7 8	9 10	11	12	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28	29	30	31

DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and the contents of each word masked (Logical AND Function) with the contents of the mask register R4. The least significant memory word is masked first. The resulting masked doubleword is transferred to the General Purpose Register (GPR) specified by R and R+1. R+1 is GPR one greater than specified by R.

SUMMARY EXPRESSION

(EWL+1)&(R4) → R+1

(EWL)&(R4) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R,R+1) is greater than zero
CC3: (R,R+1) is less than zero
CC4: (R,R+1) is equal to zero

EXAMPLE

Memory Location: 00200
Hex Instruction: B3 00 02 F2 (R=6, X=I=0)

Before Execution

PSWR	GPR4	GPR6	GPR7
00000200	3F3F3F3F	12345678	9ABCDEF0

Memory Word 002F0	Memory Word 002F4
AE69D10C	63B208F0

After Execution

PSWR	GPR4	GPR6	GPR7
20000204	3F3F3F3F	2E29110C	23320830

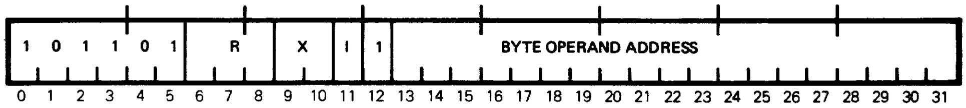
Memory Word 002F0	Memory Word 002F4
AE69D10C	63B208F0

Note

The contents from memory word 02F4 are ANDed with the contents of GPR4 and the result is transferred to GPR6. CC2 is set as the result is greater than zero.

LOAD NEGATIVE BYTE

B408



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and 24 zeros appended to the most significant end to form a word. The two's complement of this word is then taken and transferred to the General Purpose Register (GPR) specified by R.

SUMMARY EXPRESSION

$-[0_{0-23}, (EBL)] \rightarrow R$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 0D000
Hex Instruction: B4 88 D1 02 (R=1, X=I=0)

Before Execution	PSWR	GPR1	Memory Byte 0D102
	0000D000	00000000	3A
After Execution	PSWR	GPR1	Memory Byte 0D102
	1000D004	FFFFFFC6	3A

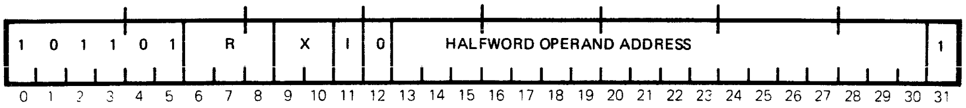
Note

The contents from memory byte 0D102 are prefixed with 24 zeros to form a word; the result is negated and transferred to GPR1. CC3 is set to indicate a value less than zero.

LNH
d,*m,x

LOAD NEGATIVE HALFWORD

B400



DEFINITION The halfword in memory specified by the Effective Halfword Address (EHA) is accessed and the sign bit (bit 16) extended 16 bit positions to the left to form a word. The two's complement of this word is then taken and transferred to the General Purpose Register (GPR) specified by R.

SUMMARY EXPRESSION $-[(EHL)_{SE}] \rightarrow R$

CONDITION CODE RESULTS
CC1: Always zero
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE
Memory Location: 08000
Hex Instruction: B6 00 84 03 (R=4, X=I=0)

Before Execution	PSWR	GPR4	Memory Halfword 08402
	40008000	12345678	960C

After Execution	PSWR	GPR4	Memory Halfword 08402
	20008004	000069F4	960C

Note The contents from memory halfword 08402 are sign extended and negated. The result is transferred to GPR4, and CC2 is set.

LNW
d,*m,x

LOAD NEGATIVE WORD

B400

1	0	1	1	0	1	R	X	I	0	WORD OPERAND ADDRESS	0	0
0	1	2	3	4	5	6 7 8	9 10	11	12	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29	30	31

DEFINITION The word in memory, specified by the Effective Word Address (EWA), is accessed and its two's complement taken and transferred to the General Purpose Register (GPR) specified by R.

SUMMARY EXPRESSION -(EWL) → R

CONDITION CODE RESULTS

CC1: Arithmetic Exception
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 00500
Hex Instruction: B6 80 06 C8 (R=5, X=I=0)

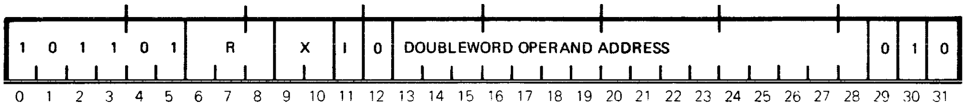
Before Execution	PSWR 08000500	GPR5 00000000	Memory Word 006C8 185E0D76
After Execution	PSWR 10000504	GPR5 E7A1F28A	Memory Word 006C8 185E0D76

Note The contents from memory word 006C8 are negated and transferred to GPR5, CC3 is set.

LND
d,*m,x

LOAD NEGATIVE DOUBLEWORD

B400



DEFINITION

The doubleword in memory, specified by the Effective Double Word Address (EDA), is accessed and its two's complement formed. The least significant memory word is complement first and the result transferred to ghe General Purpose Register (GPR) specified by R+1. R+1 is GPR one greater than specified by R. The most significant by R.

SUMMARY EXPRESSION

-(EDL)→R,R+1

CONDITION CODE RESULTS

CC1: Arithmetic Exception
CC2: (R,R+1) is greater than zero
CC3: (R,R+1) is less than zero
CC4: (R,R+1) is equal to zero

EXAMPLE

Memory Location: 02344
Hex Instruction: B5 00 24 A2 (R=2, X=I=0)

Before Execution

PSWR	GPR2	GPR3
00002344	01234567	89ABCDEF

Memory Word 024A0	Memory Word 024A4
00000000	00000001

After Execution

PSWR	GPR2	GPR3
10002348	FFFFFFFF	FFFFFFFF

Memory Word 024A0	Memory Word 024A4
00000000	00000001

Note

The doubleword obtained from the contents of memory words 024A0 and 024A4 is negated, and the result is transferred to GPR2 and GPR3. CC3 is set.

LOAD IMMEDIATE

C800

1 1 0 0 1 0	R	0 0 0	0	0 0 0	IMMEDIATE OPERAND
0 1 2 3 4 5	6 7 8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

DEFINITION The halfword immediate operand, contained in the Instruction Word (IW), is sign-extended (bit 16 extended 16 positions to the left) to form a word. This word is transferred to the General Purpose Register (GPR) specified by R.

SUMMARY EXPRESSION $(IW_{16-31})SE \rightarrow R$

CONDITION CODE RESULTS
CC1: Always zero
CC2: (R_{0-31}) is greater than zero
CC3: (R_{0-31}) is less than zero
CC4: (R_{0-31}) is equal to zero

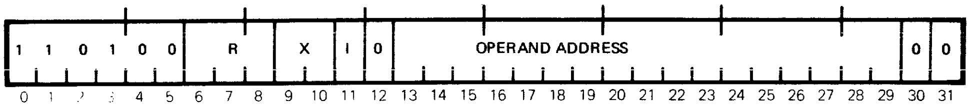
EXAMPLE
Memory Location: 0630C
Hex Instruction: C8 80 F0 B5 (R=1)

Before Execution	PSWR 0000630C	GPR1 12345678
After Execution	PSWR 10006310	GPR1 FFFFF0B5

Note The halfword operand is sign-extended and the result transferred to GPR1. CC3 is set.

LOAD EFFECTIVE ADDRESS

D000



DEFINITION

The effective address (bit 12 through bit 31) of the LEA instruction is generated in the same manner as in all other memory reference instructions and then is transferred to bit positions 12 through 31 of the General Purpose Register (GPR) specified by R.

Notes

1. If I=X=0, the entire 32-bit instruction word is transferred to the GPR specified by R.

2. If I=0 and X≠0, bit positions 0 through 11 of the GPR specified by R will contain the sum of bit positions 0 through 11 of the instruction word and bit positions 0 through 11 of the index register specified by X.

3. If I=1, bit positions 0 through 11 of the GPR specified by R will contain the sum of bit positions 0 through 11 of the last word of the indirect chain and bit positions 0 through 11 of the index register specified (if any) in the last word of the indirect chain.

4. In cases 2 and 3 above, an additional bit may be added to bit position 11 of the GPR specified by R as a result of overflow in the sum of the address and the index values.

SUMMARY EXPRESSION

EA → R_{12-31}

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE 1

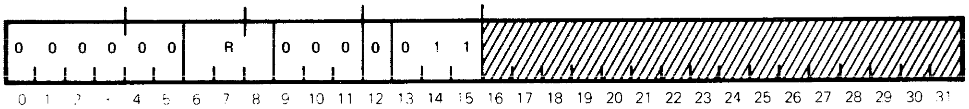
Memory Location: 1000
Hex Instruction: DO 804000 (R=1, X=I=0)

Before Execution	PSWR 08001000	GPR1 00000000	Memory Word 4000 AC881203
After Execution	08001004	DO 804000	AC881203

EXAMPLE 2	Memory Location:	1000	
	Hex Instruction:	D0 904000 (R=I=1, X=0)	
Before Execution	PSWR	GPR1	Memory Word 4000
	08001000	01000100	ACA81203 (R=X=1, I=0)
After Execution	08001004	ADA81303	ACA81203

LOAD CONTROL SWITCHES

0003



DEFINITION

The contents of Control Switches (CS) 0 through 12 are transferred to bit positions 0 through 12 of the General Purpose Register (GPR) specified by R. Bit positions 13 through 31 of the GPR specified by R are cleared to zeros.

SUMMARY EXPRESSION

$(CS_{0-12}) \rightarrow R_{0-12}$

$0 \rightarrow R_{13-31}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R_{0-31}) is greater than zero
CC3: (R_{0-31}) is less than zero
CC4: (R_{0-31}) is equal to zero

EXAMPLE

Memory Location: 06002
Hex Instruction: 03 83 (R=7)

Before Execution	PSWR 00006002	GPR7 FFFFFFFF	Control Switches 0, 6 set
After Execution	PSWR 10006004	GPR7 82000000	

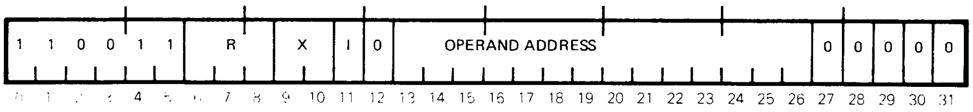
Note

Bit positions 0 and 6 of GPR7 are set and all other bits are cleared. Condition code three (CC3) is set.

LF
d,*m,x

LOAD FILE

CC00



DEFINITION

This instruction is used to load from one to eight General Purpose Registers (GPR). The word in memory, specified by the Effective Word Address (EWA) contained in the Instruction Word (IW), is accessed and transferrred to the GPR specified by R. Next, the EQA and the GPR address are incremented. The next sequential memory word is then transferred to the next sequential GPR. This successive transfer process is continued until GPR7 is loaded from memory.

NOTE

The EWA must be specified such that, when incremented, no carry will be propagated from bit position 27. Therefore, if all eight registers are to be loaded, bit positions 27 through 29 must be equal to zero initially.

SUMMARY EXPRESSION

(EWL) → R

(EWL)+1 → R+1
.   .
.   .
.   .
(EWL+N) → R7

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 00300
Hex Instruction: CE 00 02 00 (R=4, X=1=0)

Before Execution

PSWR	GPR4	GPR5	GPR6	GPR7
08000300	00000000	00000000	00000000	00000000

Memory Word 00200	Memory Word 00204	Memory Word 00208
00000001	00000002	00000003

Memory Word 0020C
00000004

After Execution

PSWR	GPR4	GPR5	GPR6	GPR7
08000304	00000001	00000002	00000003	00000004

Memory Word 00200	Memory Word 00204	Memory Word 00208
00000001	00000002	00000003

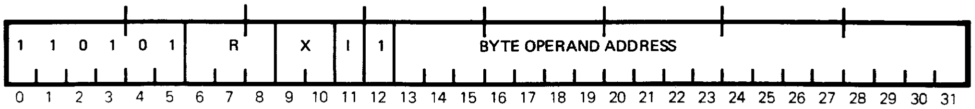
Memory Word 0020C
00000004

Note The contents for memory word 00200 are transferred to GPR4, memory word 00204 to GPR5, memory word 00208 to GPR6, and memory word 0020C to GPR7.

STB
s,*m,x

STORE BYTE

D408



DEFINITION

The least significant byte (bit 24 through bit 31) of the General Purpose Register (GPR) specified by R is transferred to the memory byte location specified by the Effective Byte Address (EBA) contained in the Instruction Word. The other three bytes of the memory word containing the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

$(R_{24-31}) \rightarrow$ EBL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 03708
Hex Instruction: D4 88 3A 13 (R=1, X=I=0)

Before Execution	PSWR 10003708	GPR1 01020304	Memory Byte 03A13 78
After Execution	PSWR 1000370C	GPR1 01020304	Memory Byte 03A13 04

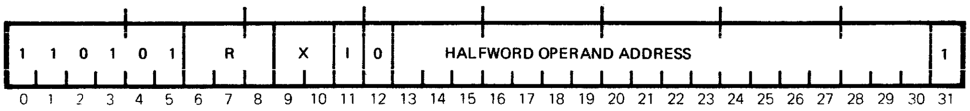
Note

The contents from bits 24-31 of GPR1 are transferred to memory byte 03A13.

STH
s,*m,x

STORE HALFWORD

D400



DEFINITION

The least significant halfword (bit 16 through bit 31) of the General Purpose Register (GPR) specified by R is transferred to the memory halfword location specified by the Effective Halfword Address (EHA) contained in the instruction word. The other halfword of the memory word containing the halfword specified by the EHA remains unchanged.

SUMMARY EXPRESSION

$(R_{16-31}) \rightarrow \mathrm{EHL}$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 082A4
Hex Instruction: D6 00 83 13 (R=4, X=I=0)

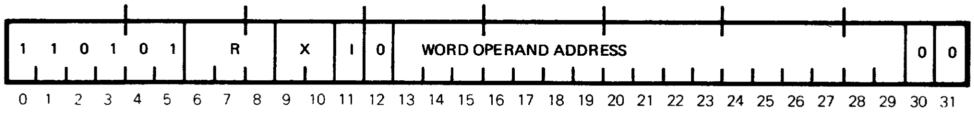
Before Execution	PSWR 000082A4	GPR4 01020304	Memory Halfword 08312 A49C
After Execution	PSWR 000082A8	GPR4 01020304	Memory Halfword 08312 0304

Note

The contents from the right halfword of GPR4 are transferred to memory halfword 08312.

STORE WORD

D400



DEFINITION

The word located in the General Purpose Register (GPR) specified by R is transferred to the memory word location specified by the Effective Word Address contiained in the instruction.

SUMMARY EXPRESSION

(R) → EWL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 03904
Hex Instruction: D7 00 3B 3C (R=6, X=I=0)

Before Execution	PSWR 10003904	GPR6 0485A276	Memory Word 03B3C 00000000
After Execution	PSWR 10003908	GPR6 0485A276	Memory Word 03B3C 0485A276

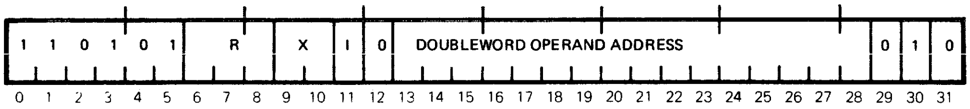
Note

The contents from GPR6 are transferred to memory word 03B3C.

STD
s,*m,x

STORE DOUBLEWORD

D400



DEFINITION

The doubleword located in the General Purpose Register (GPR), specified by R and R+1 (R+1 is GPR one greater than specified by R), is transferred to the memory doubleword location specified by the Effective Doubleword Address (EDA). The word contained in the GPR specified by R+1 is transferred to the least significant word of the doubleword memory location first.

SUMMARY EXPRESSION

(R+1) → EWL+1

(R) → EWL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 0596C
Hex Instruction: D7 00 5C 4A (R=6, X=I=0)

Before Execution

PSWR	GPR6	GPR7
2000596C	E24675C2	5923F8E8

Memory Word 05C48	Memory Word 05C4C
0A400729	8104A253

After Execution

PSWR	GPR6	GPR7
20005970	E24675C2	5923F8E8

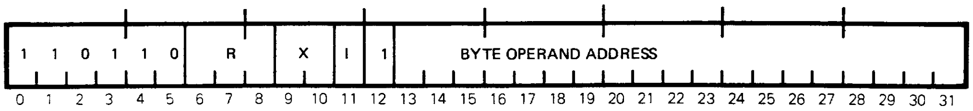
Memory Word 05C48	Memory Word 05C4C
E24675C2	5923F8E8

Note

The contents from GPR6 are transferred to memory word 05C48 and the contents from GPR7 to memory word 05C4C.

STORED MASKED BYTE

D808



DEFINITION

The least significant byte (bit 24 through bit 31) of the General Purpose Register (GPR) specified by R is masked (Logical AND Function) with the least significant byte of the mask register R4. The result byte is transferred to the memory byte location specified by the Effective Byte Address (EBA) contained in the instruction word. The other three bytes of the memory word containing the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

$(R_{24-31})\&(R4_{24-31}) \rightarrow$ EBL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 01D80
Hex Instruction: D8 08 1E 91 (R=0, X=I=0)

Before Execution	PSWR 10001D80	GPR0 AC089417	GPR4 0000FFFC	Memory Byte 01E91 94
After Execution	PSWR 10001D84	GPR0 AC089417	GPR4 0000FFFC	Memory Byte 01E91 14

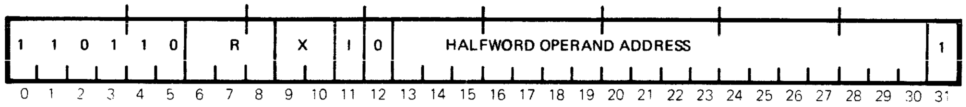
Note

The right-most byte of GPR0 is ANDed with the right-most byte of GPR4. The result is transferred to memory byte 01E91.

STMH
s,*m,x

STORE MASKED HALFWORD

D800



DEFINITION

The least significant halfword (bit 16 through bit 31) of the General Purpose Register (GPR) specified by R is masked (Logical AND Function) with the least significant halfword of the mask register R4. The result halfword is transferred to the memory halfword location specified by the Effective Halfword Address (EHA) contained in the instruction word. The other halfword of the memory word containing the halfword specified by the EHA remains unchanged.

SUMMARY EXPRESSION

$(R_{16-31})\&(R4_{16-31}) \rightarrow$ EHL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 01000
Hex Instruction: DA 80 11 AF (R=5, X=I=0)

Before Execution	PSWR 20001000	GPR4 00003FFC	GPR5 716A58AB	Memory Halfword 011AD 0000
After Execution	PSWR 20001004	GPR4 00003FFC	GPR5 716A58AB	Memory Halfword 011AD 18A8

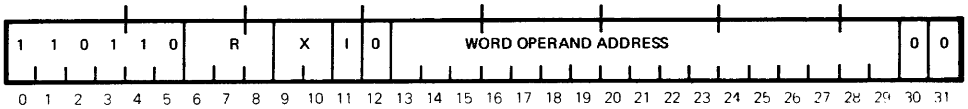
Note

The right-most halfword of GPR5 is ANDed with the right-most halfword of GPR4, and the result is transferred to memory halfword 011AD.

STMW
s,*m,x

STORE MASKED WORD

D800



DEFINITION

The word located in the General Purpose Register (GPR) specified by R is masked (Logical AND Function) with the contents of the mask register R4. The result word is transferred to the memory word location specified by the effective word address.

SUMMARY EXPRESSION

(R)&(R4) → EWL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 04000
Hex Instruction: DB 00 43 7C (R=6, X=I=0)

	PSWR	GPR4	GPR6	Memory Word 0437C
Before Execution	08004000	00FF00FF	718C3594	12345678
After Execution	08004004	00FF00FF	718C3594	008C0094

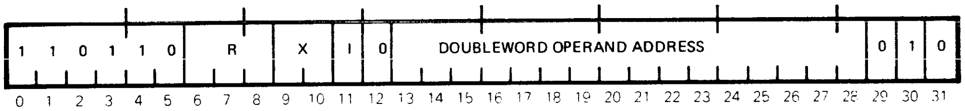
Note

The contents from GPR6 are ANDed with the contents from GPR4. The result is transferred to memory word 0437C.

STMD
s,*m,x

STORE MASKED DOUBLEWORD

D800



DEFINITION

Each word of the doubleword located in the General Purpose Register (GPR) specified by R and R+1 is masked (Logical AND Function) with the contents of the mask register R4. R+1 is GPR one greater than specified by R. The resulting doubleword is transferred to the memory doubleword location specified by the Effective Doubleword Address (EDA) contained in the instruction.

SUMMARY EXPRESSION

(R+1)&(R4) → EWL+1

(R)&(R4) → EWL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 0A498
Hex Instruction: DB 00 A6 52 (R=6, X=I=0)

Before Execution

PSWR	GPR4	GPR6	GPR7
1000A498	0007FFFC	AC88A819	988B1407

Memory Word 0A650	Memory Word 0A654
51CD092	AE69D10C

After Execution

PSWR	GPR4	GPR6	GPR7
1000A49C	0007FFFC	AC88A819	988B1407

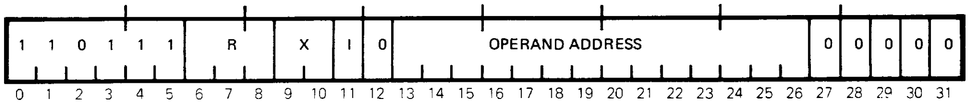
Memory Word 0A650	Memory Word 0A654
0000A818	00031404

Note

The contents from GPR6 and ANDed with the contents from GPR4 and the result transferred to memory word 0A650. The contents from GPR7 are ANDed with the contents from GPR4 and the result transferred to memory word 0A654.

STORE FILE

DC00



DEFINITION

This instruction is used to transfer the contents of from one to eight General Purpose Registers (GPR) to the specified memory locations. The contents of the GPR specified by R are transferred to the memory location specified by the Effective Word Address (EWA). The next sequential GPR is then transferred to the next sequential memory location. This successive transfer process is continued until GPR7 is loaded into memory.

NOTE

The EWA must be specified such that, when incremented, no carry will be propagated from bit position 27. Therefore, if all eight General Purpose Registers are transferred, bit positions 27 through 29 must be equal to zero initially.

SUMMARY EXPRESSION

(R) → EWL

(R+1) → EWL+1
. .
. .
. .
(R7) → EWL+N

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 02000
Hex Instruction: DE 00 21 00 (R=4, X=I=0)

Before Execution

PSWR	GPR4	GPR5	GPR6	GPR7
40002000	11111111	22222222	33333333	44444444

Memory Word 02100	Memory Word 02104
00210000	00210400
Memory Word 02108	**Memory Word 0210C**
00210800	00210C00

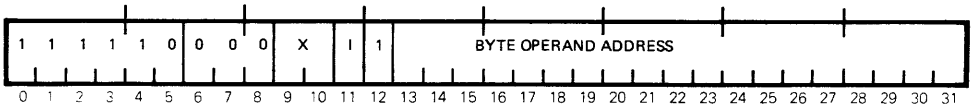
After Execution	PSWR	GPR4	GPR5	GPR6	GPR7
	40002004	11111111	22222222	33333333	44444444

Memory Word 02100	Memory Word 02104
11111111	22222222
Memory Word 02108	Memory Word 0210C
33333333	44444444

Note The contents from GPR4 are transferred to memory word 02100, that of GPR5 to 02104, that of GPR6 to 02108, and that of GPR7 to 0210C.

ZERO MEMORY BYTE

F808



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is cleared to zero. The other three bytes of the memory word containing the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

0 → EBL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 00308
Hex Instruction: F8 08 04 9F

	PSWR	Memory Byte 0049F
Before Execution	10000308	6C
After Execution	1000030C	00

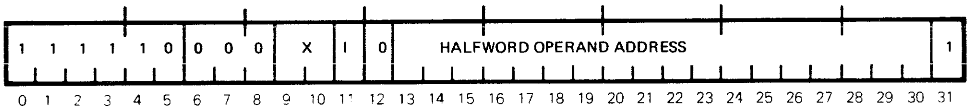
Note

The contents from memory byte 0049F are cleared to zero.

ZMH
*m,x

ZERO MEMORY HALFWORD

F800



DEFINITION The halfword in memory specified by the Effective Halfword Address (EHA) is cleared to zero. The remaining halfword containing the 16-bit location in memory specified by EHA remains unchanged.

SUMMARY EXPRESSION 0 → EHL

CONDITION CODE RESULTS
CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE
Memory Location: 2895C
Hex Instruction: F8 00 2A 42 7 (X=I=0)

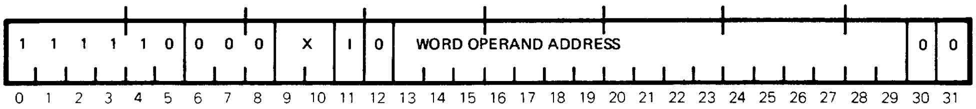
Before Execution	PSWR 0802895C	Memory Halfword 2A426 9AE3
After Execution	PSWR 08028960	Memory Halfword 2A426 0000

Note The contents from memory halfword 2A426 are cleared to zero.

ZMW
*m,x

ZERO MEMORY WORD

F800



DEFINITION The word in memory specified by the Effective Word Address (EWA) is cleared to zero.

SUMMARY EXPRESSION 0 → EWL

CONDITION CODE RESULTS
CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE
Memory Location: 05A14
Hex Instruction: F8 00 5F 90 (X=I=0)

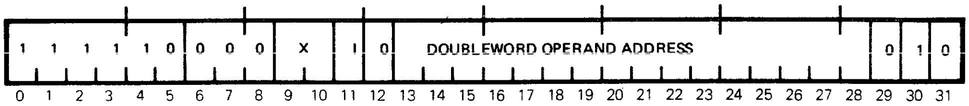
Before Execution	PSWR 00005A14	Memory Word 05F90 12345678
After Execution	PSWR 00005A18	Memory Word 05F90 00000000

Note The contents from memory word 05F90 are cleared to zero.

ZMD
*m,x

ZERO MEMORY DOUBLEWORD

F800



DEFINITION The doubleword in memory specified by the Effective Doubleword Address (EDA) is cleared to zero.

SUMMARY EXPRESSION 0 → EWL

0 → EWL+1

CONDITION CODE RESULTS
CC1: No change
CC2: No change
CC3: No change
CC4: No change

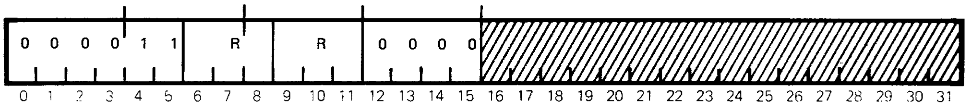
EXAMPLE Memory Location: 15B3C
Hex Instruction: F8 01 5D 6A

Before Execution	PSWR 10015B3C	Memory Word 15D68 617E853C	Memory Word 15D6C A2976283
After Execution	PSWR 10015B40	Memory Word 15D68 00000000	Memory Word 15D6C 00000000

Note The contents from memory words 15D68 and 15D6C are both cleared to zero.

ZERO REGISTER

0C00



DEFINITION The word located in the General Purpose Register (GPR), specified by R (bit 6 through bit 8) is logically exclusive ORed with the word located in the GPR specified by R (bit 9 through bit 11) resulting in zero. This result is then transferred to the GPR specified by R. The contents of the two R fields must specify the same GPR.

SUMMARY EXPRESSION (R)+(R) → R

CONDITION CODE RESULTS
CC1: Always 0
CC2: Always 0
CC3: Always 0
CC4: Always 1

EXAMPLE
Memory Location: 309A6
Hex Instruction: 3C 90 (R=1)

	PSWR	GPR1
Before Execution	100309A6	8495A6B7
After Execution	080309A8	00000000

Note The contents from GPR1 are cleared to zero, and CC4 is set.

BRANCH INSTRUCTIONS

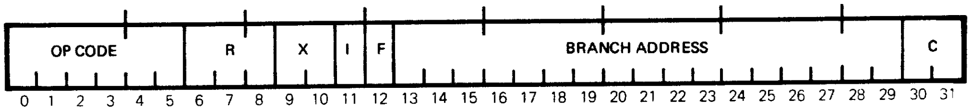
General Description

Branching Instructions provide the capability of testing for certain conditions and branching to another address if these conditions are found to be as specified by the instruction. This allows for referencing of other subroutines, repeating segments of programs, or returning to the next instruction within a sequence.

Instruction Format

The Branch Instruction group uses the following instruction format.

Memory Reference



Bits 0-5 define the Operation Code.
Bits 6-8 varies in usage as follows:

Instruction	Contents/Usage
BU, BFT	000
BCT, BCF	D field
BIB, BIH, BIW, BID	Register Number
BL	001
BRI	010

Bits 9-10 designate one of three Index Registers.
Bit 11 indicates if an indirect addressing operation is to be performed.
Bit 12 is zero.
Bits 13-30 specifies the branch address when X and I fields are zero.
Bit 31 is zero.

Condition Code Utilization

Condition code results during branching operations are unique in that they reflect the state of the indirect bit of the instruction and also the state of bits 1, 2, 3 and 4 indirect address obtained from the specified memory location.

BRANCH Programming

The usual procedure for calling a subroutine is to execute a BL whose effective address is the starting location of the routine. Since PC + 1 is saved in GPR 0, a subsequent return can be made to the location following the BL simply by executing a TRSW 0. Not that the PSW including PC + 1 word is saved in GPR 0. Hence, the subroutine can be re-entrant (pure), i.e. memory is not modified by the act of calling it. If we wish to use GPR 0 in the subroutine, we can store the return address in a convenient location in memory, say location B, with an STW 0, B and then return with a BU *B.

Consider a move subroutine that we wish to use to move fifty words beginning at TAB. The routine begins at MOVE, whose address is stored in C.MOVE. Our main program would contain this:

```
BL       *C.MOVE

...                         ; Return here
```

We use GPR 1 as an index register for counting through the table and GPR 5 to output the data. The starting address of the table is in TAB 1. The subroutine might look something like this:

```
MOVE   LW    1,  CNT       Set up GPR 1 with length of table
       LW    5,  TAB, 1    Send word into GPR 5
       STW   5,  TAB1, 1   Store word in new buffer
       SUI   1,  4         Decrement count by one word addr.
       BCT   2,  MOVE+1W   Is job done
       TRSW  0

CN T   DATAB 50
```

Argument Passing

Suppose we have an arithmetic subroutine that operates on arguments in GPR 5 and GPR 6, leaving the result in GPR 6. The subroutine call looks like this:

```
BL       SQRT      Call with arguments in GPR 5 and GPR 6

. . .
```

and subroutine looks like this:

```
SQRT .                         Arithmetic operations
     .
     .
     TRSW 0                    Return to Call + 1 word
```

In the preceding example, the calling program would have to load the general purpose registers before calling the subroutine. It is often convenient for the program to supply the arguments (or the addresses of the locations that contain them) along with the call and have the subroutine take care of the data transfers. With this method, the program gives the arguments in the two memory locations following the BL.

```
     BL        SQRT
     ...                      Argument 1
     ...                      Argument 2
     ...                      Return here with result in GPR 6
```

The return is made to the location following the second argument with the result in GPR 6.

```
SQRT      TRR       0,1
          LD        6,0,1     Pick up Arguments 1 and 2
          . .
          . .
          ADI       0,8       Increment return address by
                              2 words
          TRSW      0         Return to Call + 3 words
```

An alternate method which would allow up to six arguments to be passed per instruction would be to utilize the LF, load file, instruction as follows:

```
SQRT      TRR       0,1
          LF        2,0,1     Pick up Arguments 1 thru 6
          . .
          . .
          ADI       0,24      Increment return address by
                              6 words
          TRSW      0         Return to Call +7 words
```

The next method passes an address list instead of arguments following the BL; otherwise, it is identical to the method described above.

```
          BL        SQRT
          . .                 Address of Argument 1
          . .                 Address of Argument 2
          .
          .
          .
SQRT      TRR       0,1
          LW        6,*0,1    Pick up Argument 1
          ADI       1,4
          LW        7,*0,1    Pick up Argument 1
          .
          .
          ADI       0,8       Add 2 words to PC
          TRSW      0         Return to Call +3 words
```

The next method is the same as the previous example except that argument 1 is a table and the result replaces the second argument in memory:

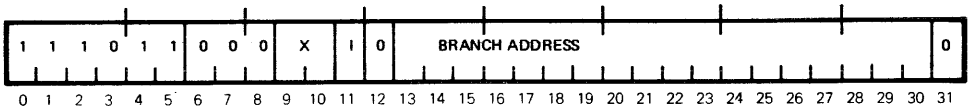
```
          BL        SQRT
          ...                 Address of Argument 1
          ...                 Address of Argument 2 and result
          .
          .
          .

SQRT      TRR       0,3       Pick up base address of table,
                              Argument 1
          TRR       0,1
          ABR       29,1
          LW        6,*0,1    Pick up Argument 2
          .
          .
          .
          STW       6,*0,1    Store Result
          ADI       0,8
          TRSW      0         Return to Call +3 words
```

The final method is similar to the previous versions except that GPR 1 through GPR 7 are left undisturbed:

```
SQRT      STF       0, SAVE   Save general purpose registers
          TRR       0,1
          LW        6,*0,1    Pick up Arguments
          ADI       1,4
          LW        7,*0,1
          .
          .
          .
          ST        6,*0,1    Store result
          LF        0, SAVE   Restore general purpose registers
          ADI       0,8
          TRSW      0         Return to Call +3
SAVE      RES       8W        Temporary storage for general
                              purpose registers
```

BU
*m,x

BRANCH UNCONDITIONALLY
ECOO



DEFINITION

The Effective Address (bit 13 through bit 30), contained in the instruction, is transferred to the corresponding bit positions in the Program Status Word Register (PSWR). This causes program control to be transferred to any word or half-word location in memory. Bit positions 1 through 12 of the PSWR remain unchanged if the Indirect bit is equal to ZERO. If the Indirect bit of the instruction word is equal to ONE, bit positions 1 through 12 of the last memory word in the Indirect chain are transferred to the corresponding bit positions of the PSWR. Bit 0 (privileged state bit) of the PSWR remains unchanged.

SUMMARY EXPRESSION

$EA \rightarrow PSWR_{13-30}$, IF I=0

$(EWL) \rightarrow PSWR_{1-30}$, IF I=1

CONDITION CODE RESULTS

If the Indirect bit is equal to zero, the condition code remains unchanged.

CC1: I is equal to ONE and (EWL_1) is equal to ONE
CC2: I is equal to ONE and (EWL_2) is equal to ONE
CC3: I is equal to ONE and (EWL_3) is equal to ONE
CC4: I is equal to ONE and (EWL_4) is equal to ONE

EXAMPLE 1

Memory Location: 0100
Hex Instruction: EC 00 14 12 (X=I=0)

Before Execution

PSWR
20001000

After Execution

PSWR
20001412

Note

The contents of bits 13 through 30 of the instruction replace the corresponding portion of the PSWR. The Condition Code remains unchanged.

EXAMPLE 2

Memory Location: 01000
Hex Instruction: EC 10 14 12 (X=0, I=1)

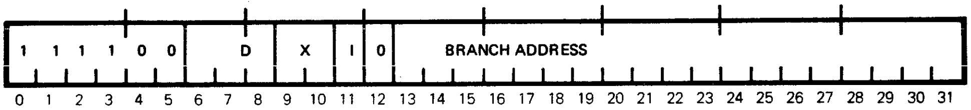
Before Execution	PSWR 8001000	Memory Word 01412 700015AC
After Execution	PSWR F00015AC	Memory Word 01412 700015AC

Note The contents of bits 1 through 30 of memory word 01412 replace the previous contents of bits 1 through 30 of the PSWR.

BCF
v,*m,x

BRANCH CONDITION FALSE

F000



DEFINITION

The Effective Address (bit 13 through bit 30), contained in the instruction, is transferred to the corresponding bit positions in the Program Status Word Register (PSWR) if the condition specified by the D field (bit 6 through 8 of the instruction) is present. The seven specifiable conditions are tabulated below. If the condition is not as specified, the next instruction in sequence is executed. If the Indirect bit of the instruction word is equal to one and the branch occurs, bit positions 1 through 12 of the last memory word in the indirect chain are transferred to the corresponding bit positions of the PSWR.

D Field (Hex.)	Branch Condition (Branch if):
1	CC1=0
2	CC2=0
3	CC3=0
4	CC4=0
5	CC2 and CC4 both =0
6	CC3 and CC4 both =0
7	CC1 and CC2 and CC3 and CC4 all =0

CONDITION CODE RESULTS

The resulting condition code remains unchanged if the Indirect bit (bit 11) is equal to zero.

CC1: I is equal to ONE and (EWL_1) is equal to ONE
CC2: I is equal to ONE and (EWL_2) is equal to ONE
CC3: I is equal to ONE and (EWL_3) is equal to ONE
CC4: I is equal to ONE and (EWL_4) is equal to ONE

EXAMPLE

Memory Location: 02094
Hex Instruction: F1 00 21 4C ($C_1C_2C_3$=2,X=I=0)

Before Execution

PSWR
10002094

After Execution

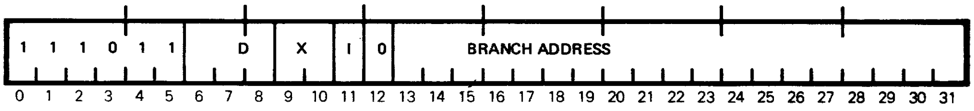
PSWR
1000214C

Note

Condition Code bit 2 is not set. The effective address, in this case bits 13 through 30 of the instruction, is transferred to the PSWR.

BRANCH CONDITION TRUE

EC00



DEFINITION

The Effective Address (bit 13 through 30) contained in the instruction is transferred to the corresponding bit positions in the Program Status Word Register (PSWR), if the current condition code is TRUE to the condition specified by the D field (bit 6 through bit 8). The seven specifiable conditions are tabulated in the table below. If the Indirect bit of the instruction word is equal to one, bit positions 1 through 12 of the last memory word in the indirect chain are transferred to the corresponding bit positions of the PSWR.

D Field (Hex.)	Branch Condition (Branch if):
1	CC1=1
2	CC2=1
3	CC3=1
4	CC4=1
5	CC2 v CC4=1
6	CC3 v CC4=1
7	CC1 v CC2 v CC3 v CC4=1

CONDITION CODE RESULTS

The resulting condition code remains unchanged if the Indirect bit (bit 11) is equal to zero.

CC1: I is equal to ONE and (EWL$_1$) is equal to ONE
CC2: I is equal to ONE and (EWL$_2$) is equal to ONE
CC3: I is equal to ONE and (EWL$_3$) is equal to ONE
CC4: I is equal to ONE and (EWL$_4$) is equal to ONE

EXAMPLE

Memory Location: 01000
Hex Instruction: EC 80 14 12 (Condition=1,X=I=0)

Before Execution

PSWR
50001000

After Execution

PSWR
50001412

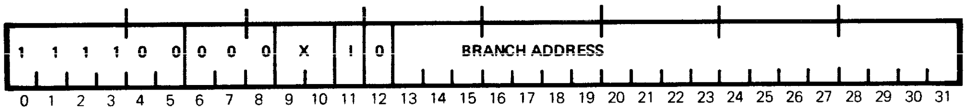
Note

The contents of bits 13-30 of the instruction are transferred to bits 13-30 of the PSWR.

BFT
*m,x

BRANCH FUNCTION TRUE

F000



DEFINITION

The Effective Address (bit 13 through bit 30) contained in the instruction is transferred to the corresponding bit positions in the Program Status Word Register (PSWR) if the function bit in the mask register (R4) for the minterm (one of the 16 possible combinations of the four condition code bits) which corresponds to the current condition code is equal to one. The function F is defined by the 16 least significant bits of the mask register. All 16 minterms of the four variables A=CC1, B=CC2, C=CC3, D=CC4 are defined below.

$$F = \bar{A}\bar{B}\bar{C}\bar{D}\ R4_{16}\mathrm{v}\ \bar{A}\bar{B}\bar{C}D\ R4_{17}\mathrm{v}\ \bar{A}\bar{B}C\bar{D}\ R4_{18}\mathrm{v}\ \bar{A}\bar{B}CD\ R4_{19}$$

$$\bar{A}B\bar{C}\bar{D}\ R4_{20}\mathrm{v}\ \bar{A}B\bar{C}D\ R4_{21}\mathrm{v}\ \bar{A}BC\bar{D}\ R4_{22}\mathrm{v}\ \bar{A}BCD\ R4_{23}$$

$$A\bar{B}\bar{C}\bar{D}\ R4_{24}\mathrm{v}\ A\bar{B}\bar{C}D\ R4_{25}\mathrm{v}\ A\bar{B}C\bar{D}\ R4_{26}\mathrm{v}\ A\bar{B}CD\ R4_{27}$$

$$AB\bar{C}\bar{D}\ R4_{28}\mathrm{v}\ AB\bar{C}D\ R4_{29}\mathrm{v}\ ABC\bar{D}\ R4_{30}\mathrm{v}\ ABCD\ R4_{31}$$

Therefore, any logical function of the four variables stored in the condition code register can be evaluated by storing the proper 16-bit function code in the mask register. The next instruction in sequence is executed if the function is equal. to zero. If the Indirect bit of the instruction word is equal to one, bit positions 1 through 12 of the last memory word in the indirect chain are transferred to the corresponding bit positions of the PSWR.

SUMMARY EXPRESSION

If F=1 & I=0, $EA_{13-30} \rightarrow PSWR_{13-30}$

If F=1 & I=1, $EA_{1-30} \rightarrow PSWR_{1-30}$

If F=0, $PSWR_{13-30} + 1_{29} \rightarrow PSWR_{13-30}$

CONDITION CODE RESULTS

The resulting condition code remains unchanged if the indirect bit (bit 11) is equal to zero.

CC1: I=1 and EA_1=1
CC2: I=1 and EA_2=1
CC3: I=1 and EA_3=1
CC4: I=1 and EA_4=1

EXAMPLE	Memory Location:	01000
	Hex Instruction:	F0 00 20 00 (X= I=0)

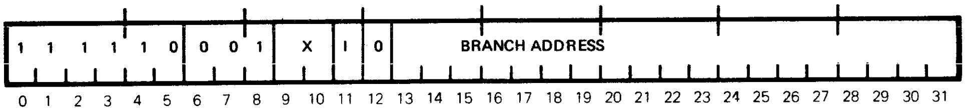
Before Execution	PSWR	GPR4
	70001000	00000002
After Execution	PSWR	GPR4
	700020000	00000002

Note Bit 30 of GPR4 defines a function for which CC1=CC2=CC3=1, CC4=0. This function is true, so a branch is effected.

BL
*m,x

BRANCH AND LINK

F880



DEFINITION

The contents of the Program Status Register (PSWR) are incremented by one word and transferred to General Purpose Regsiter 0. If the Indirect bit of the instruction word is equal to zero, the Effective Address (bit 13 through bit 30) is transferred to the corresponding bit positions of the PSWR. Bit positions 1 through 12 of the PSWR remain unchanged. If the indirect bit of the instruction word is equal to zero, bit positions 1 through 12 of the PSWR remain unchanged. If the indirect bit of the instruction word is equal to one, bit positions 1 through 12 of the last memory word in the indirect chain are also transferred to the corresponding bit positions of the PSWR. Bit 0 (privileged state bit) of the PSWR remains unchanged.

SUMMARY EXPRESSION

(PSWR) → R0

EA → PSWR_{13-30}, if I=0

EWL_{1-12}, EA → PSWR_{1-30}, if I=1

CONDITION CODE RESULTS

If the indirect bit is equal to zero, the condition code remains unchanged.

CC1: I is equal to ONE and (EWL_1) is equal to ONE
CC2: I is equal to ONE and (EWL_2) is equal to ONE
CC3: I is equal to ONE and (EWL_3) is equal to ONE
CC4: I is equal to ONE and (EWL_4) is equal to ONE

EXAMPLE

Memory Location: 0894C
Hex Instruction: F8 80 A3 78 (X=I=0)

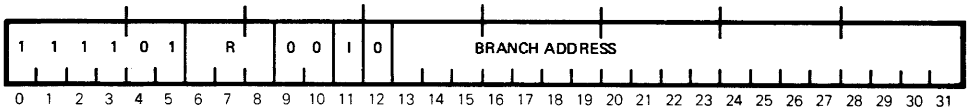
	PSWR	GPR0
Before Execution	1000894C	12345678
After Execution	1000A378	10008950

Note

The contents of the PSWR are transferred to GPR0. The contents of bits 13-30 of the instruction are transferred to bits 13-30 of the PSWR.

BRANCH AFTER INCREMENTING BYTE

F400



DEFINITION

The contents of the General Purpose Register specified by R are incremented in bit position 31. If the result is NON-ZERO the Effective Address (EA) is transferred to the Program Status Word Register (PSWR) bit positions 13 through 30 and bit positions 1 through 12 of the PSWR remain unchanged. If the result is equal to zero after incrementing, the next instruction is executed.

SUMMARY EXPRESSION

$(R) + 1_{31} \rightarrow R$

$EA \rightarrow PSWR_{13-30}$, if result ≠ 0

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 1B204
Hex Instruction: F4 01 B1 A8 (R=0, I=0)

Before Execution

PSWR	GPR0
2001B204	FFFFFFFF

After Execution

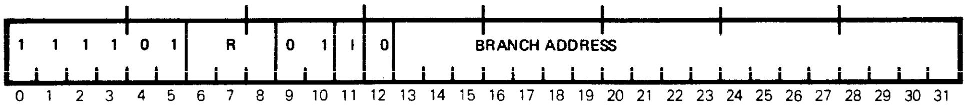
PSWR	GPR0
2001B208	00000000

Note

The contents of the GPR0 are incremented by a one at bit position 31. Since the result is zero, no branch occurs.

BRANCH AFTER INCREMENTING HALFWORD

F420 d, m



DEFINITION

The contents of the General Purpose Register specified by R are incremented in bit position 30. If the result is NON-ZERO the Effective Address (EA) is transferred to the Program Status Word Register (PSWR) bit positions 13 through 30 and bit positions 1 through 12 of the PSWR remain unchanged. If the result is equal to zero after incrementing, the next instruction is executed.

