INDUSTRIAL
PRODUCTS
DIVISION

PROGRAMMER'S REFERENCE MANUAL
MODEL 960 PROCESS CONTROL COMPUTER

TEXAS INSTRUMENTS
INCORPORATED
HOUSTON, TEXAS

PROGRAMMER'S REFERENCE MANUAL

MODEL 960 PROCESS CONTROL COMPUTER

PART NO. 214091-9701
REVISED 1 SEPTEMBER 1970

This manual contains all latest revisions as of this printing date. Manuals furnished with the equipment will contain all latest revisions.

INDUSTRIAL
PRODUCTS
DIVISION

TEXAS INSTRUMENTS
INCORPORATED

P.O. BOX 66027 HOUSTON, TEXAS 77006

CABLE: TEXINS

PRINTED IN U.S.A.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

APPENDICES

SECTION I

GENERAL DESCRIPTION

1-1. SCOPE OF MANUAL.

This manual contains program reference data for the Texas Instruments Model 960 Process Control Computer. A brief description of the equipment is presented to aid in understanding the programming system. Related programming data is given where appropriate to assist the programmer in creative application of the computer.

1-2. EQUIPMENT CAPABILITIES.

The programmer's job is simplified by the unique structure of the computer. External operations may be directly related to internal computer functions. Operational software is easily established in the language normally used to describe process control functions. The computer is a versatile tool for process automation which readily adapts to a wide variety of applications. These applications may be discrete or continuous operations. Typical applications include tool operation, fabrication and automatic assembly, material handling, environmental control, and data acquisition. The computer can perform inspections and issue status reports. Pertinent data can be displayed to an operator at the job site or relayed to a central computer facility to provide accurate data for management decisions.

Perhaps the most worthy characteristic of the computer is its real-time process control capabilities. Real-time process control requires a computer with fast efficient context switching , easy manipulation of bits and bit-fields, and easy exchange of data between the computer and external devices. The TI 960 computer solves a great many automation problems with the following features:

a. Dual Mode Operation. The dual-mode feature permits fast context switching. While running in one mode with one set of registers and execution counter, control can be switched to a second mode with identical capabilities. This not only provides a new programming environment, but frequently avoids the need to save the status of the old environment. Mode switching can be accomplished under interrupt or programmed instruction control.

b. Real-Time Clocks. Many process control functions are time critical. The computer's optional 1-millisecond resolution interval timers perform many tight, time critical functions. These optional timers are desirable where process functions must occur at specific instants or must occur after a specific time delay.

c. Data Input/Output. The Communication Register Unit (CRU) provides a changeable, efficient interface with controlled external devices. Four CRU modules, each with 16 input and 16 output lines, can be installed inside the Central Processing Unit (CPU) enclosure. Additional CRU modules may be added to expand the CRU capacity to 256 modules in a separate enclosure. Total capability would be, therefore, 4096 input and 4096 output lines. A variety of CRU functions are available, such as binary data modules, analog-to-digital modules, digital-to-analog modules, stepping motor controller modules, and pulse generator modules.

1-3. COMPUTER CHARACTERISTICS.

The computer block diagram (Figure 1-1) shows the basic internal function relationships.

a. The core memory can store from 4096 to 65,536 17-bit words (16 data bits plus 1 parity bit).

b. The Central Processing Unit (CPU) can address the core memory, perform arithmetic and logic functions, and sequence and control the exchange of information between the core memory and other elements of the computer. The CPU features an arithmetic unit and a read-only memory controller.

c. The Communication Register Unit (CRU) controls the exchange of information between the computer and external operations.

d. The Direct Memory Access Channel (DMAC) interfaces the computer with high-speed automatic computer peripherals, such as disc storage units, line printers, and magnetic tape units. By using a separate controller for each device, concurrent operation of high speed peripherals is achieved.

1-4. SPECIFICATIONS.

Core memory built in modules of 4096 words each

Minimum storage capacity, 4096 words

Maximum storage capacity, 65,536 words

Data word length, 16 information bits plus 1 parity bit

Cycle time, 980 nanoseconds

Instruction word length, 32 bits

16 virtual registers for arithmetic, index, or mask operations

Execution time

Load: 5.0 microseconds

Store: 5.3 microseconds

Add: 6.0 microseconds

Set CRU bit: 4.6 microseconds

Load register in CRU: 7.0 to 12.0 microseconds (1-16 bits)

Memory protect system for variable amounts of memory

Single and double address logic

Direct addressing of entire core memory, by word or bit

Indirect addressing with pre-indexing or post-indexing

Displacement index registers

3 MegaHertz clock

Three levels of priority interrupt

Input/Output System

Communication Register Unit with 3 million bits per second burst rate, output; 1.5 million bits per second burst rate, input

Direct Memory Access Channel with 1 million words per second burst rate, input or output