SUMMARY EXPRESSION

$(R) + 1_{30} \rightarrow R$

$EA \rightarrow PSWR_{13-30}$, if result $\neq 0$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 039A0
Hex Instruction: F5 20 39 48 (R=2, I=0)

	PSWR	GPR2
Before Execution	100039A0	FFFFD72A
After Execution	10003948	FFFFD72C

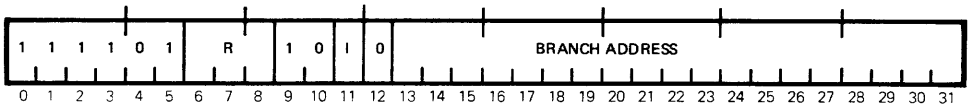
Note

The contents of GPR2 are incremented by one in bit position 30. The result is replaced in GPR2 and a branch occurs to address 03948.

BIW
d,m

BRANCH AFTER INCREMENTING WORD

F440



DEFINITION

The contents of the General Purpose Register specified by R are incremented in bit position 29. If the result is NON-ZERO the Effective Address (EA) is transferred to the Program Status Word Register (PSWR) bit positions 13 through 30 and bit positions 1 through 12 of the PSWR remain unchanged. If the result is equal to zero after incrementing, the next instruction is executed.

SUMMARY EXPRESSION

$(R) + 1_{29} \rightarrow R$

$EA \rightarrow PSWR_{13-30}$, if result ≠ 0

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 04A38
Hex Instruction: F7 40 4B 2C (R=6, I=0)

	PSWR	GPR6
Before Execution	60004A38	FFFFDC18
After Execution	60004B2C	FFFFDC1C

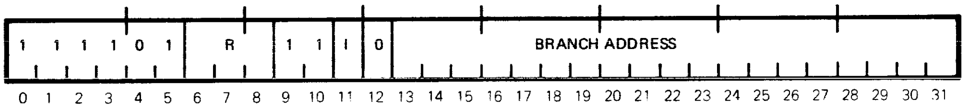
Note

The content of GPR6 is incremented by a one at bit position 29, and the result is transferred to GPR6. The Effective Address of the BIW instruction, 04B2C, replaces the previous contents of the PSWR, bits 12-30.

BID
d,m

BRANCH AFTER INCREMENTING DOUBLEWORD

F460



DEFINITION

The contents of the General Purpose Register specified by R are incremented in bit position 28. If the result is NON-ZERO the Effective Address (EA) is transferred to the Program Status Word Register (PSWR) bit positions 13 through 30 and bit positions 1 through 12 of the PSWR remain unchanged. If the result is equal to zero after incrementing, the next instruction is executed.

SUMMARY EXPRESSION

$(R) + 1_{28} \rightarrow R$

$EA \rightarrow PSWR_{13-30}$, if result ≠ 0

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 0930C
Hex Instruction: F5 E0 91 A6 (R=3, I=0)

Before Execution

PSWR	GPR3
0800930C	FFFFFFF8

After Execution

PSWR	GPR3
08009310	00000000

Note

The content of GPR3 is incremented by one at bit position 28 and replaced. Since the result is zero, no branch occurs.

BRI
*m,x

BRANCH AND RESET INTERRUPT

F900

0	1	2	3	4	5	6	7	8	9–10	11	12	13–31
1	1	1	1	1	0	0	1	0	X	I	0	BRANCH ADDRESS

DEFINITION

Execution of the Branch and Reset Interrupt (BRI) instruction resets the active condition for the highest active interrupt level. Any request signals which are received on an interrupt level that is in the active condition will be held; when the active condition of the interrupt level is reset by execution of a BRI, that interrupt will be immediately serviced again if any such requests are being held. The Effective Address (bit 13 through bit 30) is transferred to the corresponding bit positions in the Program Status Word Register (PSWR). If the indirect bit of the instruction word is equal to zero, bit positions 1 through 12 of the PSWR remain unchanged. If the indirect bit is equal to one, bit positions 0 through 12 of the last memory word in the indirect chain are also transferred to the corresponding bit positions of the PSWR. Transferring into bit position zero of the PSWR causes the operation state of the computer to be set to privileged if the bit is equal to one and unprivileged if the bit is equal to zero. Therefore, the operation state present at the time of occurrence of an interrupt can be restored by the Branch and Reset Interrupt instruction used to return program control to the interrupt program.

SUMMARY EXPRESSION

$EA \rightarrow PSWR_{13-30}$, if I=0

$EWL_{0-12}, EA \rightarrow PSWR_{0-30}$, if I=1

CONDITION CODE RESULTS

CC1: I is equal to ONE and (EWL_1) is equal to ONE
CC2: I is equal to ONE and (EWL_2) is equal to ONE
CC3: I is equal to ONE and (EWL_3) is equal to ONE
CC4: I is equal to ONE and (EWL_4) is equal to ONE

EXAMPLE

Memory Location: 081B4
Hex Instruction: F9 10 81 3C

Before Execution

PSWR	Memory Word 0813C
400081B4	04015D68

After Execution

PSWR	Memory Word 0813
04015D68	04015D68

Note

The highest active interrupt level is reset, and the content of memory word 0813C is transferred to the PSWR.

COMPARE INSTRUCTIONS

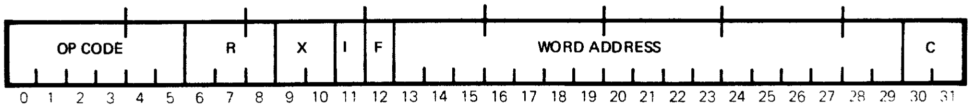
General Description

Compare Instructions provide the capability of comparing data contained in memory and General Purpose Registers. These operations can be performed on bytes, halfwords, words, or doublewords. Provisions have also been made to allow the result of compare operations to be masked with the contents of the mask register before final testing.

Instruction Format

The compare instruction group uses three instruction formats.

Memory Reference

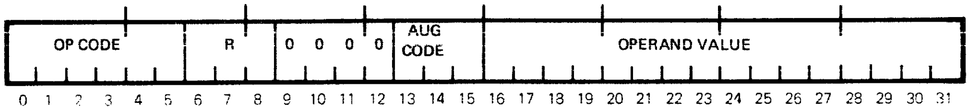


Bits 0-5 define the Operation Code.
Bits 6-8 designate a General Purpose Register Address (0 through 7).
Bits 9-10 designate one of three Index Registers.
Bit 11 indicates if an indirect addressing operation is to be performed.
Bits 12-31 specify the address of the operand when X and I fields are equal to zero.

Note

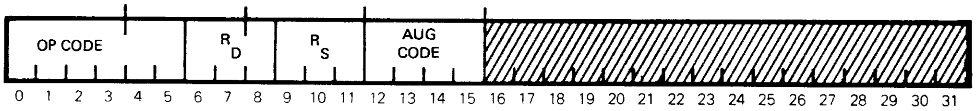
Additional information on the memory reference instruction format is included with the Load/Store instruction formats.

Immediate



Bits 0-5 define the Operation Code.
Bits 6-8 designate a General Purpose Register address (0 through 7).
Bits 9-12 unassigned.
Bits 13-15 define Augmenting Operation Code.
Bits 16-31 contain the 16-bit operand value.

Inter-Register



Bits 0-5 define the Operation Code.
Bits 6-8 designate the register to contain the result of the operation.
Bits 9-11 designate the register which contains the source operand.
Bits 12-15 define the Augmenting Operation Code.

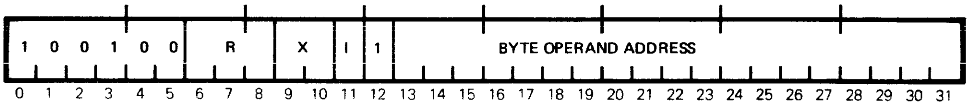
Condition Code Utilization

A Condition Code is set during most compare instructions to indicate if the operation produced a greater than, less than, or equal to zero result.

CAMB
d,*m,x

COMPARE ARITHMETIC WITH MEMORY BYTE

9008



DEFINITION

The byte in memory, specified by the Effective Byte Address (EBA), is accessed, right justified, and subtracted algebraically from the word located in the General Purpose Register (GPR) specified by R. The result of the subtraction causes one of the Condition Code bits, 2 through 4, to set. The contents of the GPR specified by R and the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

(R) - (EBL) → SCC_{2-4}

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R) is greater than (EBL)
CC3: (R) is less than (EBL)
CC4: (R) is equal to (EBL)

EXAMPLE

Memory Location:	01000
Hex Instruction:	90 88 10 B5 (R = 1, X = I = 0)

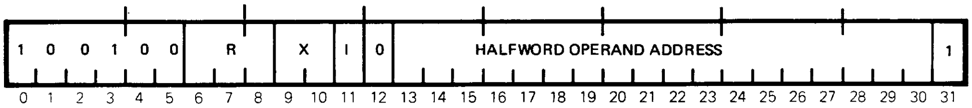
Before Execution	PSWR	GPR1	Memory Byte 010B5
	08001000	000000B6	C7
After Execution	PSWR	GPR1	Memory Byte 010B5
	1001004	000000B6	C7

Note

CC3 is set, which indicates that the contents of GPR1 are less than the contents of memory byte 010B5.

COMPARE ARITHMETIC WITH MEMORY HALFWORD

9000



DEFINITION

The halfword in memory, specified by the Effective Halfword Address (EHA), is accessed and the sign bit is extended 16 bits to the left, to form a word. The resulting word is subtracted algebraically from the word located in the General Purpose Register (GPR) specified by R. The result of the subtraction causes one of the Condition Code bits, 2 through 4, to be set. The word located in the GPR specified by R and the halfword specified by the EHA remain unchanged.

SUMMARY EXPRESSION

$(R) - (EHL)_{SE} \rightarrow SCC_{2-4}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R) is greater than (EHL)
CC3: (R) is less than (EHL)   SE
CC4: (R) is equal to (EHL) SE
SE

EXAMPLE

Memory Location: 0379C
Hex Instruction: 92 00 39 77 (R=4, X=I=0)

Before Execution	PSWR 0800379C	GPR4 00008540	Memory Halfword 03976 8640
After Execution	PSWR 200037A0	GPR4 00008540	Memory Halfword 03976 8640

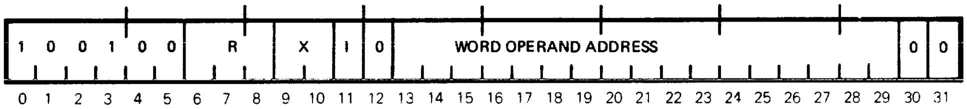
Note

The contents of GPR4 are greater than the contents of memory halfword 03976 (a negative value). CC2 is set.

CAMW
d,*m,x

COMPARE ARITHMETIC WITH MEMORY WORD

9000



DEFINITION

The word in memory, specified by the Effective Word Address (EWA), is accessed and subtracted algebraically from the word located in the General Purpose Register (GPR) specified by R. The result of the subtraction causes one of the Condition Code bits, 2 through 4, to be set. The word located in the GPR specified by R and word specified by the EWA remain unchanged.

SUMMARY EXPRESSION

$(R) - (EWL) \rightarrow SCC_{2-4}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R) is greater than (EWL)
CC3: (R) is less than (EWL)
CC4: (R) is equal to (EWL)

EXAMPLE

Memory Location: 05B20
Hex Instruction: 93 00 5C 78 (R=6, X=I=0)

Before Execution	PSWR 40005B20	GPR6 9E03B651	Memory Word 05C78 A184F207
After Execution	PSWR 10005B24	GPR6 9E03B651	Memory Word 05C78 A184F207

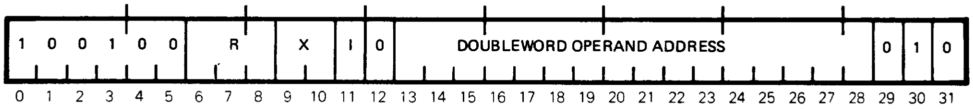
Note

The contents of the GPR6 are less than the contents of memory word 05C78. CC3 is set.

CAMD
d,*m,x

COMPARE ARITHMETIC WITH MEMORY DOUBLEWORD

9000



DEFINITION

The doubleword in memory, specified by the Effective Doubleword Address (EDA), is accessed and subtracted algebraically from the doubleword, located in the General Purpose Register (GPR), specified by R and R+1. R+1 is GPR one greater than specified by R. The result of the subtraction causes one of the Condition Code bits, 2 through 4, to be set. The doubleword located in the GPR specified by R and R+1 and the doubleword specified by the EDA remain unchanged.

SUMMARY EXPRESSION

$(R, R+1) - (EDL) \rightarrow SCC_{2-4}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R, R+1) is greater than (EDL)
CC3: (R, R+1) is less than (EDL)
CC4: (R, R+1) is equal to (EDL)

EXAMPLE

Memory Location: 27C14
Hex Instruction: 92 02 7F 52 (R=4, X=I=0)

Before Execution

PSWR	GPR4	GPR5
20027C14	7AE0156D	47B39208

Memory Word 27F50	Memory Word 27F54
7AE0156D	47B39208

After Execution

PSWR	GPR4	GPR5
08027C18	7AE0156D	47B39208

Memory Word 27F50	Memory Word 27F54
7AE0156D	47B39208

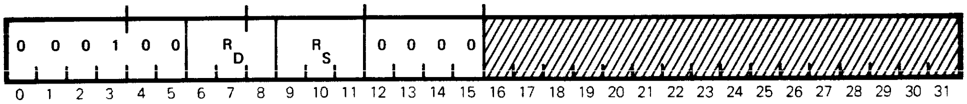
Note

The doubleword obtained from GPR4 and GPR5 is equal to that obtained from the memory words 27F50 and 27F54. CC4 is set.

CAR
s,d

COMPARE ARITHMETIC WITH REGISTER

1000



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_S is subtracted algebraically from the word located in the GPR specified by R_D. The result of the subtraction causes one of the Condition Code bits, 2 through 4, to be set. The words specified by R_S and R_D remain unchanged.

SUMMARY EXPRESSION

$(R_D) - (R_S) \rightarrow SCC_{2-4}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R_D) is greater than (R_S)
CC3: (R_D) is less than (R_S)
CC4: (R_D) is equal to (R_S)

EXAMPLE

Memory Location: 0B3C2
Hex Instruction: 10 10 (R_D = 0, R_S = 1)

Before Execution	PSWR 0800B3C2	GPR0 58DF620A	GPR1 6A92B730
After Execution	PSWR 1000B3C4	GPR0 58DF620A	GPR1 6A92B730

Note

The contents of GPR0 are less than the contents of GPR1. CC3 is set.

COMPARE IMMEDIATE

C805

1 1 0 0 1 0	R	0 0 0	0 1 0 1	IMMEDIATE OPERAND
0 1 2 3 4 5	6 7 8	9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

DEFINITION

The sign bit (bit 16) of the immediate operand is extended 16 bit positions to the left to form a word. This word is subtracted from the word located in the General Purpose Register (GPR) specifed by R. The result of the subtraction causes one of the Condition Code bits, 2 through 4, to be set. The word located in the GPR specified by R and the immediate operand (bit 16 through 31) remain unchanged.

SUMMARY EXPRESSION

$(R) - (IW_{16-31})_{SE} \rightarrow SCC_4$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R) is greater than $(IW_{16-31})_{SE}$

CC3: (R) is less than $(IW_{16-31})_{SE}$

CC4: (R) is equal to $(IW_{16-31})_{SE}$

EXAMPLE

Memory Location: 0A794
Hex Instruction: C8 85 71 A2 (R=1)

	PSWR	GPR1
Before Execution	400A794	00005719
After Execution	1000A798	00005719

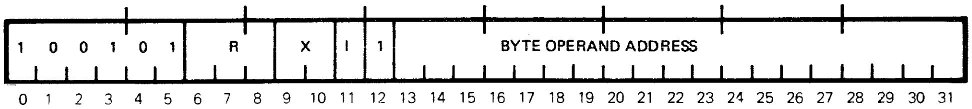
Note

The contents of GPR1 are less than the immediate operand. CC3 is set.

CMMB
d,*m,x

COMPARE MASKED WITH MEMORY BYTE

9408



DEFINITION

The byte in memory, specified by the Effective Byte Address (EBA), is accessed and 24 zeros are appended to the most significant end to form a word. This word is logically compared (exclusive OR function) with the word located in the General Purpose Register specified by R. The resulting word is then masked (logical AND function) with the contents of the mask register, R4. The masked result is tested and Condition Code bit 4 set if all 32 bits equal zero. The word located in the GPR specified by R, and the byte specified by the EBA, remain unchanged.

SUMMARY EXPRESSION

$[(R) \oplus 0_{0-23}, (EBL)] \& (R4) \rightarrow SCC_4$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: Result is equal to zero

EXAMPLE

Memory Location: 00800
Hex Instruction: 94 08 09 17 (R=0, X=I=0)

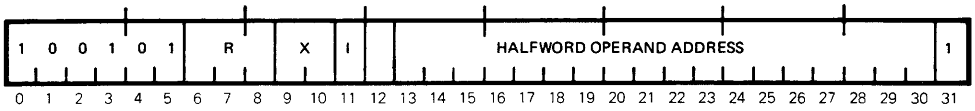
Before Execution	PSWR 10000800	GPR0 000000A1	GPR4 000000F0	Memory Byte 00917 A9
After Execution	PSWR 08000804	GPR0 000000A1	GPR4 000000F0	Memory Byte 00917 A9

Note

The contents of GPR0 and memory byte 00917 are identical in those bit positions specified by the contents of GPR4. CC4 is set.

COMPARE MASKED WITH MEMORY HALFWORD

9400



DEFINITION

The halfword in memory, specified by the Effective Halfword Address (EHA), is accessed and the sign (bit 16) is extended 16 bits to the left to form a word. The resulting word is logically compared (exclusive OR function) with the word located in the General Purpose Register (GPR) specified by R. The resulting word is then masked (logical AND function) with the contents of the mask register R4. The masked result is tested and Condition Code bit 4 set if all 32 bits equal zero. The word located in the GPR specified by R and the halfword specified by the EHA remain unchanged.

SUMMARY EXPRESSION

$[(R) \oplus (EHL)_{SE}] \& (R4) \rightarrow SCC_4$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: Result is equal to zero

EXAMPLE

Memory Location: 061B8
Hex Instruction: 95 00 62 93 (R=2)

Before Execution	PSWR 100061B8	GPR2 09A043B6	GPR4 00004284	Memory Halfword 06292 46FC
After Execution	PSWR 080061BC	GPR2 09A043B6	GPR4 00004284	Memory Halfword 06292 46FC

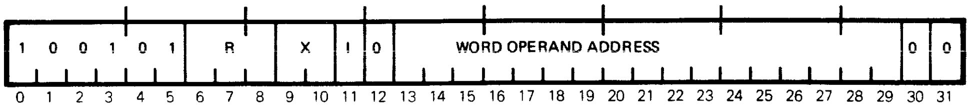
Note

The contents of GPR2 and memory halfword 06292 are identical in those bit positions specified by the contents of GPR4. CC4 is set.

CMMW
d,*m,x

COMPARE MASKED WITH MEMORY WORD

9400



DEFINITION

The word in memory, specified by the Effective Word Address (EWA), is accessed and logically compared (exclusive OR function) with the word located in the General Purpose Register (GPR) specified by R. The result of the comparison is then masked (logical AND function) with the contents of the mask register R4. The masked result is tested and Condition Code bit 4 set, if all 32 bits equal zero. The word located in the GPR specified by R and the word specified by the EWA remain unchanged.

SUMMARY EXPRESSION

$[(R) \oplus (EWL)]$ & $(R4) \rightarrow SCC_{2-4}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: Result is equal to zero

EXAMPLE

Memory Location: 13A74
Hex Instruction: 97 01 3C 94 (R=6, X=I=0)

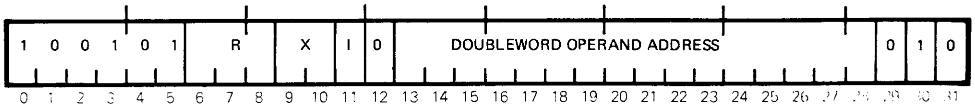
Before Execution	PSWR 08013A74	GPR4 00FFFF00	GPR6 132A1C04	Memory Word 13C94 472A3D04
After Execution	PSWR 00013A78	GPR4 00FFFF00	GPR6 132A1C04	Memory Word 13C94 472A3D04

Note

The contents of GPR6 and memory word 13C94 are not equal within the bit positions specified by the contents of GPR4.

COMPARE MASKED WITH MEMORY DOUBLEWORD

9400



DEFINITION

The doubleword in memory, specified by the Effective Doubleword Address (EDA), is accessed and compared (exclusive OR function) with the doubleword located in the General Purpose Register (GPR) specified by R and R+1. R+1 is GPR one greater than wpecified by R. Each result from the comparison is then masked (logical AND function) with the contents of the mask register R4. The doubleword masked result is tested and Condition Code bit 4 set, if all 64 bits equal zero. The doubleword located in the GPR specified by R and R+1 and the doubleword specified by the EDA remain unchanged.

SUMMARY EXPRESSION

$[(R) \oplus (EWL)] \ \& \ (R4), [(R+1) \oplus (EWL+1)] \ \& \ (R4) \rightarrow SCC_4$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: Result is equal to zero

EXAMPLE

Memory Location: 03000
Hex Instruction: 97 00 31 BA (R=6, X=I=0)

Before Execution

PSWR	GPR4	GPR6	GPR7
10003000	000FFFFF	FFF3791B	890A45D6

Memory Word 031B8	Memory Word 031BC
0003791B	890A45C2

After Execution

PSWR	GPR4	GPR6	GPR7
00003004	000FFFFF	FFF3791B	890A45D6

Memory Word 031B8	Memory Word 031BC
0003791B	890A45C2

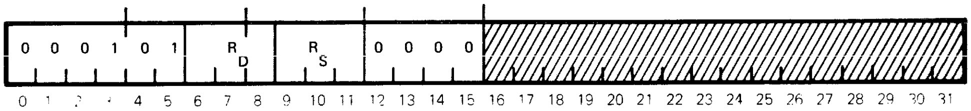
Note

The contents of GPR7 and memory word 031BC differ within the bit positions specified by the contents of GPR4.

CMR
s,d

COMPARE MASKED WITH REGISTER

1400



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_D is logically compared (exclusive OR function) with the word located in the GPR specified by R_S. The result of the comparison is then masked (logical AND function) with the contents of mask register R4. The result is tested and Condition Code bit 4 set, if all 32 bits equal zero. The words specifie; by R_S and R_D remain unchanged.

SUMMARY EXPRESSION

$[(R_D) \oplus (R_S)]$ & $(R4) \rightarrow SCC_4$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: Result is equal to zero

EXAMPLE

Memory Location: 050D2
Hex Instruction: XXXX14 A0 (R_D=1, R_S=2)

Before Execution	PSWR 100050D2	GPR1 583C94A2	GPR2 0C68C5F6	GPR4 AAAAAAAA
After Execution	PSWR 080050D4	GPR1 583C94A2	GPR2 0C68C5F6	GPR4 AAAAAAAA

Note

The contents of GPR1 and GPR2 are identical within the bit positions specified by the contents of GPR4. CC4 is set.

LOGICAL INSTRUCTIONS

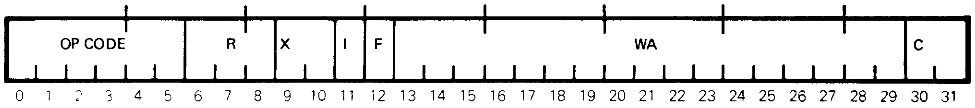
General Description

The Logical Instruction group provides the capability of performing AND, OR, and EXCLUSIVE OR operations on bytes, halfwords, and doublewords contained in memory and general purpose registers. Provisions have also been made to allow the result of Register-to-Register OR and EXCLUSIVE OR operations to be masked with the contents of the mask register (R4) before final storage.

Instruction Formats

The Logical Instruction group uses the following two instruction formats.

Memory Reference



Bits 0-5 define the Operation Code.

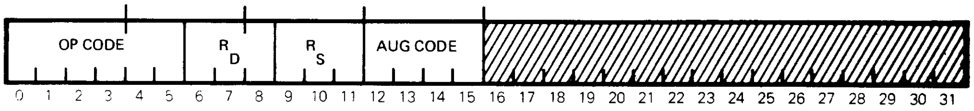
Bits 6-8 designate a General Purpose Register address (0 through 7).

Bits 9-10 designate one of three Index Registers.

Bit 11 indicates if an indirect addressing operation is to be performed.

Bits 12-31 specify the address of the operand when X and I fields are equal to zero.

Inter-Register



Bits 0-5 define the Operation Code.

Bits 6-8 designate the register to contain the result of the operation.

Bits 9-11 designate the register which contains the source operand.

Bits 12-15 define the Augmenting Operation Code.

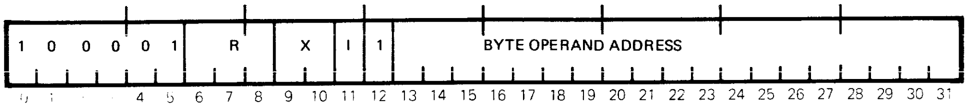
Condition Code Utilization

A condition code is set during execution of most Logical Instructions to indicate if the result of that operation was greater than, less than, or equal to zero.

ANMB
d,*m,x

AND MEMORY BYTE

8408



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and logically ANDed with the least significant byte (bit 24 through 31) of the General Purpose Register (GPR) specified by R. The result is transferred to bit positions 24 through 31 of the GPR specified by R. Bit positions 0 through 23 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(EBL)\&(R_{24-31}) \rightarrow R_{24-31}$

R_{0-23} Unchanged

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{24-31} is greater than zero

CC3: Always zero
CC4: R_{24-31} is equal to zero

EXAMPLE

Memory Location: 00200
Hex Instruction: 84 88 03 73 (R=1,X=I=0)

Before Execution	PSWR	GPR1	Memory Byte 00373
	00000200	36AC718F	C7
After Execution	PSWR	GPR1	Memory Byte 00373
	20000204	36AC7187	C7

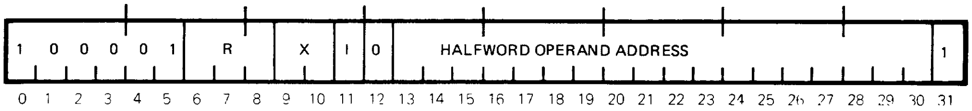
Note

The contents from memory byte 00373 are ANDed with the rightmost byte of GPR1, and the result replaces that byte in GPR1. CC2 is set.

ANMH
d,*m,x

AND MEMORY HALFWORD

8400



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed and logically ANDed with the least significant halfword (bit 16 through bit 31) of the General Purpose Register (GPR) specified by R. The result is transferred to bit positions 16 through 31 of the GPR specified by R. Bit positions 0 through 15 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(EHL)\&(R_{16-31}) \rightarrow R_{16-31}$

R_{0-15} Unchanged

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{16-31} is greater than zero

CC3: Always zero
CC4: R_{16-31} is equal to zero

EXAMPLE

Memory Location: 01000
Hex Instruction: 87 00 12 A3 (R=6,X=I=0)

Before Execution	PSWR 40001000	GPR6 4F638301	Memory Halfword 012A2 70F6
After Execution	PSWR 08001004	GPR6 4F630000	Memory Halfword 012A2 70F6

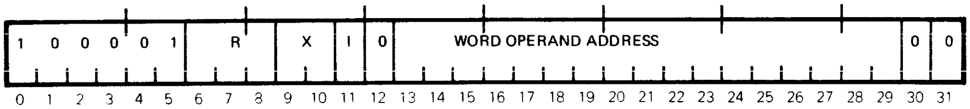
Note

The contents from memory halfword 012A2 are ANDed with the right halfword of GPR6, and the result replaces the halfword in GPR6. CC4 is set.

ANMW
d,*m,x

AND MEMORY WORD

8400



DEFINITION

The word in memory specified by the Effective Word Address (EWA) is accessed and logically ANDed with the word located in the General Purpose Register (GPR) specified by R.

SUMMARY EXPRESSION

(EWL)&(R) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{0-31} is greater than zero

CC3: R_{0-31} is less than zero

CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 00F1C
Hex Instruction: 87 80 0F D0 (R=7,X=I-0)

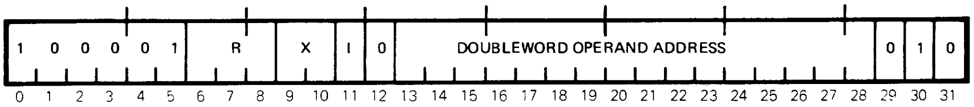
Before Execution	PSWR 08000F1C	GPR7 F0F0F0F0	Memory Word 00FD0 9ED13854
After Execution	PSWR 10000F20	GPR7 90D03050	Memory Word 00FD0 9ED13854

Note

The contents from memory word 00FD0 are ANDed with the contents from GPR7, and the result replaces the contents of that register. CC3 is set.

AND MEMORY DOUBLEWORD

8400



DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and logically ANDed with the doubleword located in the General Purpose Register (GPR) specified by R and R+1. R+1 is the GPR one greater than specified by R. The result doubleword is transferred to the GPR specified by R and R+1.

SUMMARY EXPRESSION

(EWL+1)&(R+1) → R+1

(EWL)&(R) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R,R+1) is greater than zero
CC3: (R,R+1) is less than zero
CC4: (R,R+1) is equal to zero

EXAMPLE

Memory Location: 00674
Hex Instruction: 86 00 08 1A (R=4,X=I=0)

Before Execution

PSWR	GPR4	GPR5
00000674	9045C64A	32B08F00

Memory Word 00818	Memory Word 0081C
684A711C	8104A2BC

After Execution

PSWR	GPR4	GPR5
20000678	00404008	00008200

Memory Word 00818	Memory Word 0081C
684A711C	8104A2BC

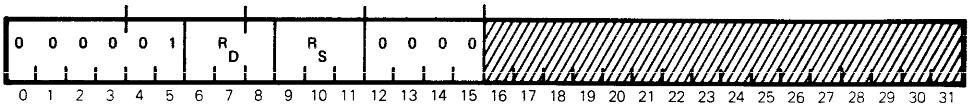
Note

The contents from memory word 00818 are ANDed with the contents from GPR4, and the result replaces the contents of GPR4. The contents from memory word 0081C are ANDed with the contents from GPR5, and the result replaces the contents of GPR5. CC2 is set.

ANR
s,d

AND REGISTER AND REGISTER

0400



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_D is logically ANDed with the word located in the GPR specified by R_S. The resulting word is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S)\&(R_D) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

EXAMPLE

Memory Location: 03812
Hex Instruction: 04 FO (R_D=1, R_S=7)

	PSWR	GPR1	GPR7
Before Execution	40003812	AC881101	000FFFFF
After Execution	20003814	00081101	000FFFFF

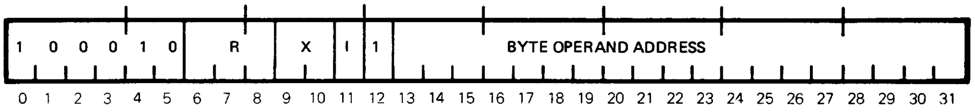
Note

The contents from GPR1 and GPR7 are ANDed, and the result is transferred to GPR1. CC2 is set.

ORMB
d,*m,x

OR MEMORY BYTE

8808



DEFINITION

The byte in memory, specified by the Effective Byte Address (EBA), is accessed and logically ORed with the least significant byte (bit 24 through bit 31) of the General Purpose Register (GPR) specified by R. The resulting byte is transferred to bit positions 24 through 31 of the GPR specified by R. Bit positions 0 through 23 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(ELB)v(R_{24-31}) \rightarrow R_{24-31}$

R_{0-23} Unchanged

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 00600
Hex Instruction: 88 88 08 A3 (R=1,X=I=o)

Before Execution	PSWR 00000600	GPR1 40404040	Memory Byte 8A3 3C
After Execution	PSWR 20000604	GPR1 4040407C	Memory Byte 8A3 3C

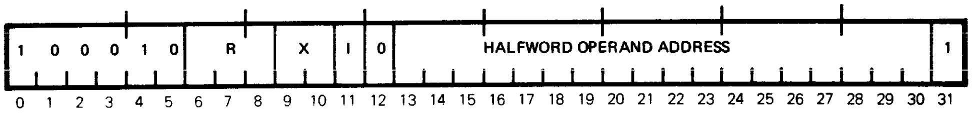
Note

The contents from memory byte 8A3 are logically ORed to the right-most byte of GPR1, and the result replaces that byte in GPR2. CC2 is set.

ORMH
d,*m,x

OR MEMORY HALFWORD

8800



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed and logically ORed with the least significant halfword (bit 16 through bit 31) of the General Purpose Register (GPR) specified by R. The resulting halfword is transferred to bit positions 16 through 31 of the GPR specified by R. Bit positions 0 through 15 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(EHL)v(R_{16-31} \rightarrow R_{16-31}$

R_{0-15} Unchanged

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 018AC
Hex Instruction: 8B 00 19 45 (R=6,X=I=0)

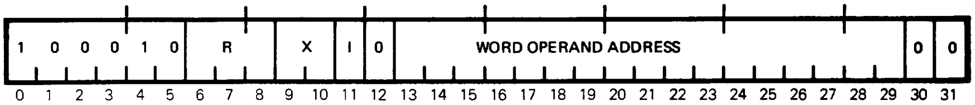
Before Execution	PSWR 000018AC	GPR6 BD71A4C6	Memory Halfword 01944 45F3
After Execution	PSWR 100018B0	GPR6 BD71E5F7	Memory Halfword 01944 45F3

Note

The contents from memory halfword 01944 are ORed with the right halfword from GPR6, and the result replaces that halfword in GPR6. CC3 is set.

OR MEMORY WORD

8800



DEFINITION

The word in memory specified by the Effective Word Address (EWA) is accessed and logically ORed with the word located in the General Purpose Register (GPR) specified by R. The result is transferred to the GPR specified by R.

SUMMARY EXPRESSION

(EWL)v(R) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 05000
Hex Instruction: 89 80 52 0C (R=3,X=I=0)

	PSWR	GPR3	Memory Word 0520C
Before Execution	40005000	88888888	0EDC4657
After Execution	10005004	8EDCCEDF	0EDC4657

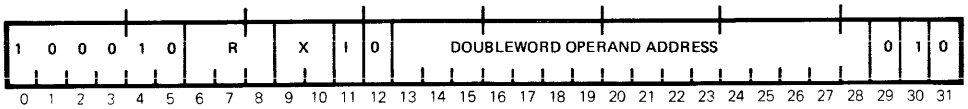
Note

The contents from memory word 0520C are ORed with the contents from GPR3, and the result is transferred to that register. CC3 is set.

ORMD
d,*m,x

OR MEMORY DOUBLEWORD

8800



DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and logically ORed with the doubleword located in the General Purpose Register (GPR) specified by R and R+1. R+1 is GPR one greater than specified by R. The result is transferred to the GPR specified by R and R+1.

SUMMARY EXPRESSION

(EWL+1)v(R+1) → R+1

(EWL)V(R) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R,R+1) is greater than zero
CC3: (R,R+1) is less than zero
CC4: (R,R+1) is equal to zero

EXAMPLE

Memory Location: 00B68
Hex Instruction: 8B 00 0C 32 (R=6, X=I=0)

Before Execution

PSWR	GPR6	GPR7
10000B68	002A0031	001D0039

Memory Word 00C30	Memory Word 00C34
18004C00	09002400

After Execution

PSWR	GPR6	GPR7
20000B6C	182A4C31	091D2439

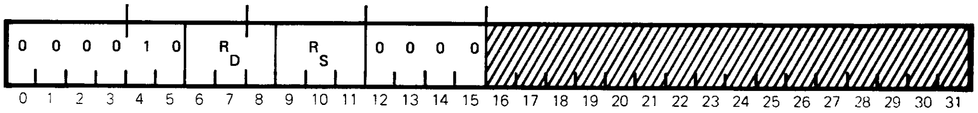
Memory Word 00C30	Memory Word 00C34
18004C00	09002400

Note

The contents from memory word 00C30 are ORed with the contents from GPR6, and the result is transferred to GPR6. The contents from memory word 00C34 are ORed with GPR7, and the result is transferred to GPR7. CC2 is set.

OR REGISTER AND REGISTER

0800



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_D is logically ORed with the word located in the GPR specified by R_S. The result is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S) v (R_D) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

EXAMPLE

Memory Location: 00F8A
Hex Instruction: 08 A0 (R_D=1, R_S=2)

Before Execution	PSWR	GPR1	GPR2
	40000F8A	0001D63F	8880000
After Execution	PSWR	GPR1	GPR2
	10000F8C	888D63F	8880000

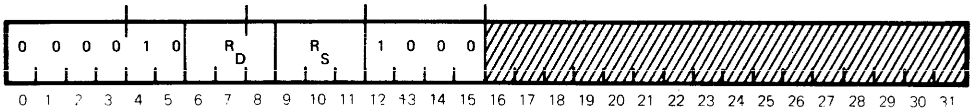
Note

The contents from GPR1 and GPR2 are ORed, and the result is transferred to GPR1. CC3 is set.

ORRM
s,d

OR REGISTER AND REGISTER MASKED

0808



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_D is logically ORed with the word located in the GPR specified by R_S. The resulting word is then masked (logical AND function) with the contents of the Mask Register R4. The result is then transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S)v(R_D)\ \&(R4) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

EXAMPLE

Memory Location: 03956
Hex Instruction: 0B 58 (R_D=6, R_S=5)

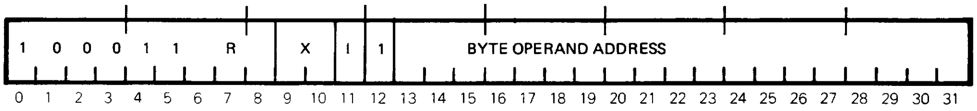
Before Execution	PSWR 08003956	GPR4 EEEEEEEE	GPR5 37735814	GPR6 2561CA95
After Execution	PSWR 10003958	GPR4 EEEEEEEE	GPR5 37735814	GPR6 2662CA84

Note

The contents from GPR5 and GPR6 are ORed, then the result is ANDed with the contents from GPR4 and transferred to GPR6. CC3 is set.

EXCLUSIVE OR MEMORY BYTE

8C08



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and logically exclusive ORed with the least significant byte (bit 24 through bit 31) of the General Purpose Register (GPR) specified by R. The result is transferred to bit positions 24 through 31 of the GPR specified by R. Bits 0 through 23 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(EBL) \oplus (R_{24-31}) \rightarrow R_{24-31}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location:	012F8
Hex Instruction:	8C 08 13 A1 (R=0,X=I=0)

Before Execution	PSWR	GPR0	Memory Byte 013A1
	000012F8	D396F458	A9
After Execution	PSWR	GPR0	Memory Byte 013A1
	100012FC	D396F4F1	A9

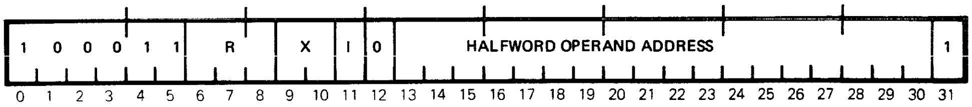
Note

The contents from memory byte 013A1 are exclusive ORed with the rightmost byte from GPR0 and the result replaces that byte in GPR0. CC3 is set.

EOMH
d,*m,x

EXCLUSIVE OR MEMORY HALFWORD

8C00



DEFINITION

The halfword in memory, specified by the Effective Halfword Address (EHA), is accessed and logically exclusive ORed with the least significant halfword (bit 16 through bit 31) of the General Purpose Register (GPR) specified by R. The result is transferred to bit positions 16 through 31 of the GPR specified by R. Bit positions 0 through 15 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(EHL) \oplus (R_{16-31}) \rightarrow R_{16-31}$

R_{0-15} Unchanged

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 00958
Hex Instruction: 8E 80 0A 41 (R=5, X=I=0)

Before Execution	PSWR 40000958	GPR5 96969696	Memory Halfword 00A40 5CAB
After Execution	PSWR 1000095C	GPR5 9696CA3D	Memory Halfword 00A40 5CAB

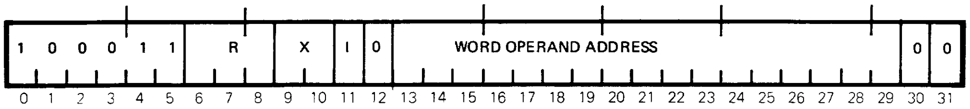
Note

The contents from memory halfword 00A40 are exclusive ORed with the right halfword from GPR5, and the result replaces that halfword in GPR5. CC3 is set.

EOMW
d,*m,x

EXCLUSIVE OR MEMORY WORD

8C00



DEFINITION

The word in memory, specified by the Effective Word Address (EWA), is accessed and logically exclusive ORed with the word located in the General Purpose Register (GPR) specified by R. The result is transferred to the GPR specified by R.

SUMMARY EXPRESSION

(EWL) ⊕ (R) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 185BC
Hex Instruction: 8F 81 86 94 (R=7, X=I=0)

Before Execution	PSWR 010185BC	GPR7 13579BDF	Memory Word 18694 22222222
After Execution	PSWR 200185C0	GPR7 3175B9FD	Memory Word 18694 22222222

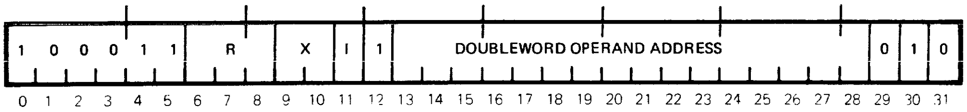
Note

The contents from memory word 18694 are exclusive ORed with the contents from GPR7. The result replaces the contents of GPR7. CC2 is set.

EOMD
d,*m,x

EXCLUSIVE OR MEMORY DOUBLEWORD

8C00



DEFINITION The doubleword in memory, specified by the Effective Doubleword Address (EDA), is accessed and logically exclusive ORed with the doubleword located in the General Purpose Register (GPR) specified by R and R+1. R+1 is one GPR greater than specified by R. The result is transferred to the GPR specified by R and R+1.

SUMMARY EXPRESSION

$(EWL+1) \oplus (R+1) \rightarrow R+1$

$(EWL) \oplus (R) \rightarrow R$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R,R+1) is greater than zero
CC3: (R,R+1) is less than zero
CC4: (R,R+1) is equal to zero

EXAMPLE

Memory Location: 00448
Hex Instruction: 8F 00 05 3A (R=6, X=I=0)

Before Execution

PSWR	GPR6	GPR7
00000448	00FFFF00	00FFF000

Memory Word 00538
482144C0

Memory Word 0053C
2881433A

After Execution

PSWR	GPR6	GPR7
2000044C	48DEBBC0	287EB33A

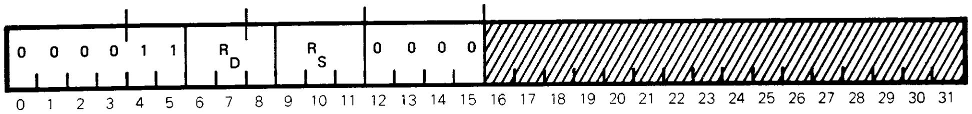
Memory Word 00538
482144C0

Memory Word 0053C
2881433A

Note The contents from memory word 00538 and GPR6 are exclusive ORed and the result is transferred to GPR6. The contents from memory word 0053C and GPR7 are exclusive ORed and the result is transferred to GPR7. CC2 is set.

EXCLUSIVE OR REGISTER AND REGISTER

0C00



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_D is logically exclusive ORed with the word located in the GPR specified by R_S. The result is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S) \oplus (R_D) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

EXAMPLE

Memory Location: 0139E
Hex Instruction: 0F E0 (R_D=7, R_S=6)

	PSWR	GPR6	GPR7
Before Execution	0100139E	33333333	55555555
After Execution	200013A0	33333333	66666666

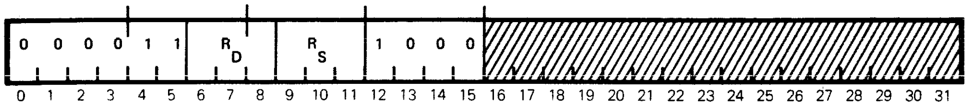
Note

The contents from GPR6 and GPR7 are exclusive ORed and the result is transferred to GPR7. CC2 is set.

EORM
s,d

EXCLUSIVE OR REGISTER AND REGISTER MASKED

0C08



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_D is logically exclusive ORed with the word located in the GPR specified by R_S. The resulting word is then masked (logical AND function) with the contents of the Mask Register, R4. The result is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S) \oplus (R_D)$ & $(R4) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

EXAMPLE

Memory Location: 25A32
Hex Instruction: 0F E8 (R_D=7, R_S=6)

Before Execution	PSWR 00025A32	GPR4 00FEDF00	GPR6 9725A2C8	GPR7 6C248237
After Execution	PSWR 08025A34	GPR4 00FEDF00	GPR6 9725A2C8	GPR7 00000000

Note

The contents from GPR6 and GPR7 are exclusive ORed. The result is ANDed with the contents from GPR4 and transferred to GPR7. CC4 is set.

REGISTER TRANSFER INSTRUCTIONS

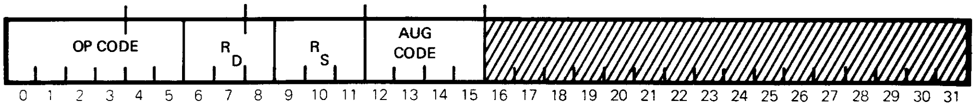
General Description

The Register Transfer Instruction group provides the capability to perform transfer or exchange of information between registers. Provisions have also been made in some instructions to allow two's complement, one's complement, and mask operations to be performed during execution.

Instruction Formats

The following Basic Instruction Format is used by the Register Transfer Instruction group.

Inter-Register



Bits 0-5 define the Operation Code.

Bits 6-8 designate the register to contain the result of the operation.

Bits 9-11 designate the register which contains the source operand.

Bits 12-15 define the Augmenting Operation Code.

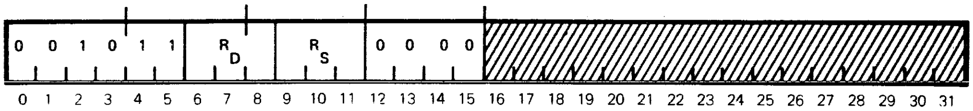
Condition Code Utilization

A Condition Code is set during execution of most Register Transfer Instructions to indicate if the contents of the Destination Register (R_D) are greater than, less than, or equal to zero.

TRR
s,d

TRANSFER REGISTER TO REGISTER

2C00



DEFINITION — The word located in the General Purpose Register (GPR) specified by R_S is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION — $(R_S) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

EXAMPLE

Memory Location: 00206
Hex Instruction: 2C A0 (R_D=1, R_S=2)

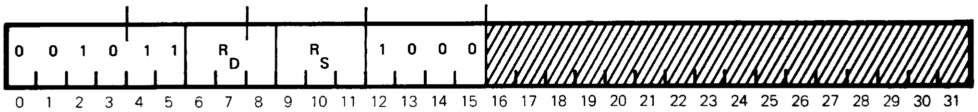
Before Execution	PSWR 00000206	GPR1 00000000	GPR2 00803AB
After Execution	PSWR 20000208	GPR1 00803AB	GPR2 00803AB

Note — The content of GPR2 is transferred to GPR1, and CC2 is set.