Dimensions, stand alone unit

12.0 inches wide

26.0 inches deep

20.0 inches high

Power consumption, 420 watts, average

Normal operating temperature

32°F to 122°F

0°C to 50°C

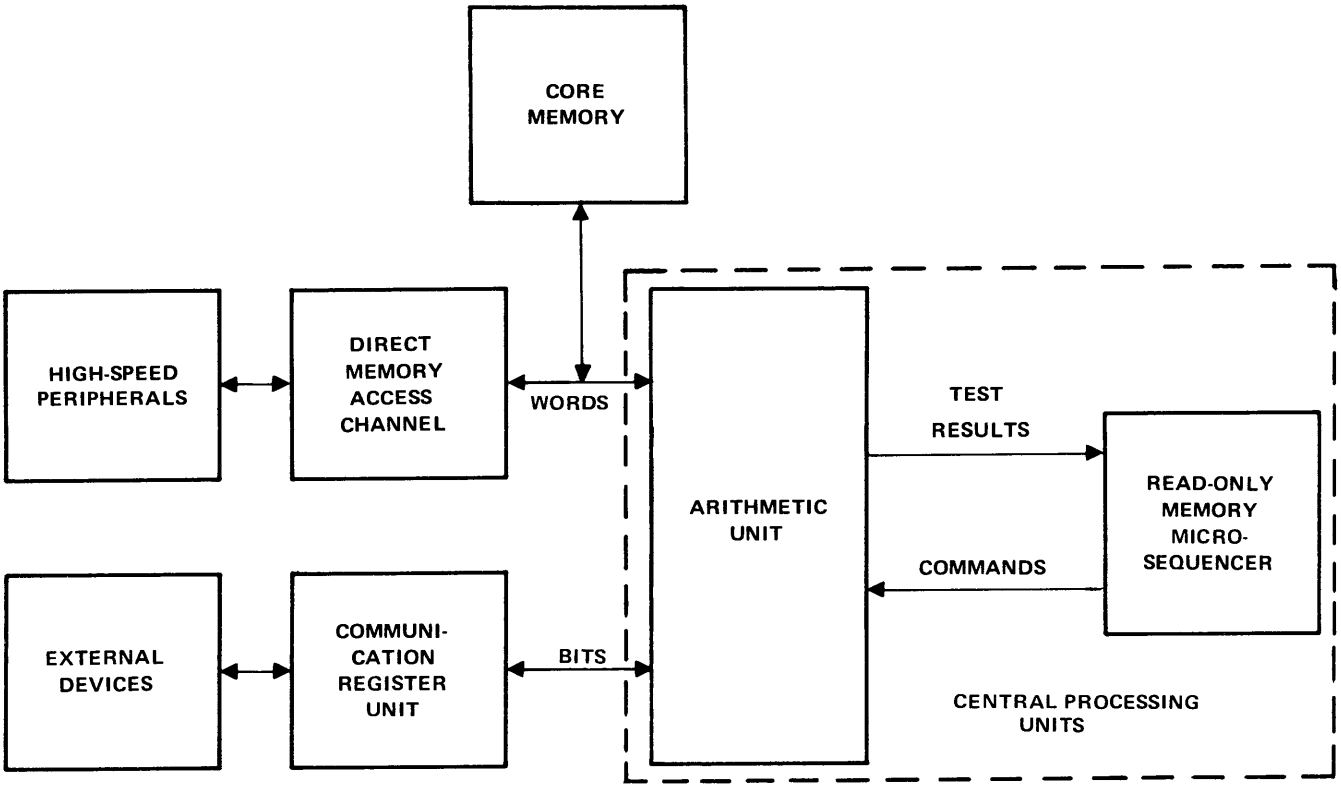


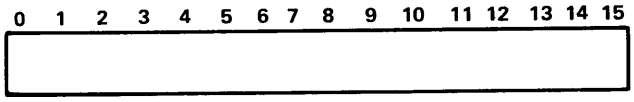
FIGURE 1-1. TI 960 COMPUTER BLOCK DIAGRAM

SECTION II

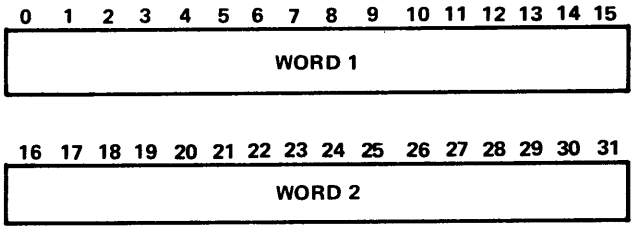
SYSTEM ORGANIZATION

2-1. INFORMATION FORMAT.

The basic element of information is a 16-bit word. Bit positions are numbered from 0 through 15.



A machine instruction occupies two words of memory in which the bit positions are numbered 0 through 31.



A fixed point integer occupies one word in memory, represented in binary form with the sign bit in position 0. A positive sign is indicated by a zero. Negative integers are represented in a two's complement form. Thus, the range of integers representable in a word of memory is from -2^{15} through $2^{15}-1$ or from −32,768 through 32,767.

2-2. CORE MEMORY.

The basic unit of core memory information is a 16-bit word plus a parity bit. The CPU may directly address all of the core memory. The addressing capability accommodates a maximum core memory size of 65,536 words. Core memory is modular and is available in up to 16 blocks of 4096 words. 8192 words can be installed within the CPU enclosure.

2-3. DEDICATED CORE MEMORY LOCATIONS.

A file of 16 virtual registers is implemented in the computer. Normally, eight are available to Supervisor Mode programs and the other eight are available to Worker Mode programs. All 16 registers are available in either mode using alternate mode addressing. The 16 registers and special functions assigned to each are listed in Table 2-1.

Other memory locations are required for interrupt and I/O status information. The following locations are reserved for functions listed below.

Memory Address$_{16}$	Function
90-91	Internal Interrupt
92-93	External Interrupt
94-95	Communication Register Unit Interrupt
96-97	Direct Memory Access Interrupt Status
98-99	Status, Device Controller 0
9A-9B	Status, Device Controller 1
9C-9D	Status, Device Controller 2
9E-9F	Status, Device Controller 3
A0-A1	Status, Device Controller 4
A2-A3	Status, Device Controller 5
A4-A5	Status, Device Controller 6
A6-A7	Status, Device Controller 7

2-4. PROTECTED MEMORY.

Locations 0-7F are protected memory locations reserved for bootstrap programs. The memory protect area can be expanded as an option (see below). Writing in protected locations can be affected only by manual intervention.