TRRM
s,d

TRANSFER REGISTER TO REGISTER MASKED

2C08



DEFINITION. The word located in the General Purpose Register (GPR) specified by R_S is masked (logical AND function) with the contents of the Mask Register R4. The result word is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION $(R_S)\&(R4) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

EXAMPLE

Memory Location: 00206
Hex Instruction: 2C A8 (R_D=1, R_S=2)

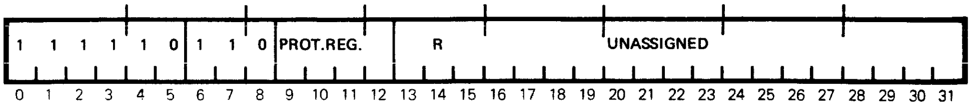
Before Execution	PSWR 00000206	GPR1 00000000	GPR2 000803AB	GPR4 0007FFFD
After Execution	PSWR 20000208	GPR1 000003A9	GPR2 000803AB	GPR4 0007FFFD

Note The content of GPR2 is ANDed with GPR4 and the result is transferred to GPR1. CC2 is set.

TRP
s,p

TRANSFER REGISTER TO PROTECT REGISTER

FB00



DEFINITION

The word located in the General Purpose Register (GPR) specified by R is transferred to the Protect Register specified by the protect register field (bit 9 through bit 12) contained in the Instruction Word (IW). The Protect Register Address is the same as the four high order memory address bits used to specify all memory locations within a given module.

SUMMARY EXPRESSION

(R) → PR

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 0050C
Hex Instruction: FB0F (R=7, Protect Register=1)

Before Execution	PSWR	GPR7	Protect Register 1
	80000050C	0000FFFE	0000
After Execution	PSWR	GPR7	Protect Register 1
	800000510	0000FFFE	FFFE

Note

The content of bits 16-31 of GPR7 is transferred to Protect Register 1. The protection status of Memory Module 1 is established such that a program operating in the unprivileged state can store information only in locations 2000 through 21FF without generating a Privilege Violation trap.

TRANSFER PROTECT REGISTER TO REGISTER

FB80

0 1 2 3 4 5	6 7 8	9 10 11 12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
1 1 1 1 1 0	1 1 1	PROT. REG.	R	UNASSIGNED

DEFINITION

The word located in the Protect Register specified by the protect register field (bit 9 through bit 12) is transferred to the General Purpose Register (GPR) specified by R. The Protect Register Address is the same as the four high order memory address bits used to specify all memory locations within a given module.

SUMMARY EXPRESSION

(PR) → R

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 0050C
Hex Instruction: FB8F (R=7, Protect Register=1)

Before Execution	PSWR 0000050C	GPR7 00000000	Protect Register 1 FFFE
After Execution	PSWR 00000510	GPR7 0000FFFE	Protect Register 1 FFFE

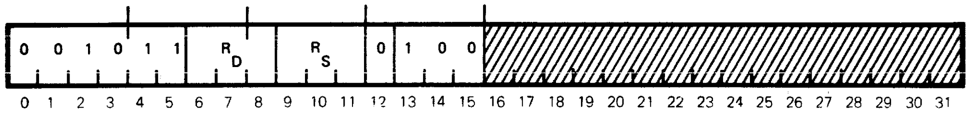
Note

The contents of Protect Register 1 is transferred to bits 16-31 of GPR7. This value defines the protection status of Memory Module 1.

TRN
s,d

TRANSFER REGISTER NEGATIVE

2C04



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_S is two's complemented and transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$-(R_S) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

EXAMPLE

Memory Location: 00AAE
Hex Instruction: 2F E4 (R_D=7, R_S=6)

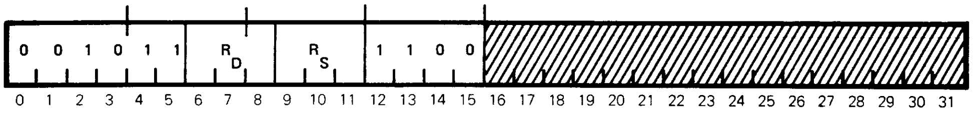
Before Execution	PSWR	GPR6	GPR7
	00000AAE	00000FFF	12345678
After Execution	PSWR	GPR6	GPR7
	10000AB0	00000FFF	FFFFF001

Note

The content of GPR6 is negated and transferred to GPR7. CC3 is set.

TRANSFER REGISTER NEGATIVE MASKED

2C0C



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_S is two's complemented and then masked (logical AND function) with the contents of the Mask Register R4. The result word is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$-(R_S)\&(R4) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

EXAMPLE

Memory Location: 00AAE
Hex Instruction: 2F EC (R_D=7, R_S=6)

Before Execution	PSWR 00000AAE	GPR4 7FFFFFFF	GPR6 00000FFF	GPR7 12345678
After Execution	PSWR 20000AB0	GPR4 7FFFFFFF	GPR6 00000FFF	GPR7 7FFFF001

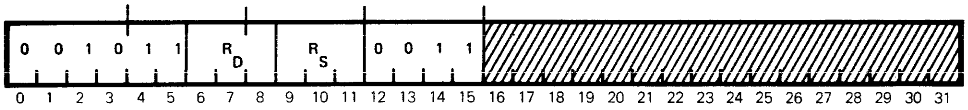
Note

The content of GPR6 is negated; the result is ANDed with the content of GPR4 and transferred to GPR7. CC2 is set.

TRC
s,d

TRANSFER REGISTER COMPLEMENT

2C03



DEFINITION The word located in the General Purpose Register (GPR) specified by R_S is one's complemented and transferred to the GPR specified by R_D.

SUMMARY EXPRESSION $(\overline{R_S}) \rightarrow R_D$

CONDITION CODE RESULTS
CC1: Always zero
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

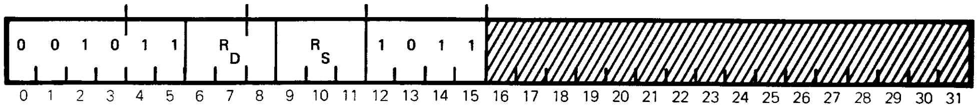
EXAMPLE
Memory Location: 01001
Hex Instruction: 2F E3 (R_D=7, R_S=6)

Before Execution	PSWR 0800100A	GPR6 55555555	GPR7 00000000
After Execution	PSWR 1000100C	GPR6 55555555	GPR7 AAAAAAAA

Note The content of GPR6 is complemented and transferred to GPR7. CC3 is set.

TRANSFER REGISTER COMPLEMENT MASKED

2C0B



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_S is one's complemented and then masked (logical AND function) with the contents of the Mask Register R4. The result is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$\overline{(R_S)}\&(R4) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

EXAMPLE

Memory Location: 0100A
Hex Instruction: 2F EB (R_D=7, R_S=6)

Before Execution	PSWR	GPR4	GPR6	GPR7
	0800100A	00FFFF00	55555555	00000000
After Execution	PSWR	GPR4	GPR6	GPR7
	2000100C	00FFFF00	55555555	00AAAA00

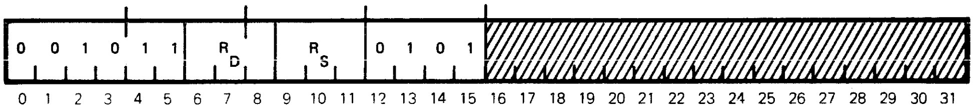
Note

The content of GPR6 is complemented and then ANDed with the content of GPR4. The result is transferred to GPR7, and CC2 is set.

XCR
s,d

EXCHANGE REGISTERS

2C05



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_S is exchanged with the word located in the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S) \rightarrow R_D$

$(R_D) \rightarrow R_S$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Original (R_D) is greater than zero
CC3: Original (R_D) is less than zero
CC4: Original (R_D) is equal to zero

EXAMPLE

Memory Location: 02002
Hex Instruction: 2C A5 (R_D=1, R_S=2)

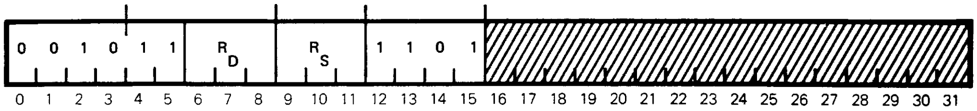
	PSWR	GPR1	GPR2
Before Execution	40002002	00000000	AC8823C1
After Execution	08002004	AC8823C1	00000000

Note

The contents of GPR1 and GPR2 are exchanged. CC4 is set.

EXCHANGE REGISTERS MASKED

2COD



DEFINITION

The contents of the General Purpose Register (GPR) specified by R_S and R_D are each masked (logical AND function) with the contents of the Mask Register R4. The results of both masked operations are exchanged.

SUMMARY EXPRESSION

$(R_S)\&(R4) \rightarrow R_D$

$(R_D)\&(R4) \rightarrow R_S$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Original (R_D) and (R4) is greater than zero
CC3: Original (R_D) and (R4) is less than zero
CC4: Original (R_D) and (R4) is equal to zero

EXAMPLE

Memory Location: 02002
Hex Instruction: 2C AD (R_D=1, R_S=2)

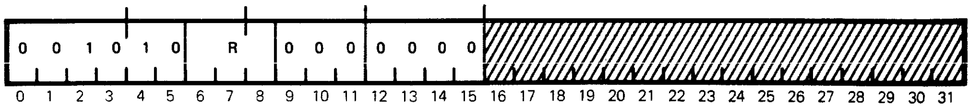
Before Execution	PSWR 40002002	GPR1 6B000000	GPR2 AC8823C1	GPR4 000FFFFF
After Execution	PSWR 08002004	GPR1 000823C1	GPR2 00000000	GPR4 000FFFFF

Note

The contents of GPR1 and GPR2 are each ANDed with the content of GPR4. The results of the masking operation are exchanged and transferred to GPR2 and GPR1, respectively. CC4 is set.

TRANSFER REGISTER TO PSWR

2800



DEFINITION

Bit positions 1 through 30 of the General Purpose Register specified by R are transferred to the corresponding bit positions 1 through 30 of the Program Status Word Register.

SUMMARY EXPRESSION

R_{1-30} $PSWR_{1-30}$

CONDITION CODE RESULTS

CC1: (R_1) is equal to ONE
CC2: (R_2) is equal to ONE
CC3: (R_3) is equal to ONE
CC4: (R_4) is equal to ONE

EXAMPLE

Memory Location: 0069E
Hex Instruction: 28 00 (R=0)

Before Execution	PSWR 6000069E	GPRO A0000B4C
After Execution	PSWR 20000B4C	GPRO A0000B4C

Note

The contents of GPRO, bits 1-30, are transferred to the PSWR, bits 1-30. Bit 0 of GPRO is ignored.

SHIFT OPERATION INSTRUCTIONS

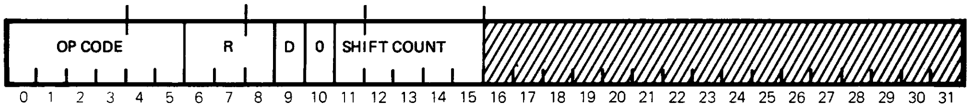
General Description

This group of instructions provides the capability to perform Arithmetic, Logical, and Circular Left or Right shift operations on the contents of words or doublewords contained in General Purpose Registers. Provisions have also been made to allow Normalize operations to be performed on the contents of words or doublewords contained in General Purpose Registers.

Instruction Formats

The following two instruction formats are used by the Shift Instruction group.

Shift Instruction



Bits 0-5 define the Operation Code.

Bits 6-8 designate a General Purpose Register address (0 through 7).

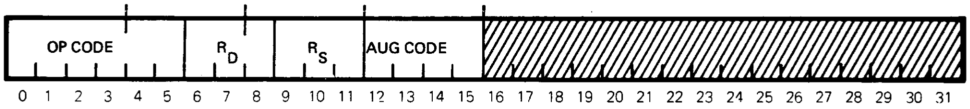
Bit 9 designates direction.

D=1 designates shift left
D=0 designates shift right

Bit 10 unassigned.

Bits 11-15 define the number of shifts to be made.

Inter-Register



Bits 0-5 define the Operation Code.

Bits 6-8 designate the register to contain the result of the operation.

Bits 9-11 designate the register which contains the source operand.

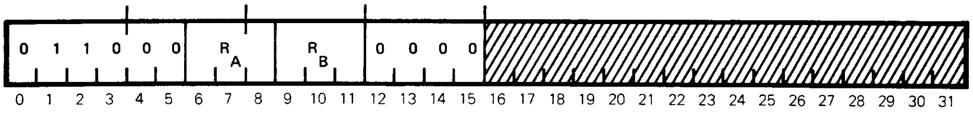
Bits 12-15 define the Augmenting Operation Code.

Condition Code Utilization

Most Shift Instructions leave the Condition Code unchanged.

NORMALIZE

6000



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_A is shifted left, four bit positions at a time, until the contents are normalized for the base 16 exponent [$(1 > (R_A) \geq 1/16)$ the contents of R_A are less than one or equal to or greater than 1/16]. The exponent is set to 40_{16} and is decremented once for each group of four shifts performed. When normalization is complete, the exponent is stored in bit positions 25 through 31 of the GPR specified by R_B. Bit positions 0 through 24 of the GPR specified by R_B are cleared to zeros. If the contents of the GPR specified by R_A are equal to zero, the exponent stored in bit positions 25 through 31 of the GPR specified by R_B will equal zero and no shifting will be performed.

Note

The normalized result must be converted to the format defined on page 5-159 prior to use by the floating-point arithmetic unit or standard FORTRAN floating-point subroutines. In addition, a test must be made for minus full scale (1XXX XXXX 0000 0000 --- 0000) and a conversion made to (1YYY YYYY 1111 0000 --- 0000), where YYY YYYY is one less than XXX XXXX.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 00D32
Hex Instruction: 63 10 (R_A=6, R_B=1)

Before Execution	PSWR	GPR1	GPR6
	20000D32	12345678	0002E915
After Execution	PSWR	GPR1	GPR6
	20000D34	0000003D	2E915000

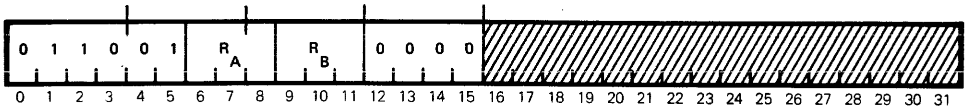
Note

The content of GPR6 is normalized by three left shifts of four bits each. The exponent is determined by decrementing 40_H once for each shift and transferred to GPR1.

NORD
d,s

NORMALIZE DOUBLE

6400



DEFINITION

The doubleword located in the General Purpose Register (GPR) specified by R_A and R_A+1 i shifted left, four bit positions at a time, until the contents are normalized for the base 16 exponent [(1 > $(R_A,R_A+1) \geq 1/16$ the contents of R_A and R_A+1 are less than one or equal to or greater than 1/16]. R_A+1 is GPR one greater than specified by R_A. The exponent of the doubleword is set to 40_{16} and is decremented once for each group of four shifts performed. When normalization is complete, the exponent is stored in bit positions 25 through 31 of the GPR specified by R_B. Bit positions 0 through 24 of the GPR specified by R_B are cleared to zeros. If the content of the doubleword specified by R_A and R_A+1 is equal to zero, the exponent stored in bit positions 25 through 31 of the GPR specified by R_B will equal zero and no shifting will be performed.

Note

The normalized result must be converted to the format defined on page 5-159 prior to use by the floating-point arithmetic unit or standard FORTRAN floating-point subroutines. In addition, a test must be made for minus full scale (1XXX XXXX 0000 0000 --- 0000) and a conversion made to (1YYY YYYY 1111 0000 --- 0000), where YYY YYYY is one less than XXX XXXX.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 0046E
Hex Instruction: 67 10 (R_A=6, R_B=1)

Before Execution	PSWR 1000046E	GPR1 9ABCDEF0	GPR6 FFFFFFFF	GPR7 FF3AD915
After Execution	PSWR 10000470	GPR1 00000037	GPR6 F3AD9150	GPR7 00000000

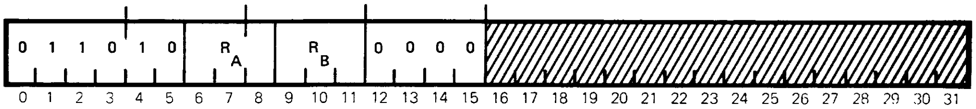
Note

The doubleword obtained from the contents of GPR6 and GPR7 is normalized by nine left shifts of four bit positions each. The result is returned to GPR6 and GPR7, and the exponent (40_H-9) is transferred to GPR1.

SCZ
d,s

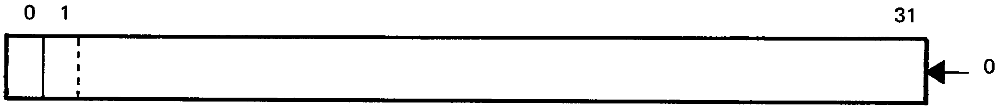
SHIFT AND COUNT ZEROS

6800



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_A is shifted left, one position at a time, until the sign (bit 0) changes from a zero to a one. When the sign bit does change; the contents are shifted left one more bit position and the total number of shifts minus one placed in bit positions 27 through 31 of the GPR specified by R_B. Bit positions 0 through 26 of the GPR specified by R_B are set to zeros. The shift count specifies the most significant bit position 0 through 31 of register R_A that was equal to one.



NOTES

1. If the contents of the GPR specified by R_A are equal to zero, the shift count placed in bit positions 27 through 31 of the GPR specified by R_B is zero and Condition Code bit 4 is set to one.

2. If the sign (bit 0) of the GPR specified by R_A is equal to one, the shift count placed in bit positions 27 through 31 of the GPR specified by R_B is zero and Condition Code Bit 4 is set to zero.

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: R_A 0-31 is equal to zero

EXAMPLE

Memory Location: 0399E
Hex Instruction: 6A 20 (R_A=4, R_B=2)

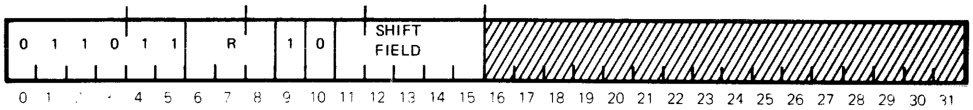
Before Execution	PSWR 2000399E	GPR2 12345678	GPR4 00300611
After Execution	PSWR 000039A0	GPR2 0000000A	GPR4 80308800

Note The content of GPR4 is left shifted 10 bits when 0=1. It is then shifted one more place, and the zero count of 10 (A_H) is transferred to GPR2.

SLA
d,v

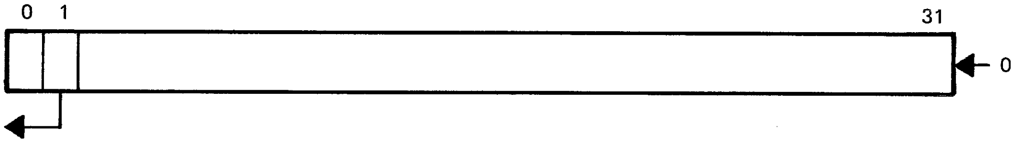
SHIFT LEFT ARITHMETIC

6C40



DEFINITION

Bit positions 1 through 31 of the General Purpose Register (GPR) specified by R are shifted left the number of bit positions specified by the shift field (bit 11 through bit 15) contained in the instruction word. Bit position 0 (sign bit) of the GPR specified R remains unchanged. Condition Code bit 1 is set to one if any bit shifted out of position 1 differs from the sign bit position 0.



CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: Always 0
CC3: Always 0
CC4: Always 0

EXAMPLE

Memory Location: 00106
Hex Instruction: 6F 4C (R=6, Shift Count=12_{10})

Before Execution	PSWR 10000106	GPR6 000013AD
After Execution	PSWR 00000108	GPR6 013AD000

Note

The content of GPR6 is left shifted twelve bit positions with zeros filled from the right. The result is transferred to GPR6.

EXAMPLE 2

Memory Location: 00106
Hex Instruction: 6F 4C (R=6, Shift Count=12_{10})

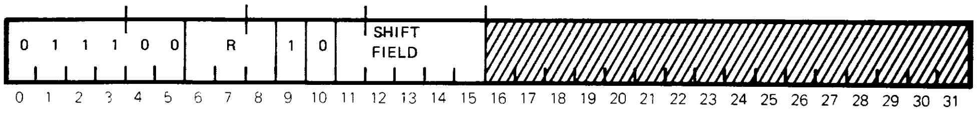
Before Execution	PSWR 10000106	GPR6 001FAD58
After Execution	PSWR 40000108	GPR6 7AD58000

Note

Overflow occurs and is indicated by CC1.

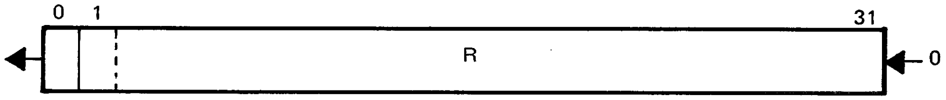
SHIFT LEFT LOGICAL

7040



DEFINITION

The word located in the General Purpose Register (GPR) specified by R is shifted left the number of bit positions specified by the shift field (bit 11 through bit 15) contained in the instruction word.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 00812
Hex Instruction: 73 D4 (R=7, Shift Count=20_{10})

Before Execution

PSWR A0000812
GPR7 12345678

After Execution

PSWR A0000814
GPR7 67800000

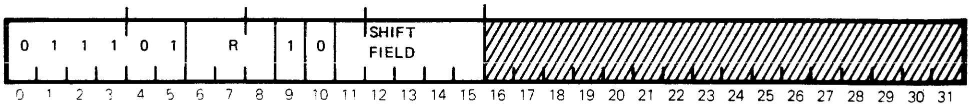
Note

The content of GPR7 is left shifted 20 bits and replaced.

SLC
d,v

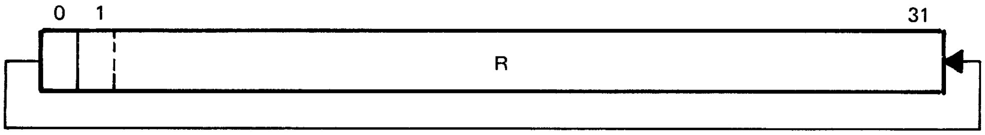
SHIFT LEFT CIRCULAR

7440



DEFINITION

The word located in the General Purpose Register (GPR) specified by R is shifted left the number of bit positions specified by the shift field (bit 11 through bit 15) contained in the instruction word. Bits shifted out of bit position 0 are shifted into bit position 31.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:	001FA
Hex Instruction:	77 CF (R=7, Shift Field=16_{10})

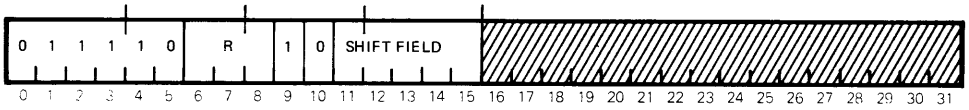
Before Execution	PSWR 000001FA	GPR7 12345678
After Execution	PSWR 000001FC	GPR7 56781234

Note

The content of GPR7 is shifted left in a circular manner for 16 bit positions.

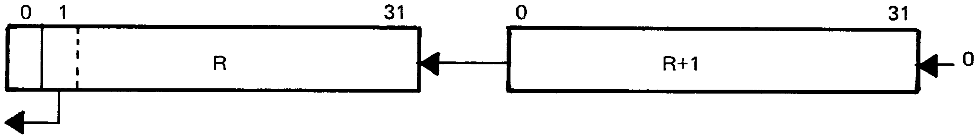
SHIFT LEFT ARITHMETIC DOUBLE

7840



DEFINITION

The doubleword located in the General Purpose Register (GPR) specified by R and R+1 is shifted left the number of bit positions specified by the shift field (bit 11 through bit 15) contained in the instruction word. R+1 is GPR one greater than specified by R. The sign (bit 0) of the GPR specified by R remains unchanged. Condition Code bit 1 is set to ONE if any bit shifted out of position 1 differs from the sign bit, position 0.



CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: Always zero
CC3: Always zero
CC4: Always zero

EXAMPLE

Memory Location:	02DF6
Hex Instruction:	7A 58 (R=4, Shift Field=24_{10})

Before Execution	PSWR 80002DF6	GPR4 FFFFFFA3	GPR5 9A178802
After Execution	PSWR 80002DF8	GPR4 A39A1788	GPR5 02000000

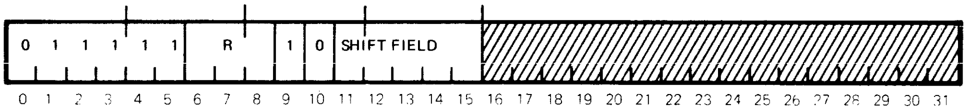
Note

The doubleword obtained from the contents of GPR4 and GPR5 is left-shifted 24 bit positions, with zeros filled from the right. The result is returned to GPR4 and GPR5.

SLLD
d,v

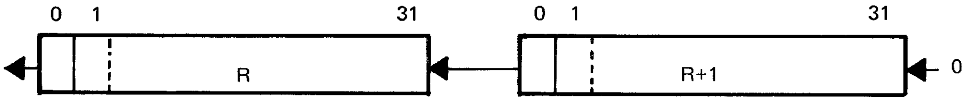
SHIFT LEFT LOGICAL DOUBLE

7C40



DEFINITION

The doubleword located in the General Purpose Register (GPR) specified by R and R+1 is shifted left the number of bit positions specified by the shift field (bit 11 through bit 15) contained in the instruction word. R+1 is GPR one greater than specified by R.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: NO change

EXAMPLE

Memory Location: 001FE
Hex Instruction: 7F 58 (R=6, Shift Field=24)

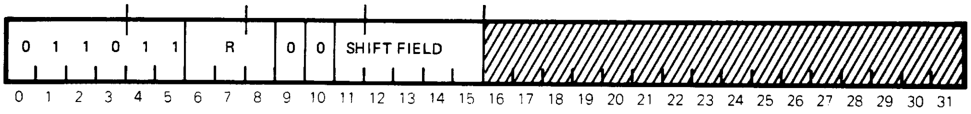
Before Execution	PSWR 100001FE	GPR6 01234567	GPR7 89ABCDEF
After Execution	PSWR 10000200	GPR6 6789ABCD	GPR7 EF000000

Note

The doubleword obtained from GPR6 and GPR7 is left-shifted 24 bit positions with zeros filled from the right. The result is returned to GPR6 and GPR7.

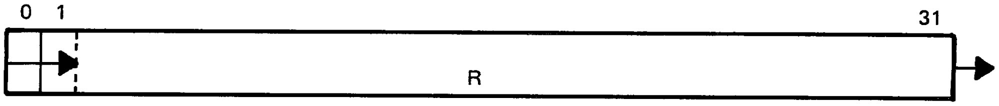
SHIFT RIGHT ARITHMETIC

6C00



DEFINITION

The word located in the General Purpose Register (GPR) specified by R is shifted right the number of bit positions specified by the shift field (bit 11 through bit 15) contained in the instruction word. Bit position 0 (sign bit) is shifted into bit position 1 on each shift. The sign bit (bit 0) remains unchanged.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location:	00372
Hex Instruction:	6D 0A (R=4, Shift Field=10_{10})

Before Execution	PSWR 10000372	GPR4 B69825F1
After Execution	PSWR 10000374	GPR4 FFEDA609

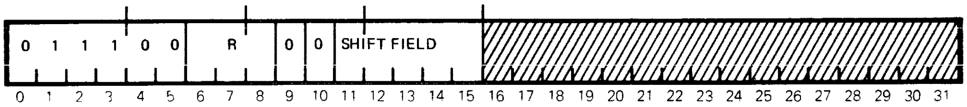
Note

The content of GPR4 is shifted right 10 bit positions. Since that value is negative, a one is entered into bit position one with each shift.

SRL
d,v

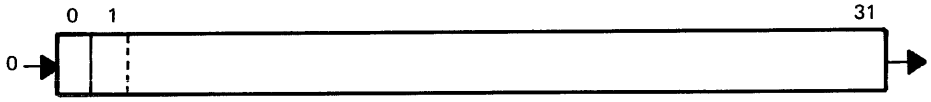
SHIFT RIGHT LOGICAL

7000



DEFINITION

The word located in the General Purpose Register (GPR) specified by R is shifted right the number of bit positions specified by the shift field (bit 11 through bit 15) contained in the instruction word.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 00372
Hex Instruction: 72 0A (R=4, Shift Field=10_{10})

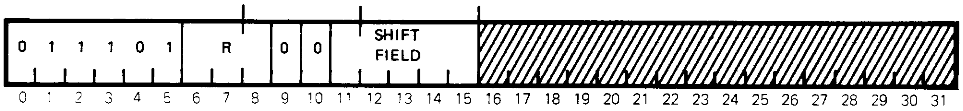
	PSWR	GPR4
Before Execution	10000372	B69825F1
After Execution	10000374	002DA609

Note

The content of GPR4 is shifted right 10 bit positions, with zeros filled from the left.

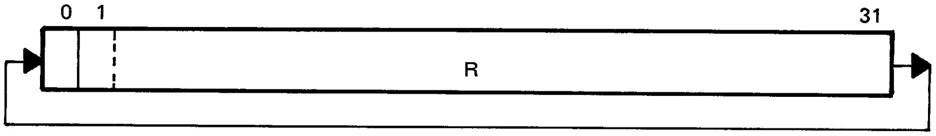
SHIFT RIGHT CIRCULAR

7400



DEFINITION

The word located in the General Purpose Register (GPR) specified by R is shifted right the number of bit positions specified by the shift field (bit 11 through bit 15) contained in the instruction word. Bits shifted out of bit position 31 are shifted into bit position 0.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 00372
Hex Instruction: 76 0C (R=4, Shift Field=12_{10})

Before Execution	PSWR	GPR4
	20000372	01234567
After Execution	PSWR	GPR4
	20000374	56701234

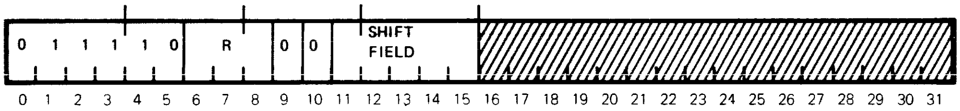
Note

The content of GPR4 is shifted right by 12 bit positions in a circular manner and replaced in GPR4.

SRAD
d,v

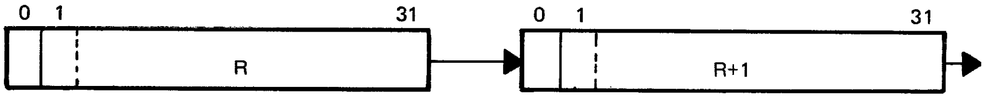
SHIFT RIGHT ARITHMETIC DOUBLE

7800



DEFINITION

The doubleword located in the General Purpose Register (GPR) specified by R and R+1 is shifted right the number of bit positions specified by the shift field (bit 11 through bit 15) contained in the instruction word. R+1 is GPR one greater than specified by R. The sign (bit 0) of the GPR specified by R remains unchanged.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 02B46
Hex Instruction: 7B 18 (R=6, Shift Field=24_{10})

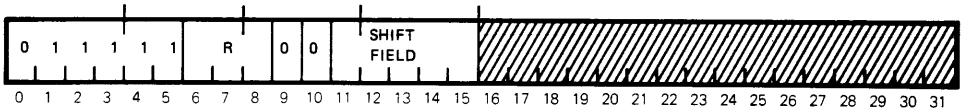
Before Execution	PSWR	GPR6	GPR7
	20002B46	8E2A379B	58C1964D
After Execution	PSWR	GPR6	GPR7
	20002B48	FFFFFF8E	2A379B58

Note

The doubleword obtained from the contents of GPR6 and GPR7 is shifted right 24 bit positions, with the sign extended 24 bits from the left. The result is transferred to GPR6 and GPR7.

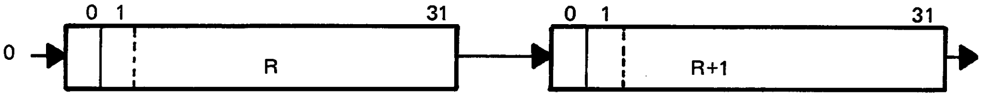
SHIFT RIGHT LOGICAL DOUBLE

7C00



DEFINITION

The doubleword located in the General Purpose Register (GPR) specified by R and R+1 is shifted right the number of bit positions specified by the shift field (bit 11 through bit 15) contained in the instruction word. R+1 is GPR one greater than specified by R.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

EXAMPLE

Memory Location: 02B46
Hex Instruction: 7F 18 (R=6, Shift Field=24_{10})

Before Execution	PSWR 20002B46	GPR6 8E2A379B	GPR7 58C1964D
After Execution	PSWR 20002B48	GPR6 0000008E	GPR7 2A379B58

Note

The doubleword obtained from the contents of GPR6 and GPR7 is shifted right 24 bit positions, with zeros filled from the left. The result is transferred to GPR6 and GPR7.

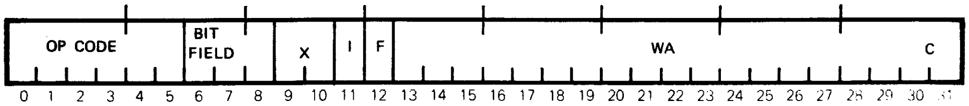
BIT MANIPULATION INSTRUCTIONS

General Description

The Bit Manipulation Instruction group provides the capability to SET, ZERO, or ADD a bit to a specified bit location within a specified byte of a memory location or General Purpose Register. Provisions have also been made to test a bit in memory or a General Purpose Register by transferring the contents of that bit position to the Condition Code Register.

The Bit Manipulation Instruction group uses the following two instruction formats.

Memory Reference



Bits 0-5 define the Operation Code.

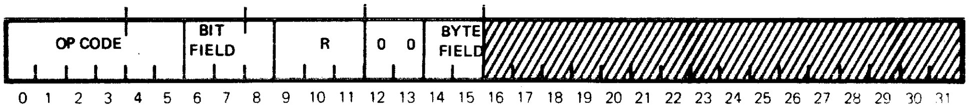
Bits 6-8 specify a bit (0 through 7).

Bits 9-10 designate one of three Index Registers.

Bit 11 indicates if an indirect addressing operation is to be performed.

Bits 12-31 specify the address of the operand when X and I fields are equal to zero.

Inter-Register



Bits 0-5 define the Operation Code.

Bits 6-8 specify a bit (0 through 7).

Bits 9-11 designate a General Purpose Register address (0 through 7).

Bits 12-13 unassigned

Bits 14-15 specify a byte (0 through 3).

Condition Code Utilization

A Condition Code is set during execution of Set Bit, Zero Bit, and Test Bit operations if the bit being operated on was equal to one. During Add Bit operations, a Condition Code is set to indicate if the execution of the instruction caused an arithmetic exception, a greater than zero, less than zero, or equal to zero result.

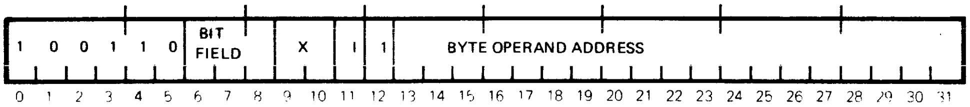
Interprocessor Semaphores

When two processors share memory and other resources, it is necessary that a simple positive method be provided for dynamically reserving/releasing shared memory pages and the other shared resources. The Set Bit or Zero Bit instructions (SBM, SBR, ZBM, ZBR) are used for this purpose. If both processors attempt to set (or zero) the same semaphore bit at the same time, one processor will actually access the memory location before the other processor by virtue of the shared memory bus design. The first processor to access the bit will copy the previous contents of the bit into its condition code register before setting (or clearing) the bit. On the very next memory cycle the other processor will copy the state of the bit as set by the first processor into its condition code register and then set (or clear) the bit again. Both processors then execute Branch on Condition Code instructions to test the status of the bit prior to changing it. The first processor will find the bit previously not set (or set) indicating that it was able to reserve the resource which the user has associated with the bit. The second processor will find the bit already set (or not set) indicating that the resource is currently reserved by the other processor and that subsequent attempts should be made.

SBM
c,*m,x

SET BIT IN MEMORY

9808



DEFINITION

The byte in memory, specified by the Effective Byte Address (EBA), is accessed and the specified bit (bit field) within the byte set to a one. All other bits within the byte remain unchanged. The resulting byte is replaced in the location specified by the EBA. Condition Code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2 and the original status of the specified bit of the byte specified by the EBA is transferred to CC1.

NOTE

Since the contents of the condition code register are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the general-purpose registers can be stored in the condition code register for a combined conditional branch test.

SUMMARY EXPRESSION

(CC3) → CC4
(CC2) → CC3
(CC1) → CC2
(EBL_{SBL}) → CC1
1 EBL_{SBL}

CONDITION CODE RESULTS

CC1: Equal to ONE, if EBL_{SBL}=1
CC2: Equal to ONE, if CC1 was 1
CC3: Equal to ONE, if CC2 was 1
CC3: Equal to ONE, if CC3 was 1

EXAMPLE

Memory Location: 01000
Hex Instruction: 98 88 14 03 (bit field = 1)

Before Execution

PSWR 20001000
Memory Byte 01403 1A

After Execution

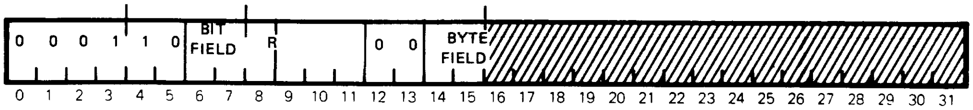
PSWR 10001004
Memory Byte 01403 5A

Note

Bit 1 of memory byte 01403 is set to a one.

SET BIT IN REGISTER

1800



DEFINITION

The specified bit (bit field) of the specified byte (byte field) in the General Purpose Register (GPR) specified by R is set to a one. All other bits, within the GPR specified by R, remain unchanged. Condition Code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2 and the original status of the specified bit of the specified byte in register R is transferred to CC1.

NOTE

Since the contents of the condition code register are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the general-purpose registers can be stored in the condition code register for a combined conditional branch test.

SUMMARY EXPRESSION

$(CC3) \rightarrow CC4$
$(CC2) \rightarrow CC3$
$(CC1) \rightarrow CC2$
$(R_{SBL}) \rightarrow CC1$
$1 \rightarrow EBL_{SBL}$

CONDITION CODE RESULTS

CC1: Equal to ONE, if $R_{SBL} = 1$
CC2: Equal to ONE, if CC1 was 1
CC3: Equal to ONE, if CC2 was 1
CC4: Equal to ONE, if CC3 was 1

EXAMPLE

Memory Location:	01002
Hex Instruction:	XXXX1B 82 (bit field=7, R=0, byte field=2)

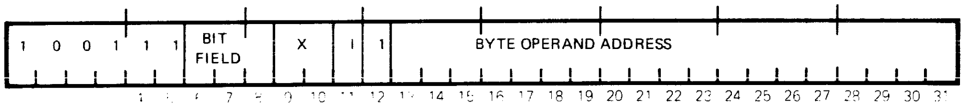
Before Execution	PSWR	GPR0
	10001002	0374B891
After Execution	PSWR	GPR0
	08001004	0374B991

Note Bit 23 of GPR0 is set to one.

ZBM
c,*m,x

ZERO BIT IN MEMORY

9C08



DEFINITION

The byte in memory, specified by the Effective Byte Address (EBA), is accessed and the specified bit (bit field) within the byte set to a zero. All other bits within the byte remain unchanged. The resulting byte is replaced in the location specified by the EBA. Condition Code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2 and the original status of the specified bit of the byte specified by the EBA is transferred to CC1.

Note

Since the contents of the condition code register are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the general-purpose registers can be stored in the condition code register for a combined conditional branch test.

SUMMARY EXPRESSION

$(CC3) \rightarrow CC4$
$(CC2) \rightarrow CC3$
$(CC1) \rightarrow CC2$
$(EBL_{SBL}) \rightarrow CC1$
$0 \rightarrow EBL_{SBL}$

CONDITION CODE RESULTS

CC1: Equal to ONE, if $EBL_{SBL} = 1$
CC2: Equal to ONE, if CC1 was 1
CC3: Equal to ONE, if CC2 was 1
CC4: Equal to ONE, if CC3 was 1

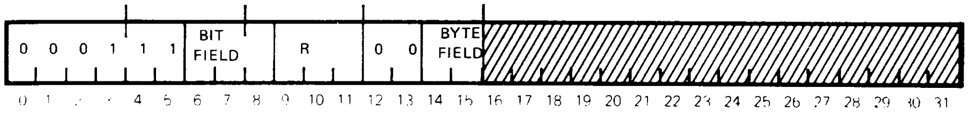
EXAMPLE

Memory Location: 1F684
Hex Instruction: 9E 8A 01 22 (bit field=5)

Before Execution	PSWR 1001F684	Memory Byte 20122 34
After Execution	PSWR 4801F688	Memory Byte 20122 30

ZERO BIT IN REGISTER

1C00



DEFINITION The specified bit (bit field) of the specified byte (byte field) in the General Purpose Register (GPR) specified by R is set a zero. All other bits within the GPR specified by R remain unchanged. Condition Code bit (CC3) is transferred to CC2 and the original status of the specified bit of the specified byte in register R is transferred to CC1.

NOTE Since the contents of the condition code are shifted to the next highest position before the bit is loaded into CC1, any four bits in memory or the general-purpose registers can be stored in the condition code register for a combined conditional branch test.

SUMMARY EXPRESSION

(CC3) → CC4
(CC2) → CC3
(CC1) → CC2
$(R_{SBL}) \rightarrow CC1$
$0 \rightarrow EBL_{EBL}$

CONDITION CODE RESULTS

CC1: Equal to ONE, if $R_{SBL} = 1$
CC2: Equal to ONE, if CC1 was 1
CC3: Equal to ONE, if CC2 was 1
CC4: Equal to ONE, if CC3 was 1

EXAMPLE

Memory Location: 00C56
Hex Instruction: 1C51 (bit field=0, R=5, byte field=1)

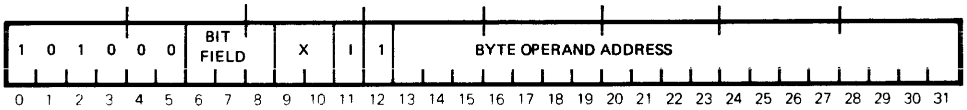
	PSWR	GPR5
Before Execution	10000C56	76A43B19
After Execution	48000C58	76243B19

Note Bit 8 of GPR5 is cleared to zero. CC4 is set.

ABM
c,*m,x

ADD BIT IN MEMORY

A008



DEFINITION

The byte in memory, specified by the Effective Byte Address (EBA), is accessed and a ONE added to the bit position specified by the bit field. The addition is performed on the entire memory word containing the byte specified by the EBA. Therefore, a carry may be propagated left, to the sign bit. The resulting word is transferred to the memory word location containing the byte specified by the EBA.

SUMMARY EXPRESSION

$(EBL)+1_{SBL} \rightarrow EBL$

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: (EWL) is greater than zero
CC3: (EWL) is less than zero
CC4: (EWL) is equal to zero

EXAMPLE

Memory Location: 03000
Hex Instruction: A2 08 31 92 (bit field=4, X=I=0)

Before Execution	PSWR 00003000	Memory Word 03190 51A3F926
After Execution	PSWR 20003004	Memory Word 03190 51A40126

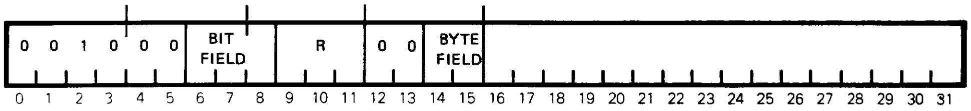
Note

A one is added to bit position 20_{10} of memory word 03190 (byte 2, bit 4) which propagates a carry left to bit position 13_{10}. The result is returned to memory word 03190 and CC2 is set.

ABR
d,b

ADD BIT IN REGISTER

2000



DEFINITION

A ONE is added to the specified bit (bit field) of the specified byte (byte field) in the General Purpose Register (GPR) specified by R. The addition is performed on the entire word of the GPR specified by R. Therefore, a carry may be propagated left to the sign bit. The result is then transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(R)+1_{SBL} \rightarrow R$

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 0184E
Hex Instruction: 21 61 (bit field=2, R=6, byte field=1)

	PSWR	GPR6
Before Execution	0800184E	3BE9AC48
After Execution	20001850	3C09AC48

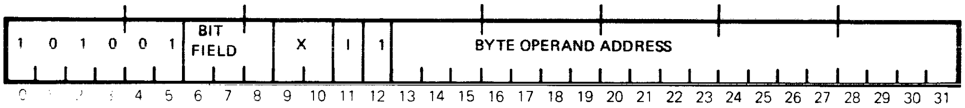
Note

A one is added to bit position 10_{10} to the contents of GPR6 and the result is replaced in GPR6. CC2 is set.

TBM
c,*m,x

TEST BIT IN MEMORY

A408



DEFINITION

The specified bit in memory is transferred to the Condition Code register. Condition Code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2 and the specified bit (bit field) of the byte, specified by the Effective Byte Address (EBA), is transferred to CC1.

NOTE

Since the contents of the Condition Code register are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the general-purpose registers can be stored in the Condition Code register for a combined conditional branch test.

SUMMARY EXPRESSION

$(CC3) \rightarrow CC4$
$(CC2) \rightarrow CC3$
$(CC1) \rightarrow CC2$
$(EBL_{SBL}) \rightarrow CC1$

CONDITION CODE RESULTS

CC1: R_{SBL} is equal to ONE
CC2: CC1 was equal to ONE
CC3: CC2 was equal to ONE
CC4: CC3 was equal to ONE

EXAMPLE

Memory Location:	05A38
Hex Instruction:	A6 08 5B 21 (bit field=4, X=I=0)

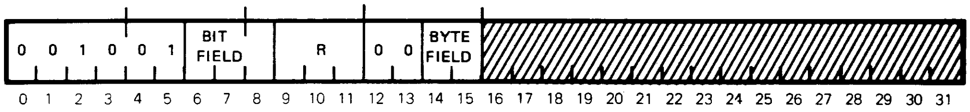
Before Execution	PSWR 10005A38	Memory Byte 05B21 29
After Execution	PSWR 48005A3C	Memory Byte 05B21 29

Note

Bit 4 of memory byte 05B21 is transferred to CC1. CC3 is transferred to CC4.

TEST BIT IN REGISTER

2400



DEFINITION The specified bit in the General Purpose Register (GPR) specified by R is transferred to the Condition Code register. Condition Code bit 3 (CC3) is transferred to CC4, (CC2) is transferred to CC3, (CC1) is transferred to CC2 and the specified bit (bit field), of the specified byte (byte field) in the GPR specified by R, is transferred to CC1.

NOTE Since the contents of the Condition Code register are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the general-purpose registers can be stored in the Condition Code register for a combined conditional branch test.

SUMMARY EXPRESSION
(CC3) → CC4
(CC2) → CC3
(CC1) → CC2
(R_{SBL}) → CC1

CONDITION CODE RESULTS
CC1: R_{SBL} was equal to ONE
CC2: CC1 was equal to a ONE
CC3: CC2 was equal to a ONE
CC4: CC3 was equal to a ONE

EXAMPLE
Memory Location: 01982
Hex Instruction: 25 D3 (bit field=3, R=5, byte field=3)

Before Execution	PSWR 18001982	GPR5 81A2C64D
After Execution	PSWR 08001984	GPR5 81A2C64D

Note CC2 through CC4 is right shifted by one bit position. CC1 is cleared to zero since bit 27_{10} of GPR5 is zero.

FIXED-POINT ARITHMETIC INSTRUCTIONS

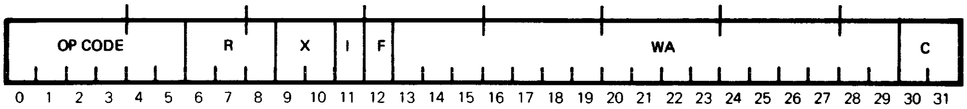
General Description

The Fixed-Point Arithmetic group is used to perform add, subtract, multiply, divide, and sign control functions on bytes, halfwords, words, and doublewords contained in memory and general purpose registers. Provisions have also been made to allow the result of a register-to-register add or subtract to be masked before final storage.

Instruction Formats

The Fixed-Point Arithmetic instructions use the following three instruction formats.

Memory Reference



Bits 0-5 define the Operation Code.

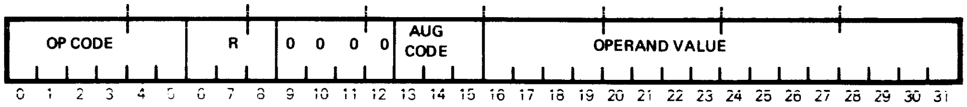
Bits 6-8 designate a General Purpose Register address (0 through 7).

Bits 9-10 designate one of three Index Registers.

Bit 11 designates if an Indirect Addressing operation is to be performed.

Bits 12-31 specify the address of the operand when X and I fields are equal to zero.

Immediate



Bits 0-5 define the Operation Code.

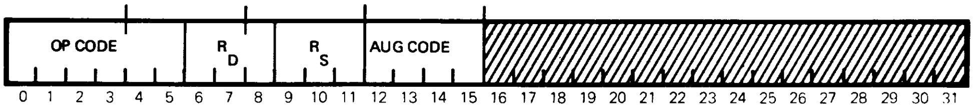
Bits 6-8 designate a General Purpose Register address (0 through 7).

Bits 9-12 unassigned.

Bits 13-15 defines Augmenting Operation Code.

Bits 16-31 contain the 16-bit operand value.

Inter-Register



Bits 0-5 define the Operation Code.

Bits 6-8 designate the register to contain the result of the operation.

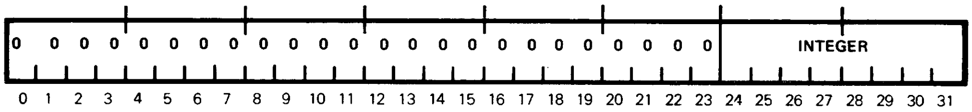
Bits 9-11 designate the register which contains the source operand.

Bits 12-15 define the Augmenting Operation Code.

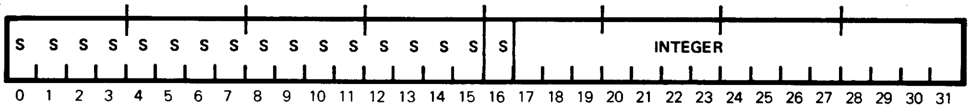
Data Formats

The Fixed-Point Arithmetic Instructions use the following data formats.

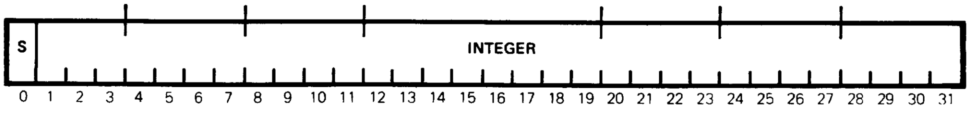
Byte



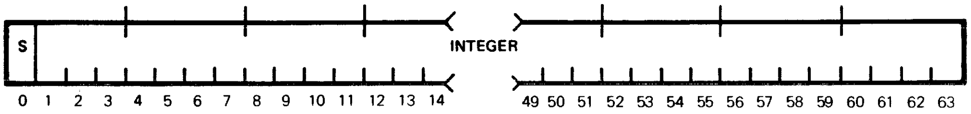
Halfword
(Sign Extended)



Fullword



Doubleword



Condition Code Utilization

Execution of most Fixed-Point Arithmetic Instructions causes a condition code to be set to indicate if the result of the operation was greater than, less than, or equal to zero. Arithmetic exceptions produced by an arithmetic operation are also reflected by the condition code results.

Treatment of Signed Numbers

To perform logical operations, the hardware interprets operands as logical words. For fixed point arithmetic operations, operands are treated as unsigned numbers. Logical and arithmetic operations can be performed on any of the data types available in the SEL 32, bytes, 16-bit halfwords, 32-bit words, and 64-bit doublewords. A program executing on the SEL 32, however, has the opportunity to interpret any of the available data types as a two's complement notation number. It is a property of two's complement arithmetic that operations on signed numbers using two's complement conversions are identical to operations on unsigned numbers; in other words, the hardware treats the sign as the most significant magnitude bit. Consider a general purpose register that contains:

1	0	0	0	0	0	0	1

As an unsigned number, this would be equivalent to:

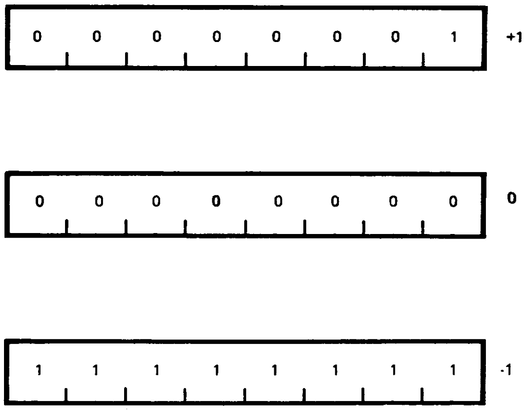
$$81_{16} \quad = \quad 129_{10}$$

whereas, interpreted as a signed number using two's complement notation, it would be:

$$7E_{16} \quad = \quad 126_{10}$$

Insofar as processor operation is concerned, it makes no difference which way the programmer interprets data. However, the programmer is aided in the use of two's complement notation by the Condition Code (CC) bits of the program Status Word (PSW), which are generally set based on two's complement notation.

Numbers in two's complement notation are symetrical in magnitude around a zero representation so all even numbers, both positive and negative, will end in 0, the all odd numbers in 1 (binary word containing all 1's represents - 1).



If one's complement were used for negatives, a negative number could be read by attaching significance to the 0's instead of the 1's.

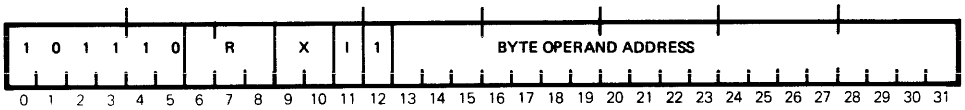
In two's complement notation, each number is one greater than the complement of the positive number of the same magnitude, so a negative number can be read by attaching significance to the rightmost 1 and to the zeros to the left of it (the negative number of the largest magnitude has a 1 only in the sign position). Assuming a binary integer, 1's may be discarded at the left in a negative integer in the same way that leading zeros may be dropped from a positive integer.

Associated with the arithmetic logic unit is a four bit condition code register, which forms the CC portion of the PSW. These CC bits are altered during all arithmetic/logical operations and data transfers. The CC bits indicate such conditions as arithmetic exception, overflow, zero, and positive or negative magnitude.

ADMB
d,*m,x

ADD MEMORY BYTE

B808



DEFINITION The byte in memory, specified by the Effective Byte Address (EBA), is accessed and 24 zeros appended to the most significant end to form a word. This word is algebraically added to the contents of the General Purpose Register (GPR) specified by R. The result word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION 0_{0-23},(EBL)+(R) → R

CONDITION CODE RESULTS
CC1: Arithmetic exception
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE
Memory Location: 00800
Hex Instruction: BA 08 09 15 (R-4, X=I=0)

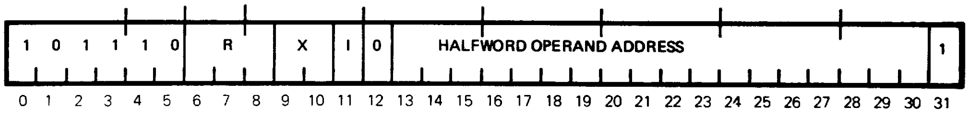
Before Execution	PSWR 10000800	GPR4 00000099	Memory Byte 00915 8A
After Execution	PSWR 20000804	GPR4 00000123	Memory Byte 00915 8A

Note The contents from memory byte 00915, with zeros prefixed, are added to the contents from GPR4, and the result is transferred to GPR4. CC2 is set.

ADMH
d,*m,x

ADD MEMORY HALFWORD

B800



DEFINITION

The halfword in memory, specified by the Effective Halfword Address (EHA), is accessed and the sign (bit 16) extended 16 bits to the left to form a word. This word is algebraically added to the contents of the General Purpose Register (GPR) specified by R. The result word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(EHL)_{SE}+(R) \rightarrow R$

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 40D68
Hex Instruction: BB 84 10 97 (R=7, X=I=0)

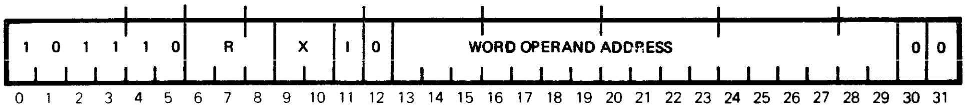
Before Execution	PSWR 20040D68	GPR7 000006C4	Memory Halfword 41096 8C42
After Execution	PSWR 10040D6C	GPR7 FFFF9306	Memory Halfword 41096 8C42

Note

The contents from memory halfword 41096, with sign extension are added to the contents from GPR7, and the result replaces the contents from GPR7. CC3 is set.

ADD MEMORY WORD

B800



DEFINITION
The word in memory, specified by the Effective Word Address (EWA), is accessed and algebraically added to the contents of the General Purpose Register (GPR) specified by R. The result word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION
(EWL)+(R) → R

CONDITION CODE RESULTS
CC1: Arithmetic exception
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE
Memory Location: 00D50
Hex Instruction: BB 00 11 AC (R=6, X=I=0)

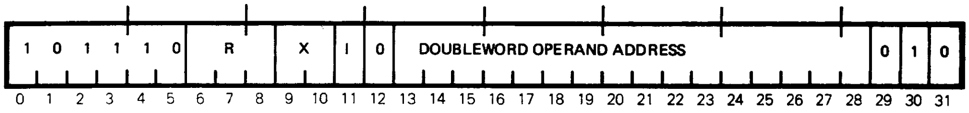
	PSWR	GPR6	Memory Word 011AC
Before Execution	40000D50	0037C1F3	004FC276
After Execution	20000D54	00878469	004FC276

Note
The contents from memory word 011AC are added to the contents from GPR6. The result is transferred to GPR6, and CC2 is set.

ADMD
d,*m,x

ADD MEMORY DOUBLEWORD

B800



DEFINITION

The doubleword in memory, specified by the Effective Doubleword Address (EDA), is accessed and algebraically added to the contents of the General Purpose Register (GPR) specified by R and R+1. R+1 is GPR one greater than specified by R. The contents of the GPR specified by R+1 are added to the contents of the least significant word of the doubleword first. The contents of the GPR specified by R are added to the contents of the most significant word of the doubleword last. The result doubleword is transferred to the GPR specified by R and R+1.

SUMMARY EXPRESSION

(EWL + 1) + (R+1)   R+1 + Carry

(EWL) + (R) + Carry   R

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: (R, R+1) is greater than zero
CC3: (R, R+1) is less than zero
CC4: (R, R+1) is equal to zero

EXAMPLE

Memory Location: 08E3C
Hex Instruction: BA 00 92 52 (R=4, X=I=0)

Before Execution

PSWR	GPR4	GPR5
08008E3C	000298A1	815BC63E

Memory Word 09250	Memory Word 09254
3B69A07E	7F3549A4

After Execution

PSWR	GPR4	GPR5
20008E40	3B6C3920	00913FE2

Memory Word 09250	Memory Word 09254
3B69A07E	7F3579A4

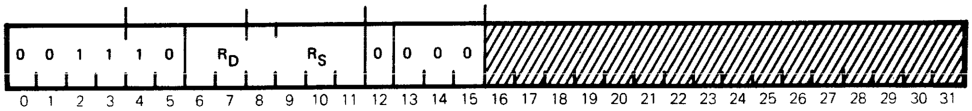
Note

The doubleword obtained from the contents from memory words 09250 and 09254 is added to the doubleword obtained from the contents from GPR4 and GPR5. The result is transferred to GPR4 and GPR5, and CC2 is set.

ADR
s,d

ADD REGISTER TO REGISTER

3800



DEFINITION The word located in the General Purpose Register (GPR), specified by R_D, is algebraically added to the word located in the GPR specified by R_S. The result word is then transferred to the GPR specified by R_D.

SUMMARY EXPRESSION $(R_S+R_D) \rightarrow R_D$

CONDITION CODE RESULTS
CC1: Arithmetic exception
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

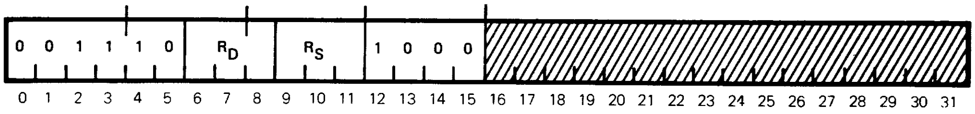
EXAMPLE
Memory Location: 03FA2
Hex Instruction: 3B 70 (R_D=6, R_S=7)

Before Execution	PSWR 08003FA2	GPR6 FF03C67D	GPR7 045C6E3F
After Execution	PSWR 20003FA4	GPR6 036034BC	GPR7 045C6E3F

Note The contents from GPR6 and GPR7 are added and the result is transferred to GPR6. CC2 is set.

ADD REGISTER TO REGISTER MASKED

3808



DEFINITION The word located in the General Purpose Register (GPR) specified by R_D is algebraically added to the word located in the GPR specified by R_S. The sum of this addition is masked (Logical AND Function) with the contents of the mask register R4. The result word is then transferred to the GPR specified by R_D.

SUMMARY EXPRESSION $(R_S)+(R_D)\ \&(R4) \rightarrow R_D$

CONDITION CODE RESULTS
CC1: Arithmetic exception
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

EXAMPLE
Memory Location: 16A9A
Hex Instruction: 3B 78 (R_D=6, R_S=7)

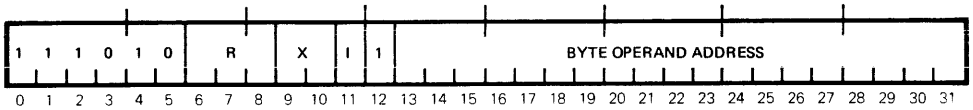
Before Execution	PSWR 40016A9A	GPR4 007FFFFC	GPR6 004FC276	GPR7 0037C1F3
After Execution	PSWR 20016A9C	GPR4 0007FFFC	GPR6 00078468	GPR7 0037C1F3

Note The contents from GPR6 and GPR7 are added; the result is ANDed with the contents of GPR4 and transferred to GPR6. CC2 is set.

ARMB
s,*m,x

ADD REGISTER TO MEMORY BYTE

E808



DEFINITION The byte in memory specified by the Effective Byte Address (EBA) is accessed and algebraically added to the least significant byte (bit 24 through 31) of the General Purpose Register (GPR) specified by R. The result is then transferred to the memory byte location specified by the EBA. The other three bytes in the word which contains the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION $(R_{24-31})+(EBL) \rightarrow EBL$

CONDITION CODE RESULTS
CC1: Undefined
CC2: Undefined
CC3: Undefined
CC4: (EBL) is equal to zero

EXAMPLE
Memory Location: 01A64
Hex Instruction: EB 08 1A 97 (R=6, X=I=0)

Before Execution	PSWR	GPR6	Memory Byte 01A97
	00001A64	0000004A	39
After Execution	PSWR	GPR6	Memory Byte 01A97
	00001A68	0000004A	83

Note The contents from GPR6, bits 24-31, and memory byte 01A97 are added and the result is transferred to memory byte 01A97.

ADD REGISTER TO MEMORY HALFWORD

E800

0	1	2	3	4	5	6–8	9–10	11	12	13–31
1	1	1	0	1	0	R	X	I	0	HALFWORD OPERAND ADDRESS

DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA), is accessed and algebraically added to the least significant halfword (bit 16 through bit 31) of the General Purpose Register (GPR) specified by R. The result is then transferred to the memory halfword location specified by the EHA. The other halfword of the word which contains the halfword specified by the EHA remains unchanged.

SUMMARY EXPRESSION

$(R_{16-31})+(EHA) \rightarrow EHL$

CONDITION CODE RESULTS

CC1: Undefined
CC2: Undefined
CC3: Undefined
CC4: (EHL) is equal to zero

EXAMPLE

Memory Location: 200B4
Hex Instruction: EA 82 09 19 (R=5, X=I=0)

Before Execution	PSWR 000200B4	GPR5 FFFF8C42	Memory Halfword 20918 06C4
After Execution	PSWR 000200B8	GPR5 FFFF8C42	Memory Halfword 20918 9306

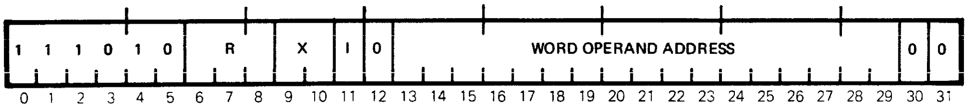
Note

The contents from GPR5, bits 16-31, and memory halfword 20918 are added, and the result is transferred to memory halfword 20918.

ARMW
s,*m,x

ADD REGISTER TO MEMORY WORD

E800



DEFINITION: The word in memory specified by the Effective Word Address (EWA) is accessed and algebraically added to the word located in the General Purpose Register (GPR) specified by R. The result word is then transferred to the memory word location specified by the EWA.

SUMMARY EXPRESSION: (R)+(EWL) → EWL

CONDITION CODE RESULTS:
CC1: Arithmetic exception
CC2: (EWL) is greater than zero
CC3: (EWL) is less than zero
CC4: (EWL) is equal to zero

EXAMPLE:
Memory Location: 03000
Hex Instruction: EB 80 31 00 (R=7, X=I=0)

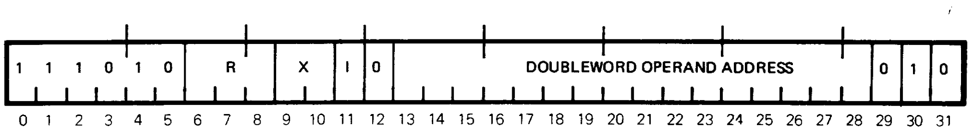
Before Execution	PSWR 08003000	GPR7 245C6E3F	Memory Word 03100 FF03C67D
After Execution	PSWR 20003004	GPR7 245C6E3F	Memory Word 03100 236034BC

Note: The contents from GPR7 and memory word 03100 are added, and the result is transferred to memory word 03100. CC2 is set.

ARMD
s,*m,x

ADD REGISTER TO MEMORY DOUBLEWORD

E800



DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and algebraically added to the doubleword located in the General Purpose Register (GPR) specified by R and R+1. R+1 is GPR one greater than specified by R. The contents of the GPR specified by R+1 are added to the contents of the least significant word of the doubleword first. The result doubleword is transferred to the memory doubleword location specified by the EDA.

SUMMARY EXPRESSION

(R+1)+EQL+1) → EWL+1+Carry

(R)+(EWL)+Carry → EWL

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: (EDL) is greater than zero
CC3: (EDL) is less than zero
CC4: (EDL) is equal to zero

EXAMPLE

Memory Location: 0819C
Hex Instruction: EB 00 83 AA (R=6, X=I=0)

Before Execution

PSWR	GPR6	GPR7
4000819C	01A298A1	F15BC63E

Memory Word 083A8	Memory Word 083AC
3B69A07E	7F3579A4

After Execution

PSWR	GPR6	GPR7
200081A0	01A298A1	F15BC63E

Memory Word 083A8	Memory Word 083AC
3D0C3920	70913FE2

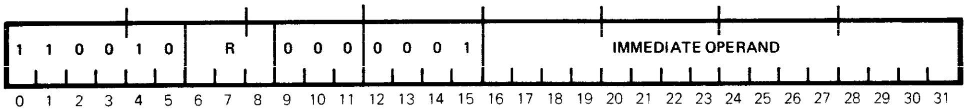
Note

The doubleword obtained from GPR6 and GPR7 is added to the doubleword from memory words 083A8 and 083AC. The result is transferred to memory words 083A8 and 083AC. CC2 is set.

ADI
d,v

ADD IMMEDIATE

C801



DEFINITION

The sign of the least significant bit (bit 16 through bit 31) of the Instruction Word is extended 16 bits to the left to form a word. This word is algebraically added to the word located in the General Purpose Register (GPR) specified by R. The result word is transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(IW_{16-31})_{SE}+(R) \rightarrow R$

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 00D88
Hex Instruction: C8 01 86 B2 (R=0)

Before Execution

PSWR	GPR0
20000D88	0000794E

After Execution

PSWR	GPR0
08000D8C	00000000

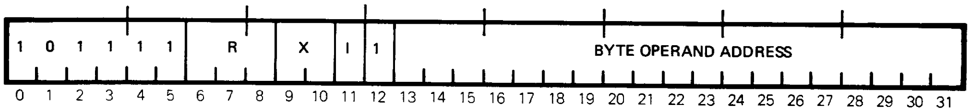
Note

The immediate operand, sign extended, is added to the contents of the GPR0, and the result replaces the previous contents of GPR0. CC4 is set.

SUMB
d,*m,x

SUBTRACT MEMORY BYTE

BC08



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and 24 zeros appended to the most significant end to form a word. This word is algebraically subtracted from the word located in the General Purpose register (GPR) specified by R. The result word is transferred (GPR) specified by R.

SUMMARY EXPRESSION

$(R)-[0_{0-23},(EBL)] \rightarrow R$

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 01000
Hex Instruction: BC 88 12 01 (R=1, X=I=0)

Before Execution	PSWR 40001000	GPR1 0194A7F2	Memory Byte 01201 9A
After Execution	PSWR 20001004	GPR1 0194A758	Memory Byte 01201 9A

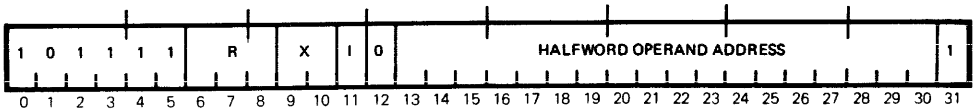
Note

The contents from memory byte 01201, with 24 zeros prefixed, are subtracted from the contents from GPR1. The result is transferred to GPR1 and CC2 is set.

SUMH
d,*m,x

SUBTRACT MEMORY HALFWORD

BC00



DEFINITION The halfword in memory specified by the Effective Halfword Address is accessed and the sign (bit 16) extended 16 bits to the left to form a word. This word is algebraically subtracted from the word located in the General Purpose Register (GPR) specified by R. The result word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION $(R)-(EHL)_{SE} \rightarrow R$

CONDITION CODE RESULTS
CC1: Arithmetic exception
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE
Memory Location: 01604
Hex Instruction: BF 00 18 77 (R=6, X=I=0)

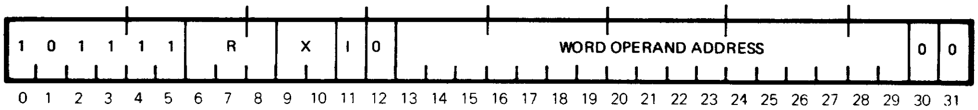
Before Execution	PSWR 10001604	GPR6 00024CB3	Memory Halfword 01876 34C6
After Execution	PSWR 20001608	GPR6 000217ED	Memory Halfword 01876 34C6

Note The contents from memory halfword 01876, sign extended, are subtracted from the contents from GPR6. The result is transferred to GPR6, and CC2 is set.

SUMW
d,*m,x

SUBTRACT MEMORY WORD

BC00



DEFINITION The word in memory specified by the Effective Word Address is accessed and algebraically subtracted from the word located in the General Purpose Register (GPR) specified by R. The result word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION (R)-(EWL) → R

CONDITION CODE RESULTS
CC1: Arithmetic exception
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE
Memory Location: 6C208
Hex Instruction: BC 86 F9 14 (R=1, X=I=0)

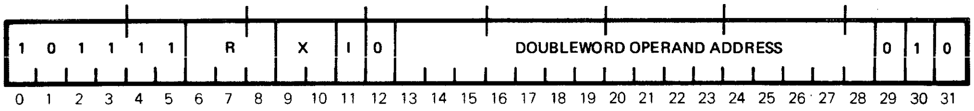
Before Execution	PSWR 0406C208	GPR1 00A6264D	Memory Word 6F914 00074BC3
After Execution	PSWR 2006C20C	GPR1 009EDA8A	Memory Word 6F914 00074BC3

Note The contents from memory word 6F914 are subtracted from the contents from GPR1, and the result is transferred to GPR1. CC2 is set.

SUMD
d,*m,x

SUBTRACT MEMORY DOUBLEWORD

BC00



DEFINITION

The doubleword in memory specified by the Effective Doubleword Address (EDA) is accessed and algebraically subtracted for the doubleword located in the General Purpose Register (GPR) specified by R and R+1. R+1 is GPR one greater than specified by R. The word located in the GPR specified by R+1 is subtracted from the least significant word of the doubleword first. The result doubleword is transferred to the GPR specified by R and R+1.

SUMMARY EXPRESSION

(R+1)-(EWL+1) → R+1-Borrow

(R)-(EWL)-Borrow → R

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: (R, R+1) is greater than zero
CC3: (R, R+1) is less than zero
CC4: (R, R+1) is equal to zero

EXAMPLE

Memory Location: 03000
Hex Instruction: BF 00 31 02 (R=6, X=I=0)

Before Execution

PSWR	GPR6	GPR7
10003000	5AD983B7	C833D509

Memory Word 03100	Memory Word 03104
153B0492	5BE87A16

After Execution

PSWR	GPR6	GPR7
20003004	459E7F25	6C4B5AF3

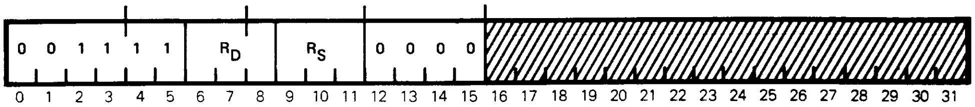
Memory Word 03100	Memory Word 03104
153B0492	5BE87A16

Note

The doubleword obtained from memory words 03100 and 03104 is subtracted from the doubleword contained in GPR6 and GPR7. The result is transferred to GPR6 and GPR7. CC2 is set.

SUBTRACT REGISTER FROM REGISTER

3C00



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_S is algebraically subtracted from the word located in the GPR specified by R_D. The result word is then transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_D)-(R_S) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

EXAMPLE

Memory Location: 106AE
Hex Instruction: 3C A0 (R_D=1, R_S=2)

	PSWR	GPR1	GPR2
Before Execution	100106AE	12345678	12345678
After Execution	080106B0	00000000	12345678

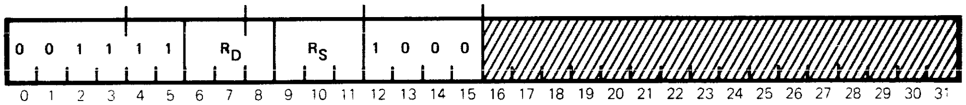
Note

The contents from GPR2 are subtracted from the contents from GPR1. The result is replaced in GPR1, and CC4 is set.

SURM
s,d

SUBTRACT REGISTER FROM REGISTER MASKED

3C08



DEFINITION The word located in the General Purpose Register (GPR) specified by R_S is algebraically subtracted from the word located in the GPR specified by R_D. The difference of this subtraction is then masked (Logical AND Function) with the contents of the mask register R4. The result word is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION $(R_D)-(R_S)$ &(R4) → R_D

CONDITION CODE RESULTS
CC1: Arithmetic exception
CC2: (R_D) is greater than zero
CC3: (R_D) is less than zero
CC4: (R_D) is equal to zero

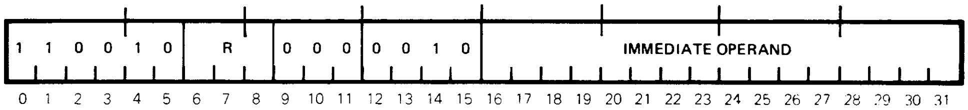
EXAMPLE
Memory Location: 00496
Hex Instruction: 3F 58 (R_D=6, R_S=5)

	PSWR	GPR4	GPR5	GPR6
Before Execution	10000496	00FFFF00	00074BC3	00A6264D
After Execution	20000498	00FFFF00	00074BC3	009EDA00

Note The contents from GPR5 are subtracted from the contents from GPR6. The result is ANDed with the contents from GPR4, and then transferred to GPR6. CC2 is set.

SUBTRACT IMMEDIATE

C802



DEFINITION

The sign of the least significant half (bit 16 through bit 31) of the Instruction Word is extended 16 bits to the left to form a word. This word is algebraically subtracted from the word located in the General Purpose Register (GPR) specified by R. The result word is transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(R)-(W_{16-31})_{SE} \rightarrow R$

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 019B8
Hex Instruction: CB 82 83 9A (R=7)

	PSWR	GPR7
Before Execution	100019B8	FFFF839A
After Execution	080019BC	00000000

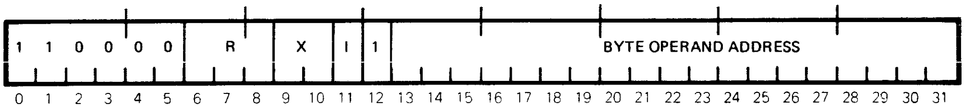
Note

The immediate operand, with sign extension, is subtracted from the contents of GPR7. The result is transferred to GPR7, and CC4 is set.

MPMB
d,*m,x

MULTIPLY BY MEMORY BYTE

C008



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and 24 zeros appended to the most significant end to form a word. This word is algebraically multiplied by the word located in the General Purpose Register (GPR) specified by R+1, one greater than GPR specified by R. The double-precision result is transferred to the GPR specified by R and R+1.

NOTES

1. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.

2. GPR specified by R must have an even address.

SUMMARY EXPRESSION

0_{0-23},(EBA)x(R+1) → R,R+1

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R, R+1) is greater than zero
CC3: (R, R+1) is less than zero
CC4: (R, R+1) is equal to zero

EXAMPLE

Memory Location: 2BA28
Hex Instruction: C0 0A C3 D9

Before Execution

PSWR	GPR0	GPR1
00002BA28	12345678	6F90C859

Memory Byte 2C3D9
40

After Execution

PSWR	GPR0	GPR1
2002BA2C	0000001B	E4321640

Memory Byte 2C3D9
40

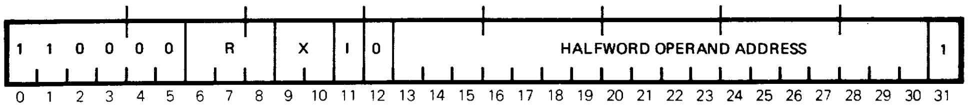
Note

The contents of memory byte 2C3D9, with zeros prefixed, are multiplied by the contents from GPR1. The result is transferred to GPR0 and GPR1. CC2 is set.

MPMH
d,*m,x

MULTIPLY BY MEMORY HALFWORD

C000



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed and the sign (bit 16) extended 16 bits to the left to form a word. This word is algebraically multiplied by the word located in the General Purpose REgister specified by R+1, one greater than GPR specified by R. The double-precision result is transferred to the GPR specified by R and R+1.

NOTES

1. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.

2. GPR specified by R must have an even address.

SUMMARY EXPRESSION

$(EHL)_{SE}x(R+1) \rightarrow R,R+1$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R, R+1) is greater than zero
CC3: (R, R+1) is less than zero
CC4: (R, R+1) is equal to zero

EXAMPLE

Memory Location: 096A4
Hex Instruction: C1 00 9B 57 (R=2, X=I=0)

Before Execution	PSWR 080096A4	GPR2 12345678	GPR3 00000003	Memory Halfword 09B56 FFFD
After Execution	PSWR 100096A8	GPR2 FFFFFFFF	GPR3 FFFFFFF7	Memory Halfword 09B56 FFFD

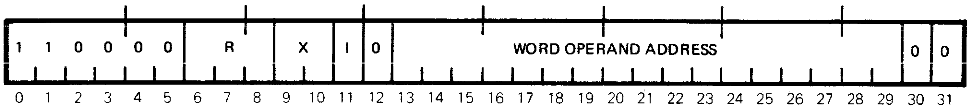
Note

The contents from GPR3 are multiplied by the contents from memory halfword 09B56. The doubleword result is transferred to GPR2 and GPR3. CC3 is set.

MPMW
d,*m,x

MULTIPLY BY MEMORY WORD

C000



DEFINITION The word in memory specified by the Effective Word Address (EWA), is accessed and algebraically multiplied by the word located in the General Purpose Register (GPR) specified by R+1, one greater than GPR specified by R. The double-precision result is transferred to the GPR specified by R and R+1.

NOTES
1. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.
2. GPR specified by R must have an even address.

SUMMARY EXPRESSION (EWL)x(R+1) → R,R+1)

CONDITION CODE RESULTS
CC1: Always zero
CC2: (R, R+1) is greater than zero
CC3: (R, R+1) is less than zero
CC4: (R, R+1) is equal to zero

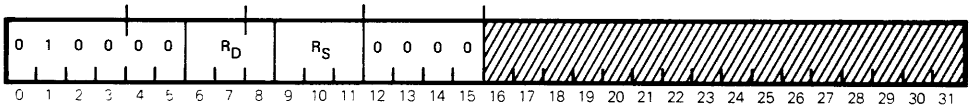
EXAMPLE
Memory Location: 04AC8
Hex Instruction: C3 00 4B 1C (R=6, X=I=0)

Before Execution	PSWR 10004AC8	GPR6 00000000	GPR7 80000000	Memory Word 04B1C 80000000
After Execution	PSWR 20004ACC	GPR6 40000000	GPR7 00000000	Memory Word 04B1C 80000000

Note The contents from GPR7 and memory word 04B1C are multiplied, and the result is transferred to GPR6 and GPR7. CC2 is set.

MULTIPLY REGISTER BY REGISTER

4000



DEFINITION

The word located in the General Purpose Register (GPR) specified by R_S is algebraically multiplied by the word located in the GPR specified by R_D+1. R_D+1 is GPR one greater than specified by R_D. The double-precision result is transferred to the GPR specified by R_D and R_D+1.

NOTES

1. The multiplicand regiser R_S can be any register, including register R_D+1; however, R_D must be an even-numbered register.

2. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.

SUMMARY EXPRESSION

$(R_S)x(R_D+1) \rightarrow R_D,R_D+1$

CONDITION CODE RESULTS

CC1: Always zero
CC2: (R_D,R_D+1) is greater than zero
CC3: (R_D,R_D+1) is less than zero
CC4: (R_D,R_D+1) is equal to zero

EXAMPLE

Memory Location: 0098E
Hex Instruction: 40 10 (R_D=0,R_S=1)

Before Execution	PSWR 1000098E	GPR0 00000000	GPR1 0000000F
After Execution	PSWR 20000990	GPR1 00000000	GPR1 000000E1

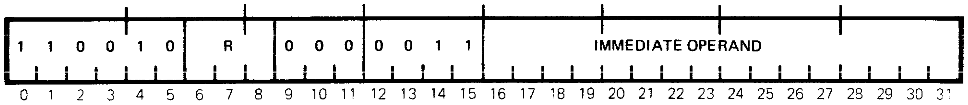
Note

The content from GPR1 is multiplied by itself, and the doubleword product is transferred to GPR0 and GPR1. CC2 is set.

MPI
d,v

MULTIPLY IMMEDIATE

C803



DEFINITION The sign of the least significant half (bit 16 through bit 31) of the Instruction Word (IW) is extended 16 bits to the left to form a word. This word is algebraically multiplied by the word located in the General Purpose Register (GPR) specified by R+1. R+1 is GPR one greater than specified by R. The result is transferred to the GPR specified by R and R+1.

NOTES

1. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.

2. GPR specified by R must have an even address.

SUMMARY EXPRESSION $(IW_{16-31})_{SE}x(R+1) \rightarrow R,R+1$

CONDITION CODE RESULTS
CC1: Always zero
CC2: (R,R+1) is greater than zero
CC3: (R,R+1) is less than zero
CC4: (R,R+1) is equal to zero

EXAMPLE
Memory Location: 00634
Hex Instruction: CB 03 01 00 (R=6, X=I=0)

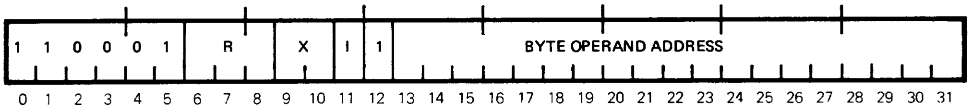
	PSWR	GPR6	GPR7
Before Execution	20000634	12345678	F37A9B15
After Execution	10000638	FFFFFFF3	7A9B1500

Note The immediate operand, sign extended, is multiplied by the contents from GPR7. The result is transferred to GPR6 and GPR7. CC3 is set.

DVMB
d,*m,x

DIVIDE BY MEMORY BYTE

C408



DEFINITION

The byte in memory specified by the Effective Byte Address (EBA) is accessed and 24 zeros appended to the most significant end to form a word. This word is algebraically divided into the doubleword located in the General Purpose Register (GPR) specified by R and R+1. R+1 is GPR one greater than specified by R. The result quotient is then transferred to the GPR specified by R+1 and the remainder to the GPR specified by R. The sign of the GPR specified by R (Remainder) is set to the original sign of the dividend and the sign of the GPR specified by R+1 (Quotient) will be algebraic product of the original signs of the dividend and the divisor except when the absolute value of the dividend is less than the absolute value of the divisor, in which case the resulting quotient (GPR specified by R+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the value of the quotient exceeds 32 bits. If an arithmetic exception occurs, the original dividend will be restored in the GPR specified by R and R+1.

2. GPR specified by R must have an even address.

SUMMARY EXPRESSION

$(R,R1)/[0_{0-23},(EBL)] \rightarrow R+1$

Remainder → R

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: $(R+1_{0-31})$ is greater than zero
CC3: $(R+1_{0-31})$ is less than zero
CC4: $(R+1_{0-31})$ is equal to zero

EXAMPLE

Memory Location: 03000
Hex Instruction: C4 08 30 BF (R=0, X=I=0)

	PSWR	GPR0	GPR1	Memory Byte 030BF
Before Execution	10003000	00000000	00000139	04
After Execution	20003004	00000001	0000004E	04

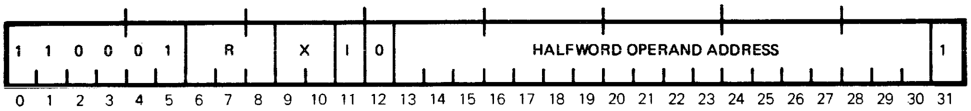
Note

The doubleword contents of GPR0 and GPR1 are divided by the content of memory byte 030BF, with 24 zeros prefixed. The quotient is transferred to GPR1 and the remainder to GPR0. CC2 is set.

DVMH
d,*m,x

DIVIDE BY MEMORY HALFWORD

C400



DEFINITION

The halfword in memory specified by the Effective Halfword Address (EHA) is accessed and the sign extended 16 bits to the left to form a word. This word is algebraically divided into the doubleword located in the General Purpose Register (GPR) specified by R and R+1. R+1 is GPR one greater than specified by R. The result quotient is then transferred to the GPR specified by R+1 and the remainder of the GPR specified by R. The sign of the GPR specified by R (Remainder) is set to the original sign of the dividend and the sign of the GPR specified by R+1 (Quotient) will be algebraic product of the original signs of the dividend and the divisor, except when the absolute value of the dividend is less than the absolute value of the divisor, in which case the resulting quotient (GPR specified by R+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the value of the quotient exceeds 32 bits. If an arithmetic exception occurs, the original dividend will be restored in the GPR specified by R and R+1.

2. The GPR specified by R must have an even address.

SUMMARY EXPRESSION

$(R,R+1)/(EHL)_{SE} \rightarrow R+1$

Remainder → R

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: $R+1_{0-31}$ is greater than zero
CC3: $R+1_{0-31}$ is less than zero
CC4: $R+1_{0-31}$ is equal to zero

EXAMPLE

Memory Location: 05A94
Hex Instruction: C7 00 5D 6B (R=6, X=I=0)

Before Execution	PSWR	GPR6	GPR7	Memory Halfword 05D6A
	08005A94	00000000	0000003B	FFF8
After Execution	PSWR	GPR6	GPR7	Memory Word 05D6A
	10005A98	00000005	FFFFFFF9	FFF8

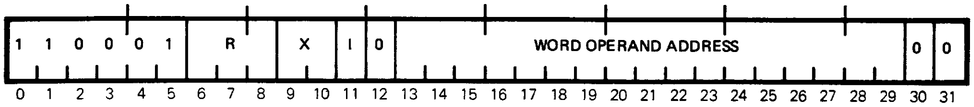
Note

The doubleword content of GPR6 and GPR7 is divided by the content of memory halfword 05D6A with sign extension. The quotient is transferred to GPR7 and the remainder to GPR6. CC3 is set.

DVMW
d,*m,x

DIVIDE BY MEMORY WORD

C400



DEFINITION

The word in memory specified by the Effective Word Address (EWA) is accessed and algebraically divided into the doubleword located in the General Purpose Register (GPR) specified by R and R+1. R+1 is GPR one greater than specified by R. The result quotient is then transferred to the GPR specified by R+1 and the remainder to the GPR specified by R. The sign of the GPR specified by R (Remainder) is set to the original sign of the dividend, and the sign of the GPR specified by R+1 (Quotient) will be the algebraic produce of the original signs of the dividend and the divisor except when the absolute value of the dividend is less than the absolute value of the divisor, in which case the resulting quotient (GPR specified by R+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the value of the quotient exceeds 32 bits. If an arithmetic exception occurs, the original dividend will be restored in the GPR specified by R and R+1.

2. The GPR specified by R must have an even address.

SUMMARY EXPRESSION

(R,R+1)/(EWL) → R+1

Remainder → R

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: $R+1_{0-31}$ is greater than zero
CC3: $R+1_{0-31}$ is less than zero
CC4: $R+1_{0-31}$ is equal to zero

EXAMPLE

Memory Location: 078C0
Hex Instruction: C6 00 7B 5C (R=4, X=I=0)

	PSWR	GPR4	GPR5	Memory Word 07B5C
Before Execution	400078C0	00000000	039A20CF	FC000000
After Execution	080078C4	039A20CF	00000000	FC000000

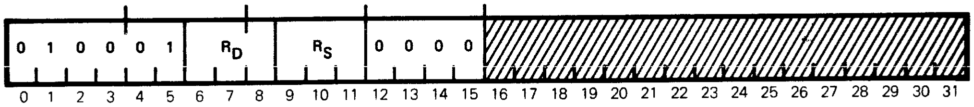
Note

The doubleword obtained from GPR4 and GPR5 is divided by the content of memory word 07B5C. The quotient is transferred to GPR5 and the remainder to GPR4. CC4 is set.

DVR
s,d

DIVIDE REGISTER BY REGISTER

4400



DEFINITION The word located in the General Purpose Register (GPR) specified by R_S is algebraically divided into the doubleword located in the GPR specified by R_D and R_D+1. R_D+1 is GPR one greater than specified by R_D. The result quotient is then transferred to the GPR specified by R_D+1 and the remainder to the GPR specified by R_D. The sign of the GPR specified by R_D (Remainder) is set to the original sign of the dividend and the sign of the GPR specified by R_D+1 (Quotient) will be the algebraic produce of the original signs of the dividend and the divisor, except when the absolute value of the dividend is less than than the absolute value of the divisor, in which case the resulting quotient (GPR specified by R_D+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the value of the quotient exceeds 32 bits. If an arithmetic exception occurs, the original dividend will be restored in the GPR specified by R and R+1.

2. The GPR specified by R_D must have an even address.

3. R_S must not equal R_D or R_D+1.

SUMMARY EXPRESSION $(R_D,R_D+1)/R_S \rightarrow R_D+1$

Remainder → R_D

CONDITION CODE RESULTS
CC1: Arithmetic exception
CC2: R_D+1_{0-31} is greater than zero
CC3: R_D+1_{0-31} is less than zero
CC4: R_D+1_{0-31} is equal to zero

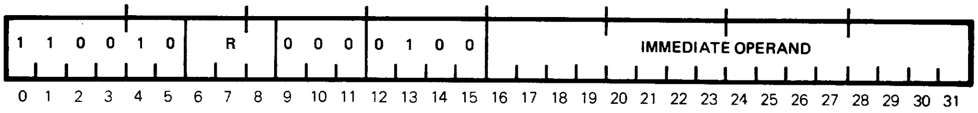
EXAMPLE
Memory Location: 04136
Hex Instruction: 47 20 (R_D=6,R_S=2)

	PSWR	GPR2	GPR6	GPR7
Before Execution	10004136	0000000A	00000000	000000FF
After Execution	20004138	0000000A	00000005	00000019

Note The doubleword obtained from GPR6 and GPR7 is divided by the contents of GPR2. The quotient is transferred to GPR7 and the remainder to GPR6. CC2 is set.