Optional Memory Protect Limits

Memory Address$_{16}$	
0-FF	excluding memory locations 80-A7
0-1FF	excluding memory locations 80-A7
0-3FF	excluding memory locations 80-A7

TABLE 2-1. REGISTER FILE

REGISTER NUMBER	SUPERVISOR REGISTER$_{16}$	WORKER REGISTER$_{16}$	FUNCTIONAL USE BY ARITHMETIC INSTRUCTIONS	FUNCTIONAL USE BY BIT AND FIELD MANIPULATING INSTRUCTIONS
0	80	88	General Arithmetic Reg. Index Register 0	General Register
1	81	89	General Arithmetic Reg. Index Register 1	General Register
2	82	8A	General Arithmetic Reg. Index Register 2	General Register
3	83	8B	General Arithmetic Reg. Index Register 3	General Register
4	84	8C	General Arithmetic Reg. Index Register 4	Base of Machine Data
5	85	8D	General Arithmetic Reg. Index Register 5	Base of Machine Procedures
6	86	8E	General Arithmetic Reg. Index Register 6	Base of Software Flag Areas
7	87	8F	General Arithmetic Reg. Index Register 7	Base of CRU Address

Optional Memory Protect Limits

Memory Address$_{16}$	
0-7FF	excluding memory locations 80-A7
0-FFF	excluding memory locations 80-A7
0-1FFF	excluding memory locations 80-A7
0-3FFF	excluding memory locations 80-A7
0-7FFF	excluding memory locations 80-A7
0-FFFF	excluding memory locations 80-A7

2-5. CENTRAL PROCESSING UNIT (CPU).

Programs are normally executed in one of two modes. Instructions in either mode have independent access to a general file of eight registers. Programs of either mode can provide arithmetic, logical, and control functions typical of general purpose computers. Programs designed for bit or bit-field processing can also be executed in either mode. These instructions can address any bit in memory or any bit or bit-field in the Communications Register Unit.

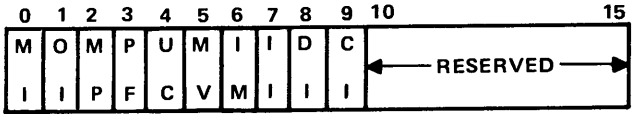
In the Supervisor Mode, the CPU is executing instructions via the Program Counter (PC) and utilizing the Supervisor Mode Register file for register referencing instructions. In the Worker Mode, instructions are executed via the Event Counter (EC) and utilize the Worker Mode register file for register referencing instructions. Any instruction can be executed in either mode. Mode changing is under program and interrupt control.

2-6. PROGRAM STATUS BLOCK.

The control conditions for program execution in the CPU are defined by the PC or EC and the status register. Instruction addressing is controlled by the PC when the computer is in the Supervisor Mode and by the EC when the computer is in the Worker Mode. At the completion of each instruction, depending on the mode in which the computer is executing, the PC or the EC is incremented by two. Program control instructions can modify the PC or the EC in other ways. The PC or the EC always contains the address of the next instruction to be executed. The PC and EC are implemented in live registers which can be addressed by special instructions.

The status register, used to hold the condition of the computer at any time and to enable or disable interrupts, is also implemented as a live register. Shown below is a functional chart of the bits in the status register.

2-7. STATUS REGISTER.



MI – Mode Indicator
If MI is 0, execution is in the Supervisor Mode, if a 1, execution is in the Worker Mode.

OI – Overflow Indicator
If OI is set to 1, an overflow condition exists.

MP – Memory Parity Indicator
If MP is a 1, a parity error has occurred.

PF – Power Failure Indicator
If PF is a 1, a power failure is indicated.

UC – Undefined Code
If UC is a 1, an undefined operation code has been detected.

MV – Memory Violation
If MV is a 1, an attempt has been made to alter protected memory.

IM – Index Mode
If IM is a 0, pre-indexing is performed; if it is a 1, post-indexing is done.

II – Internal Interrupt Mask
If II is a 0, the internal interrupt is enabled.

DI – DMAC Interrupt Mask
If DI is a 0, the DMAC interrupt is enabled.

CI – CRU Interrupt Mask
If CI is a 0, the CRU interrupt is enabled.

Bits 10-15 are reserved for future use.

2-8. PRIORITY INTERRUPT SYSTEM.

The computer features a priority interrupt system that provides added program control of Input/Output operations, provides immediate response to abnormal conditions, and allows immediate recognition of special external conditions.

The interrupt system gives the programmer flexible interrupt control of external devices. Implemented in the interrupt system are three levels: internal interrupts, first priority; Communications Register Unit interrupts, second priority; and direct memory access channel interrupts, third priority.

2-8.1 INTERNAL INTERRUPT. An internal interrupt provides immediate attention to any of the following conditions:

a. Memory Parity Error. This interrupt condition protects the user against a possible data transmission error or misread instruction. When this interrupt occurs, bit 2 (MP) of the status register is set to 1.

b. Change of Power System Status. This interrupt signal tells the computer that power has just been restored or that power loss is imminent. The power loss condition sets bit 3 (PF) of the status register to 1. When power is restored, the status register is cleared and bit 3 (PF) is set to 0.

c. Undefined Code Execution. An attempted execution of an undefined operation code results in an internal interrupt and bit four (UC) of the status register being set to 1.

d. Memory Violation. An attempt to alter protected memory with memory protect on results in an internal interrupt and sets bit 5 (MV) of the Status register to 1.

When an internal interrupt occurs, the interrupt mask bits of the status register are automatically set to 1, assuring that complete corrective response can be made for the condition. The internal interrupt causes a transfer of control to memory location 90_{16} where a Store Status and Branch (SSB) instruction would normally reside. The status register saved by this instruction shows the status of the interrupt mask bits just prior to the interrupt, that is, before the interrupt mask bits are set to 1 by the interrupt. The stored contents of the Status Register also show the condition(s) causing the interrupt. This allows determination of the cause(s) of the interrupt by examining bits 2, 3, 4, or 5 of the stored contents of the Status Register. The contents of the PC or the EC are also captured. The internal

interrupt is re-enabled by setting bit 7 (II) of the status register to a 0.

2-8.2 DIRECT MEMORY ACCESS CHANNEL INTERRUPT. The direct memory access channel (DMAC) interrupt is taken when any device connected to a channel port changes its status storage memory location. The DMAC interrupt causes a transfer of control to memory location 92_{16} where normally a Store Status and Branch instruction will reside. To identify the particular device causing the interrupt, a special memory location reserved for DMAC interrupts can be examined. When a DMAC interrupt is taken, bit 8 (DI) of the Status Register is set to 1, disabling DMAC interrupts.

2-8.3 COMMUNICATIONS REGISTER UNIT INTERRUPT. The Communications Register Unit interrupt occurs when a CRU interrupt line goes to 1. CRU interrupt modules containing eight interrupt lines are optionally available. The particular line causing the interrupt can be identified by scanning the CRU interrupt lines. When the interrupt occurs, bits 8 and 9 (DI and CI) of the status register are automatically set to 1, disabling other CRU interrupts and the DMAC interrupt. The CRU interrupt causes transfer of control to memory location 94_{16}.

2-9. INTERRUPT LINKAGE.

Two consecutive, fixed memory locations are associated with each interrupt. When an interrupt is taken, the instruction in the respective interrupt location is executed. The interrupt location will normally contain a Store Status and Branch instruction which will store either the PC or the EC and the status register and then cause a branch to an interrupt program sequence. After the interrupt is serviced, a Load Status Block instruction will restore the computer to the state it was in immediately prior to the interrupt. Saving and restoring general registers used in interrupt subroutines is the responsibility of the programmer.

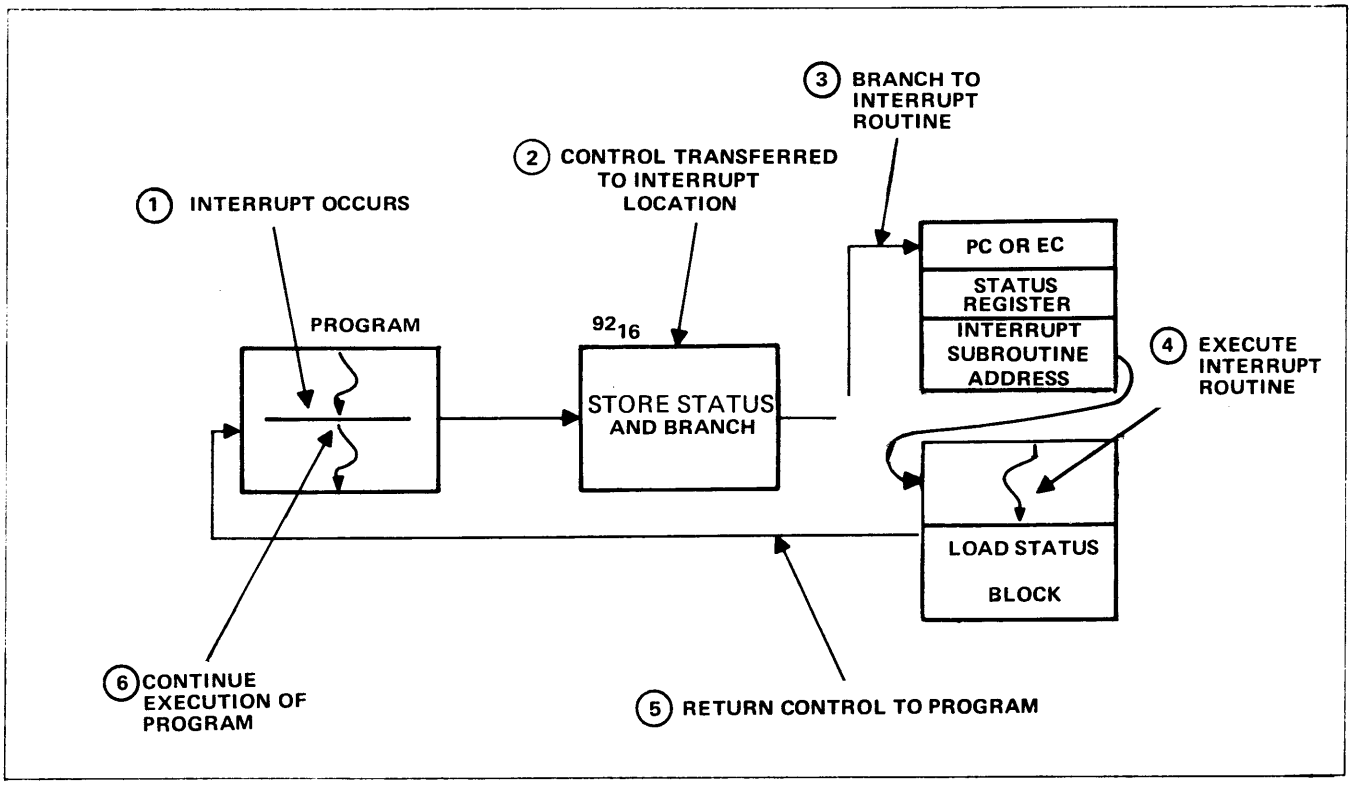


FIGURE 2-1. TYPICAL LINKAGE TO INTERRUPT ROUTINE

SECTION III

PROGRAMMING SYSTEM

3-1. INTRODUCTION.