DIVIDE IMMEDIATE

C804



DEFINITION

The sign of the least significant half (bit 16 through bit 31) of the Instruction Word (IW) is extended 16 bits to the left to form a word. This word is algebraically divided into the doubleword located in the General Purpose Register (GPR) specified by R and R+1. R-1 is GPR one greater than specified by R. The resulting quotient is then transferred to the GPR specified by R+1 and the remainder to the GPR specified by R. The sign of the GPR specified by R (Remainder) is set to the original sign of the dividend and the sign of the GPR specified by R+1 (Quotient) will be the algebraic product of the original signs of the dividend and the divisor, except when the absolute value of the dividend is less than the absolute value of the divisor, in which case the resulting quotient (GPR specified by R+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the value of the quotient exceeds 32 bits. If an arithmetic exception occurs, the original dividend will be restored in the GPR specified by R and R+1.

2. The GPR specified by R must have an even address.

SUMMARY EXPRESSION

$(R,R+1)/(IW_{16-31})_{SE} \rightarrow R+1$

Remainder → R

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: $R+1_{0-31}$ is greater than zero
CC3: $R+1_{0-31}$ is less than zero
CC4: $R+1_{0-31}$ is equal to zero

EXAMPLE

Memory Location: 08000
Hex Instruction: C9 04 FF FD (R=2)

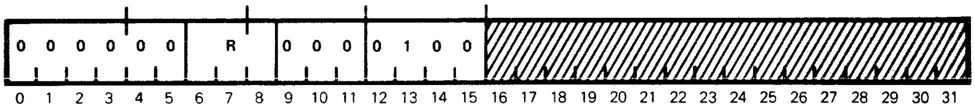
	PSWR	GPR2	GPR3
Before Execution	04008000	00000000	000001B7
After Execution	10008004	00000001	FFFFFF6F

Note

The doubleword obtained from GPR2 and GPR3 is divided by the immediate operand, sign extended. The quotient is transferred to GPR3 and the remainder to GPR2. CC3 is set.

EXTEND SIGN

0004



DEFINITION The sign (bit 0) of the contents of the General Purpose Register (GPR), specified by R+1, is extended through all 32 bit positions of the GPR specified by R.

SUMMARY EXPRESSION $(R+1_0) \rightarrow R_{0-31}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

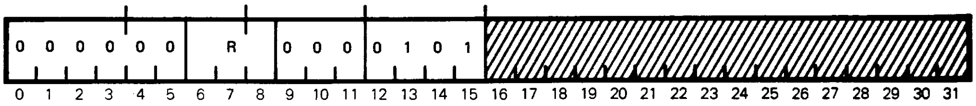
Memory Location: 0083A
Hex Instruction: 00 84 (R=1)

	PSWR	GPR1	GPR2
Before Execution	0800083A	0000B074	8000C361
After Execution	1000083C	FFFFFFFF	8000C361

Note Bits 0 through 31 of GPR1 are set to ones. CC3 is set.

ROUND REGISTER

0005



DEFINITION

The contents of the General Purpose Register (GPR) specified by R are incremented by ONE if bit position zero of the GPR specified by R+1 is equal to ONE. R+1 is GPR one greater than specified by R.

SUMMARY EXPRESSION

$(R)+1,\text{if}(R+1_0)=1$

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: R_{0-31} is greater than zero
CC3: R_{0-31} is less than zero
CC4: R_{0-31} is equal to zero

EXAMPLE

Memory Location: 00FFE
Hex Instruction: 03 75 (R=6)

	PSWR	GPR6	GPR7
Before Execution	40000FFE	783A05B2	92CD061F
After Execution	20001000	783A05B3	92CD061F

Note

The contents of GPR6 is incremented by one, and the result is returned to GPR6. CC2 is set.

FLOATING POINT ARITHMETIC INSTRUCTION

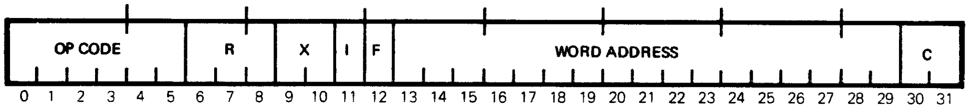
General Description

The Floating-Point Arithmetic Instructions provide the capability to add, subtract, multiply, or divide operands of large magnitude with precise results. A floating-point number is made up of three parts: a sign, a fraction, and an exponent. The sign applies to the fraction and denotes a positive or negative value. The fraction is a binary number with an assumed radix point between the sign bit and the most significant bit. The exponent is a seven-bit binary power to which the base 16 is raised. The quantity that the floating-point number represents is obtained by multiplying the fraction by the number 16 raised to the power represented by the exponent.

Instruction Format

The following Instruction Format is used for all floating-point operations:

Memory Reference



Bits 0-5 define the Operation Code.

Bits 6-8 designate a General Purpose Register address (0 through 7).

Bits 9-10 designate one of three Index Registers.

Bit 11 indicates if an indirect addressing operation is to be performed.

Bits 12-31 specify the address of the operand directly when X and I fields are equal to zero. If X is not equal to zero, indirect addressing is specified. Bit 12 (F) is used as an augment bit by the Floating-Point Instruction.

Condition Code Utilization

Execution of all floating-point arithmetic instructions causes a condition code to be set to indicate if the result of the operation was greater than, less than, or equal to zero. Arithmetic exceptions produced by a floating-point operation are also reflected by the condition code results.

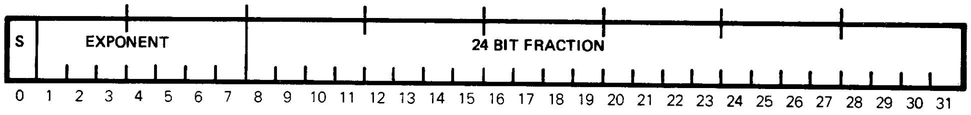
The meaning of the condition codes differ for the execution of the floating-point instructions. CC1 is set by an Arithmetic Exception condition (underflow or overflow). To differentiate between these exceptions, CC4 is also set when the overflow condition occurs. In both instances, either

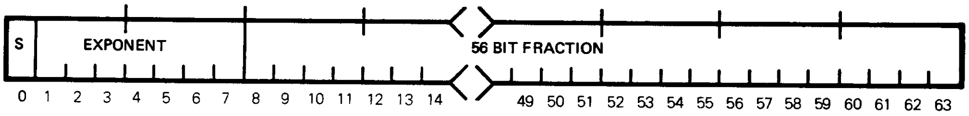
CC2 or CC3 is used to indicate the state of what would have been the sign of the resultant fraction had the arithmetic exception not occured. The following is a table reflecting the possible condition code settings.

Condition Code				Definition
CC1	CC2	CC3	CC4	
1	0	0	0	Arithmetic exception
0	1	0	0	Positive fraction
0	0	1	0	Negative
0	0	0	1	Zero fraction
1	1	0	0	Exponent Underflow, positive fraction
1	0	1	0	Exponent Underflow, negative fraction
1	1	0	1	Exponent Overflow, positive fraction
1	0	1	1	Exponent Overflow, negative fraction

Floating-Point Arithmetic Operands

A floating-point number can be represented in two different formats, word and doubleword. These two formats are the same except that the doubleword contains an extra eight hexadecimal digits of significance in the fraction. These two formats are shown below.





The floating-point number, in either format, is made up of three parts: a sign, a fraction, and an exponent. The sign bit (bit 0) applies to the fraction and denotes a positive or negative value. The fraction is a hexadecimal normalized number with a radix point to the left of the highest order fraction bit (bit 8). The exponent (bits 1-7) is a seven bit binary number to which the base 16 is raised.

Negative exponents are carried in the two's complement format. In order to remove the sign and therefore enable exponents to be compared directly, both positive and negative exponents are biased up by 40_{16} (excess 64_{10} notation). The quantity that a floating-point number represents is obtained by multiplying the fraction by the number 16_{10} raised to the power of the exponent minus 40_{16}.

A positive floating-point number is converted to a negative floating-point number by taking the two's complement of the positive fraction and the one's complement of the biased exponent. If the minus one case is ruled illegitimate, all floating-point numbers can be converted from positive to negative and from negative to positive by merely taking the two's complement of the number in floating-point format. Signed numbers in the floating-point format can then be compared directly one with another by using the COMPARE ARITHMETIC class of instructions.

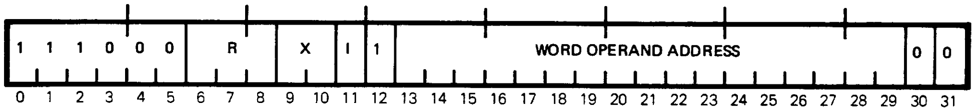
All floating-point operands must be normalized before being operated upon by a floating-point instruction. A positive floating-point number is normalized when the value of the fraction is less than one and greater than or equal to one-sixteenth ($1 > F \geq 1/16$). A negative floating-point number is normalized when the value of the fraction is greater than minus one and less than or equal to minus one-sixteenth ($-1 < F \leq -1/16$). All floating-point answers are normalized by the CPU. If a floating-point operation results in a minus one of the form 1 XXX XXXX 0000...0000, the CPU will convert that result to a legitimate normalized floating-point number of the form 1 YYY YYYY 1111 0000...0000, where YYY YYYY is one less than XXX XXXX.

A hexadecimal guard digit is appended to the least significant hexadecimal digit of the floating-point word operands by the CPU. This guard digit is carried throughout all floating-point word computations. The most significant bit of the guard digit is used as the basis for rounding by the CPU at the end of every floating-point word computation.

ADFW
d,*m,x

ADD FLOATING-POINT WORD

E008



DEFINITION

The floating-point operand in memory is accessed. If either of the floating-point numbers are negative, the one's complement of the base 16 exponent (bits 1-7) is taken of the negative number. Both exponents are then stripped of their 40_{16} bias and algebraically compared. If the two exponents are equal, the signed fractions of the two numbers are algebraically added. If the exponents differ, and the difference is equal to a greater than one, or equal to or less than six ($1 \leq$ exponent difference ≤ 6) the fraction of the operand containing the smaller exponent is shifted right, one hexadecimal digit. After exponent equalization, the fractions are added algebraically. The normalized and rounded sum of the two fractions is placed in bit positions 0 and 8 through 31 of GPR d. The resulting exponent is biased up by 40_{16} and, if the resulting fraction is negative, the one's complement of the exponent is placed in bit positions 1 through 7 of GPR d.

NOTES

1. If the resulting fraction equals zero, the exponent and fraction are set to zero, in GPR d.

2. Operands are expected to be normalized.

SUMMARY EXPRESSION

(R)+(EWL) → (R)

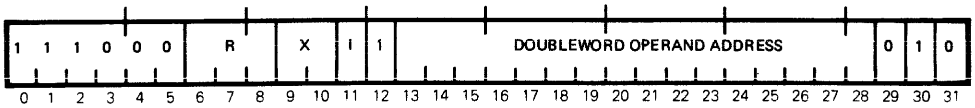
CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: $R_{0,8-31}$ is greater than zero
CC3: $R_{0,8-31}$ is less than zero
CC4: $R_{0,8-31}$ is equal to zero

ADFD
d,*m,x

ADD FLOATING-POINT DOUBLEWORD

E008



DEFINITION

The floating-point operand in memory is accessed. If either of the floating-point numbers are negative, the one's complement of the base 16 exponent (bits 1-7) is taken of the negative number. Both exponents are then stripped of their 40_{16} bias and algebraically compared. If the two exponents are equal, the signed fractions of the two numbers are algebraically added. If the exponents differ, and the difference is equal to a greater than one, or equal to or less than six ($1 \leq$ exponent difference ≤ 6) the fraction of the operand containing the smaller exponent is shifted right, one hexidecimal digit. After exponet equalization, the fractions are added algebraically. The normalized and rounded sum of the two fractions is placed in bit positions 0 and 8 through 63 of GPR d + 1. The resulting exponent is biased up by 40_{16} and, if the resulting fraction is negative, the one's complement of the exponent is placed in bit positions 1 through 7 of GPR d.

NOTES

1. If the resulting fraction equals zero, the exponent and fraction are set to zero, in GPR +1.

2. Operands are expected to be normalized.

SUMMARY EXPRESSION

(R),(R+1)+(EWL),(EWL+1) → (R),(R+1)

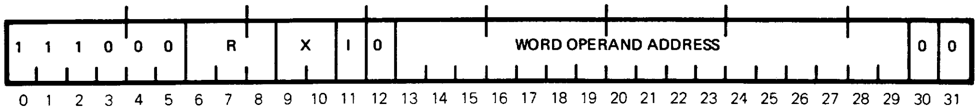
CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: $R_{0,8-31}$ is greater than zero
CC3: $R_{0,8-31}$ is less than zero
CC4: $R_{0,8-31}$ is equal to zero

SUFW
d,*m,x

SUBTRACT FLOATING-POINT WORD

E000



DEFINITION

The floating-point operand in memory is accessed. If either the floating-point number in the GPR or memory is negative, the one's complement of the basic 16 exponent (bits 1-7) is taken. Both exponents are then stripped of their 40_{16} bits and algebraically compared. If the two exponents are equal, the 24-bit signed fractions are algebraically subtracted (i.e. the memory operand is subtracted from the GPR or GPRs). If the exponents differ, and the difference is equal to or greater than one, or equal to or less than six ($1 \leq$ exponent difference ≤ 6), the fraction of the operand containing the smaller exponent is shifted right one hexadecimal digit at a time until the exponents are equalized. The exponent of this operand is effectively incremented by one each time the fraction is shifted right one hexadecimal unit. After exponent equalization, the fractions are subtracted algebraically. The normalized and rounded difference between the two fractions is placed in bit positions 0 and 8 through 31 of GPR d. The resulting exponent is biased up by 40_{16} and, if the resulting fraction is negative, the one's complement of the exponent is placed in bit position 1 through 7 of GPR d.

NOTES

1. If the resulting fraction is equal to zero, the exponent and fraction are set to zero in the GPR or GPRs.

2. Operands are expected to be normalized.

SUMMARY EXPRESSION

(R)-(EWL) → (R)

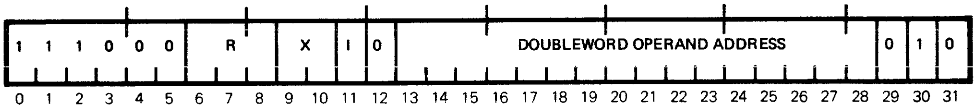
CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: $R_{0,8-31}$ is greater than zero
CC3: $R_{0,8-31}$ is less than zero
CC4: $R_{0,8-31}$ is equal to zero

SUFD
d,*m,x

SUBTRACT FLOATING-POINT DOUBLEWORD

E000



DEFINITION The floating-point operand in memory is accessed. If either the floating-point number in the GPR or memory is negative, the one's complement of the basic 16 exponent (bits 1-7) is taken. Both exponents are then stripped of their 40_{16} bits and algebraically compared. If the two exponents are equal, the 24-bit signed fractions are algebraically subtracted (i.e. the memory operand is subtracted from the GPR or GPRs). If the exponents differ, and the difference is equal to or greater than one, or equal to or less than six ($1 \leq$ exponent difference ≤ 6), the fraction of the operand containing the smaller exponent is shifted right one hexadecimal digit at a time until the exponents are equalized. The exponent of this operand is effectively incremented by one each time the fraction is shifted right one hexadecimal digit. After exponent equalization, the fractions are subtracted algebraically. The normalized and rounded difference between the two fractions is placed in bit positions 0 and 8 through 63 of GPR d + 1. The resulting exponent is biased up by 40_{16} and, if the resulting fraction is negative, the one's complement of the exponent is placed in bit position 1 through 7 of GPR d.

NOTES

1. If the resulting fraction is equal to zero, the exponent and fraction are set to zero in the GPR or GPRs.

2. Operands are expected to be normalized.

SUMMARY EXPRESSION (R),(R+1)-(EWL),(EWL+1) → (R),(R+1)

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: $R_{0,8-31}$ is greater than zero
CC3: $R_{0,8-31}$ is less than zero
CC4: $R_{0,8-31}$ is equal to zero

MPFW
d,*m,x

MULTIPLY FLOATING-POINT WORD

E408

0	1	2	3	4	5	6-8	9-10	11	12	13-29	30	31
1	1	1	0	0	1	R	X	I	1	WORD OPERAND ADDRESS	0	0

DEFINITION

The floating-point operand fraction is multiplied by the fraction of GPR d. If either one or both of the floating-point numbers are negative, the exponent of the negative number is changed to its one's complement. Both exponents are then stripped of their 40_{16} bias and algebraically added. The normalized and rounded product of the multiplication is placed in bits 0 and 8 through 31 of GPR d. The resulting exponent is biased up by 40_{16} and, if the resulting fraction is negative, the one's complement of the resulting exponent is placed in bits 1 through 7 of GPR d.

NOTE

Operands are expected to be normalized.

SUMMARY EXPRESSION

$(EWL_{0,8-31})x(R_{0,8-31}) \rightarrow R_{0,8-31}$

$(EWL_{1-7})+(R_{1-7}) \rightarrow R_{1-7}$

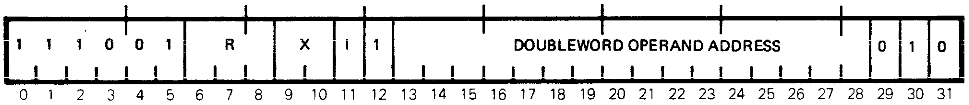
CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: $R_{0,8-31}$ is greater than zero
CC3: $R_{0,8-31}$ is less than zero
CC4: $R_{0,8-31}$ is equal to zero

MPFD
d,*m,x

MULTIPLY FLOATING-POINT DOUBLEWORD

E408



DEFINITION

The floating-point operand fraction is multiplied by the fraction of GPR d + 1. If either one or both of the floating-point numbers are negative, the exponent of the negative number is changed to its one's complement. Both exponents are then stripped of their 40_{16} bias and algebraically added. The normalized and rounded product of the multiplication is placed in bits 0 and 8 through 63 of GPR d + 1. The resulting exponent is biased up by 40_{16} and, if the resulting fraction is negative, the one's complement of the resulting exponent is placed in bits 1 through 7 of GPR d.

NOTE

Operands are expected to be normalized.

SUMMARY EXPRESSION

$(EWL_{0,8-31}, EWL{+}1_{0-31})x(R_{0,8-31}, R{+}1_{0-31})$

$\rightarrow R_{0,8-31}, R{+}1_{0-31}$

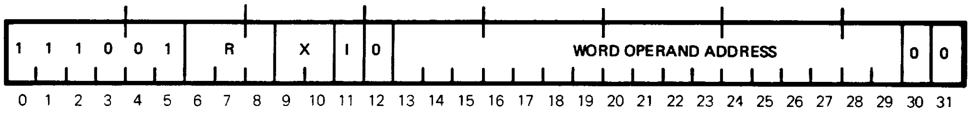
$(EWL_{1-7})+(R_{1-7}) \rightarrow R_{1-7}$

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: $R_{0,8-31}$ is greater than zero
CC3: $R_{0,8-31}$ is less than zero
CC4: $R_{0,8-31}$ is equal to zero

DIVIDE FLOATING-POINT WORD

E400



DEFINITION

The floating-point operand in memory (divisor) is accessed and the fraction is divided into the fraction of GPR d. If either one or both of the floating-point numbers are negative, the one's complement of the exponent is taken. Both exponents are then stripped of the 40_{16} bias and the exponent of the divisor is subtracted algebraically from the exponent of the dividend. The normalized and rounded quotient is placed in bit 0 and bit positions 8 through 31 of the GPR d. The resulting exponent is biased up to 40_{16} and, if the resulting fraction is negative, its one's complement is placed in bits 1 through 7 of GPR d.

NOTE

Operands are expected to be normalized.

SUMMARY EXPRESSION

$(R_{0,8-31})/(EWL_{0,8-31}) \rightarrow R_{0,8-31}$

$(R_{1-7})-(EWL_{1-7}) \rightarrow R_{1-7}$

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: $R_{0,8-31}$ is greater than zero
CC3: $R_{0,8-31}$ is less than zero
CC4: $R_{0,8-31}$ is equal to zero

DVFD
d,*m,x

DIVIDE FLOATING-POINT DOUBLEWORD

E400

0	1	2	3	4	5	6 7 8	9 10	11	12	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28	29	30	31
1	1	1	0	0	1	R	X	I	0	DOUBLEWORD OPERAND ADDRESS	0	1	0

DEFINITION The floating-point operand in memory (divisor) is accessed and the fraction is divided into the fraction of GPR d + 1. If either one or both of the floating-point numbers are negative, the one's complement of the exponent is taken. Both exponents are then stripped of the 40_{16} bias and the exponent of the divisor is subtracted algebraically from the exponent of the dividend. The normalized and rounded quotient is placed in bit 0 and bit positions 8 through 63 of the GPR d + 1. The resulting exponent is biased up to 40_{16} and, if the resulting fraction is negative, it's one's complement is placed in bits 1 through 7 of GPR d.

NOTE Operands are expected to be normalized.

SUMMARY EXPRESSION

$(R_{0,8-31}, R+1_{0-31})/(EWL_{0,8-31}, EWL+1_{0-31})$

$\rightarrow R_{0,8-31}, R+1_{0-31}$

$(R_{1-7})-(EWL_{1-7}) \rightarrow R_{1-7}$

CONDITION CODE RESULTS

CC1: Arithmetic exception
CC2: $R_{0,8-31}$ is greater than zero
CC3: $R_{0,8-31}$ is less than zero
CC4: $R_{0,8-31}$ is equal to zero

CONTROL INSTRUCTIONS

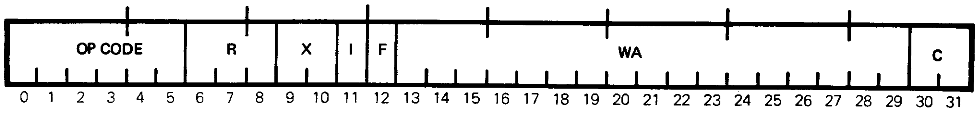
General Description

This group of Instructions allows the mainframe to perform Execute, NO OP., Halt and Wait operations.

Instruction Formats

Control Instructions use the Memory Reference and Inter-Register Instruction formats. It should be noted that several of the Control Instructions vary the Basic Inter-Register format in that certain portions are not used and are left blank.

Memory Reference



Bits 0-5 define the Operation Code.

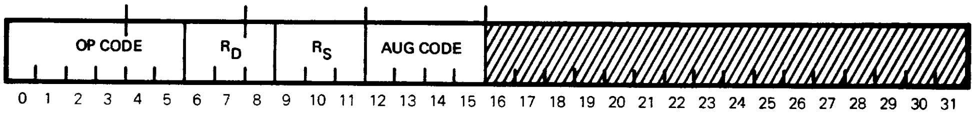
Bits 6-8 designate a General Purpose Register address (0 through 7).

Bits 9-10 designate one of three Index Registers.

Bit 11 indicates if an indirect addressing operation is to be performed.

Bits 12-31 specify the address of the operand when X and I fields are equal to zero.

Inter-Register



Bits 0-5 define the Operation Code.

Bits 6-8 designate the register to contain the result of the operation.

Bits 9-11 designate the register which contains the source operand.

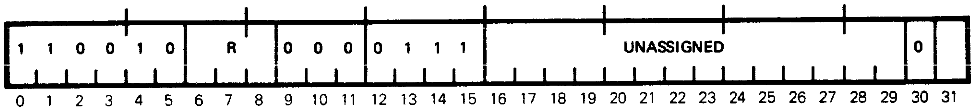
Bits 12-15 define the Augmenting Operation Code.

Condition Code Utilization

Condition Code results for Execute operations will be dependent upon the Instruction that was performed. All other control operations leave the current Condition Code unchanged.

EXECUTE REGISTER

C807



DEFINITION

The word located in the General Purpose Register (GPR) specified by R is transferred to the instruction register to be executed as the next instruction. Providing this instruction is not a branch, the next instruction executed (following execution of the instruction in register R) is contained in the sequential memory location following the EXR instruction. If the GPR specified by R does contain a branch instruction, the Program Status Word Register is changed accordingly.

NOTES

1. If two halfword instructions are contained in the GPR specified by R, only the left halfword instruction is executed.

2. An unimplemented instruction trap is generated if an EXR instruction attempts to execute an unimplemented instruction or another execute instruction.

SUMMARY EXPRESSION

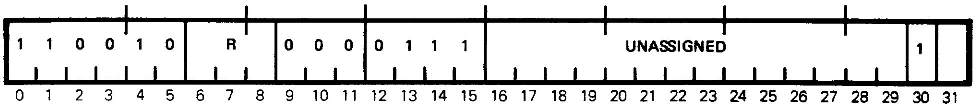
(R) → I

CONDITION CODE RESULTS

Defined by the executed instruction.

EXECUTE REGISTER RIGHT

C807



DEFINITION The contents of the least significant halfword (bit 16 through bit 31) of the General Purpose Register (GPR) specified by R are transferred to the most significant halfword position (bit 0 through bit 15) of the instruction register to be executed as the next instruction. Providing this halfword instruction is not a branch, the next instruction executed (following execution of the halfword instruction transferred to the instruction register) is contained in the sequential memory location following the EXRR instruction. If the instruction transferred to the instruction register is a branch instruction, the Program Status Word Register is changed accordingly.

NOTE An unimplemented instruction trap is generated if an EXRR instruction attempts to execute an unimplemented instruction or another execute instruction.

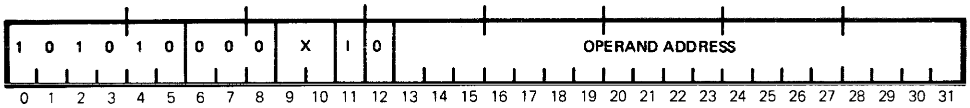
SUMMARY EXPRESSION $(R_{16-31}) \rightarrow I_{0-15}$

CONDITION CODE RESULTS Defined by the executed instruction.

EXM
*m,x

EXECUTE MEMORY

A800



DEFINITION

The word in memory specified by the Effective Address (EA) is accessed and executed as the next instruction. Providing tis instruction is not a branch, the next instruction executed (following execution of the instruction specified by the EA) is contained in the next sequential memory location following the EXM instruction. If the instruction in memory specified by the EA is a branch instruction, the Program Status Word Register is changed accordingly.

NOTES

1. If two halfword instructions are contained in the memory location specified by the EA, bit 30 of the EA determines which halfword instruction is executed. When bit 30 equals 0, left halfword is used. When bit 30 equals 1, right halfword is used.

2. An unimplemented instruction trap is generated if an EXM instruction attempts to execute an unimplemented instruction or another execute instruction.

SUMMARY EXPRESSION

$(EWL_{0-31}) \rightarrow I$, if $EA_{30}=0$

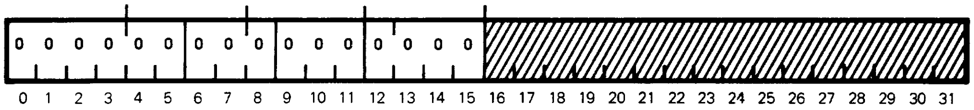
$(EWL_{16-31}) \rightarrow I$, if $EA_{30}=1$

CONDITION CODE RESULTS

Defined by the executed instruction.

HALT

0000



DEFINITION

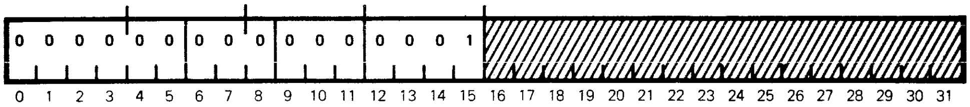
The execution of this instruction causes computer operation to be stopped. This includes input/output transfers and the servicing of priority interrupts.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

WAIT

0001

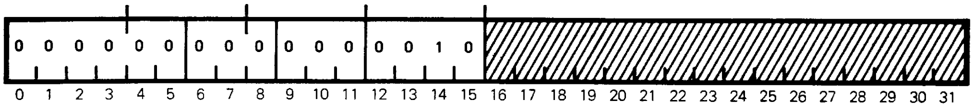


DEFINITION The execution of this instruction causes the completion of subsequent instructions to be stopped. Input/output transfers and priority interrupt servicing continue.

CONDITION CODE RESULTS
CC1: No change
CC2: No change
CC3: No change
CC4: No change

NO OPERATION

0002



DEFINITION

This instruction does not perform any operation.

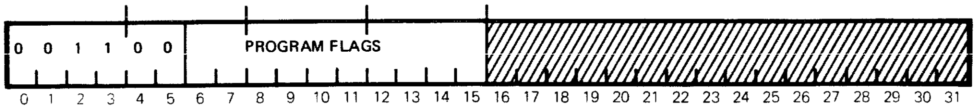
CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

CALM

CALL MONITOR

3000



DEFINITION

The execution of this instruction causes an interrupt request signal to be applied to interrupt priority level 27_{16}. Bit positions 6 through 15 of the instruction word may be used to contain program flags which can be examined by the interrupt service routine.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

INTERRUPT INSTRUCTIONS

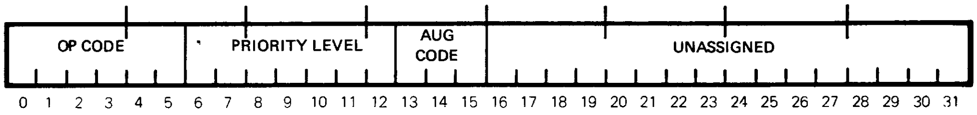
General Description

The Interrupt Control Instruction group provides the availability to permit selective Enable, Disable, Request, Activate, and Deactivate operations to be performed on any addressed interrupt level.

Instruction Formats

The following instruction format is used for all Interrupt Control operations.

Interrupt Control



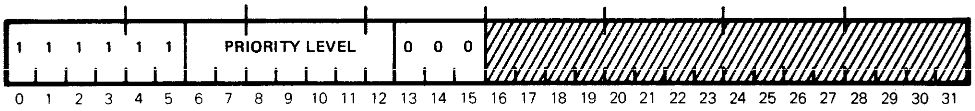
Bits 0-5	define the Operation Code.
Bits 6-12	define the binary priority level number of the interrupt being commanded.
Bits 13-15	define the Augmenting Operation Code.
Bits 16-31	unassigned.

Condition Code Utilization

All Interrupt Control Instructions leave the current Condition Code unchanged.

ENABLE INTERRUPT

FC00



DEFINITION

The priority interrupt level specified by the priority level field (bit 6 through bit 12) contained in the Instruction Word (IW) is conditioned to respond to an interrupt signal.

NOTE

This instruction does not operate with priority levels 0_{16} through 11_{16}.

INSTRUCTION PRIORITY LEVEL FIELD

Bits 6 through 12	Priority Level (Hex.)
0000000	00
0000001	01
0000010	02
-	-
-	-
-	-
-	-
1111110	7E
1111111	7F

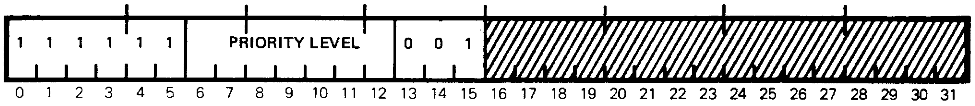
CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

DI
V

DISABLE INTERRUPT

FC01



DEFINITION

The priority interrupt level specified by the priority level field (bit 6 through bit 12) contained in the Instruction Word (IW) is disabled and will not respond to an interrupt signal.

NOTES

1. Any unserviced request signal at this level is cleared by execution of this instruction.

2. This instruction does not operate with priority levels 0_{16} through 11_{16}.

INSTRUCTION PRIORITY LEVEL FIELD

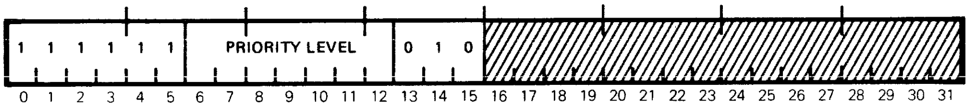
Bits 6 through 12	Priority Level (Hex.)
0000000	00
0000001	01
0000010	02
- - -	- - -
1111110	7E
1111111	7F

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

REQUEST INTERRUPT

FC02



DEFINITION

An interrupt request signal is applied to the interrupt level specified by the priority level field (bit 6 through bit 12) contained in the Instruction Word (IW). This signal simulates the signal generated by the internal or external condition which is connected to the specified level.

NOTE

This instruction does not operate with priority levels 2_{16} through 11_{16}.

INSTRUCTION PRIORITY LEVEL FIELD

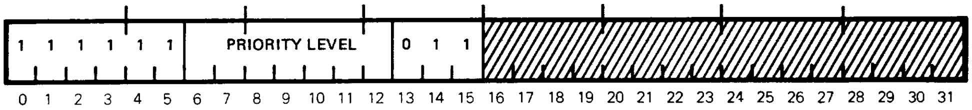
Bits 6 through 12	Priority Level (Hex.)
0000000	00
0000001	01
0000010	02
- - -	- - -
1111110	7E
1111111	7F

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

ACTIVATE INTERRUPT

FC03



DEFINITION

A signal is applied to set the active condition in the priority interrupt level specified by the priority level field (bit 6 through bit 12) contained in the Instruction Word (IW). The active level is set in the specified level regardless of whether or not that level is enabled. This condition prohibits this level and any lower levels not already in service from being serviced until this level is deactivated. However, request signals occurring at this or lower levels are stored for subsequent servicing.

NOTE

This instruction does not operate with priority levels 2_{16} through 11_{16}.

INSTRUCTION PRIORITY LEVEL FIELD

Bits 6 through 12	Priority Level (Hex.)
0000000	00
0000001	01
0000010	02
- - -	- - -
1111110	7E
1111111	7F

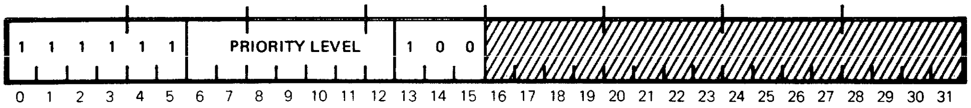
CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

DAI
V

DEACTIVATE INTERRUPT

FC04



DEFINITION

A signal is applied to reset the active condition for the priority interrupt level specified by the priority level field (bit 6 through bit 12) contained in the instruction word. Execution of the Deactivate Interrupt instruction does not clear any request signals on the specified level or any other level.

NOTE

This instruction does not operate with priority levels 2_{16} through 11_{16}.

INSTRUCTION PRIORITY LEVEL FIELD

Bits 6 through 12	Priority Level (Hex.)
0000000	00
0000001	01
0000010	02
- - -	- - -
1111110	7E
1111111	7F

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

INPUT/OUTPUT INSTRUCTIONS

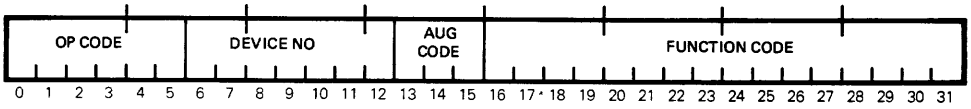
General Description

The Input/Output Instructions provide the capability to perform Command or Test operations to attached peripheral devices. Both the Command and the Test Instruction causes a 16-bit function code to be sent to the device specified by the instruction.

Instruction Formats

The following instruction format is used by both input/output instructions.

Input/Output



Bits 0-5	define the Operation Code.
Bits 6-12	designate the device number.
Bits 13-15	define the Augmenting Operation Code.
Bits 16-31	contain the 16-bit function code.

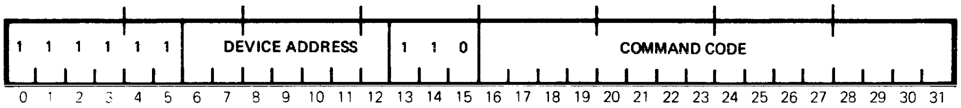
Condition Code Utilization

The Condition Code is set during execution of a test device instruction to indicate the result of the test being performed. The command device instruction leaves the current Condition Code unchanged.

CD
h,f

COMMAND DEVICE

FC06



DEFINITION

The contents of the command code field (bit 16 through bit 31) are transferred to the Device Controller Channel specified by the device address contained in bit positions 6 through 12 of the Instruction Word.

SUMMARY EXPRESSION

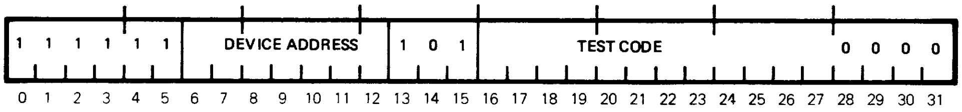
$IW_{16-31} \rightarrow DCC_N$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

TEST DEVICE

FC05



DEFINITION

The contents of the test code field (bit 16 through bit 27) are transferred to the Device Controller Channel (DCC), specified by the device address contained in bit positions 6 through 12 of the Instruction Word. The device test defined by the test code is performed in the DCC, and the test results are stored in condition code bits 1 to 4 (CC_{1-4}).

NOTES

1. A TD having a unique test code is available with most peripheral devices. Execution of a TD with this code causes a snapshot of all device and DCC status to be stored in memory. The individual peripheral device reference manuals define the operation of this instruction with each device.

2. Bits 28-31 of the Instruction Word must be zeros since the four-bit condition code is transferred to the CP over these bits of the I/O bus.

SUMMARY EXPRESSION

$IW_{16-31} \rightarrow DCC_N$

Test Results $\rightarrow CC_{1-4}$

CONDITION CODE RESULTS

Test results defined for specific peripheral device.

SECTION VI

CONTROL PANELS

GENERAL

The SEL 32 has two control panels; the Turnkey Panel and the System Control Panel. The Turnkey Panel is basic to system operation, however the System Control Panel is optional and provides many additional capabilities. This section describes each of these control panels, their function and operation.

TURNKEY PANEL

The Turnkey Panel provides the minimum system control and monitoring to perform system start-up and user to system communication. This communication is initiated and controlled through a combination of the panel Attention button and teletype query-response. A diagram of the Turnkey Panel is shown in figure 6-1.

FUNCTIONAL CONTROLS

The Turnkey Panel provides the following functional controls:

Panel Lock

The Panel Lock is a two position rotary keyswitch having an unlocked and locked position. The key can be removed in either position. When the switch is in the unlocked position, all panel keys on both the Turnkey Panel and System Control Panel, when present, are operational. In the locked position, all panel keys are disabled except for the Attention key and the System Control Panel buttons associated with loading and reading the Control Switches. The Attention and Control Switch loading are operational at all times.

Run/Halt

If the central processor is in a halt mode, and this button is depressed, the CPU will go into the run mode and begin executing instructions from the location specified in the Program Status Word.

If the central processor is in the run mode and this button is depressed, the CPU will go into the halt mode. In the halt mode, the processor no longer executes instructions from memory and instead is placed in a micro-routine which monitors selected panel support functions.

System Reset

When the system is in the halt mode, depressing this button initializes all appropriate logic in all SEL bus devices.

Attention

Depressing the Attention button causes an interrupt to occur at the Attention Interrupt level, priority level 13_{16}.

Initial Program Load

When the system is in the halt mode, depressing this button puts the CPU in the Initial Program Load mode. This initiates the microprogram loading sequence which consists of reading a dedicated device address and then reading from the specified device. The device number is entered through the System Control Panel, when present.

Power

The Power Switch is a two position latching pushbutton. This pushbutton provides the capability to power the system on or off. The state of the power is determined by the run and halt displays. When the power is on, either the run or halt display is on. When the power is off, all displays on the panel will be off.

Clock Override

When the Clock Override button is depressed, the override condition is activated and no further interrupts from the real-time clock or the interval timer will be permitted. A second depression of this button will deactivate the clock override.

DISPLAYS

The Turnkey Panel displays consist of single bit light emitting diodes (LED's). These displays are described below.

Parity Error

The Parity Error display, when lit, indicates that a memory parity error has occurred between the CPU and memory.

Interrupt Active

This display is on if any, I/O or external, interrupt is in the active state.

Wait

This display in on when the CPU is in the Wait state. In the Wait state, no instructions from memory are executed, however I/O operations continue to completion.

Run

The Run display is on (lit) when the CPU is in the run mode. In the run mode the processor is executing instructions fetched from memory.

Halt

The Halt display is on when the processor in in the halt mode. In the halt mode, no instructions are fetched from memory for execution. Also, all I/O operation is stopped.

Clock Override

This display is on when the clock override condition is active. The display is not lit when the clock override is inactive.

SYSTEM CONTROL PANEL

The System Control Panel is an optional panel which complements the functions provided by the Turnkey Panel. The System Control Panel provides the capability to selectively store/read data in memory or registers.

The System Control Panel functions as a separate input/output device on the SEL bus. It consists of two assemblies; the panel and the panel controller.

The following discussion provides a detailed description of each control switch and indicator on the Systems Control Panel. Following the control and indicator descriptions are the detail operating instructions that provide a step by step procedure for using the display and load functions provided by the System Control Panel.

Figure 6-2 provides an illustration of the System Control Panel. This figure is intended for use with the following discussions to provide the operator with the physical location of the controls and indicators on the System Control Panel.

CONTROLS

The controls for the System Control Panel consist of two keyboards, referred to as the Hexadecimal (HEX) Keyboard and the Function Keyboard as illustrated in figure 6-2. The following discussions of the Hex and Function Keyboards describes the keys contained on the two keyboards.

HEX KEYBOARD

The Hex Keyboard is used to either enter data into the B Display or to enter the source/destination of the panel function to be performed. The dual function of each Hex Keyboard Key is indicated by the upper and lower case values printed on each Hex Keyboard Key.

The upper case values are used when data is entered into the B Display. The upper case values are enabled by first depressing the Function Keyboard, KEYBOARD Key. The Function Keyboard KEYBOARD Key causes the B Display to be cleared and the KEYBOARD indicator to illuminate. When the KEYBOARD indicator is illuminated, all entries from the HEX Keyboard are interpreted as data and are entered into the B Display by left shifting the contents of the B Display by four bit places and entering the hex value of the depressed key into the four least significant bit positions (hex digit) of the B Display. If the 32-bit capacity of the B Display is exceeded, the most significant four bits of the B Display are shifted out of the display and lost and the new digit is loaded into the least significant bit positions.

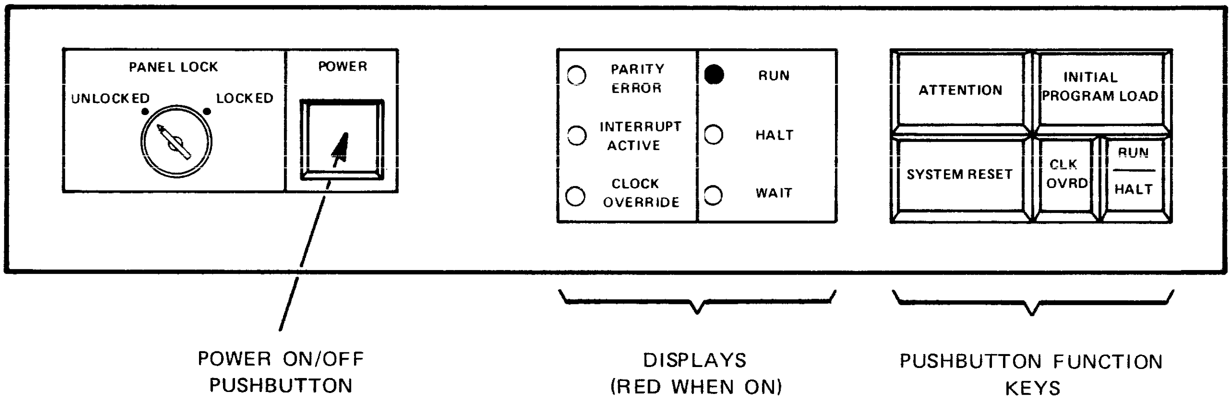


Figure 6-1. Turnkey Panel Diagram

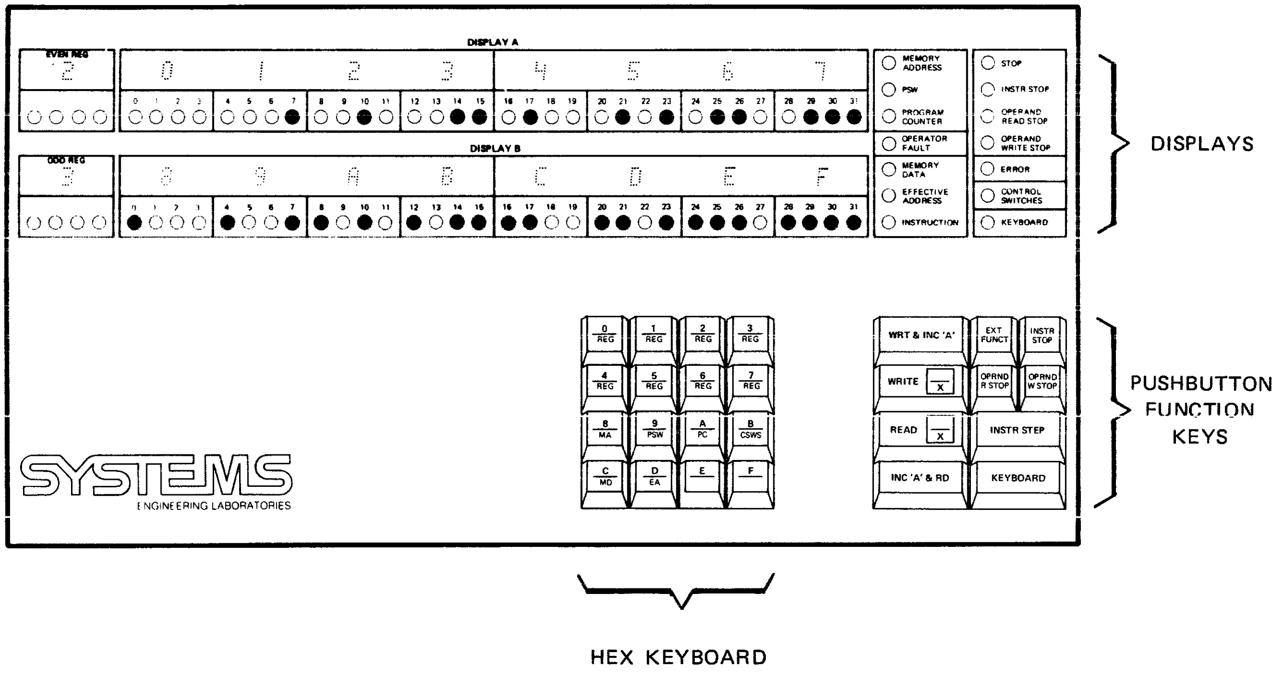


Figure 6-2. System Control Panel Diagram

The lower case values of the Hex Keyboard are used to specify the source/destination of a function to be performed by the Systems Control Panel. The lower case values are enabled by first depressing the Function Keyboard WRITE $\overline{X}$ Key or the READ $\overline{X}$ Key. The WRITE $\overline{X}$ or READ $\overline{X}$ keys cause the subsequent entry from the Hex Keyboard to be interpreted as the source/destination of the write or read function. When a source/destination is entered in the Hex Keyboard, it causes a corresponding miscellaneous indicator to illuminate on the Systems Control Panel. The Hex Keyboard keys that cause a miscellaneous indicator to illuminate are listed as follows:

a. The $\frac{0}{\text{REG}}$, $\frac{2}{\text{REG}}$, $\frac{4}{\text{REG}}$, and $\frac{6}{\text{REG}}$ keys cause the Even Register Hex Indicator.to indicate the hex value of the even register addressed.

b. The $\frac{1}{\text{REG}}$, $\frac{3}{\text{REG}}$, $\frac{5}{\text{REG}}$, and $\frac{7}{\text{REG}}$ keys cause the Odd Register Hex Indicator to indicate the hex value of the odd register addressed.

c. The $\frac{8}{\text{MA}}$ key causes the MEMORY ADDRESS Indicator to illuminate.

d. The $\frac{9}{\text{PSW}}$ key causes the PSW (Program Status Register) indicator to illuminate.

e. The $\frac{A}{\text{PC}}$ key causes the PROGRAM COUNTER indicator to illuminate.

f. The $\frac{B\ \text{key}}{\text{CSWS}}$ causes the CONTROL SWITCHES indicator to illuminate.

g. The $\frac{C}{\text{MD}}$ key causes the MEMORY DATA indicator to illuminate.

h. The $\frac{D}{\text{EA}}$ key causes the EFFECTIVE ADDRESS indicator to illuminate.