The Programming System (PS) provides a simplified approach to a wide range of process control applications. Functionally, the programming system offers the programmer symbolic programming, simplified updating and corrections, relocatable programs, segment and subroutine linking, and debugging facilities. Major components of the Programming System are:

Symbolic Assembly Language (SAL)

Linking and Relocating Loader (LRL)

Source Maintenance Routines (SMR).

3-2. SYMBOLIC ASSEMBLY LANGUAGE.

Using SAL, the programmer may implement his programs either as standalone units, or may construct programs from modules of four basic segment types providing him a more flexible system. The four segment types are:

a. Procedures (PSEG). The procedure segment is normally the main body of the program and contains computer instructions. It is the action part of a program. Symbol attributes in SAL assist the programmer in making procedures re-entrant.

b. Data (DSEG). Data segments are used to provide storage, I/O Buffers, and constants for procedure segments.

c. Flags (FSEG). A flag segment allows the programmer to symbolically address the memory bit by bit.

d. Communication Register Segment (BSEG). This segment simplifies the assignment and use of symbolic addresses for references to bit lines in the Communication Register Unit, either by field or by individual bit.

One or more of the four segments is used to construct a program. Once the program is assembled by the SAL assembler, it may either be loaded into core memory by the LRL for execution or transformed into linked, relocatable object format by the LRL. The structure of a typical program is shown in Figure 3-1.

The SAL assembler creates the necessary data required by the LRL to perform its functions of linking the different segments, relocating segments, and completing the asembly process for external symbols.

The SAL programmer has at his disposal flexible address modification capabilities, a comprehensive set of machine instructions that make available the full capabilities of the machine hardware, and a powerful set of assembler control directives.

Indexing – Indexing is specified by the presence of a register number (0-7) or symbol after the address operand.
Example: L 2,N,3

Note: The underlined entries in the operand field of example instructions are used by the programmer to specify the appropriate action.

Indirect – Indirect addressing is specified by placing an asterisk (*) before the address operand.
Example: L 2,*N

Immediate – Immediate addressing is accomplished by using an alternate mnemonic of the operation, e.g., the immediate counterpart of the Load Register instruction (L) is LA (Load Address).
Example: LA 2,N

Relative – When the "at" symbol (@) precedes the address operand, the relative address of the operand is used rather than the absolute address.
Example: L 2,@N,5

Alternate Mode Registers – The presence of the pound sign (#) preceding the operand list causes the registers of the inactive mode to be used in the execution of that instruction.
Example: L #2,N

For maximum flexibility in address modification, any combination of the attributes may be used. Examples of some of the possible combinations are as follows:

a. L #2,N

b. L 2,*N,3

c. L #2,*@N,3

d. LA 2,@N,3

Relative Addressing – As previously stated, the program structure is divided into four basic groups to provide easy updating and program correction, simplified programming of re-entrant routines and independent assembly of segments. A hardware feature which makes this approach attractive is the automatic use of specified base registers in certain machine instructions. For example, when referencing a software flag in the Software Flag Segment (FSEG) with a Software Flag instruction, the value of the symbol representing the software flag is automatically biased with the contents of the Software Flag Base Register during execution of the instruction. This then creates the absolute bit address of the software flag. So, as the assembler is building an instruction that utilizes automatic base registers, the displacement of the symbol relative to the origin of the segment in which it was defined is specified rather than the absolute address of the symbol. For those instructions not utilizing automatic base registers, the relative value of the symbol rather than its absolute value may be specified with the use of the relative attribute, the "at" symbol, '@'. The relative attribute may also be used with certain assembler directives. For example:

DATA @SYMBOL

This directive would cause a data word to be initialized with the value of the displacement of SYMBOL relative to the origin of the segment in which it was defined. If the relative attribute were not used in the example, the data word would be initialized with the absolute address of SYMBOL rather than its relative value.

3-3. LINKING AND RELOCATING LOADER .

The LRL is used to link program Segments resulting in relocatable programs. The LRL is a two pass operation. During the second pass, LRL can optionally relocate and load the program for execution or it can produce a relocatable object tape for subsequent loading by a less complex loader.

3-4. SOURCE MAINTENANCE ROUTINE.

The SMR provides the user a means for easy updating and correcting of source programs, either on tape or cards; for building source programs; and for listing of source programs.

3-5. ADDITIONAL SOFTWARE.

In addition to the Programming System, a library of software is available with the TI 960. Included in the library are:

Programming Support Monitor (PSM)

Process Automation Monitor (PAM)

Performance Assurance Tests (PAT) and Debugging Aids

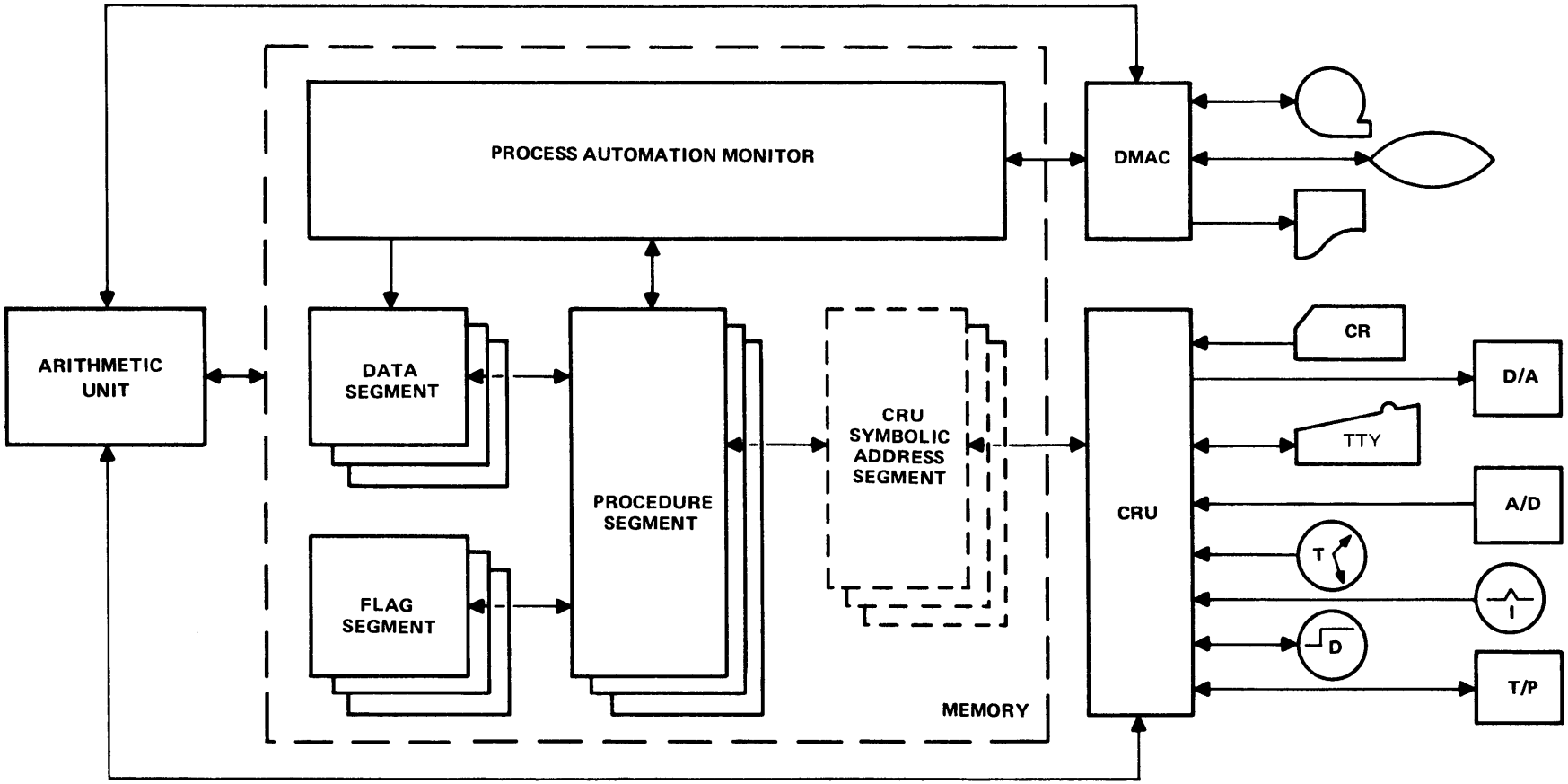


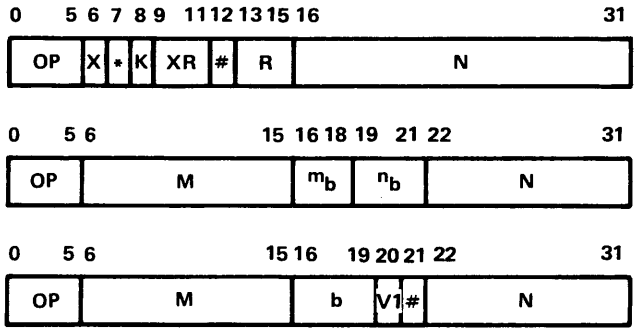
FIGURE 3-1. TYPICAL PROGRAM STRUCTURAL CONFIGURATION, BLOCK DIAGRAM

SECTION IV

MACHINE INSTRUCTIONS

4-1 SCOPE

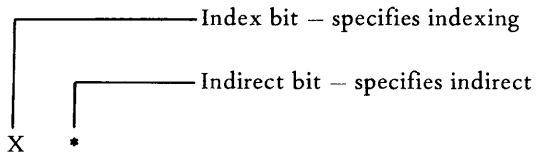
Machine instructions are implemented in three basic formats. Each instruction requires two consecutive memory locations. The formats are defined as follows:



A description of each of the fields is as follows:

- OP The operation code field of an instruction.
- * The bit of the instruction used to specify indirect addressing.
- X The bit of the instruction used to specify that indexing is to be done.
- K Immediate operand indicator.
- XR The field of the instruction used to specify an index register.
- # The field of the instruction used to specify alternate mode registers.
- R The field of the instruction that specifies a register in the register file or specifies a shift count.
- N The field of an instruction used as an address field.
- M The field of an instruction used as an address field in two address instructions.
- M_b The field of instruction used to specify a base register to be used with the M address field.
- N_b The field of an instruction used to specify a base register to be used with the N address field.
- V1 A bit used as an immediate value in flag and bit instructions.
- b The field used to specify a flag address within a memory word or the number of bits in a communication register.

The '*' and 'X' fields, used to specify address modifications, have the following meanings:



X	*	
0	0	Direct operand
0	1	Indirect operand
1	0	Indexed operand
1	1	Indirect, indexed operand

Address modifications, as specified by the X and * fields, are performed using the XR and the N fields. If indirect addressing and indexing are both specified, the indexing is done before indirect address if bit 6 (IM) of the status register is zero. If it is a one, the indexing is done after the indirect addressing. Any of the eight registers of each Mode Register File may be used as an index register.