FUNCTION KEYBOARD

The Function Keyboard sets the function to be performed by the System Control Panel according to the key that is depressed. The functions that can be selected by the Function Keyboard keys are as follows:

WRITE $\boxed{X}$ Key

The WRITE $\overline{X}$ key causes the operand in the B Display to be stored in the destination specification by a subsequent depression of a Hex Keyboard key. The lower case value of the Hex Keyboard key describes the destination of the operand and the miscellaneous indicator that will illuminate. The use of the Hex Keyboard $\frac{D}{EA}$ key is prohibited for a destination of a write function.

If the Hex Keyboard $\frac{C}{MD}$ key is depressed, the contents of the A Display (which must contain a valid Memory Address, PSW, or Program Counter Value) is used to address memory and the operand in the B Display is stored at that memory address.

READ $\boxed{X}$ Key

The Read $\overline{X}$ key causes the operand specified by a subsequent depression of a Hex Keyboard key to be loaded into either the A or B Display. The lower case value of the Hex Keyboard key describes the source of the operand and the miscellaneous indicator that will illuminate. The use of the Hex Keyboard $\frac{8}{MA}$ key is prohibited as a source of a read function.

If the Hex Keyboard $\frac{C}{MD}$ key is depressed, the contents of the A Display (which must contain a valid Memory Address, PSW, or Program Counter value) is used to address memory and the contents of the addressed memory location is loaded into the B Display.

WRT& INC "A" Key

The WRT & INC "A" key causes the operand in the B Display to be stored in the memory location addressed by the A Display and then the A Display is incremented by four (one memory word). If the A Display contains a PSW or Program Counter value, the incremented value of the A Display is also loaded into the CPU Program Status Word register. The A Display must contain a value memory address and theB Display must contain the operand to be stored in memory. The WRT & INC "A" key is used for write functions to sequential memory locations.

INC "A" & RD Key

The INC "A" & RD key causes the address in the A Display to be incremented by four (one memory word) and the updated address is used to address memory. The contents of the addressed memory location is then loaded into the B Display. If the A Display contains a PSW or Program Counter value, the incremented value of the A Display is loaded into the CPU Program Status Word register. The A Display must contain a valid memory address. The INC "A" & RD Key is used for read functions of sequential memory locations.

INSTR STOP Key

The INSTR STOP key causes the Instruction Stop function to become active or inactive. If the Instruction Stop function was previously active and the INSTR STOP indicator was illuminated, the Function Keyboard INSTR STOP key causes the function to go inactive and the indicator to Turn OFF. If the Instruction Stop function was previously inactive and the INSTR STOP indicator was OFF, the Function Keyboard INSTR STOP key causes the Instruction Stop function to become active or INSTR STOP indicator to illuminate, and the memory address in the B Display to be loaded into the interface Compare register. When the CPU fetches an instruction form the memory location specified by the Compare register, the STOP indicates or will illuminate and the CPU will halt. The B Display must be loaded with the instruction address by way of the Hex Keyboard prior to depressing the INSTR STOP key on the Function Keyboard.

OPRND R STOP Key

The OPRND R STOP key causes the Operand Read Stop function to become active or inactive. If the Operand Read Stop function was previously active and the OPERAND READ STOP indicator was illuminated, the Function Keyboard OPRND R STOP key causes the function to go inactive and the indicator to turn OFF. If the Operand Read Stop was previously inactive, the Function Keyboard OPRND R STOP key causes the Operand Read Stop function to become active, the OPERAND READ STOP indicator to illuminate, and the memory address in the B Display to be loaded into the interface Compare register. When the CPU reads an operand from the specified memory location, the STOP indicator will illuminate and the CPU will halt. The B Display must be loaded with the operand memory address by way of the Hex Keyboard prior to depressing the OPRND R STOP key.

OPRND W STOP Key

The OPRND W STOP key causes the Operand Write Stop function to become active or inactive. If the Operand Write Stop function was previously active and the OPERAND WRITE STOP indicator was illuminated, the Function Keyboard OPRND W STOP key causes the function to go inactive and the indicator to turn OFF. If the Operand Write Stop was previously inactive the Function Keyboard OPRND W STOP key causes the Operand Write Stop function to become active, and the OPERAND WRITE STOP indicator to illuminate, and the memory address in the B Display to be loaded into the interface Compare register. When the CPU stores an operand in the specified memory location, the STOP indicator will illuminate and the CPU will halt. The B Display must be loaded with the operand memory address by way of the Hex Keyboard prior to depressing the OPRND W STOP key.

INSTR STEP Key

The INSTR STEP key causes both the A and B Displays and all miscellaneous indicators except the Instruction and Operand STOP indicators to be cleared and then causes the CPU to execute one software instruction that is addressed by the CPU Program Status Word register. After the one instruction has been executed, the CPU comes to a halt.

KEYBOARD Key

The KEYBOARD key causes the B Display to be cleared, the KEYBOARD indicator to illuminate, and any subsequent Hex Keyboard entries to be interpreted at their upper case values and inserted into the four right most bit positions of the B Display. The KEYBOARD key is normally used to clear the B Display prior to entering an operand into the B Display from the Hex Keyboard.

EXT FUNCT Key

The EXT FUNCT key provides a set of additional functions to the System Control Panel. These Extended Functions can only be used in a maintenance environment. Therefore, the Extended Function operations are documented in the Technical Manual provided for the SYSTEMS Control Panel.

INDICATORS

The following discussions describe each indicator or group of indicators on the Systems Control Panel. The indicators are divided into three functional groups for discussion purposes and these groups are referred to as the A Display, B Display and miscellaneous indicators. Figure 6-2 illustrates the physical location of each indicator on the Systems Control Panel.

A DISPLAY

The A Display consists of thirty-two binary indicators that are divided into eight 4-bit fields for easy hexadecimal read-out. When the Hex Display option is included in the Systems Control Panel, a hex display indicator is located above each 4-bit field, providing a direct hex read-out of the contents of the 4-bit field.

The contents of the A Display is described by the miscellaneous indicators located directly to the right of the A Display or the Even Register hex display indicator to the left of the A Display. The contents of the A Display can be any of the following:

a. A Memory Address in bit positions 08 through 31.

b. The contents of the CPU Program Status Word register.

c. The Program Counter bits from the CPU Program Status Word register in display bit positions 08 through 31.

d. The contents of any of four even numbered CPU General Purpose registers.

The A Display can be loaded in either a Write or Read function as specified by the corresponding keys of the Function Keyboard. In a Write function the B Display must be loaded with an operand or address by way of the Hex Keyboard. Then the Function Keyboard WRITE $\overline{X}$ key must be depressed to specify the Write function. Next the Hex Keyboard lower case value (operand destination) must be specified by depressing one of the even numbered register keys or the MA, PSW, or PC keys. When these operations are complete, the contents of the B Display is transferred to the A Display and corresponding CPU register and one of the A Display miscellaneous indicators will illuminate to define the contents of the A Display.

In a Read function the A Display is loaded by depressing the Read $\overline{X}$ key on the Function Keyboard followed by the lower case value (Operand source) key on the Hex Keyboard. Of the Hex Keyboard keys, only an even numbered register key, or the PSW, or PC keys can be used to specify the source operand for the Read function. When the Read function is complete the operand specified by the Hex Keyboard will be loaded into the A Display and the corresponding miscellaneous indicator will illuminate to define the contents of the A Display.

When the A Display contains a Memory Address, Program Status Word, or Program Counter, the contents of the A Display can be used to address memory during memory Read or Write functions. In these types of functions the WRT & INC "A" and the INC "A" & RD keys of the Function Keyboard can be used to access memory and increment the contents of the A Display to the next sequential memory word address.

B DISPLAY

The B Display consists of thirty-two binary indicators that are divided into eight 4-bit fields for easy hexadecimal read-out. When the Hex Display option is included in the Systems Control Panel, a hex display indicator is located above each 4-bit field, providing a direct hex read-out of the contents of the 4-bit field.

The contents of the B Display is described by the miscellaneous indicators, located directly to the right of the B Display or the Odd Register hex display indicator, located to the left of the B Display. The contents of the B Display can be any of the following.

a. Keyboard data being entered from the Hex Keyboard.

b. A memory data word.

c. An Effective Address of the instruction addressed by the PSW or PC in the A Display.

d. An instruction addressed by the PSW or PC in the A Display.

e. A four bit value of a Systems Control Panel error code.

f. The contents of the CPU Control switches in bit positions 00 through 11 of the B Display.

g. The contents of any of four odd numbered CPU General Purpose registers.

The B Display can be loaded in either a Write or Read function as specified by the corresponding keys of the Function Keyboard. In a Write function the B Display is loaded from the Hex Keyboard. After an operand has been loaded into the B Display from the Hex Keyboard the Function Keyboard WRITE $\overline{X}$ must be depressed to specify the Write function.

After the Write function has been specified, the contents of the B Display can be transferred to the A Display by depressing any even numbered register key, the MA key, the PSW key, or the PC key on the Hex Keyboard to specify the operand destination. The contents of the B Display can be transferred directly to an odd numbered register, the CPU Control switch register, or to the memory location addressed by the A Display by depressing any of the four odd numbered register keys, the PCSWS key, or the MD key on the Function Keyboard to specify the operand destination. In either case when the operation is complete, the corresponding miscellaneous indicators will illuminate to define the contents of the B and/or A Display.

In a Read function the B Display is loaded by depressing the READ $\overline{X}$ key on the Function Keyboard followed by the lower case value (operand source) key on the Hex Keyboard. Of the Hex Keyboard keys, only an odd numbered register key, the CSWS key, the MD key, or the EA key can be used to specify the source operand for the B Display. When the Read function is complete, the corresponding miscellaneous indicator will illuminate to define the contents of the B Display.

MISCELLANEOUS INDICATORS

The miscellaneous indicators consist of all indicators other than the A and B Display indicators. The following paragraphs describe the miscellaneous indicators and their primary functions.

EVEN REGISTER Indicator

The EVEN REGISTER indicator consists of a hexadecimal display indicator that provides a direct read-out of the even numbered register being addressed by the Systems Control Panel. The contents of this register is displayed in the A Display. The only time the EVEN REGISTER indicator will be illuminated is when the A Display contains the contents of an even numbered register.

It should be noted that the four binary indicators directly below the EVEN REGISTER indicator do not correspond to the even register address and have no relationship to the EVEN REGISTER indicator.

ODD REGISTER Indicator

The ODD REGISTER indicator consists of a hexadecimal display indicator that provides a direct read-out of the odd numbered register being addressed by the Systems Control Panel. The contents of this register is displayed in the B Display. The only time the ODD REGISTER indicator will be illuminated is when the B Display contains the contents of an odd numbered register.

It should be noted that the four binary displays directly below the ODD REGISTER indicator do not correspond to the odd register address and have no relationship to the ODD REGISTER indicator.

MEMORY ADDRESS Indicator

The MEMORY ADDRESS INDICATOR is a one bit display that defines the contents of the A Display as a memory address. The memory address can only be loaded into the A Display with a Write function. The memory address is primarily used for memory addressing in subsequent memory read or write operations.

PSW Indicator

The PSW indicator is a one bit display that defines the contents of the A Display as the CPU Program Status Word register. The PSW can be loaded into the A Display with either a Word or a Read function. The PSW can be used to change the contents of the CPU PSW register and for memory addressing in subsequent memory read or write operations.

PROGRAM COUNTER Indicator

The PROGRAM COUNTER Indicator is a one bit display that defines the contents of the A Display as the current value of the CPU Program Counter portion of the Program Status Word register. The Program Counter can be loaded into the A Display with either a Write or a Read Function. The Program Counter can be used to change the Program Counter portion of the Program Status Word register and for memory addressing in subsequent memory read or write operations.

OPERATOR FAULT Indicator

The OPERATOR FAULT indicator is a one bit display that indicates that an operator fault has occurred on Systems Control Panel. Four types of Operator Faults can normally occur as follows:

a. More than one key was simultaneously depressed on the Function Keyboard.

b. More than one key was simultaneously depressed on the Hex Keyboard.

c. The function selected by the Function Keyboard was illogical with respect to the operand source/destination selected by the Hex Keyboard.

d. The function selected by the Function Keyboard combined with the operand source/destination specified by the Hex Keyboard can not be performed due to one of the following:

    1. The CPU Turnkey Panel is in a Locked mode and the specified function is not allowed.

    2. The CPU is in a Run mode and the specified function is not allowed.

The specific type of Operator Fault that has occurred must be determined by the Systems Control Panel operator.

MEMORY DATA Indicator

The MEMORY DATA indicator is a one bit display that defines the content of the B Display as memory data from the memory location addressed by the A Display. For the MEMORY DATA indicator to be illuminated, the A Display must contain a memory address and the MEMORY ADDRESS indicator must be illuminated. Memory data can be manually loaded into the B Display and the addressed memory location in a Write function or read from the addressed memory location in a Read function.

EFFECTIVE ADDRESS Indicator

The EFFECTIVE ADDRESS indicator is a one bit display that defines the contents of the B Display as an effective address of a software, memory reference instruction that is addressed by the contents of the A Display. The A Display must contain either a PSW or Program Counter value which is used by the CPU to access the software memory reference instruction. The CPU then computes the instructions effective address based on any indexed or indirect addressing specified by the instruction. When the addressing is complete the effective address is returned to the B Display. The effective address can only be loaded into the B Display by a Read Function.

INSTRUCTION Indicator

The INSTRUCTION indicator is a one bit display that defines the contents of the B Display as an instruction addressed by a PSW or Program Counter value in the A Display. An instruction can be manually loaded into the B Display and addressed memory location in a Write function or read into the B Display from the addressed memory location in a Read function. It should be noted that the Systems Control Panel defines the contents of any memory location as an instruction if the A Display contains a PSW or Program Counter value. If the A Display contains a Memory Address (the MEMORY ADDRESS indicator is illuminated) the contents of the addressed memory location is defined as memory data, which illuminates the MEMORY DATA indicator.

STOP Indicator

The STOP indicator is a one bit display that indicates that the CPU has been halted by the Instruction Stop, Operand Read Stop, or Operand Write Stop logic. In addition to the STOP indicator, one or more of the INSTR STOP, OPERAND READ STOP, or OPERAND WRITE STOP indicators should be illuminated to indicate the type of stop logic that is active in the Systems Control Panel. When the STOP indicator illuminates and the CPU halts, the A Display will contain the current contents of the CPU PSW register and the B Display will contain the instruction addressed by the Program Counter portion of the PSW (A Display).

INSTR STOP Indicator

The INSTR STOP indicator is a one bit display that defines the active condition of the Instruction Stop logic. When Instruction Stop is active a memory address is in the interface compare register and when the CPU fetches an instruction from that memory location the CPU will be halted and the STOP indicator will illuminate.

OPERAND READ STOP Indicator

The OPERAND READ STOP indicator is a one bit display that defines the active condition of the Operand Read Stop logic. When Operand Read Stop is active, a memory address is in the interface Compare register and when the CPU performs a memory read from that location the CPU will be halted and the STOP indicator will illuminate.

OPERAND WRITE STOP Indicator

The OPERAND WRITE STOP indicator is a one bit display that defines the active condition of the Operand Write Stop logic. When the Operand Write Stop is active, a memory address is in the intprface Compare register, and the CPU performs a memory write to that location the CPU will be halted and the STOP indicator will illuminate.

ERROR Indicator

The ERROR indicator is a one bit display that defines the contents of the B Display as a internal error code. The specific definitions of the error codes are provided in table 6-1. The internal errors exclude operator errors and include Systems Control Panel errors, CPU acknowledge errors, SEL Bus transmission errors, and memory errors.

CONTROL SWITCHES Indicator

The CONTROL SWITCHES indicator is a one bit display that defines the contents of the B Display as the CPU Control Switches. The Control Switches can be loaded into the B Display in either a Write or a Read function. In a Write function the B Display is loaded from the Hex Keyboard and then the contents of the B Display (Control Switches) is loaded into a dedicated memory location. in a Read function the Systems Control Panel reads the dedicated memory location and transfers its contents (Control Switches) to the B Display.

The specific dedicated memory address used for storage of the Control Switches is a function of the computer system configuration and CPU firmware. The dedicated address must be specified at the time the computer system is ordered.

KEYBOARD Indicator

The KEYBOARD indicator is a one bit display that defines that the upper case values (hex digits 0 through F) cna be loaded into the B Display from the Hex Keyboard. The KEYBOARD indicator illuminates in response to the KEYBOARD switch on the Function Keyboard.

Table 6-1. B Display-Error Code Definitions

B Display Contents	Error Definition
1	Hex Keyboard or Function Keyboard Change Indicator did not reset.
2	No Hex or Function Keyboard Key detected.
3	No Response from memory
4	Non-present memory
5	Parity Error in memory
6	Write/Read Compare Error (memory)
7	SEL Bus Communication Error

OPERATING INSTRUCTIONS

The following discussions provide step by step instructions for using the features provided by the Systems Control Panl. Each heading describes the function to be performed and the subsequent steps that must be performed in order to execute the function. Each discussion also provides the conditions of the Turnkey Panel that are required to execute the specific function. These two conditions are the Panel Lock switch and the RUN/HALT switch with associated indicators.

It should be noted that the Load B Display from Hex Keyboard description and the Load A Display description provide the primary functions of the Systems Control Panel that are necessary for all other functions. After these descriptions are initially presented, they are referred to by their titles in subsequent descriptions.

It should also be noted that during Write functions to CPU registers (except PSW Register) or to the memory, that the Systems Control Panel automatically performs a read after write operation to verify that the Write function was performed correctly. In these cases the operand displayed in the A or B Display is the data read out of the register or memory in the read after write operation. Thus, if the operand entered into the Systems Control Panel from the Hex Keyboard does not compare to the operand in the A or B Display following the Write function execution, it can be assumed that a malfunction occurred within in the computer system during the Write or subsequent read after write operation. If a malfunction does occur and the OPERATOR FAULT and ERROR indicators do not illuminate, the Write function should be repeated.

LOAD B DISPLAY FROM HEX KEYBOARD

a. The Turnkey Panel can be in the LOCKED or UNLOCKED mode.

b. The CPU can be in the RUN or HALT mode.

c. Depress the KEYBOARD key on the Function Keyboard.

d. Observe that the B Display clears and the KEYBOARD indicator illuminates.

e. Enter the operand into the B Display by depressing the correct hex digit key on the Hex Keyboard, one digit at a time.

f. Observe that the digit just entered from the Hex Keyboard is loaded into the four least significant bit positions of the B Display and that any previous contents of the B Display is left-shifted by four bit positions.

g. When the B Display is full or the complete operand has been entered into the B Display the operation is complete.

LOAD B (Cont'd)

h. If the 32-bit capacity of the B Display is exceeded, the four most significant bit positions of the B Display will be lost as each new digit is entered into the B Display.

i. If a mistake is made while entering the operand, depress the KEYBOARD key on the Function Keyboard and return to step d.

LOAD A DISPLAY

The Load A Display function can be divided into five sub-functions that are described separately in the following descriptions. The five sub-functions are:

a. Write Memory Address.

b. Write PSW (Program Status Word).

c. Read PSW (Program Status Word).

d. Write Program Counter.

e. Read Program Counter.

Write Memory Address

a. The Turnkey Panel must be in the UNLOCKED mode.

b. The CPU can be in a RUN or HALT mode.

c. Enter the Memory Address into the B Display from the Hex Keyboard.

d. Depress the WRITE $\frac{\ }{X}$ key on the Function Keyboard.

e. Depress the $\frac{8}{MA}$ key on the Hex Keyboard.

f. Observe that the Memory Address is transferred from the B Display to the A Display and that the MEMORY ADDRESS indicator illuminates.

g. The operation is complete. If a mistake was made during the sequence, return to step c.

Write PSW (Program Status Word)

a. The Turnkey Panel must be in the UNLOCKED mode.

b. The CPU must be in the HALT mode.

c. Enter the PSW operand into the B Display from the Hex Keyboard.

d. Depress the WRITE $\overline{\text{X}}$ key on the Function Keyboard.

e. Depress the $\frac{9}{\text{PSW}}$ key on the Hex Keyboard.

f. Observe that the PSW operand is transferred from the B Display to the A Display and that the PSW indicator illuminates. At this time the PSW operand has also been loaded into the CPU Program Status Word register.

g. The operation is complete. If a mistake was made during the sequence, return to step c.

Read PSW (Program Status Word)

a. The Turnkey Panel must be in the UNLOCKED mode.

b. The CPU must be in the HALT mode.

c. Depress the READ $\overline{\text{X}}$ key on the Function Keyboard.

d. Depress the $\frac{9}{\text{PSW}}$ key on the Hex Keyboard.

e. Observe that the Program Status Word is transferred from the CPU PSW register to the A Display and that the PSW indicator illuminates.

f. The operation is complete. If a mistake was made during the sequence return to step c.

Write Program Counter

a. The Turnkey Panel must be in the UNLOCKED mode.

b. The CPU must be in the HALT mode.

c. Enter the Program Counter value into bits 08 through 31 of the B Display from the Hex Keyboard.

d. Depress the WRITE $\overline{\text{X}}$ key on the Function Keyboard.

e. Depress the $\frac{\text{A}}{\text{PC}}$ key on the Hex Keyboard.

f. Observe that bits 13 through 31 of the B Display are transferred to the A Display and that the PROGRAM COUNTER indicator illuminates. At this time the Program Counter value has been loaded into the Program Counter portion of the CPU Program Status Word register.

g. The operation is complete. If a mistake was made during the sequence, return to step c.

Read Program Counter

a. The Turnkey Panel must be in the UNLOCKED mode.

b. The CPU must be in the HALT mode.

c. Depress the READ $\overline{\text{X}}$ key on the Function Keyboard.

d. Depress the $\frac{\text{A}}{\text{PC}}$ key on the Hex Keyboard.

e. Observe that the Program Counter value is unloaded from the CPU Program Status Word register and transferred to bits 13 through 31 of the A Display, and that the PROGRAM COUNTER indicator illuminates.

f. The operation is complete. If a mistake was made during the sequence return to step c.

WRITE MEMORY

The Write Memory sequence is dependent on a valid address (Memory Address PSW, or Program Counter value) in the A Display. This value can be set-up in the A Display by using any of the sub-functions described in the Load A Display discussion.

The type of address in the A Display determines if the CPU must be in a HALT mode for the Write Memory function to be executed. The rules for the RUN/HALT mode are as follows:

a. If a Memory Address is in the A Display the CPU can be in either the RUN or HALT mode.

b. If a PSW or Program Counter value is in the A Display the CPU must be in the HALT mode.

Write Memory (Single Address)

a. The Turnkey Panel must be in the UNLOCKED mode.

b. Select either the RUN or HALT CPU mode according to the type of operand to be used to address memory from the A Display.

c. Enter a Memory Address, PSW or Program Counter value into the A Display as described in the Load A Display discussion.

d. Enter the operand to be stored in memory into the B Display from the Hex Keyboard.

e. Depress the WRITE $\overline{\text{X}}$ key on the Function Keyboard.

f. Depress the $\frac{\text{C}}{\text{MD}}$ key on the Hex Keyboard.

g. Observe that the operand in the B Display remains unchanged and that either the MEMORY DATA or INSTRUCTION indicator illuminates as follows:

   1. If the A Display contains Memory Address, the MEMORY DATA indicator should illuminate.

   2. If the A Display contains either a PSW or Program Counter value, the INSTRUCTION indicator should illuminate.

h. The operation is complete. If a mistake occurred during the sequence, return to step b.

READ MEMORY

The Read Memory sequence is dependent on a valid address (Memory Address, PSW, or Program Counter value) in the A Display. This value can be set-up in the A Display by using any of the sub-functions described in the Load A Display discussion.

The type of address in the A Display determines if the CPU must be in a HALT mode for the Read Memory function to be executed. The rules for the RUN/HALT mode are as follows:

a. If a Memory Address is in the A Display the CPU can be in either the RUN or HALT mode.

b. If a PSW or Program Counter value is in the A Display the CPU must be in a HALT mode.

Read Memory (Single Address)

a. The Turnkey Panel must be in the UNLOCKED mode.

b. Select either the RUN or HALT mode according to the type of operand to be used to address memory in the A Display.

c. Enter a Memory Address, PSW, or Program Counter value into the A Display as described in the Load A Display discussion.

d. Depress the READ $\overline{X}$ key on the Function Keyboard.

e. Depress the $\frac{C}{MD}$ key on the Hex Keyboard.

f. Observe that the A Display is unchanged.

g. Observe that the MEMORY DATA or INSTRUCTION indicator illuminates as follows:

   1. If the A Display contains a Memory Address the MEMORY DATA indicator should illuminate.

   2. If the A Display contains a PSW or Program Counter value the INSTRUCTION indicator should illuminate.

h. The operand in the B Display should be the contents of the memory location address by the A Display.

i. The operation is complete. If a mistake has occurred during the sequence return to step b.

Read Memory (Sequential Addresses)

a. The Turnkey Panel must be in the UNLOCKED mode.

b. Select either the RUN or HALT CPU mode according to the type of operand to be used to address memory in the A Display.

c. Enter a Memory Address, PSW, or Program Counter value into the A Display as described in the Load A Display discussion.

d. Depress the INC "A" & RD key on the Function Keyboard.

e. Observe that the A Display is incremented by four to the next sequential memory address.

f. Observe that the MEMORY DATA or INSTRUCTION indicator illuminates as follows:

   1. If the A Display contains a Memory Address the MEMORY DATA indicator should illuminate.

   2. If the A Display contains a PSW or Program Counter value the INSTRUCTION indicator should illuminate.

g. The operand in the B Display should be the contents of the memory location addressed by the A Display.

h. If no mistakes occurred in the above sequence return to step d to read the next memory location.

i. If a mistake did occur, the same memory address can be reread by performing the Read Memory (Single Address) sequence from step d.

It should be noted that when using the Read Memory (Sequential Addresses) sequence, that the first address entered into the A Display will not be read. If it is desired to read the first address; first, perform the Read Memory (Single Address) sequence and then enter the Read Memory (Sequential Addresses) sequence at step d.

INSTRUCTION STEP

The Instruction Step function causes the CPU to enter the RUN mode and execute one software instruction. After the instruction has been executed the CPU returns to the HALT mode.

The sequence for the Instruction Step function is as follows:

a. The Turnkey Panel must be in the UNLOCKED mode.

b. The CPU must be in the HALT mode.

c. If the CPU Program Status Word register does not point to the instruction to be executed, load a Program Counter or PSW value into the A Display and CPU PSW register as described in the Load A Display description.

d. Depress the INSTR STEP key on the Function Keyboard.

e. Observe that the Turnkey Panel HALT indicator is illuminated.

f. The system halts with the updated PSW value in the A Display and instruction addressed by the A Display (PSW) in the B Display.

g. If it desired to execute the next instruction return to step d.

READ EFFECTIVE ADDRESS

The Read Effective Address sequence causes the CPU to fetch the instruction addressed by the Program Counter or PSW value in the A Display. The instruction fetched should be a memory reference instruction in order to generate a valid effective address. After the instruction has been fetched, the CPU calculates the instructions effective memory address by performing the indexing and indirect addressing specified by the instruction. When the address computations are complete the CPU transfers the effective address to the Systems Control Panel B Display.

The sequence for the Read Effective Address is as follows:

a. The Turnkey Panel must be in the UNLOCKED mode.

b. The CPU must be in the HALT mode.

c. Enter a PSW or Program Counter value into the A Display as described in the Load A Display discussion.

d. Depress the Read $\overline{X}$ key on the Function Keyboard.

e. Depress the $\frac{\text{D}}{\text{EA}}$ key on the Hex Keyboard.

f. Observe that the EFFECTIVE ADDRESS indicator illuminates and the effective address is loaded into the B Display.

g. The operation is complete. If a mistake occurred, return to step c.

STOP SEQUENCE

The Stop sequence includes the Instruction Stop, Operand Read Stop, and Operand Write Stop functions. Each of the three functions is provided with its own key on the Function Keyboard and its own indicator to indicate when that function is active.

The sequence for the Stop functions is as follows:

a. The Turnkey Panel must be in the UNLOCKED mode.

b. The CPU can be in either the RUN or HALT mode.

c. Enter the memory stop address into the B Display from the Hex Keyboard.

d. Depress the INSTR STOP, OPRND R STOP, or OPRND W STOP key on the Function Keyboard.

e. Observe that the indicator for the stop function selected by the Function Keyboard illuminates.

f. If the CPU is in a RUN mode and the specified memory location is accessed in the correct operating mode (instruction fetch, operand read, or operand write) the following events should occur.

   1. The Turnkey Panel HALT indicator should illuminate.

   2. The STOP indicator should illuminate.

   3. The current contents of the CPU PSW register should appear in the A Display and the PSW indicator should illuminate.

   4. The instruction addressed by the Program Counter portion of the PSW should appear in the B Display and the INSTRUCTION indicator should illuminate.

g. If it is desired to clear any active Stop function perform the following steps.

   1. Depress the Function Keyboard key that corresponds to the function to be cleared.

   2. Observe that the correspond Stop function indicator turns OFF.

It should be noted that when using the Stop function, that multiple Stop functions can be set-up by entering the Stop functions sequentially; however, if different stop address are entered with each Stop function the most recently entered stop address will be used for all active Stop functions.

CONTROL SWITCHES SEQUENCE

The Control Switches sequence is used to set-up or monitor the CPU Control Switches that are stored in a dedicated memory location. The Control Switches sequence is divided into the Write Control Switches function which sets-up the Control Switches in the dedicated memory location or the Read Control Switches function that reads out the contents of the dedicated memory location.

Write Control Switches

a. The Turnkey Panel can be in the LOCKED or UNLOCKED mode.

b. The CPU can be in the RUN or HALT mode.

c. Enter the Control Switch configuration into bit positions 00 through 12 of the B Display from the Hex Keyboard.

d. Depress the WRITE $\overline{X}$ key on the Function Keyboard.

e. Depress the $\frac{B}{CSWS}$ key on the Hex Keyboard.

f. Observe that the CONTROL SWITCHES indicator illuminates. At this time the contents of the B Display has been transferred to the Control Switches dedicated memory location.

g. The operation is complete. If a mistake occurred return to step c.

Read Control Switches

a. The Turnkey Panel can be in the LOCKED or UNLOCKED mode.

b. The CPU can be in the RUN or HALT mode.

c. Depress the READ $\overline{X}$ key on the Function Keyboard.

d. Depress the $\frac{B}{CSWS}$ key on the Hex Keyboard.

e. Observe that the CONTROL SWITCHES indicator illuminates and the contents of the Control Switches dedicated memory location is transferred to the B Display.

f. The operation is complete. If a mistake occurred return to step c.

INITIAL PROGRAM LOAD SEQUENCE

The Initial Program Load (IPL) sequence is a joint function of the Systems Control Panel and the Turnkey Panel. The Turnkey Panel provides the IPL key that initiates the IPL sequence and the Systems Control Panel provides the provision of loading a peripheral device address of the device containing the IPL "bootstrap" program.

The IPL sequence is as follows:

a. The Turnkey Panel must be in an UNLOCKED mode.

b. The CPU must be in a HALT mode.

c. Depress the SYSTEM RESET key on the Turnkey Panel.

d. Enter the peripheral device address of the IPL device into the B Display from the Hex Keyboard. Note: If an all zeros device address is entered into the B Display, the CPU firmware will default to a firmware specified IPL device address.

e. When the IPL sequence is complete, the CPU will be in the HALT mode and any changes in the software program can be made at this time.

f. The operation is complete. Refer to the software description of the "bootstrap" program for operating instructions of the "bootstrap" program.

OPERATOR FAULT INDICATORS

The Systems Control Panel is equipped with an OPERATOR FAULT indicator that illuminates when the Panel detects an operator fault condition. The specific type of fault that has occurred is not defined by the Systems Control Panel; however these faults include the following:

a. Depressing more than one key simultaneously on either the Hex Keyboard or Function Keyboard.

b. The function selected was illogical with respect to the operand source or destination.

c. The function selected can not be performed due to the LOCKED condition of the Turnkey Panel or the RUN mode in the CPU.

When an OPERATOR FAULT indication occurs the indicator can be cleared by depressing any key on the Function or Hex Keyboards. The key that is depressed can be also used to retry the function which resulted in the Operator Fault. If a retry of the function results in another Operator Fault, the operator must insure that the Turnkey Panel Panel Lock switch is in the correct position and that the CPU is in the correct RUN or HALT mode. If the Turnkey Panel setting are correct, the operator must insure that the procedure used for the function is correct as defined in the Operating Instructions discussion.

ERROR INDICATORS

The Systems Control Panel is equipped with an ERROR indicator that illuminates when a Panel error is detected. The specific type of error is defined by the error code in the B Display. Table 6-1 defines each of the error codes used with the Systems Control Panel.

When an ERROR indication occurs the indicator can be cleared by depressing any key on the Hex or Function Keyboards. The key that is depressed can be also used to retry the function which resulted in the error.

APPENDIX A

INSTRUCTION SET

(FUNCTIONALLY GROUPED)

SEL 32 instructions are listed alphabetically by mnemonic code within one of the following functional groupings:

- Load/Store Instructions
- Branch Instructions
- Compare Instructions
- Logical Instructions
- Register Transfer Instructions
- Shift Operation Instructions
- Bit Manipulation Instructions
- Fixed-Point Arithmetic Instructions
- Floating-Point Arithmetic Instructions
- Control Instructions
- Interrupt Instructions
- Input/Output Instructions

Each entry includes the following information:

- Instruction Mnemonic
- Operand Format
- Operation Code
- Instruction Function

The following symbols are used to denote required entries for operand formats:

b	-	Bit Number In General Register (0-31)
c	-	Bit Number In Memory Byte
d	-	Destination General Register (0-7)
f	-	Function
m	-	Memory Address
n	-	Channel Or Device Number
p	-	Protect Register Number
s	-	Source General Register (0-7)
v	-	Value of Operand For Immediate, Shift, and Condition Code Instructions
x	-	Index Register (1-3)
*	-	Indirect Addressing

Halfword instructions are denoted by an asterisk (*) preceding the instruction mnemonic.

LOAD/STORE INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
LB	d,*m,x	AC08	5-7	Load Byte
LD	d,*m,x	AC00	5-10	Load Doubleword
LH	d,*m,x	AC00	5-8	Load Halfword
LW	d,*m,x	AC00	5-9	Load Word
LEA	d,*m,x	D000	5-20	Load Effective Address
LF	d,*m,x	CC00	5-23	Load File
LI	d,v	C800	5-19	Load Immediate
LMB	d,*m,x	B008	5-11	Load Masked Byte
LMD	d,*m,x	B000	5-14	Load Masked Doubleword
LMH	d,*m,x	B000	5-12	Load Masked Halfword
LMW	d,*m,x	B000	5-13	Load Masked Word
LNB	d,*m,x	B408	5-15	Load Negative Byte
LND	d,*m,x	B400	5-18	Load Negative Doubleword
LNH	d,*m,x	B400	5-16	Load Negative Halfword
LNW	d,*m,x	B400	5-17	Load Negative Word
STB	s,*m,x	D408	5-25	Store Byte
STD	s,*m,x	D400	5-28	Store Doubleword
STH	s,*m,x	D400	5-26	Store Halfword
STW	s,*m,x	D400	5-27	Store Word
STF	s,*m,x	DC00	5-33	Store File
STMB	s,*m,x	D808	5-29	Store Masked Byte
STMD	s,*m,x	D800	5-32	Store Masked Doubleword
STMH	s,*m,x	D800	5-30	Store Masked Halfword
STMW	s,*m,x	D800	5-31	Store Masked Word
ZMB	*m,x	F808	5-35	Zero Memory Byte
ZMD	*m,x	F800	5-38	Zero Memory Doubleword
ZMH	*m,x	F800	5-36	Zero Memory Halfword
ZMW	*m,x	F800	5-37	Zero Memory Word
*ZR	d	0C00	5-39	Zero Register
*LCS	d	0003	5-22	Load Control Switches

*Indicates Halfword Instruction

BRANCH INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
BCF	v,*m,x	F000	5-46	Branch Condition False
BCT	v,*m,x	EC00	5-47	Branch Condition True
BFT	*m,x	F000	5-48	Branch Function True
BIB	d,m	F400	5-51	Branch After Incrementing Byte
BID	d,m	F460	5-54	Branch After Incrementing Doubleword
BIH	d,m	F420	5-52	Branch After Incrementing Halfword
BIW	d,m	F440	5-53	Branch After Incrementing Word
BL	*m,x	F880	5-50	Branch and Link
BRI	*m,x	F900	5-55	Branch and Reset Interrupt
BU	*m,x	EC00	5-44	Branch Unconditionally

COMPARE INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
CAMB	d,*m,x	9008	5-58	Compare Arithmetic with Memory Byte
CAMD	d,*m,x	9000	5-61	Compare Arithmetic with Memory Doubleword
CAMH	d,*m,x	9000	5-59	Compare Arithmetic with Memory Halfword
CAMW	d,*m,x	9000	5-60	Compare Arithmetic with Memory Word
*CAR	s,d	1000	5-62	Compare Arithmetic with Register
CI	d,v	C805	5-63	Compare Immediate
CMMB	d,*m,x	9408	5-64	Compare Masked with Memory Byte
CMMD	d,*m,x	9400	5-67	Compare Masked with Memory Doubleword
CMMH	d,*m,x	9400	5-65	Compare Masked with Memory Halfword
CMMW	d,*m,x	9400	5-66	Compare Masked with Memory Word
*CMR	s,d	1400	5-68	Compare Masked with Register

*Indicates Halfword Instruction

LOGICAL INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
ANMB	d,*m,x	8408	5-70	AND Memory Byte
ANMD	d,*m,x	8400	5-73	AND Memory Doubleword
ANMH	d,*m,x	8400	5-71	AND Memory Halfword
ANMW	d,*m,x	8400	5-72	AND Memory Word
*ANR	s,d	0400	5-74	AND Register and Register
EOMB	d,*m,x	8C08	5-81	Exclusive OR Memory Byte
EOMD	d,*m,x	8C00	5-84	Exclusive OR Memory Doubleword
EOMH	d,*m,x	8C00	5-82	Exclusive OR Memory Halfword
EOMW	d,*m,x	8C00	5-83	Exclusive OR Memory Word
*EOR	s,d	0C00	5-85	Exclusive OR Register and Register
*EORM	s,d	0C08	5-86	Exclusive OR Register and Register Masked
ORMB	d,*m,x	8808	5-75	OR Memory Byte
ORMD	d,*m,x	8800	5-78	OR Memory Doubleword
ORMH	d,*m,x	8800	5-76	OR Memory Halfword
ORMW	d,*m,x	8800	5-77	OR Memory Word
*ORR	s,d	0800	5-79	OR Register and Register
ORRM	s,d	0808	5-80	OR Register and Register Masked

REGISTER TRANSFER INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
*XCR	s,d	2C05	5-96	Exchange Registers
*XCRM	s,d	2C0D	5-97	Exchange Registers Masked
TPR	r,p	FB80	5-91	Transfer Protect Register to Register
*TRC	s,d	2C03	5-94	Transfer Register Complement
*TRCM	s,d	2C0B	5-95	Transfer Register Complement Masked
*TRN	s,d	2C04	5-92	Transfer Register Negative
*TRNM	s,d	2C0C	5-93	Transfer Register Negative Masked
TRP	s,p	FB00	5-90	Transfer Register to Protect Register
*TRR	s,d	2C00	5-88	Transfer Register to Register
*TRRM	s,d	2C08	5-89	Transfer Register to Register Masked
*TRSW	s	2800	5-98	Transfer Register to PSWR

*Indicates Halfword Instruction

SHIFT OPERATION INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
*NOR	d,s	6000	5-101	Normalize
*NORD	d,s	6400	5-102	Normalize Double
*SCZ	d,s	6800	5-103	Shift and Count Zeros
*SLA	d,v	6C40	5-104	Shift Left Arithmetic
*SLAD	d,v	7840	5-107	Shift Left Arithmetic Double
*SLC	d,v	7440	5-106	Shift Left Circular
*SLL	d,v	7040	5-105	Shift Left Logical
*SLLD	d,v	7C40	5-108	Shift Left Logical Double
*SRA	d,v	6C00	5-109	Shift Right Arithmetic
*SRAD	d,v	7800	5-112	Shift Right Arithmetic Double
*SRC	d,v	7400	5-111	Shift Right Circular
*SRL	d,v	7000	5-110	Shift Right Logical
*SRLD	d,v	7C00	5-113	Shift Right Logical Double

BIT MANIPULATION INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
ABM	c,*m,x	A008	5-120	Add Bit in Memory
*ABR	d,b	2000	5-121	Add Bit in Register
SBM	c,*m,x	9808	5-116	Set Bit in Memory
*SBR	d,b	1800	5-117	Set Bit in Register
TBM	c,*m,x	A408	5-122	Test Bit in Memory
*TBR	d,b	2400	5-123	Test Bit in Register
ZBM	c,*m,x	9C08	5-118	Zero Bit in Memory
*ZBR	d,b	1C00	5-119	Zero Bit in Register

FIXED-POINT ARITHMETIC INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
ADI	d,v	C801	5-138	Add Immediate
ADMB	d,*m,x	B808	5-128	Add Memory Byte
ADMD	d,*m,x	B800	5-131	Add Memory Doubleword
ADMH	d,*m,x	B800	5-129	Add Memory Halfword
ADMW	d,*m,x	B800	5-130	Add Memory Word
*ADR	s,d	3800	5-132	Add Register to Register
*ADRM	s,d	3808	5-133	Add Register to Register Masked

*Indicates Halfword Instruction

FIXED-POINT ARITHMETIC INSTRUCTIONS (Cont'd)

Mnemonic	Operand Format	Op Code	Page	Instruction Function
ARMB	s,*m,x	E808	5-134	Add Register to Memory Byte
ARMD	s,*m,x	E800	5-137	Add Register to Memory Doubleword
ARMH	s,*m,x	E800	5-135	Add Register to Memory Halfword
ARMW	s,*m,x	E800	5-136	Add Register to Memory Word
SUI	s,v	C802	5-145	Subtract Immediate
SUMB	d,*m,x	BC08	5-139	Subtract Memory Byte
SUMD	d,*m,x	BC00	5-142	Subtract Memory Doubleword
SUMH	d,*m,x	BC00	5-140	Subtract Memory Halfword
SUMW	d,*m,x	BC00	5-141	Subtract Memory Word
*SUR	s,d	3C00	5-143	Subtract Register from Register
*SURM	s,d	3C08	5-144	Subtract Register from Register Masked
MPMH	d,*m,x	C000	5-147	Multiply by Memory Halfword
MPMW	d,*m,x	C000	5-148	Multiply by Memory Word
*MPR	s,d	4000	5-149	Multiply Register by Register
MPI	d,v	C803	5-150	Multiply Immediate
MPMB	d,*m,x	C008	5-146	Multiply by Memory Byte
DVI	d,v	C804	5-155	Divide Immediate
DVMB	d,*m,x	C408	5-151	Divide by Memory Byte
DVMH	d,*m,x	C400	5-152	Divide by Memory Halfword
DVMW	d,*m,x	C400	5-153	Divide by Memory Word
*DVR	s,d	4400	5-154	Divide Register by Register
*ES	d	0004	5-156	Extend Sign
*RND	d	0005	5-157	Round Register

FLOATING POINT ARITHMETIC INSTRUCTIONS

Mnemonic	Operand	Op Code	Page	Instruction Function
ADFD	d,*m,x	E008	5-162	Add Floating Point Doubleword
ADFW	d,*m,x	E008	5-161	Add Floating Point Word
SUFD	d,*m,x	E000	5-164	Subtract Floating Point Doubleword
SUFW	d,*m,x	E000	5-163	Subtract Floating Point Word
MPFD	d,*m,x	E408	5-166	Multiply Floating Point Doubleword
MPFW	d,*m,x	E408	5-165	Multiply Floating Point Word
DVFD	d,*m,x	E400	5-168	Divide Floating Point Doubleword
DVFW	d,*m,x	E400	5-167	Divide Floating Point Word

*Indicates Halfword Instruction

CONTROL INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
*CALM	v	3000	5-176	Call Monitor
EXM	*m,x	A800	5-172	Execute Memory
EXR	s	C807	5-170	Execute Register
EXRR	s	C807	5-171	Execute Register Right
*HALT	-	0000	5-173	Halt
*NOP	-	0002	5-175	No Operation
*WAIT	-	0001	5-174	Wait

INTERRUPT INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
AI	v	FC03	5-181	Activate Interrupt
DAI	v	FC04	5-182	Deactivate Interrupt
DI	v	FC01	5-179	Disable Interrupt
EI	v	FC00	5-178	Enable Interrupt
RI	v	FC02	5-180	Request Interrupt

INPUT/OUTPUT INSTRUCTIONS

Mnemonic	Operand Format	Op Code	Page	Instruction Function
CD	n,f	FC06	5-184	Command Device
TD	n,f	FC05	5-185	Test Device

*Indicates Halfword Instruction

APPENDIX B
INSTRUCTION SET
(ALPHABETIZED MNEMONIC CODES)

SEL 32 instructions are listed alphabetically by mnemonic code. Each entry includes the following information:

- Instruction mnemonic
- Operand Format
- Operation Code
- Instruction Function

The following symbols are used to denote required entries for operand formats:

b	-	Bit Number In General Register (0-31)
c	-	Bit Number In Memory Byte
d	-	Destination General Register (0-7)
f	-	Function
m	-	Memory Address
n	-	Channel Or Device Number
p	-	Protect Register Number
s	-	Source General Register (0-7)
v	-	Value Of Operand For Immediate, Shift, and Condition Code Instructions
x	-	Index Register (1-3)
*	-	Indirect Addressing

Halfword instructions are denoted by an asterisk (*) preceding the instruction mnemonic.