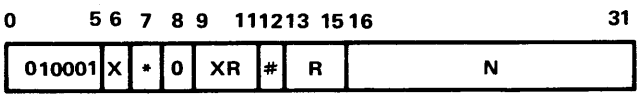
Required entries in instruction operands are underlined.

4-2 LOAD AND STORE OPERATIONS

4-2.1 LOAD REGISTER.

Mnemonic: L

Operand: #R,*@N,XR



The effective operand is loaded into the specified register.

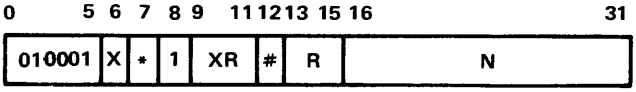
Overflow: No

Mode Switching: No

4-2.2 LOAD REGISTER WITH EFFECTIVE ADDRESS.

Mnemonic: LA

Operand: #R,*@N,XR



The effective address is loaded into the specified register.

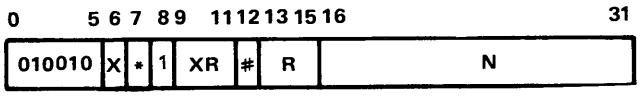
Overflow: No

Mode Switching: No

4-2.3 STORE REGISTER.

Mnemonic: ST

Operand: #R,*@N,XR



The contents of the specified register are stored in the memory location specified by the effective address.

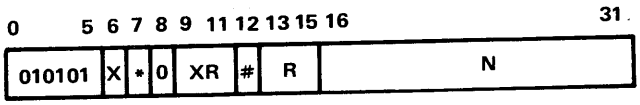
Overflow: No

Mode Switching: No

4-2.4 LOAD ONES TALLY.

Mnemonic: LOT

Operand: #R,*@N,XR



The binary ones in the effective operand are counted. The result is placed in the specified register.

Overflow: No

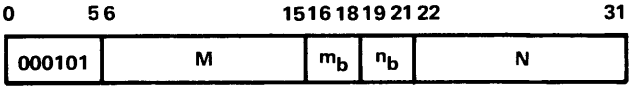
Mode Switching: No

4-2.5 MOVE MEMORY TO MEMORY.

Mnemonic: MOV

Operand: (M,m_b), (N,n_b) Option 1, Explicit base register definition.

NAMEM, NAMEN Option 2, Base register determined by segment class in which symbol is defined.



The contents of the register specified by m_b are added to M to obtain the effective address of operand 1. Likewise the contents of the register specified by n_b are added to N to obtain the effective address of operand 2.

Operand 2 is replaced by operand 1.

Execution in the supervisor mode uses supervisor mode registers for m_b and n_b: worker mode uses worker mode registers.

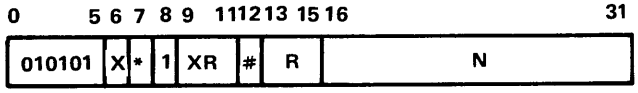
Overflow: No

Mode Switching: No

4-2.6 LOAD ONES TALLY OF EFFECTIVE ADDRESS.

Mnemonic: LOTA

Operand: #R,*@N,XR



The binary ones in the effective address are counted. The result is placed in the specified register.

Overflow: No

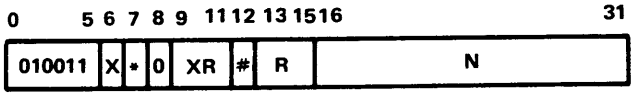
Mode Switching: No

4-3 ARITHMETIC OPERATIONS

4-3.1 ADD TO REGISTER.

Mnemonic: A

Operand: #R,*@N,XR



The effective operand is added to the contents of register R and the result is placed in register R.

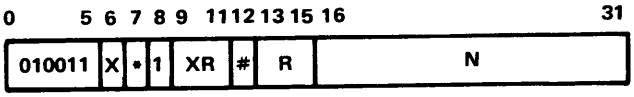
Overflow: Yes

Mode Switching: No

4-3.2 ADD EFFECTIVE ADDRESS TO REGISTER.

Mnemonic: AA

Operand: #R,*@N,XR



The effective address is added to the contents of register R and the result is placed in register R.

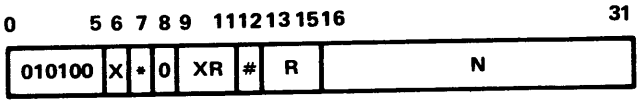
Overflow: Yes

Mode Switching: No

4-3.3 SUBTRACT FROM REGISTER.

Mnemonic: S

Operand: #R,*@N,XR



The effective operand is subtracted from the contents of register R. The result is placed in register R.

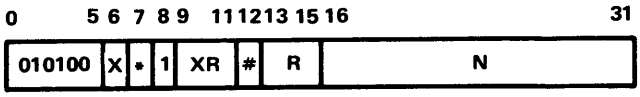
Overflow: Yes

Mode Switching: No

4-3.4 SUBTRACT EFFECTIVE ADDRESS FROM REGISTER.

Mnemonic: SA

Operand: #R,*@N,XR



The effective address is subtracted from the contents of register R. The result is placed in register R.

Overflow: Yes

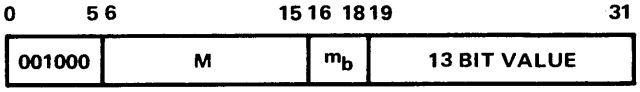
Mode Switching: No

4-3.5 ADD TO MEMORY IMMEDIATE.

Mnemonic: AMI

Operand: (M,m_b), VALUE — Option 1, Explicit base register definition.

NAMEM, VALUE — Option 2, Base register determined by segment class in which symbol is defined.



The 13 bit two's complement value is added to the memory location addressed by M plus the contents of the register specified by m_b. The result is placed in the same memory location.