Mnemonic	Operand Format	Op Code	Page	Instruction Function
ABM	c,*m,x	A008	5-120	Add Bit in Memory
*ABR	d,b	2000	5-121	Add Bit in Register
ADFD	d,*m,x	E008	5-162	Add Floating-Point Doubleword
ADFW	d,*m,x	E008	5-161	Add Floating-Point Word
ADI	d,v	C801	5-138	Add Immediate
ADMB	d,*m,x	B808	5-128	Add Memory Byte
ADMD	d,*m,x	B800	5-131	Add Memory Doubleword
ADMH	d,*m,x	B800	5-129	Add Memory Halfword
ADMW	d,*m,x	B800	5-130	Add Memory Word
*ADR	s,d	3800	5-132	Add Register to Registpr
*ADRM	s,d	3808	5-133	Add Register to Register Masked

*Indicates Halfword Instruction

Mnemonic	Operand Format	Op Code	Page	Instruction Function
AI	v	FC03	5-181	Activate Interrupt
ANMB	d,*m,x	8408	5-70	AND Memory Byte
ANMD	d,*m,x	8400	5-73	AND Memory Doubleword
ANMH	d,*m,x	8400	5-71	AND Memory Halfword
ANMW	d,*m,x	8400	5-72	AND Memory Word
*ANR	s,d	0400	5-74	AND Register and Register
ARMB	s,*m,x	E808	5-134	Add Register to Memory Byte
ARMD	s,*m,x	E800	5-137	Add Register to Memory Doubleword
ARMH	s,*m,x	E800	5-135	Add Register to Memory Halfword
ARMW	s,*m,x	E800	5-136	Add Register to Memory Word
BCF	v,*m,x	F000	5-46	Branch Condition False
BCT	v,*m,x	EC00	5-47	Branch Condition True
BFT	*m,x	F000	5-48	Branch Function True
BIB	d,m	F400	5-51	Branch After Incrementing Byte
BID	d,m	F460	5-54	Branch After Incrementing Doubleword
BIH	d,m	F420	5-52	Branch After Incrementing Halfword
BIW	d,m	F440	5-53	Branch After Incrementing Word
BL	*m,x	F880	5-50	Branch and Link
BRI	*m,x	F900	5-55	Branch and Reset Interrupt
BU	*m,x	EC00	5-44	Branch Unconditionally
*CALM	v	3000	5-176	Call Monitor
CAMB	d,*m,x	9008	5-58	Compare Arithmetic with Memory Byte
CAMD	d,*m,x	9000	5-61	Compare Arithmetic with Memory Doubleword
CAMH	d,*m,x	9000	5-59	Compare Arithmetic with Memory Halfword
CAMW	d,*m,x	9000	5-60	Compare Arithmetic with Memory Word
*CAR	s,d	1000	5-62	Compare Arithmetic with Register
CD	n,f	FC06	5-184	Command Device
CI	d,v	C805	5-63	Compare Immediate
CMMB	d,*m,x	9408	5-64	Compare Masked with Memory Byte
CMMD	d,*m,x	9400	5-67	Compare Masked with Memory Doubleword
CMMH	d,*m,x	9400	5-65	Compare Masked with Memory Halfword
CMMW	d,*m,x	9400	5-66	Compare Masked with Memory Word

*Indicates Halfword Instruction

Mnemonic	Operand Format	Op Code	Page	Instruction Function
*CMR	s,d	1400	5-68	Compare Masked with Register
DAI	v	FC04	5-182	Deactivate Interrupt
DI	v	FC01	5-179	Disable Interrupt
DVFD	d,*m,x	E400	5-168	Divide Floating-Point Doubleword
DVFW	d,*m,x	E400	5-167	Divide Floating-Point Word
DVI	d,v	C804	5-155	Divide Immediate
DVMB	d,*m,x	C408	5-151	Divide by Memory Byte
DVMH	d,*m,x	C400	5-152	Divide by Memory Halfword
DVMW	d,*m,x	C400	5-153	Divide by Memory Word
*DVR	s,d	4400	5-154	Divide Register by Register
EI	v	FC00	5-178	Enable Interrupt
EOMB	d,*m,x	8C08	5-81	Exclusive OR Memory Byte
EOMD	d,*m,x	8C00	5-84	Exclusive OR Memory Doubleword
EOMH	d,*m,x	8C00	5-82	Exclusive OR Memory Halfword
EOMW	d,*m,x	8C00	5-83	Exclusive OR Memory Word
*EOR	s,d	0C00	5-85	Exclusive OR Register and Register
*EORM	s,d	0C08	5-86	Exclusive OR Register and Register Masked
*ES	c	0004	5-156	Extend Sign
EXM	*m,x	A800	5-172	Execute Memory
EXR	s	C807	5-170	Execute Register
EXRR	s	C807	5-171	Execute Register Right
*HALT		0000	5-173	Halt
LB	d,*m,x	AC08	5-7	Load Byte
LD	d,*m,x	AC00	5-10	Load Doubleword
LH	d,*m,x	AC00	5-8	Load Halfword
LW	d,*m,x	AC00	5-9	Load Word
*LCS	d	0003	5-22	Load Control Switches
LEA	d,*m,x	D000	5-20	Load Effective Address
LF	d,*m,x	CC00	5-23	Load File
LI	d,v	C800	5-19	Load Immediate
LMB	d,*m,x	B008	5-11	Load Masked Byte
LMD	d,*m,x	B000	5-14	Load Masked Doubleword
LMH	d,*m,x	B000	5-12	Load Masked Halfword
LMW	d,*m,x	B000	5-13	Load Masked Word
LNB	d,*m,x	B408	5-15	Load Negative Byte
LND	d,*m,x	B400	5-18	Load Negative Doubleword
LNH	d,*m,x	B400	5-16	Load Negative Halfword
LNW	d,*m,x	B400	5-17	Load Negative Word
MPFD	d,*m,x	E408	5-166	Multiply Floating-Point Doubleword
MPFW	d,*m,x	E408	5-165	Multiply Floating Point Word
MPI	d,v	C803	5-150	Multiply Immediate
MPMB	d,*m,x	C008	5-146	Multiply by Memory Byte

*Indicate Halfword Instruction

Mnemonic	Operand Format	Op Code	Page	Instruction Format
MPMH	d,*m,x	C000	5-147	Multiply by Memory Halfword
MPMW	d,*m,x	C000	5-148	Multiply by Memory Word
*MPR	s,d	4000	5-149	Multiply Register by Register
*NOP		0002	5-175	No Operation
*NOR	d,s	6000	5-101	Normalize
*NORD	d,s	6400	5-102	Normalize Double
ORMB	d,*m,x	8808	5-75	OR Memory Byte
ORMD	d,*m,x	8800	5-78	OR Memory Doubleword
ORMH	d,*m,x	8800	5-76	OR Memory Halfword
ORMW	d,*m,x	8800	5-77	OR Memory Word
*ORR	s,d	0800	5-79	OR Register and Register
*ORRM	s,d	0808	5-80	OR Register and Register Masked
RI	v	FC02	5-180	Request Interrupt
*RND	d	0005	5-157	Round Register
SBM	c,*m,x	9808	5-116	Set Bit in Memory
*SBR	d,b	1800	5-117	Set Bit in Register
*SCZ	d,s	6800	5-103	Shift and Count Zeros
*SLA	d,v	6C40	5-104	Shift Left Arithmetic
*SLAD	d,v	7840	5-107	Shift Left Arithmetic Double
*SLC	d,v	7440	5-106	Shift Left Circular
*SLL	d,v	7040	5-105	Shift Left Logical
*SLLD	d,v	7C40	5-108	Shift Left Logical Double
*SRA	d,v	6C00	5-109	Shift Right Arithmetic
*SRAD	d,v	7800	5-112	Shift Right Arithmetic Double
*SRC	d,v	7400	5-111	Shift Right Circular
*SRL	d,v	7000	5-110	Shift Right Logical
*SRLD	d,v	7C00	5-113	Shift Right Logical Double
STB	s,*m,x	D408	5-25	Store Byte
STD	s,*m,x	D400	5-28	Store Doubleword
STH	s,*m,x	D400	5-26	Store Halfword
STW	s,*m,x	D400	5-27	Store Word
STF	s,*m,x	DC00	5-33	Store File
STMB	s,*m,x	D808	5-29	Store Masked Byte
STMD	s,*m,x	D800	5-32	Store Masked Doubleword
STMH	s,*m,x	D800	5-30	Store Masked Halfword
STMW	s,*m,x	D800	5-31	Store Masked Word
SUFD	d,*m,x	E000	5-164	Subtract Floating Point Doubleword
SUFW	d,*m,x	E000	5-163	Subtract Floating Point Word
SUI	d,v	C802	5-145	Subtract Immediate
SUMB	d,*m,x	BC08	5-139	Subtract Memory Byte
SUMD	d,*m,x	BC00	5-142	Subtract Memory Doubleword
SUMH	d,*m,x	BC00	5-140	Subtract Memory Halfword
SUMW	d,*m,x	BC00	5-141	Subtract Memory Word
*SUR	s,d	3C00	5-143	Subtract Register from Register

*Indicates Halfword Instruction

Mnemonic	Operand Format	Op Code	Page	Instruction Function
*SURM	s,d	3C08	5-144	Subtract Register from Register Masked
TBM	c,*m,x	A408	5-122	Test Bit in Memory
*TBR	d,b	2400	5-123	Test Bit in Register
TD	n,f	FC05	5-185	Test Device
TPR	d,p	FB80	5-91	Transfer Protect Register to Register
*TRC	s,d	2C03	5-94	Transfer Register Complement
*TRCM	s,d	2C0B	5-95	Transfer Register Complement Masked
*TRN	s,d	2C04	5-92	Transfer Register Negative
*TRNM	s,d	2C0C	5-93	Transfer Register Negative Masked
TRP	s,p	FB00	5-90	Transfer Register to Protect Register
*TRR	s,d	2C00	5-88	Transfer Register to Register
*TRRM	s,d	2C08	5-89	Transfer Register to Register Masked
*TRSW	s	2800	5-98	Transfer Register to PSWR
*WAIT		0001	5-174	Wait
*XCR	s,d	2C05	5-96	Exchange Registers
*XCRM	s,d	2C0D	5-97	Exchange Registers Masked
ZBM	c,*m,x	9C08	5-118	Zero Bit in Memory
*ZBR	d,b	1C00	5-119	Zero Bit in Register
ZMB	*m,x	F808	5-35	Zero Memory Byte
ZMD	*m,x	F800	5-38	Zero Memory Doubleword
ZMH	*m,x	F800	5-36	Zero Memory Halfword
ZMW	*m,x	F800	5-37	Zero Memory Word
*ZR	d	0C00	5-39	Zero Register

*Indicates Halfword Instruction

APPENDIX C
INSTRUCTION SET
(OPERATION CODE ORDER)

SEL 32 instructions are listed numerically in operation code order (0000 through FC06). Each entry includes the following information:

- Operation Code
- Instruction Mnemonic
- Instruction Function

Halfword instructions are denoted by an asterisk (*) preceding the instruction mnemonic.

Op Code	Mnemonic	Instruction Function
0000	*HALT	Halt
0001	*WAIT	Wait
0002	*NOP	No Operation
0003	*LCS	Load Control Switches
0004	*ES	Extend Sign
0005	*RND	Round Register
0400	*ANR	AND Register and Register
0800	*ORR	OR Register and Register
0808	*ORRM	OR Register and Register Masked
0C00	*EOR	Exclusive OR Register and Register
0C00	*ZR	Zero Register
0C08	*EORM	Exclusive OR Register and Register Masked
1000	*CAR	Compare Arithmetic with Register
1400	*CMR	Compare Masked with Register
1800	*SBR	Set Bit in Register
1C00	*ZBR	Zero Bit in Register
2000	*ABR	Add Bit in Register
2400	*TBR	Test Bit in Register
2800	*TRSW	Transfer Register to PSWR
2C00	*TRR	Transfer Register to Register
2C03	*TRC	Transfer Register Complement
2C04	*TRN	Transfer Register Negative
2C05	*XCR	Exchange Registers
2C08	*TRRM	Transfer Register to Register Masked
2C0B	*TRCM	Transfer Regsiter Complement Masked
2C0C	*TRNM	Transfer Register Negative Masked
2C0D	*XCRM	Exchange Registers Masked

Op Code	Mnemonic	Instruction Function
3000	*CALM	Call Monitor
3800	*ADR	Add Register to Register
3808	*ADRM	Add Register to Register Masked
3C00	*SUR	Subtract Register from Register
3C08	*SURM	Subtract Register from Register Masked
4000	*MPR	Multiply Register by Register
4400	*DVR	Divide Register by Register
6000	*NOR	Normalize
6400	*NORD	Normalize Double
6800	*SCZ	Shift and Count Zeros
6C00	*SRA	Shift Right Arithmetic
6C40	*SLA	Shift Left Arithmetic
7000	*SRL	Shift Right Logical
7040	*SLL	Shift Left Logical
7400	*SRC	Shift Right Circular
7440	*SLC	Shift Left Circular
7800	*SRAD	Shift Right Arithmetic Double
7840	*SLAD	Shift Left Arithmetic Double
7C00	*SRLD	Shift Right Logical Double
7C40	*SLLD	Shift Left Logical Double
8400	ANMD	AND Memory Doubleword
8400	ANMH	AND Memory Halfword
8400	ANMW	AND Memory Word
8408	ANMB	AND Memory Byte
8800	ORMD	OR Memory Doubleword
8800	ORMH	OR Memory Halfword
8800	ORMW	OR Memory Word
8808	ORMB	OR Memory Byte
8C00	EOMD	Exclusive OR Memory Doubleword
8C00	EOMH	Exclusive OR Memory Halfword
8C00	EOMW	Exclusive OR Memory Word
8C08	EOMB	Exclusive OR Memory Byte
9000	CAMD	Compare Arithmetic with Memory Doubleword
9000	CAMH	Compare Arithmetic with Memory Halfword
9000	CAMW	Compare Arithmetic with Memory Word
9008	CAMB	Compare Arithmetic with Memory Byte
9400	CMMD	Compare Masked with Memory Doubleword
9400	CMMH	Compare Masked with Memory Halfword
9400	CMMW	Compare Masked with Memory Word
9408	CMMB	Compare Masked with Memory Byte
9808	SBM	Set Bit in Memory
9C08	ZBM	Zero Bit in Memory
A008	ABM	Add Bit in Memory
A408	TBM	Test Bit in Memory
A800	EXM	Execute Memory
AC00	LD	Load Doubleword
AC00	LH	Load Halfword
AC00	LW	Load Word
AC08	LB	Load Byte

Op Code	Mnemonic	Instruction Function
B000	LMD	Load Masked Doubleword
B000	LMH	Load Masked Halfword
B000	LMW	Load Masked Word
B008	LMB	Load Masked Byte
B400	LND	Load Negative Doubleword
B400	LNH	Load Negative Halfword
B400	LNW	Load Negative Word
B408	LNB	Load Negative Byte
B800	ADMD	Add Memory Doubleword
B800	ADMH	Add Memory Halfword
B800	ADMW	Add Memory Word
B808	ADMB	Add Memory Byte
BC00	SUMD	Subtract Memory Doubleword
BC00	SUMH	Subtract Memory Halfword
BC00	SUMW	Subtract Memory Word
BC08	SUMB	Subtract Memory Byte
C000	MPMH	Multiply By Memory Halfword
C000	MPMW	Multiply By Memory Word
C008	MPMB	Multiply By Memory Byte
C400	DVMH	Divide By Memory Halfword
C400	DVMW	Divide By Memory Word
C408	DVMB	Divide By Memory Byte
C800	LI	Load Immediate
C801	ADI	Add Immediate
C802	SUI	Subtract Immediate
C803	MPI	Multiply Immediate
C804	DVI	Divide Immediate
C805	CI	Compare Immediate
C807	EXR	Execute Register
C807	EXRR	Execute Register Right
CC00	LF	Load File
D000	LEA	Load Effective Address
D400	STD	Store Doubleword
D400	STH	Store Halfword
D400	STW	Store Word
D408	STB	Store Byte
D800	STMD	Store Masked Doubleword
D800	STMH	Store Masked Halfword
D800	STMW	Store Masked Word
D808	STMB	Store Masked Byte
DC00	STF	Store File
E000	SUFD	Subtract Floating-Point Doubleword
E000	SUFW	Subtract Floating-Point Word
E008	ADFD	Add Floating-Point Doubleword
E008	ADFW	Add Floating-Point Word
E400	DVFD	Divide Floating-Point Doubleword
E400	DVFW	Divide Floating-Point Word
E408	MPFD	Multiply Floating-Point Doubleword

Op Code	Mnemonic	Instruction Function
E408	MPFW	Multiply Floating-Point Word
E800	ARMD	Add Register to Memory Doubleword
E800	ARMH	Add Register to Memory Halfword
E800	ARMW	Add Register to Memory Word
E808	ARMB	Add Register to Memory Byte
EC00	BCT	Branch Condition True
EC00	BU	Branch Unconditionally
F000	BCF	Branch Condition False
F000	BFT	Branch Function True
F400	BIB	Branch After Incrementing Byte
F420	BIH	Branch After Incrementing Halfword
F440	BIW	Branch After Incrementing Word
F460	BID	Branch After Incrementing Doubleword
F800	ZMD	Zero Memory Doubleword
F800	ZMH	Zero Memory Halfword
F800	ZMW	Zero Memory Word
F808	ZMB	Zero Memory Byte
F880	BL	Branch and Link
F900	BRI	Branch and Reset Interrupt
FC00	EI	Enable Interrupt
FB00	TRP	Transfer Register to Protect Register
FB80	TPR	Transfer Protect Register to Register
FC01	DI	Disable Interrupt
FC02	RI	Request Interrupt
FC03	AI	Activate Interrupt
FC04	DAI	Deactivate Interrupt
FC05	TD	Test Device
FC06	CD	Command Device

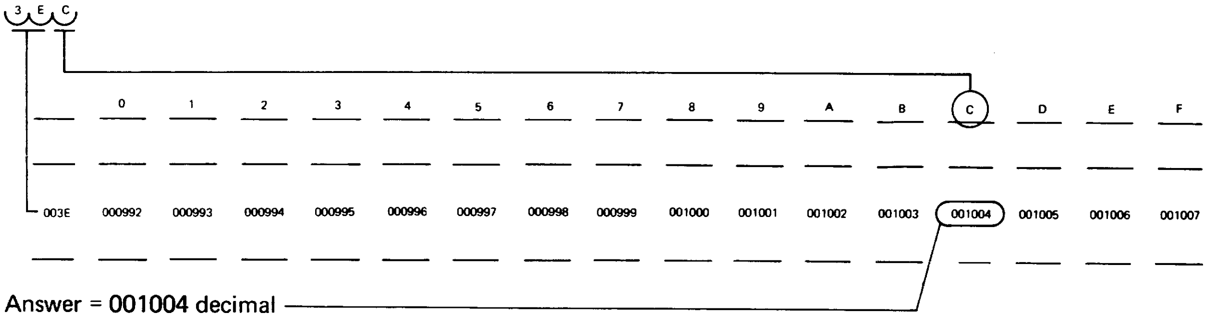
APPENDIX D
HEXADECIMAL-DECIMAL CONVERSION TABLE

The following table contains the necessary information for direct conversion of decimal and hexadecimal numbers in these ranges:

Hexadecimal	Decimal
00000 to 01FFF	000000 to 008191

To convert a hexadecimal number to a decimal value, locate all but the last digit of the hexadecimal value in the left-most column of the table, then follow that line of figures to the right to the column under the last digit of the hexadecimal value. At this intersection is the decimal value of the hexadecimal number.

Example: Convert hexadecimal 3EC to decimal.



Answer = 001004 decimal

For decimal to hexadecimal conversion as in the example, first find the decimal value (1004) in the table, then construct the hexadecimal value from the hexadecimal characters above the column and in the left-most column.

For numbers outside the range of the table, add the following values to the table figures:

Hexadecimal	Decimal
3000	12288
4000	16384
5000	20480
6000	24576
7000	28672
8000	32768
9000	36864
A000	40960
B000	45056
C000	49152
D000	52248
E000	57344
F000	61440

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	000000	000001	000002	000003	000004	000005	000006	000007	000008	000009	000010	000011	000012	000013	000014	000015
0001	000016	000017	000018	000019	000020	000021	000022	000023	000024	000025	000026	000027	000028	000029	000030	000031
0002	000032	000033	000034	000035	000036	000037	000038	000039	000040	000041	000042	000043	000044	000045	000046	000047
0003	000048	000049	000050	000051	000052	000053	000054	000055	000056	000057	000058	000059	000060	000061	000062	000063
0004	000064	000065	000066	000067	000068	000069	000070	000071	000072	000073	000074	000075	000076	000077	000078	000079
0005	000080	000081	000082	000083	000084	000085	000086	000087	000088	000089	000090	000091	000092	000093	000094	000095
0006	000096	000097	000098	000099	000100	000101	000102	000103	000104	000105	000106	000107	000108	000109	000110	000111
0007	000112	000113	000114	000115	000116	000117	000118	000119	000120	000121	000122	000123	000124	000125	000126	000127
0008	000128	000129	000130	000131	000132	000133	000134	000135	000136	000137	000138	000139	000140	000141	000142	000143
0009	000144	000145	000146	000147	000148	000149	000150	000151	000152	000153	000154	000155	000156	000157	000158	000159
000A	000160	000161	000162	000163	000164	000265	000166	000167	000168	000169	000170	000171	000172	000173	000174	000175
000B	000176	000177	000178	000179	000180	000181	000182	000183	000184	000185	000186	000187	000188	000189	000190	000191
000C	000192	000193	000194	000195	000196	000197	000198	000199	000200	000201	000202	000203	000204	000205	000206	000207
000D	000208	000209	000210	000211	000212	000213	000214	000215	000216	000217	000218	000219	000220	000221	000222	000223
000E	000224	000225	000226	000227	000228	000229	000230	000231	000232	000233	000234	000235	000236	000237	000238	000239
000F	000240	000241	000242	000243	000244	000245	000246	000247	000248	000249	000250	000251	000252	000253	000254	000255

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0010	000256	000257	000258	000259	000260	000261	000262	000263	000264	000265	000266	000267	000268	000269	000270	000271
0011	000272	000273	000274	000275	000276	000277	000278	000279	000280	000281	000282	000283	000284	000285	000286	000287
0012	000288	000289	000290	000291	000292	000293	000294	000295	000296	000297	000298	000299	000300	000301	000302	000303
0013	000304	000305	000306	000307	000308	000309	000310	000311	000312	000313	000314	000315	000316	000317	000318	000319
0014	000320	000321	000322	000323	000324	000325	000326	000327	000328	000329	000330q	000331	000332	000333	000334	000335
0015	000336	000337	000338	000339	000340	000341	000342	000343	000344	000345	000346	000347	000348	000349	000350	000351
0016	000352	000353	000354	000355	000356	000357	000358	000359	000360	000361	000362	000363	000364	000365	000366	000367
0017	000368	000369	000370	000371	000372	000373	000374	000375	000376	000377	000378	000379	000380	000381	000382	000383
0018	000384	000385	000386	000387	000388	000389	000390	000391	000392	000393	000394	000395	000396	000397	000398	000399
0019	000400	000401	000402	000403	000404	000405	000406	000407	000408	000409	000410	000411	000412	000413	000414	000415
001A	000416	000417	000418	000419	000420	000421	000422	000423	000424	000425	000426	000427	000428	000429	000430	000431
001B	000432	000433	000434	000435	000436	000437	000438	000439	000440	000441	000442	000443	000444	000445	000446	000447
001C	000448	000449	000450	000451	000452	000453	000454	000455	000456	000457	000458	000459	000460	000461	000462	000463
001D	000464	000465	000466	000467	000468	000469	000470	000471	000472	000473	000474	000475	000476	000477	000478	000479
001E	000480	000481	000482	000483	000484	000485	000486	000487	000488	000489	000490	000491	000492	000493	000494	000495
001F	000496	000497	000498	000499	000500	000501	000502	000503	000504	000505	000506	000507	000508	000509	000510	000511

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0020	000512	000513	000514	000515	000516	000517	000518	000519	000520	000521	000522	000523	000524	000525	000526	000527
0021	000528	000529	000530	000531	000532	000533	000534	000535	000536	000537	000538	000539	000540	000541	000542	000543
0022	000544	000545	000546	000547	000548	000549	000550	000551	000552	000553	000554	000555	000556	000557	000558	000559
0023	000560	000561	000562	000563	000564	000565	000566	000567	000568	000569	000570	000571	000572	000573	000574	000575
0024	000576	000577	000578	000579	000580	000581	000582	000583	000584	000585	000586	000587	000588	000589	000590	000591
0025	000592	000593	000594	000595	000596	000597	000598	000599	000600	000601	000602	000603	000604	000605	000606	000607
0026	000608	000609	000610	000611	000612	000613	000614	000615	000616	000617	000618	000619	000620	000621	000622	000623
0027	000624	000625	000626	000627	000628	000629	000630	000631	000632	000633	000634	000635	000636	000637	000638	000639
0028	000640	000641	000642	000643	000644	000645	000646	000647	000648	000649	000650	000651	000652	000653	000654	000655
0029	000656	000657	000658	000659	000660	000661	000662	000663	000664	000665	000666	000667	000668	000669	000670	000671
002A	000672	000673	000674	000675	000676	000677	000678	000679	000680	000681	000682	000683	000684	000685	000686	000687
002B	000688	000689	000690	000691	000692	000693	000694	000695	000696	000697	000698	000699	000700	000701	000702	000703
002C	000704	000705	000706	000707	000708	000709	000710	000711	000712	000713	000714	000715	000716	000717	000718	000719
002D	000720	000721	000722	000723	000724	000725	000726	000727	000728	000729	000730	000731	000732	000733	000734	000735
002E	000736	000737	000738	000739	000740	000741	000742	000743	000744	000745	000746	000747	000748	000749	000750	000751
002F	000752	000753	000754	000755	000756	000757	000758	000759	000760	000761	000762	000763	000764	000765	000766	000767

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0030	000768	000769	000770	000771	000772	000773	000774	000775	000776	000777	000778	000779	000780	000781	000782	000783
0031	000784	000785	000786	000787	000788	000789	000790	000791	000792	000793	000794	000795	000796	000797	000798	000799
0032	000800	000801	000802	000803	000804	000805	000806	000807	000808	000809	000810	000811	000812	000813	000814	000815
0033	000816	000817	000818	000819	000820	000821	000822	000823	000824	000825	000826	000827	000828	000829	000830	000831
0034	000832	000833	000834	000835	000836	000837	000838	000839	000840	000841	000842	000843	000844	000845	000846	000847
0035	000848	000849	000850	000851	000852	000853	000854	000855	000856	000857	000858	000859	000860	000861	000862	000863
0036	000864	000865	000866	000867	000868	000869	000870	000871	000872	000873	000874	000875	000876	000877	000878	000879
0037	000880	000881	000882	000883	000884	000885	000886	000887	000888	000889	000890	000891	000892	000893	000894	000895
0038	000896	000897	000898	000899	000900	000901	000902	000903	000904	000905	000906	000907	000908	000909	000910	000911
0039	000912	000913	000914	000915	000916	000917	000918	000919	000920	000921	000922	000923	000924	000925	000926	000927
003A	000928	000929	000930	000931	000932	000933	000934	000935	000936	000937	000938	000939	000940	000941	000942	000943
003B	000944	000945	000946	000947	000948	000949	000950	000951	000952	000953	000954	000955	000956	000957	000958	000959
003C	000960	000961	000962	000963	000964	000965	000966	000967	000968	000969	000970	000971	000972	000973	000974	000975
003D	000976	000977	000978	000979	000980	000981	000982	000983	000984	000985	000986	000987	000988	000989	000990	000991
003E	000992	000993	000994	000995	000996	000997	000998	000999	0010000	001001	001002	001003	001004	001005	001006	001007
003F	001008	001009	001010	001011	001012	001013	001014	001015	001016	001017	001018	001019	001020	001021	001022	001023

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0040	001024	001025	001026	001027	001028	001029	001030	001031	001032	001033	001034	001035	001036	001037	001038	001039
0041	001040	001041	001042	001043	001044	001045	001046	001047	001048	001049	001050	001051	001052	001053	001054	001055
0042	001056	001057	001058	001059	001060	001061	001062	001063	001064	001065	001066	001067	001068	001069	001070	001071
0043	001072	001073	001074	001075	001076	001077	001078	001079	001080	001081	001082	001083	001084	001085	001086	001087
0044	001088	001089	001090	001091	001092	001093	001094	001095	001096	001097	001098	001099	001100	001101	001102	001103
0045	001104	001105	001106	001107	001108	001109	001110	001111	001112	001113	001114	001115	001116	001117	001118	001119
0046	001120	001121	001122	001123	001124	001125	001126	001127	001128	001129	001130	001131	001132	001133	001134	001135
0047	001136	001137	001138	001139	001140	001141	001142	001143	001144	001145	001146	001147	001148	001149	001150	001151
0048	001152	001153	001154	001155	001156	001157	001158	001159	001160	001161	001162	001163	001164	001165	001166	001167
0049	001168	001169	001170	001171	001172	001173	001174	001175	001176	001177	001178	001179	001180	001181	001182	001183
004A	001184	001185	001186	001187	001188	001189	001190	001191	001192	001193	001194	001195	001196	001197	001198	001199
004B	001200	001201	001202	001203	001204	001205	001206	001207	001208	001209	001210	001211	001212	001213	001214	001215
004C	001216	001217	001218	001219	001220	001221	001222	001223	001224	001225	001226	001227	001228	001229	001230	001231
004D	001232	001233	001234	001235	001236	001237	001238	001239	001240	001241	001242	001243	001244	001245	001246	001247
004E	001248	001249	001250	001251	001252	001253	001254	001255	001256	001257	001258	001259	001260	001261	001262	001263
004F	001264	001265	001266	001267	001268	001269	001270	001271	001272	001273	001274	001275	001276	001277	001278	001279

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0050	001280	001281	001282	001283	001284	001285	001286	001287	001288	001289	001290	001291	001292	001293	001294	001295
0051	001296	001297	001298	001299	001300	001301	001302	001303	001304	001305	001306	001307	001308	001309	001310	001311
0052	001312	001313	001314	001315	001316	001317	001318	001319	001320	001321	001322	001323	001324	001325	001326	001327
0053	001328	001329	001330	001331	001332	001333	001334	001335	001336	001337	001338	001339	001340	001341	001342	001343
0054	001344	001345	001346	001347	001348	001349	001350	001351	001352	001353	001354	001355	001356	001357	001358	001359
0055	001360	001361	001362	001363	001364	001365	001366	001367	001368	001369	001370	001371	001372	001373	001374	001375
0056	001376	001377	001378	001379	001380	001381	001382	001383	001384	001385	001386	001387	001388	001389	001390	001391
0057	001392	001393	001394	001395	001396	001397	001398	001399	001400	001401	001402	001403	001404	001405	001406	001407
0058	001408	001409	001410	001411	001412	001413	001414	001415	001416	001417	001418	001419	001420	001421	001422	001423
0059	001424	001425	001426	001427	001428	001429	001430	001431	001432	001433	001434	001435	001436	001437	001438	001439
005A	001440	001441	001442	001443	001444	001445	001446	001447	001448	001449	001450	001451	001452	001453	001454	001455
005B	001456	001457	001458	001459	001460	001461	001462	001463	001464	001465	001466	001467	001468	001469	001470	001471
005C	001472	001473	001474	001475	001476	001477	001478	001479	001480	001481	001482	001483	001484	001485	001486	001487
005D	001488	001489	001490	001491	001492	001493	001494	001495	001496	001497	001498	001499	001500	001501	001502	001503
005E	001504	001505	001506	001507	001508	001509	001510	001511	001512	001513	001514	001515	001516	001517	001518	001519
005F	001520	001521	001522	001523	001524	001525	001526	001527	001528	001529	001530	001531	001532	001533	001534	001535

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0060	001536	001537	001538	001539	001540	001541	001542	001543	001544	001545	001546	001547	001548	001549	001550	001551
0061	001552	001553	001554	001555	001556	001557	001558	001559	001560	001561	001562	001563	001564	001565	001566	001567
0062	001568	001569	001570	001571	001572	001573	001574	001575	001576	001577	001578	001579	001580	001581	001582	001583
0063	001584	001585	001586	001587	001588	001589	001590	001591	001592	001593	001594	001595	001596	001597	001598	001599
0064	001600	001601	001602	001603	001604	001605	001606	001607	001608	001609	001610	001611	001612	001613	001614	001615
0065	001616	001617	001618	001619	001620	001621	001622	001623	001624	001625	001626	001627	001628	001629	001630	001631
0066	001632	001633	001634	001635	001636	001637	001638	001639	001640	001641	001642	001643	001644	001645	001646	001647
0067	001648	001649	001650	001651	001652	001653	001654	001655	001656	001657	001658	001659	001660	001661	001662	001663
0068	001664	001665	001666	001667	001668	001669	001670	001671	001672	001673	001674	001675	001676	001677	001678	001679
0069	001680	001681	001682	001683	001684	001685	001686	001687	001688	001689	001690	001691	001692	001693	001694	001695
006A	001696	001697	001698	001699	001700	001701	001702	001703	001704	001705	001706	001707	001708	001709	001710	001711
006B	001712	001713	001714	001715	001716	001717	001718	001719	001720	001721	001722	001723	001724	001725	001726	001727
006C	001728	001729	001730	001731	001732	001733	001734	001735	001736	001737	001738	001739	001740	001741	001742	001743
006D	001744	001745	001746	001747	001748	001749	001750	001751	001752	001753	001754	001755	001756	001757	001758	001759
006E	001760	001761	001762	001763	001764	001765	001766	001767	001768	001769	001770	001771	001772	001773	001774	001775
006F	001776	001777	001778	001779	001780	001781	001782	001783	001784	001785	001786	001787	001788	001789	001790	001791

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0070	001792	001793	001794	001795	001796	001797	001798	001799	001800	001801	001802	001803	001804	001805	001806	001807
0071	001808	001809	001810	001811	001812	001813	001814	001815	001816	001817	001818	001819	001820	001821	001822	001823
0072	001824	001825	001826	001827	001828	001829	001830	001831	001832	001833	001834	001835	001836	001837	001838	001839
0073	001840	001841	001842	001843	001844	001845	001846	001847	001848	001849	001850	001851	001852	001853	001854	001855
0074	001856	001857	001858	001859	001860	001861	001862	001863	001864	001865	001866	001867	001868	001869	001870	001871
0075	001872	001873	001874	001875	001876	001877	001878	001879	001880	001881	001882	001883	001884	001885	001886	001887
0076	001888	001889	001890	001891	001892	001893	001894	001895	001896	001897	001898	001899	001900	001901	001902	001903
0077	001904	001905	001906	001907	001908	001909	001910	001911	001912	001913	001914	001915	001916	001917	001918	001919
0078	001920	001921	001922	001923	001924	001925	001926	001927	001928	001929	001930	001931	001932	001933	001934	001935
0079	001936	001937	001938	001939	001940	001941	001942	001943	001944	001945	001946	001947	001948	001949	001950	001951
007A	001952	001953	001954	001955	001956	001957	001958	001959	001960	001961	001962	001963	001964	001965	001966	001967
007B	001968	001969	001970	001971	001972	001973	001974	001975	001976	001977	001978	001979	001980	001981	001982	001983
007C	001984	001985	001986	001987	001988	001989	001990	001991	001992	001993	001994	001995	001996	001997	001998	001999
007D	002000	002001	002002	002003	002004	002005	002006	002007	002008	002009	002010	002011	002012	002013	002014	002015
007E	002016	002017	002018	002019	002020	002021	002022	002023	002024	002025	002026	002027	002028	002029	002030	002031
007F	002032	002033	002034	002035	002036	002037	002038	002039	002040	002041	002042	002043	002044	002045	002046	002047

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0080	002048	002049	002050	002051	002052	002053	002054	002055	002056	002057	002058	002059	002060	002061	002062	002063
0081	002064	002065	002066	002067	002068	002069	002070	002071	002072	002073	002074	002075	002076	002077	002078	002079
0082	002080	002081	002082	002083	002084	002085	002086	002087	002088	002089	002090	002091	002092	002093	002094	002095
0083	002096	002097	002098	002099	002100	002101	002102	002103	002104	002105	002106	002107	002108	002109	002110	002111
0084	002112	002113	002114	002115	002116	002117	002118	002119	002120	002121	002122	002123	002124	002125	002126	002127
0085	002128	002129	002130	002131	002132	002133	002134	002135	002136	002137	002138	002139	002140	002141	002142	002143
0086	002144	002145	002146	002147	002148	002149	002150	002151	002152	002153	002154	002155	002156	002157	002158	002159
0087	002160	002161	002162	002163	002164	002165	002166	002167	002168	002169	002170	002171	002172	002173	002174	002175
0088	002176	002177	002178	002179	002180	002181	002182	002183	002184	002185	002186	002187	002188	002189	002190	002191
0089	002192	002193	002194	002195	002196	002197	002198	002199	002200	002201	002202	002203	002204	002205	002206	002207
008A	002208	002209	002210	002211	002212	002213	002214	002215	002216	002217	002218	002219	002220	002221	002222	002223
008B	002224	002225	002226	002227	002228	002229	002230	002231	002232	002233	002234	002235	002236	002237	002238	002239
008C	002240	002241	002242	002243	002244	002245	002246	002247	002248	002249	002250	002251	002252	002253	002254	002255
008D	002256	002257	002258	002259	002260	002261	002262	002263	002264	002265	002266	002267	002268	002269	002270	002271
008E	002272	002273	002274	002275	002276	002277	002278	002279	002280	002281	002282	002283	002284	002285	002286	002287
008F	002288	002289	002290	002291	002292	002293	002294	002295	002296	002297	002298	002299	002300	002301	002302	002303

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0090	002304	002305	002306	002307	002308	002309	002310	002311	002312	002313	002314	002315	002316	002317	002318	002319
0091	002320	002321	002322	002323	002324	002325	002326	002327	002328	002329	002330	002331	002332	002333	002334	002335
0092	002336	002337	002338	002339	002340	002341	002342	002343	002344	002345	002346	002347	002348	002349	002350	002351
0093	002352	002353	002354	002355	002356	002357	002358	002359	002360	002361	002362	002363	002364	002365	002366	002367
0094	002368	002369	002370	002371	002372	002373	002374	002375	002376	002377	002378	002379	002380	002381	002382	002383
0095	002384	002385	002386	002387	002388	002389	002390	002391	002392	002393	002394	002395	002396	002397	002398	002399
0096	002400	002401	002402	002403	002404	002405	002406	002407	002408	002409	002410	002411	002412	002413	002414	002415
0097	002416	002417	002418	002419	002420	002421	002422	002423	002424	002425	002426	002427	002428	002429	002430	002431
0098	002432	002433	002434	002435	002436	002437	002438	002439	002440	002441	002442	002443	002444	002445	002446	002447
0099	002448	002449	002450	002451	002452	002453	002454	002455	002456	002457	002458	002459	002460	002461	002462	002463
009A	002464	002465	002466	002467	002468	002469	002470	002471	002472	002473	002474	002475	002476	002477	002478	002479
009B	002480	002481	002482	002483	002484	002485	002486	002487	002488	002489	002490	002491	002492	002493	002494	002495
009C	002496	002497	002498	002499	002500	002501	002502	002503	002504	002505	002506	002507	002508	002509	002510	002511
009D	002512	002513	002514	002515	002516	002517	002518	002519	002520	002521	002522	002523	002524	002525	002526	002527
009E	002528	002529	002530	002531	002532	002533	002534	002535	002536	002537	002538	002539	002540	002541	002542	002543
009F	002544	002545	002546	002547	002548	002549	002550	002551	002552	002553	002554	002555	002556	002557	002558	002559

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00A0	002560	002561	002562	002563	002564	002565	002566	002567	002568	002569	002570	002571	002572	002573	002574	002575
00A1	002576	002577	002578	002579	002580	002581	002582	002583	002584	002585	002586	002587	002588	002589	002590	002591
00A2	002592	002593	002594	002595	002596	002597	002598	002599	002600	002601	002602	002603	002604	002605	002606	002607
00A3	002608	002609	002610	002611	002612	002613	002614	002615	002616	002617	002618	002619	002620	002621	002622	002623
00A4	002624	002625	002626	002627	002628	002629	002630	002631	002632	002633	002634	002635	002636	002637	002638	002639
00A5	002640	002641	002642	002643	002644	002645	002646	002647	002648	002649	002650	002651	002652	002653	002654	002655
00A6	002656	002657	002658	002659	002660	002661	002662	002663	002664	002665	002666	002667	002668	002669	002670	002671
00A7	002672	002673	002674	002675	002676	002677	002678	002679	002680	002681	002682	002683	002684	002685	002686	002687
00A8	002688	002689	002690	002691	002692	002693	002694	002695	002696	002697	002698	002699	002700	002701	002702	002703
00A9	002704	002705	002706	002707	002708	002709	002710	002711	002712	002713	002714	002715	002716	002717	002718	002719
00AA	002720	002721	002722	002723	002724	002725	002726	002727	002728	002729	002730	002731	002732	002733	002734	002735
00AB	002736	002737	002738	002739	002740	002741	002742	002743	002744	002745	002746	002747	002748	002749	002750	002751
00AC	002752	002753	002754	002755	002756	002757	002758	002759	002760	002761	002762	002763	002764	002765	002766	002767
00AD	002768	002769	002770	002771	002772	002773	002774	002775	002776	002777	002778	002779	002780	002781	002782	002783
00AE	002784	002785	002786	002787	002788	002789	002790	002791	002792	002793	002794	002795	002796	002797	002798	002799
00AF	002800	002801	002802	002803	002804	002805	002806	002807	002808	002809	002810	002811	002812	002813	002814	002815

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00B0	002816	002817	002818	002819	002820	002821	002822	002823	002824	002825	002826	002827	002828	002829	002830	002831
00B1	002832	002833	002834	002835	002836	002837	002838	002839	002840	002841	002842	002843	002844	002845	002846	002847
00B2	002848	002849	002850	002851	002852	002853	002854	002855	002856	002857	002858	002859	002860	002861	002862	002863
00B3	002864	002865	002866	002867	002868	002869	002870	002871	002872	002873	002874	002875	002876	002877	002878	002879
00B4	002880	002881	002882	002883	002884	002885	002886	002887	002888	002889	002890	002891	002892	002893	002894	002895
00B5	002896	002897	002898	002899	002900	002901	002902	002903	002904	002905	002906	002907	002908	002909	002910	002911
00B6	002912	002913	002914	002915	002916	002917	002918	002919	002920	002921	002922	002923	002924	002925	002926	002927
00B7	002928	002929	002930	002931	002932	002933	002934	002935	002936	002937	002938	002939	002940	002941	002942	002943
00B8	002944	002945	002946	002947	002948	002949	002950	002951	002952	002953	002954	002955	002956	002957	002958	002959
00B9	002960	002961	002962	002963	002964	002965	002966	002967	002968	002969	002970	002971	002972	002973	002974	002975
00BA	002976	002977	002978	002979	002980	002981	002982	002983	002984	002985	002986	002987	002988	002989	002990	002991
00BB	002992	002993	002994	002995	002996	002997	002998	002999	003000	003001	003002	003003	003004	003005	003006	003007
00BC	003008	003009	003010	003011	003012	003013	003014	003015	003016	003017	003018	003019	003020	003021	003022	003023
00BD	003024	003025	003026	003027	003028	003029	003030	003031	003032	003033	003034	003035	003036	003037	003038	003039
00BE	003040	003041	003042	003043	003044	003045	003046	003047	003048	003049	003050	003051	003052	003053	003054	003055
00BF	003056	003057	003058	003059	003060	003061	003062	003063	003064	003065	003066	003067	003068	003069	003070	003071

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00C0	003072	003073	003074	003075	003076	003077	003078	003079	003080	003081	003082	003083	003084	003085	003086	003087
00C1	003088	003089	003090	003091	003092	003093	003094	003095	003096	003097	003098	003099	003100	003101	003102	003103
00C2	003104	003105	003106	003107	003108	003109	003110	003111	003112	003113	003114	003115	003116	003117	003118	003119
00C3	003120	003121	003122	003123	003124	003125	003126	003127	003128	003129	003130	003131	003132	003133	003134	003135
00C4	003136	003137	003138	003139	003140	003141	003142	003143	003144	003145	003146	003147	003148	003149	003150	003151
00C5	003152	003153	003154	003155	003156	003157	003158	003159	003160	003161	003162	003163	003164	003165	003166	003167
00C6	003268	003169	003170	003171	003172	003173	003174	003175	003176	003177	003178	003179	003180	003181	003182	003183
00C7	003184	003185	003186	003187	003188	003189	003190	003191	003192	003193	003194	003195	003196	003197	003198	003199
00C8	003200	003201	003202	003203	003204	003205	003206	003207	003208	003209	003210	003211	003212	003213	003214	003215
00C9	003216	003217	003218	003219	003220	003221	003222	003223	003224	003225	003226	003227	003228	003229	003230	003231
00CA	003232	003233	003234	003235	003236	003237	003238	003239	003240	003241	003242	003243	003244	003245	003246	003247
00CB	003248	003249	003250	003251	003252	003253	003254	003255	003256	003257	003258	003259	003260	003261	003262	003263
00CC	003264	003265	003266	003267	003268	003269	003270	003271	003272	003273	003274	003275	003276	003277	003278	003279
00CD	003280	003281	003282	003283	003284	003285	003286	003287	003288	003289	003290	003291	003292	003293	003294	003295
00CE	003296	003297	003298	003299	003300	003301	003302	003303	003304	003305	003306	003307	003308	003309	003310	003311
00CF	003312	003313	003314	003315	003316	003317	003318	003319	003320	003321	003322	003323	003324	003325	003326	003327

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00D0	003328	003329	003330	003331	003332	003333	003334	003335	003336	003337	003338	003339	003340	003341	003342	003343
00D1	003344	003345	003346	003347	003348	003349	003350	003351	003352	003353	003354	003355	003356	003357	003358	003359
00D2	003360	003361	003362	003363	003364	003365	003366	003367	003368	003369	003370	003371	003372	003373	003374	003375
00D3	003376	003377	003378	003379	003380	003381	003382	003383	003384	003385	003386	003387	003388	003389	003390	003391
00D4	003392	003393	003394	003395	003396	003397	003398	003399	003400	003401	003402	003403	003404	003405	003406	003407
00D5	003408	003409	003410	003411	003412	003413	003414	003415	003416	003417	003418	003419	003420	003421	003422	003423
00D6	003424	003425	003426	003427	003428	003429	003430	003431	003432	003433	003434	003435	003436	003437	003438	003439
00D7	003440	003441	003442	003443	003444	003445	003446	003447	003448	003449	003450	003451	003452	003453	003454	003455
00D8	003456	003457	003458	003459	003460	003461	003462	003463	003464	003465	003466	003467	003468	003469	003470	003471
00D9	003472	003473	003474	003475	003476	003477	003478	003479	003480	003481	003482	003483	003484	003485	003486	003487
00DA	003488	003489	003490	003491	003492	003493	003494	003495	003496	003497	003498	003499	003500	003501	003502	003503
00DB	003504	003505	003506	003507	003508	003509	003510	003511	003512	003513	003514	003515	003516	003517	003518	003519
00DC	003520	003521	003522	003523	003524	003525	003526	003527	003528	003529	003530	003531	003532	003533	003534	003535
00DD	003536	003537	003538	003539	003540	003541	003542	003543	003544	003545	003546	003547	003548	003549	003550	003551
00DE	003552	003553	003554	003555	003556	003557	003558	003559	003560	003561	003562	003563	003564	003565	003566	003567
00DF	003568	003569	003570	003571	003572	003573	003574	003575	003576	003577	003578	003579	003580	003581	003582	003583

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00E0	003584	003585	003586	003587	003588	003589	003590	003591	003592	003593	003594	003595	003596	003597	003598	003599
00E1	003600	003601	003602	003603	003604	003605	003606	003607	003608	003609	003610	003611	003612	003613	003614	003615
00E2	003616	003617	003618	003619	003620	003621	003622	003623	003624	003625	003626	003627	003628	003629	003630	003631
00E3	003632	003633	003634	003635	003636	003637	003638	003639	003640	003641	003642	003643	003644	003645	003646	003647
00E4	003648	003649	003650	003651	003652	003653	003654	003655	003656	003657	003658	003659	003660	003661	003662	003663
00E5	003664	003665	003666	003667	003668	003669	003670	003671	003672	003673	003674	003675	003676	003677	003678	003679
00E6	003680	003681	003682	003683	003684	003685	003686	003687	003688	003689	003690	003691	003692	003693	003694	003695
00E7	003696	003697	003698	003699	003700	003701	003702	003703	003704	003705	003706	003707	003708	003709	003710	003711
00E8	003712	003713	003714	003715	003716	003717	003718	003719	003720	003721	003722	003723	003724	003725	003726	003727
00E9	003728	003729	003730	003731	003732	003733	003734	003735	003736	003737	003738	003739	003740	003741	003742	003743
00EA	003744	003745	003746	003747	003748	003749	003750	003751	003752	003753	003754	003755	003756	003757	003758	003759
00EB	003760	003761	003762	003763	003764	003765	003766	003767	003768	003769	003770	003771	003772	003773	003774	003775
00EC	003776	003777	003778	003779	003780	003781	003782	003783	003784	003785	003786	003787	003788	003789	003790	003791
00ED	003792	003793	003794	003795	003796	003797	003798	003799	003800	003801	003802	003803	003804	003805	003806	003807
00EE	003808	003809	003810	003811	003812	003813	003814	003815	003816	003817	003818	003819	003820	003821	003822	003823
00EF	003824	003825	003826	003827	003828	003829	003830	003831	003832	003833	003834	003835	003836	003837	003838	003839

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00F0	003840	003841	003842	003843	003844	003845	003846	003847	003848	003849	003850	003851	003852	003853	003854	003855
00F1	003856	003857	003858	003859	003860	003861	003862	003863	003864	003865	003866	003867	003868	003869	003870	003871
00F2	003872	003873	003874	003875	003876	003877	003878	003879	003880	003881	003882	003883	003884	003885	003886	003887
00F3	003888	003889	003890	003891	003892	003893	003894	003895	003896	003897	003898	003899	003900	003901	003902	003903
00F4	003904	003905	003906	003907	003908	003909	003910	003911	003912	003913	003914	003915	003916	003917	003918	003919
00F5	003920	003921	003922	003923	003924	003925	003926	003927	003928	003929	003930	003931	003932	003933	003934	003935
00F6	003936	003937	003938	003939	003940	003941	003942	003943	003944	003945	003946	003947	003948	003949	003950	003951
00F7	003952	003953	003954	003955	003956	003957	003958	003959	003960	003961	003962	003963	003964	003965	003966	003967
00F8	003968	003969	003970	003971	003972	003973	003974	003975	003976	003977	003978	003979	003980	003981	003982	003983
00F9	003984	003985	003986	003987	003988	003989	003990	003991	003992	003993	003994	003995	003996	003997	003998	003999
00FA	004000	004001	004002	004003	004004	004005	004006	004007	004008	004009	004010	004011	004012	004013	004014	004015
00FB	004016	004017	004018	004019	004020	004021	004022	004023	004024	004025	004026	004027	004028	004029	004030	004031
00FC	004032	004033	004034	004035	004036	004037	004038	004039	004040	004041	004042	004043	004044	004045	004046	004047
00FD	004048	004049	004050	004051	004052	004053	004054	004055	004056	004057	004058	004059	004060	004061	004062	004063
00FE	004064	004065	004066	004067	004068	004069	004070	004071	004072	004073	004074	004075	004076	004077	004078	004079
00FF	004080	004081	004082	004083	004084	004085	004086	004087	004088	004089	004090	004091	004092	004093	004094	004095

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0100	004096	004097	004098	004099	004100	004101	004102	004103	004104	004105	004106	004107	004108	004109	004110	004111
0101	004112	004113	004114	004115	004116	004117	004118	004119	004120	004121	004122	004123	004124	004125	004126	004127
0102	004128	004129	004130	004131	004132	004133	004134	004135	004136	004137	004138	004139	004140	004141	004142	004143
0103	004144	004145	004146	004147	004148	004149	004150	004151	004152	004153	004154	004155	004156	004157	004158	004159
0104	004160	004161	004162	004163	004164	004165	004166	004167	004168	004169	004170	004171	004172	004173	004174	004175
0105	004176	004177	004178	004179	004180	004181	004182	004183	004184	004185	004186	004187	004188	004189	004190	004191
0106	004192	004193	004194	004195	004196	004197	004198	004199	004200	004201	004202	004203	004204	004205	004206	004207
0107	004208	004209	004210	004211	004212	004213	004214	004215	004216	004217	004218	004219	004220	004221	004222	004223
0108	004224	004225	004226	004227	004228	004229	004230	004231	004232	004233	004234	004235	004236q	004237	004238	004239
0109	004240	004241	004242	004243	004244	004245	004246	004247	004248	004249	004250	004251	004252	004253	004254	004255
010A	004256	004257	004258	004259	004260	004261	004262	004263	004264	004265	004266	004267	004268	004269	004270	004271
010B	004272	004273	004274	004275	004276	004277	004278	004279	004280	004281	004282	004283	004284	004285	004286	004287
010C	004288	004289	004290	004291	004292	004293	004294	004295	004296	004297	004298	004299	004300	004301	004302	004303
010D	004304	004305	004306	004307	004308	004309	004310	004311	004312	004313	004314	004315	004316	004317	004318	004319
010E	004320	004321	004322	004323	004324	004325	004326	004327	004328	004329	004330	004331	004332	004333	004334	004335
010F	004336	004337	004338	004339	004340	004341	004342	004343	004344	004345	004346	004347	004348	004349	004350	004351

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0110	004352	004353	004354	004355	004356	004357	004358	004359	004360	004361	004362	004363	004364	004365	004366	004367
0111	004368	004369	004370	004371	004372	004373	004374	004375	004376	004377	004378	004379	004380	004381	004382	004383
0112	004384	004385	004386	004387	004388	004389	004390	004391	004392	004393	004394	004395	004396	004397	004398	004399
0113	004400	004401	004402	004403	004404	004405	004406	004407	004408	004409	004410	004411	004412	004413	004414	004415
0114	004416	004417	004418	004419	004420	004421	004422	004423	004424	004425	004426	004427	004428	004429	004430	004431
0115	004432	004433	004434	004435	004436	004437	004438	004439	004440	004441	004442	004443	004444	004445	004446	004447
0116	004448	004449	004450	004451	004452	004453	004454	004455	004456	004457	004458	004459	004460	004461	004462	004463
0117	004464	004465	004466	004467	004468	004469	004470	004471	004472	004473	004474	004475	004476	004477	004478	004479
0118	004480	004481	004482	004483	004484	004485	004486	004487	004488	004489	004490	004491	004492	004493	004494	004495
0119	004496	004497	004498	004499	004500	004501	004502	004503	004504	004505	004506	004507	004508	004509	004510	004511
011A	004512	004513	004514	004515	004516	004517	004518	004519	004520	004521	004522	004523	004524	004525	004526	004527
011B	004528	004529	004530	004531	004532	004533	004534	004535	004536	004537	004538	004539	004540	004541	004542	004543
011C	004544	004545	004546	004547	004548	004549	004550	004551	004552	004553	004554	004555	004556	004557	004558	004559
011D	004560	004561	004562	004563	004564	004565	004566	004567	004568	004569	004570	004571	004572	004573	004574	004575
011E	004576	004577	004578	004579	004580	004581	004582	004583	004584	004585	004586	004587	004588	004589	004590	004591
011F	004592	004593	004594	004595	004596	004597	004598	004599	004600	004601	004602	004603	004604	004605	004606	004607

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0120	004608	004609	004610	004611	004612	004613	004614	004615	004616	004617	004618	004619	004620	004621	004622	004623
0121	004624	004625	004626	004627	004628	004629	004630	004631	004632	004633	004634	004635	004636	004637	004638	004639
0122	004640	004641	004642	004643	004644	004645	004646	004647	004648	004649	004650	004651	004652	004653	004654	004655
0123	004656	004657	004658	004659	004660	004661	004662	004663	004664	004665	004666	004667	004668	004669	004670	004671
0124	004672	004673	004674	004675	004676	004677	004678	004679	004680	004681	004682	004683	004684	004685	004686	004687
0125	004688	004689	004690	004691	004692	004693	004694	004695	004696	004697	004698	004699	004700	004701	004702	004703
0126	004704	004705	004706	004707	004708	004709	004710	004711	004712	004713	004714	004715	004716	004717	004718	004719
0127	004720	004721	004722	004723	004724	004725	004726	004727	004728	004729	004730	004731	004732	004733	004734	004735
0128	004736	004737	004738	004739	004740	004741	004742	004743	004744	004745	004746	004747	004748	004749	004750	004751
0129	004752	004753	004754	004755	004756	004757	004758	004759	004760	004761	004762	004763	004764	004765	004766	004767
012A	004768	004769	004770	004771	004772	004773	004774	004775	004776	004777	004778	004779	004780	004781	004782	004783
012B	004784	004785	004786	004787	004788	004789	004790	004791	004792	004793	004794	004795	004796	004797	004798	004799
012C	004800	004801	004802	004803	004804	004805	004806	004807	004808	004809	004810	004811	004812	004813	004814	004815
012D	004816	004817	004818	004819	004820	004821	004822	004823	004824	004825	004826	004827	004828	004829	004830	004831
012E	004832	004833	004834	004835	004836	004837	004838	004839	004840	004841	004842	004843	004844	004845	004846	004847
012F	004848	004849	004850	004851	004852	004853	004854	004855	004856	004857	004858	004859	004860	004861	004862	004863

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0130	004864	004865	004866	004867	004868	004869	004870	004871	004872	004873	004874	004875	004876	004877	004878	004879
0131	004880	004881	004882	004883	004884	004885	004886	004887	004888	004889	004890	004891	004892	004893	004894	004895
0132	004896	004897	004898	004899	004900	004901	004902	004903	004904	004905	004906	004907	004908	004909	004910	004911
0133	004912	004913	004914	004915	004916	004917	004918	004919	004920	004921	004922	004923	004924	004925	004926	004927
0134	004928	004929	004930	004931	004932	004933	004934	004935	004936	004937	004938	004939	004940	004941	004942	004943
0135	004944	004945	004946	004947	004948	004949	004950	004951	004952	004953	004954	004955	004956	004957	004958	004959
0136	004960	004961	004962	004963	004964	004965	004966	004967	004968	004969	004970	004971	004972	004973	004974	004975
0137	004976	004977	004978	004979	004980	004981	004982	004983	004984	004985	004986	004987	004988	004989	004990	004991
0138	004992	004993	004994	004995	004996	004997	004998	004999	005000	005001	005002	005003	005004	005005	005006	005007
0139	005008	005009	005010	005011	005012	005013	005014	005015	005016	005017	005018	005019	005020	005021	005022	005023
013A	005024	005025	005026	005027	005028	005029	005030	005031	005032	005033	005034	005035	005036	005037	005038	005039
013B	005040	005041	005042	005043	005044	005045	005046	005047	005048	005049	005050	005051	005052	005053	005054	005055
013C	005056	005057	005058	005059	005060	005061	005062	005063	005064	005065	005066	005067	005068	005069	005070	005071
013D	005072	005073	005074	005075	005076	005077	005078	005079	005080	005081	005082	005083	005084	005085	005086	005087
013E	005088	005089	005090	005091	005092	005093	005094	005095	005096	005097	005098	005099	005100	005101	005102	005103
013F	005104	005105	005106	005107	005108	005109	005110	005111	005112	005113	005114	005115	005116	005117	005118	005119

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0140	005120	005121	005122	005123	005124	005125	005126	005127	005128	005129	005130	005131	005132	005133	005134	005135
0141	005136	005137	005138	005139	005140	005141	005142	005143	005144	005145	005146	005147	005148	005149	005150	005151
0142	005152	005153	005154	005155	005156	005157	005158	005159	005160	005161	005162	005163	005164	005165	005166	005167
0143	005168	005169	005170	005171	005172	005173	005174	005175	005176	005177	005178	005179	005180	005181	005182	005183
0144	005184	005185	005186	005187	005188	005189	005190	005191	005192	005193	005194	005195	005196	005197	005198	005199
0145	005200	005201	005202	005203	005204	005205	005206	005207	005208	005209	005210	005211	005212	005213	005214	005215
0146	005216	005217	005218	005219	005220	005221	005222	005223	005224	005225	005226	005227	005228	005229	005230	005231
0147	005232	005233	005234	005235	005236	005237	005238	005239	005240	005241	005242	005243	005244	005245	005246	005247
0148	005248	005249	005250	005251	005252	005253	005254	005255	005256	005257	005258	005259	005260	005261	005262	005263
0149	005264	005265	005266	005267	005268	005269	005270	005271	005272	005273	005274	005275	005276	005277	005278	005279
014A	005280	005281	005282	005283	005284	005285	005286	005287	005288	005289	005290	005291	005292	005293	005294	005295
014B	005296	005297	005298	005299	005300	005301	005302	005303	005304	005305	005306	005307	005308	005309	005310	005311
014C	005312	005313	005314	005315	005316	005317	005318	005319	005320	005321	005322	005323	005324	005325	005326	005327
014D	005328	005329	005330	005331	005332	005333	005334	005335	005336	005337	005338	005339	005340	005341	005342	005343
014E	005344	005345	005346	005347	005348	005349	005350	005351	005352	005353	005354	005355	005356	005357	005358	005359
014F	006360	005361	005362	005363	005364	005365	005366	005367	005368	005369	005370	005371	005372	005373	005374	005375

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0150	005376	005377	005378	005379	005380	005381	005382	005383	005384	005385	005386	005387	005388	005389	005390	005391
0151	005392	005393	005394	005395	005396	005397	005398	005399	005400	005401	005402	005403	005404	005405	006406	005407
0152	005408	005409	005410	005411	005412	005413	005414	005415	005416	005417	005418	005419	005420	005421	005422	005423
0153	005424	005425	005426	005427	005428	005429	005430	005431	005432	005433	005434	005435	005436	005437	005438	005439
0154	005440	005441	005442	005443	005444	005445	005446	005447	005448	005449	005450	005451	005452	005453	005454	005455
0155	005466	005457	005458	005459	005460	005461	005462	005463	005464	005465	005466	005467	005468	005469	005470	005471
0156	005472	005473	005474	005475	005476	005477	005478	005479	005480	005481	005482	005483	005484	005485	005486	005487
0157	005488	005489	005490	005491	005492	005493	005494	005495	005496	005497	005498	005499	005500	005501	005502	005503
0158	005504	005505	005506	005507	005508	005509	005510	005511	005512	005513	005514	005515	005516	005517	005518	005519
0159	005520	005521	005522	005523	005524	005525	005526	005527	005528	005529	005530	005531	005532	005533	005534	005535
015A	005536	005537	005538	005539	005540	005541	005542	005543	005544	005545	005546	005547	005548	005549	005550	005551
015B	005552	005553	005554	005555	005556	005557	005558	005559	005560	005561	005562	005563	005564	005565	005566	005567
015C	005568	005569	005570	005571	005572	005573	005574	005575	005576	005577	005578	005579	005580	005581	005582	005583
015D	005584	005585	005586	005587	005588	005589	005590	005591	005592	005593	005594	005595	005596	005597	005598	005599
015E	005600	005601	005602	005603	005604	005605	005606	005607	005608	005609	005610	005611	005612	005613	005614	005615
015F	005616	005617	005618	005619	005620	005621	005622	005623	005624	005625	005626	005627	005628	005629	005630	005631

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0160	005632	005633	005634	005635	005636	005637	005638	005639	005640	005641	005642	005643	005644	005645	005646	005647
0161	005648	005649	005650	005651	005652	005653	005654	005655	005656	005657	005658	005659	005660	005661	005662	005663
0162	005664	005665	005666	005667	005668	005669	005670	005671	005672	005673	005674	005675	005676	005677	005678	005679
0163	005680	005681	005682	005683	005684	005685	005686	005687	005688	005689	005690	005691	005692	005693	005694	005695
0164	005696	005697	005698	005699	005700	005701	005702	005703	005704	005705	005706	005707	005708	005709	005710	005711
0165	005712	005713	005714	005715	005716	005717	005718	005719	005720	005721	005722	005723	005724	005725	005726	005727
0166	005728	005729	005730	005731	005732	005733	005734	005735	005736	005737	005738	005739	005740	005741	005742	005743
0167	005744	005745	005746	005747	005748	005749	005750	005751	005752	005753	005754	005755	005756	005757	005758	005759
0168	005760	005761	005762	005763	005764	005765	005766	005767	005768	005769	005770	005771	005772	005773	005774	005775
0169	005776	005777	005778	005779	005780	005781	005782	005783	005784	005785	005786	005787	005788	005789	005790	005791
016A	005792	005793	005794	005795	005796	005797	005798	005799	005800	005801	005802	005803	005804	005805	005806	005807
016B	005808	005809	005810	005811	005812	005813	005814	005815	005816	005817	005818	005819	005820	005821	005822	005823
016C	005824	005825	005826	005827	005828	005829	005830	005831	005832	005833	005834	005835	005836	005837	005838	005839
016D	005840	005841	005842	005843	005844	005845	005846	005847	005848	005849	005850	005851	005852	005853	005854	005855
016E	005856	006857	005858	005859	005860	005861	005862	005863	005864	005865	005866	005867	005868	005869	005870	005871
016F	005872	005873	005874	005875	005876	005877	005878	005879	005880	005881	005882	005883	005884	005885	005886	005887

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0170	005888	005889	005890	005891	005892	005893	005894	005895	005896	005897	005898	005899	005900	005901	005902	005903
0171	005904	005905	005906	005907	005908	005909	005910	005911	005912	005913	005914	005915	005916	005917	005918	005919
0172	005920	005921	005922	005923	005924	005925	005926	005927	005928	005929	005930	005931	005932	005933	005934	005935
0173	005936	005937	005938	005939	005940	005941	005942	005943	005944	005945	005946	005947	005948	005949	005950	005951
0174	005952	005953	005954	005955	005956	005957	005958	005959	005960	005961	005962	005963	005964	005965	005966	005967
0175	005968	005969	005970	005971	005972	005973	005974	005975	005976	005977	005978	005979	005980	005981	005982	005983
0176	005984	005985	005986	005987	005988	005989	005990	005991	005992	005993	005994	005995	005996	005997	005998	005999
0177	006000	006001	006002	006003	006004	006005	006006	006007	006008	006009	006010	006011	006012	006013	006014	006015
0178	006016	006017	006018	006019	006020	006021	006022	006023	006024	006025	006026	006027	006028	006029	006030	006031
0179	006032	006033	006034	006035	006036	006037	006038	006039	006040	006041	006042	006043	006044	006045	006046	006047
017A	006048	006049	006050	006051	006052	006053	006054	006055	006056	006057	006058	006059	006060	006061	006062	006063
017B	006064	006065	006066	006067	006068	006069	006070	006071	006072	006073	006074	006075	006076	006077	006078	006079
017C	006080	006081	006082	006083	006084	006085	006086	006087	006088	006089	006090	006091	006092	006093	006094	006095
017D	006096	006097	006098	006099	006100	006101	006102	006103	006104	006105	006106	006107	006108	006109	006110	006111
017E	006112	006113	006114	006115	006116	006117	006118	006119	006120	006121	006122	006123	006124	006125	006126	006127
017F	006128	006129	006130	006131	006132	006133	006134	006135	006136	006137	006138	006139	006140	006141	006142	006143

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0180	006144	006145	005146	006147	006148	006149	006150	006151	006152	006153	006154	006155	006156	006157	006158	006159
0181	006160	006161	006162	006163	006164	006165	006166	006167	006168	006169	006170	006171	006172	006173	006174	006175
0182	006176	006177	006178	006179	006180	006181	006182	006183	006184	006185	006186	006187	006188	006189	006190	006191
0183	006192	006193	006194	006195	006196	006197	006198	006199	006200	006201	006202	006203	006204	006205	006206	006207
0184	006208	006209	006210	006211	006212	006213	006214	006215	006216	006217	006218	006219	006220	006221	006222	006223
0185	006224	006225	006226	006227	006228	006229	006230	006231	006232	006233	006234	006235	006236	006237	006238	006239
0186	006240	006241	006242	006243	006244	006245	006246	006247	006248	006249	006250	006251	006252	006253	006254	006255
0187	006256	006257	006258	006259	006260	006261	006262	006263	006264	006265	006266	006267	006268	006269	006270	006271
0188	006272	006273	006274	006275	006276	006277	006278	006279	006280	006281	006282	006283	006284	006285	006286	006287
0189	006288	006289	006290	006291	006292	006293	006294	006295	006296	006297	006298	006299	006300	006301	006302	006303
018A	006304	006305	006306	006307	006308	006309	006310	006311	006312	006313	006314	006315	006316	006317	006318	006319
018B	006320	006321	006322	006323	006324	006325	006326	006327	006328	006329	006330	006331	006332	006333	006334	006335
018C	006336	006337	006338	006339	006340	006341	006342	006343	006344	006345	006346	006347	006348	006349	006350	006351
018D	006352	006353	006354	006355	006356	006357	006358	006359	006360	006361	006362	006363	006364	006365	006366	006367
018E	006368	006369	006370	006371	006372	006373	006374	006375	006376	006377	006378	006379	006380	006381	006382	006383
018F	006384	006385	006386	006387	006388	006389	006390	006391	006392	006393	006394	006395	006396	006397	006398	006399

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0190	006400	006401	006402	006403	006404	006405	006406	006407	006408	006409	006410	006411	006412	006413	006414	006415
0191	006416	006417	006418	006419	006420	006421	006422	006423	006424	006425	006426	006427	006428	006429	006430	006431
0192	006432	006433	006434	006435	006436	006437	006438	006439	006440	006441	006442	006443	006444	006445	006446	006447
0193	006448	006449	006450	006451	006452	006453	006454	006455	006456	006457	006458	006459	006460	006461	006462	006463
0194	006464	006465	006466	006467	006468	006469	006470	006471	006472	006473	006474	006475	006476	006477	006478	006479
0195	006480	006481	006482	006483	006484	006485	006486	006487	006488	006489	006490	006491	006492	006493	006494	006495
0196	006496	006497	006498	006499	006500	006501	006502	006503	006504	006505	006506	006507	006508	006509	006510	006511
0197	006512	006513	006514	006515	006516	006517	006518	006519	006520	006521	006522	006523	006524	006525	006526	006527
0198	006528	006529	006530	006531	006532	006533	006534	006535	006536	006537	006538	006539	006540	006541	006542	006543
0199	006544	006545	006546	006547	006548	006549	006550	006551	006552	006553	006554	006555	006556	006557	006558	006559
019A	006560	006561	006562	006563	006564	006565	006566	006567	006568	006569	006570	006571	006572	006573	006574	006575
019B	006576	006577	006578	006579	006580	006581	006582	006583	006584	006585	006586	006587	006588	006589	006590	006591
019C	006592	006593	006594	006595	006596	006597	006598	006599	006600	006601	006602	006603	006604	006605	006606	006607
019D	006608	006609	006610	006611	006612	006613	006614	006615	006616	006617	006618	006619	006620	006621	006622	006623
019E	006624	006625	006626	006627	006628	006629	006630	006631	006632	006633	006634	006635	006636	006637	006638	006639
019F	006640	006641	006642	006643	006644	006645	006646	006647	006648	006649	006650	006651	006652	006653	006654	006655

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01A0	006656	006657	006658	006659	006660	006661	006662	006663	006664	006665	006666	006667	006668	006669	006670	006671
01A1	006672	006673	006674	006675	006676	006677	006678	006679	006680	006681	006682	006683	006684	006685	006686	006687
01A2	006688	006689	006690	006691	006692	006693	006694	006695	006696	006697	006698	006699	006700	006701	006702	006703
01A3	006704	006705	006706	006707	006708	006709	006710	006711	006712	006713	006714	006715	006716	006717	006718	006719
01A4	006720	006721	006722	006723	006724	006725	006726	006727	006728	006729	006730	006731	006732	006733	006734	006735
01A5	006736	006737	006738	006739	006740	006741	006742	006743	006744	006745	006746	006747	006748	006749	006750	006751
01A6	006752	006753	006754	006755	006756	006757	006758	006759	006760	006761	006762	006763	006764	006765	006766	006767
01A7	006768	006769	006770	006771	006772	006773	006774	006775	006776	006777	006778	006779	006780	006781	006782	006783
01A8	006784	006785	006786	006787	006788	006789	006790	006791	006792	006793	006794	006795	006796	006797	006798	006799
01A9	006800	006801	006802	006803	006804	006805	006806	006807	006808	006809	006810	006811	006812	006813	006814	006815
01AA	006816	006817	006818	006819	006820	006821	006822	006823	006824	006825	006826	006827	006828	006829	006830	006831
01AB	006832	006833	006834	006835	006836	006837	006838	006839	006840	006841	006842	006843	006844	006845	006846	006847
01AC	006848	006849	006850	006851	006852	006853	006854	006855	006856	006857	006858	006859	006860	006861	006862	006863
01AD	006864	006865	006866	006867	006868	006869	006870	006871	006872	006873	006874	006875	006876	006877	006878	006879
01AE	006880	006881	006882	006883	006884	006885	006886	006887	006888	006889	006890	006891	006892	006893	006894	006895
01AF	006896	006897	006898	006899	006900	006901	006902	006903	006904	006905	006906	006907	006908	006909	006910	006911

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01B0	006912	006913	006914	006915	006916	006917	006918	006919	006920	006921	006922	006923	006924	006925	006926	006927
01B1	006928	006929	006930	006931	006932	006933	006934	006935	006936	006937	006938	006939	006940	006941	006942	006943
01B2	006944	006945	006946	006947	006948	006949	006950	006951	006952	006953	006954	006955	006956	006957	006958	006959
01B3	006960	006961	006962	006963	006964	006965	006966	006967	006968	006969	006970	006971	006972	006973	006974	006975
01B4	006976	006977	006978	006979	006980	006981	006982	006983	006984	006985	006986	006987	006988	006989	006990	006991
01B5	006992	006993	006994	006995	006996	006997	006998	006999	007000	007001	007002	007003	007004	007005	007006	007007
01B6	007008	007009	007010	007011	007012	007013	007014	007015	007016	007017	007018	007019	007020	007021	007022	007023
01B7	007024	007025	007026	007027	007028	007029	007030	007031	007032	007033	007034	007035	007036	007037	007038	007039
01B8	007040	007041	007042	007043	007044	007045	007046	007047	007048	007049	007050	007051	007052	007053	007054	007055
01B9	007056	007057	007058	007059	007060	007061	007062	007063	007064	007065	007066	007067	007068	007069	007070	007071
01BA	007072	007073	007074	007075	007076	007077	007078	007079	007080	007081	007082	007083	007084	007085	007086	007087
01BB	007088	007089	007090	007091	007092	007093	007094	007095	007096	007097	007098	007099	007100	007101	007102	007103
01BC	007104	007105	007106	007107	007108	007109	007110	007111	007112	007113	007114	007115	007116	007117	007118	007119
01BD	007120	007121	007122	007123	007124	007125	007126	007127	007128	007129	007130	007131	007132	007133	007134	007135
01BE	007136	007137	007138	007139	007140	007141	007142	007143	007144	007145	007146	007147	007148	007149	007150	007151
01BF	007152	007153	007154	007155	007156	007157	007158	007159	007160	007161	007162	007163	007164	007165	007166	007167

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01C0	007168	007169	007170	007171	007172	007173	007174	007175	007176	007177	007178	007179	007180	007181	007182	007183
01C1	007184	007185	007186	007187	007188	007189	007190	007191	007192	007193	007194	007195	007196	007197	007198	007199
01C2	007200	007201	007202	007203	007204	007205	007206	007207	007208	007209	007210	007211	007212	007213	007214	007215
01C3	007216	007217	007218	007219	007220	007221	007222	007223	007224	007225	007226	007227	007228	007229	007230	007231
01C4	007232	007233	007234	007235	007236	007237	007238	007239	007240	007241	007242	007243	007244	007245	007246	007247
01C5	007248	007249	007250	007251	007252	007253	007254	007255	007256	007257	007258	007259	007260	007261	007262	007263
01C6	007264	007265	007266	007267	007268	007269	007270	007271	007272	007273	007274	007275	007276	007277	007278	007279
01C7	007280	007281	007282	007283	007284	007285	007286	007287	007288	007289	007290	007291	007292	007293	007294	007295
01C8	007296	007297	007298	007299	007300	007301	007302	007303	007304	007305	007306	007307	007308	007309	007310	007311
01C9	007312	007313	007314	007315	007316	007317	007318	007319	007320	007321	007322	007323	007324	007325	007326	007327
01CA	007328	007329	007330	007331	007332	007333	007334	007335	007336	007337	007338	007339	007340	007341	007342	007343
01CB	007344	007345	007346	007347	007348	007349	007350	007351	007352	007353	007354	007355	007356	007357	007358	007359
01CC	007360	007361	007362	007363	007364	007365	007366	007367	007368	007369	007370	007371	007372	007373	007374	007375
01CD	007376	007377	007378	007379	007380	007381	007382	007383	007384	007385	007386	007387	007388	007389	007390	007391
01CE	007392	007393	007394	007395	007396	007397	007398	007399	007400	007401	007402	007403	007404	007405	007406	007407
01CF	007408	007409	007410	007411	007412	007413	007414	007415	007416	007417	007418	007419	007420	007421	007422	007423

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01D0	007424	007425	007426	007427	007428	007429	007430	007431	007432	007433	007434	007435	007436	007437	007438	007439
01D1	007440	007441	007442	007443	007444	007445	007446	007447	007448	007449	007450	007451	007452	007453	007454	007455
01D2	007456	007457	007458	007459	007460	007461	007462	007463	007464	007465	007466	007467	007468	007469	007470	007471
01D3	007472	007473	007474	007475	007476	007477	007478	007479	007480	007481	007482	007483	007484	007485	007486	007487
01D4	007488	007489	007490	007491	007492	007493	007494	007495	007496	007497	007498	007499	007500	007501	007502	007503
01D5	007504	007505	007506	007507	007508	007509	007510	007511	007512	007513	007514	007515	007516	007517	007518	007519
01D6	007520	007521	007522	007523	007524	007525	007526	007527	007528	007529	007530	007531	007532	007533	007534	007535
01D7	007536	007537	007538	007539	007540	007541	007542	007543	007544	007545	007546	007547	007548	007549	007550	007551
01D8	007552	007553	007554	007555	007556	007557	007558	007559	007560	007561	007562	007563	007564	007565	007566	007567
01D9	007568	007569	007570	007571	007572	007573	007574	007575	007576	007577	007578	007579	007580	007581	007582	007583
01DA	007584	007585	007586	007587	007588	007589	007590	007591	007592	007593	007594	007595	007596	007597	007598	007599
01DB	007600	007601	007602	007603	007604	007605	007606	007607	007608	007609	007610	007611	007612	007613	007614	007615
01DC	007616	007617	007618	007619	007620	007621	007622	007623	007624	007625	007626	007627	007628	007629	007630	007631
01DD	007632	007633	007634	007635	007636	007637	007638	007639	007640	007641	007642	007643	007644	007645	007646	007647
01DE	007648	007649	007650	007651	007652	007653	007654	007655	007656	007657	007658	007659	007660	007661	007662	007663
01DF	007664	007665	007666	007667	007668	007669	007670	007671	007672	007673	007674	007675	007676	007677	007678	007679

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01E0	007680	007681	007682	007683	007684	007685	007686	007687	007688	007689	007690	007691	007692	007693	007694	007695
01E1	007696	007697	007698	007699	007700	007701	007702	007703	007704	007705	007706	007707	007708	007709	007710	007711
01E2	007712	007713	007714	007715	007716	007717	007718	007719	007720	007721	007722	007723	007724	007725	007726	007727
01E3	007728	007729	007730	007731	007732	007733	007734	007735	007736	007737	007738	007739	007740	007741	007742	007743
01E4	007744	007745	007746	007747	007748	007749	007750	007751	007752	007753	007754	007755	007756	007757	007758	007759
01E5	007760	007761	007762	007763	007764	007765	007766	007767	007768	007769	007770	007771	007772	007773	007774	007775
01E6	007776	007777	007778	007779	007780	007781	007782	007783	007784	007785	007786	007787	007788	007789	007790	007791
01E7	007792	007793	007794	007795	007796	007797	007798	007799	007800	007801	007802	007803	007804	007805	007806	007807
01E8	007808	007809	007810	007811	007812	007813	007814	007815	007816	007817	007818	007819	007820	007821	007822	007823
01E9	007824	007825	007826	007827	007828	007829	007830	007831	007832	007833	007834	007835	007836	007837	007838	007839
01EA	007840	007841	007842	007843	007844	007845	007846	007847	007848	007849	007850	007851	007852	007853	007854	007855
01EB	007856	007857	007858	007859	007860	007861	007862	007863	007864	007865	007866	007867	007868	007869	007870	007871
01EC	007872	007873	007874	007875	007876	007877	007878	007879	007880	007881	007882	007883	007884	007885	007886	007887
01ED	007888	007889	007890	007891	007892	007893	007894	007895	007896	007897	007898	007899	007900	007901	007902	007903
01EE	007904	007905	007906	007907	007908	007909	007910	007911	007912	007913	007914	007915	007916	007917	007918	007919
01EF	007920	007921	007922	007923	007924	007925	007926	007927	007928	007929	007930	007931	007932	007933	007934	007935

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01F0	007936	007937	007938	007939	007940	007941	007942	007943	007944	007945	007946	007947	007948	007949	007950	007951
01F1	007952	007953	007954	007955	007956	007957	007958	007959	007960	007961	007962	007963	007964	007965	007966	007967
01F2	007968	007969	007970	007971	007972	007973	007974	007975	007976	007977	007978	007979	007980	007981	007982	007983
01F3	007984	007985	007986	007987	007988	007989	007990	007991	007992	007993	007994	007995	007996	007997	007998	007999
01F4	008000	008001	008002	008003	008004	008005	008006	008007	008008	008009	008010	008011	008012	008013	008014	008015
01F5	008016	008017	008018	008019	008020	008021	008022	008023	008024	008025	008026	008027	008028	008029	008030	008031
01F6	008032	008033	008034	008035	008036	008037	008038	008039	008040	008041	008042	008043	008044	008045	008046	008047
01F7	008048	008049	008050	008051	008052	008053	008054	008055	008056	008057	008058	008059	008060	008061	008062	008063
01F8	008064	008065	008066	008067	008068	008069	008070	008071	008072	008073	008074	008075	008076	008077	008078	008079
01F9	008080	008081	008082	008083	008084	008085	008086	008087	008088	008089	008090	008091	008092	008093	008094	008095
01FA	008096	008097	008098	008099	008100	008101	008102	008103	008104	008105	008106	008107	008108	008109	008110	008111
01FB	008112	008113	008114	008115	008116	008117	008118	008119	008120	008121	008122	008123	008124	008125	008126	008127
01FC	008128	008129	008130	008131	008132	008133	008134	008135	008136	008137	008138	008139	008140	008141	008142	008143
01FD	008144	008145	008146	008147	008148	008149	008150	008151	008152	008153	008154	008155	008156	008157	008158	008159
01FE	008160	008161	008162	008163	008164	008165	008166	008167	008168	008169	008170	008171	008172	008173	008174	008175
01FF	008176	008177	008178	008179	008180	008181	008182	008183	008184	008185	008186	008187	008188	008189	008190	008191

APPENDIX E
HEXADECIMAL CONVERSION TABLE

Converting to hexadecimal may be simplified by using the following table.

To convert $(61275)_{10}$ to hexadecimal, using the table: the table entry closest to, but not greater than, $(61275)_{10}$ is $(61184)_{10}$, which equals $(EF00)_{16}$ from the table. Subtracting 61,184 from the original number $(61275\text{-}61184)_{10}$ leaves a remainder of $(91)_{10}$, which equals $(5B)_{16}$. Therefore, $(61275)_{10} = (EF5B)_{16}$.

COLUMN

ROW	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	000 00000	001 00256	002 00512	003 00768	004 01024	005 01280	006 01536	007 01792	008 02048	009 02304	010 02560	011 02816	012 03072	013 03328	014 03584	015 03840
1	016 04096	017 04352	018 04608	019 04864	020 05120	021 05376	022 05632	023 05888	024 06144	025 06400	026 06656	027 06912	028 07168	029 07424	030 07680	031 07936
2	032 08192	033 08448	034 08704	035 08960	036 09216	037 09472	038 09728	039 09984	040 10240	041 10496	042 10752	043 11008	044 11264	045 11520	046 11776	047 12032
3	048 12288	049 12544	050 12800	051 13056	052 13312	053 13568	054 13824	055 14080	056 14336	057 14592	058 14848	059 15104	060 15360	061 15616	062 15872	063 16128
4	064 16384	065 16640	066 16896	067 17152	068 17408	069 17664	070 17920	071 18176	072 18432	073 18688	074 18944	075 19200	076 19456	077 19712	078 19968	079 20224
5	080 20480	081 20735	082 20992	083 21248	084 21504	085 21760	086 22016	087 22272	088 22528	089 22784	090 23040	091 23296	092 23552	093 23808	094 24064	095 24320
6	096 24576	097 24832	098 25088	099 25344	100 25600	101 25856	102 26112	103 26368	104 26624	105 26880	106 27136	107 27392	108 27648	109 27904	110 28160	111 28416
7	112 28672	113 28928	114 29184	115 29440	116 29696	117 29952	118 30208	119 30464	120 30720	121 30976	122 31232	123 31488	124 31744	125 32000	126 32256	127 32512
8	128 32768	129 33024	130 33280	131 33536	132 33792	133 34048	134 34304	135 34560	136 34816	137 35072	138 35328	139 35584	140 35840	141 36096	142 36352	143 36608
9	144 36864	145 37120	146 37376	147 37632	148 37888	149 38144	150 38400	151 38656	152 38912	153 39168	154 39424	155 39680	156 39936	157 40192	158 40448	159 40704
A	160 40960	161 41216	162 41472	163 41728	164 41984	165 42240	166 42496	167 42752	168 43008	169 43264	170 43520	171 43776	172 44032	173 44288	174 44544	175 44800
B	176 45056	177 45312	178 45568	179 45824	180 46080	181 46336	182 46592	183 46848	184 47104	185 47360	186 47616	187 47872	188 48128	189 48384	190 48640	191 48896
C	192 49152	193 49408	194 49664	195 49920	196 50176	197 50432	198 50688	199 50944	200 51200	201 51456	202 51712	203 51968	204 52224	205 52480	206 52736	207 52992
D	208 53248	209 53504	210 53760	211 54016	212 54272	213 54528	214 54784	215 55040	216 55296	217 55552	218 55808	219 56064	220 56320	221 56576	222 56832	223 57088
E	224 57344	225 57600	226 57856	227 58112	228 58368	229 58624	230 58880	231 59136	232 59392	233 59648	234 59904	235 60160	236 60416	237 60672	238 60928	239 61184
F	240 61440	241 61696	242 61952	243 62208	244 62464	245 62720	246 62976	247 63232	248 63488	249 63744	250 64000	251 64256	252 64512	253 64768	254 65024	255 65280

APPENDIX F
HEXADECIMAL ADDITIONS

In the following Hexadecimal Addition Table, all values represent the result of an addition of a hexadecimal character from the column across the top and the column down the left side. The result of the addition is found where the two characters to be added intersect within the table. All values above the slanted line represent the result of an addition with no carry generated; all those values below the slanted line represent the result of an addition with a carry of one generated into the next character position of the hexadecimal result.

HEXADECIMAL ADDITION TABLE

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0
2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1
3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2
4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3
5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4
6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5
7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6
8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7
9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8
A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9
B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A
C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B
D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C
E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D
F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E

APPENDIX G
NUMERICAL INFORMATION

TABLE OF POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 380 086 994 171 142 578 125

APPENDIX H
USASCII INTERCHANGE CODE SET WITH CARD PUNCH CODES

Row	Col	0	1	2	3	4	5	6	7
Bit Positions									
4	0	0	0	0	0	0	0	0	0
5	1	0	0	0	0	1	1	1	1
6	2	0	0	1	1	0	0	1	1
7	3	0	1	0	1	0	1	0	1
0000	0	NUL 12-0-9-8-1	DLE 12-11-9-8-1	SP No punch	0 0	@ 8-4	P 11-7	` 8-1	p 12-11-7
0001	1	SOH 12-9-1	DCI 11-9-1	! 12-8-7	1 1	A 12-1	Q 11-8	a 12-0-1	q 12-11-8
0010	2	STX 12-9-2	DC2 11-9-2	" 8-7	2 2	B 12-2	R 11-9	b 12-0-2	r 12-11-9
0011	3	ETX 12-9-3	DC3 11-9-3	# 8-3	3 3	C 12-3	S 0-2	c 12-0-3	s 11-0-2
0100	4	EOT 9-7	DC4 9-8-4	$ 11-8-3	4 4	D 12-4	T 0-3	d 12-0-4	t 11-0-3
0101	5	ENQ 0-9-8-5	NAK 9-8-5	% 0-8-4	5 5	E 12-5	U 0-4	e 12-0-5	u 11-0-4
0110	6	ACK 0-9-8-6	SYN 9-2	& 12	6 6	F 12-6	V 0-5	f 12-0-6	v 11-0-5
0111	7	BEL 0-9-8-7	ETB 0-9-6	' 8-5	7 7	G 12-7	W 0-6	g 12-0-7	w 11-0-6
1000	8	BS 11-9-6	CAN 11-9-8	(12-8-5	8 8	H 12-8	X 0-7	h 12-0-8	x 11-0-7
1001	9	HT 12-9-5	EM 11-9-8-1	) 11-8-5	9 9	I 12-9	Y 0-8	i 12-0-9	y 11-0-8
1010	A	LF 0-9-5	SUB 9-8-7	. 11-8-4	: 8-2	J 11-1	Z 0-9	j 12-11-1	z 11-0-9
1011	B	VT 12-9-8-3	ESC 0-9-7	+ 12-8-6	; 11-8-6	K 11-2	[12-8-2	k 12-11-2	{ 12-0
1100	C	FF 12-9-8-4	FS 11-9-8-4	, 0-8-3	< 12-8-4	L 11-3	\ 0-8-2	l 12-11-3	¦ 12-11
1101	D	CR 12-9-8-5	GS 11-9-8-5	- 11	= 8-6	M 11-4	] 11-8-2	m 12-11-4	} 11-0
1110	E	SO 12-9-8-6	RS 11-9-8-6	. 12-8-3	> 0-8-6	N 11-5	∧ 11-8-7	n 12-11-5	~ 11-0-1
1111	F	SI 12-9-8-7	US 11-9-8-7	/ 0-1	? 0-8-7	O 11-6	— 0-8-5	o 12-11-6	DEL 12-9-7

Some positions in the USASCII code chart may have a different graphic representation on various devices as:

USASCII	IBM 029
!	I
[	¢
]	!
Λ	>

Control Characters:

NUL	–	Null
SOH	–	Start of Heading (CC)
STX	–	Start of Text (CC)
ETX	–	End of Text (CC)
EOT	–	End of Transmission (CC)
ENQ	–	Enquiry (CC)
ACK	–	Acknowledge (CC)
BEL	–	Bell (audible or attention signal)
BS	–	Backspace (FE)
HT	–	Horizontal Tabulation (punch card skip)(FE)
LF	–	Line Feed (FE)
VT	–	Vertical Tabulation (FE)
FF	–	Form Feed (FE)
CR	–	Carriage Return (FE)
SO	–	Shift Out
Si	–	Shift In
DLE	–	Data Link Escape (CC)
DC1	–	Device Control 1
DC2	–	Device Control 2
DC3	–	Device Control 3
DC4	–	Device Control 4 (stop)
NAK	–	Negative Acknowledge (CC)
SYN	–	Synchronous Idle (CC)
ETB	–	End of Transmission Block (CC)
CAN	–	Cancel
EM	–	End of Medium
SS	–	Start of Special Sequence
ESC	–	Escape
FS	–	File Separator (IS)
GS	–	Group Separator (IS)
RS	–	Record Separator (IS)
US	–	Unit Separator (IS)
DEL	–	Delete
SP	–	Space (normally nonprinting)
(CC)	–	Communication Control
(FE)	–	Format Effector
(IS)	–	Information Separator