Overflow: Yes

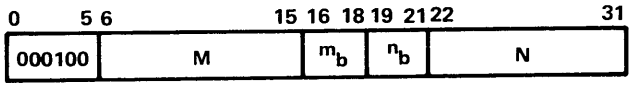
Mode Switching: No

4-3.6 COMPARE MEMORY WITH MEMORY.

Mnemonic: CM

Operand: (M,m_b), N,n_b) — Option 1, Explicit base register definition.

THIS, WITH — Option 2, Base register determined by segment class in which symbol is defined.



The contents of the memory location addressed by M plus the contents of the register specified by m_b are compared arithmetically with the contents of the memory location addressed by N plus the contents of the register specified by n_b.

If the first operand is less than the second, the next instruction in sequence is executed.

If the first operand is greater than the second, one instruction is skipped.

Two instructions are skipped if the operands are equal.

Overflow: No

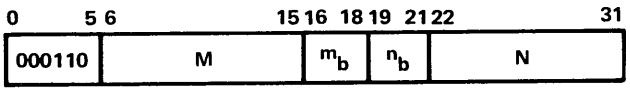
Mode Switching: No

4-3.7 COMPARE MEMORY WITH HIGH AND LOW LIMITS IN MEMORY.

Mnemonic: CML

Operand: (M,m_b), (N,n_b) — Option 1, Explicit base register definition.

NAME, LIMITS — Option 2, Base register determined by segment class in which symbol is defined.



The first operand, addressed by M plus the contents of the register specified by m_b, is compared with a lower and upper limit. The lower limit is addressed by N plus the contents of the register specified by n_b. The upper limit must occupy the next memory location after the lower limit.

If the first operand is less than the lower limit, the next instruction in sequence is executed.

If the first operand is greater than the upper limit, one instruction is skipped.

If the first operand is within the limits or equal to a limit, two instructions are skipped.

Overflow: No

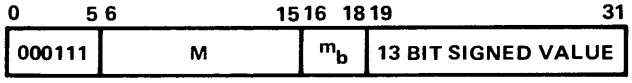
Mode Switching: No

4-3.8 COMPARE MEMORY IMMEDIATE.

Mnemonic: CMI

Operand: (M,m_b), VALUE — Option 1, Explicit base register definition.

NAMEM, VALUE — Option 2, Base register determined by segment class in which symbol is defined.



The contents of the memory location addressed by M plus the contents of the register specified by m_b are compared arithmetically to the 13 bit two's complement value in the instruction.

If the memory operand is less than the immediate value the next instruction in sequence is executed.

One instruction is skipped if the memory operand is greater than the immediate value.

Two instructions are skipped if the memory operand is equal to the immediate value.

Overflow: No

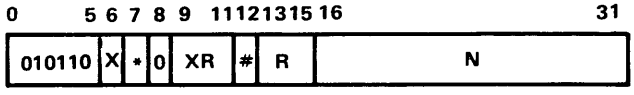
Mode Switching: No

4-4 LOGICAL OPERATIONS

4-4.1 LOGICAL AND.

Mnemonic: N

Operand: #R,*@N,XR



The contents of register R are logically ANDed bit by bit with the effective operand. The results are placed in register R.

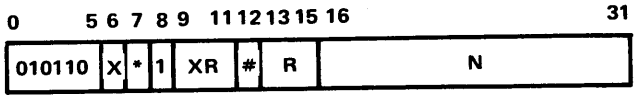
Overflow: No

Mode Switching: No

4-4.2 LOGICAL AND WITH EFFECTIVE ADDRESS.

Mnemonic: NA

Operand: #R,*@N,XR



The contents of register R are logically ANDed bit by bit with the effective address. The results are placed in register R.

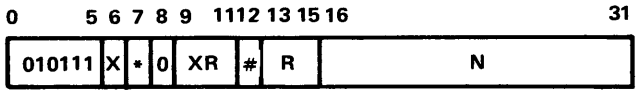
Overflow: No

Mode Switching: No

4-4.3 LOGICAL OR.

Mnemonic: OR

Operand: #R,*@N,XR



The contents of register R are logically ORed bit by bit with the effective operand. The result is placed in register R.

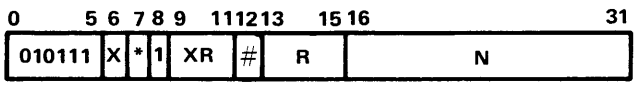
Overflow: No

Mode Switching: No

4-4.4 LOGICAL OR WITH EFFECTIVE ADDRESS.

Mnemonic: ORA

Operand: #R,*@N,XR



The contents of register R are logically ORed bit by bit with the effective address. The results are placed in register R.

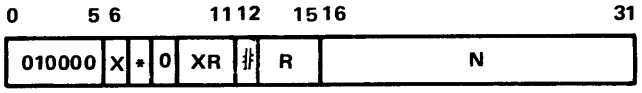
Overflow: No

Mode Switching: No

4-4.5 EXCLUSIVE OR.

Mnemonic: XOR

Operand: R,*@N,XR



The contents of register R are logically exclusive-ORed bitwise with the effective operand. The result is placed in register R. Where bits in register R are equal to bits in the effective operand zeros are placed in register R. Otherwise ones are placed in register R.

Execution Time: 8 microseconds

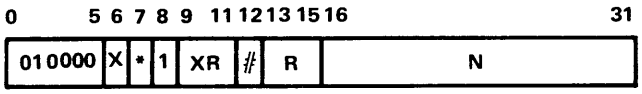
Overflow: No

Mode Switching: No

4-4.6 EXCLUSIVE OR WITH EFFECTIVE ADDRESS.

Mnemonic: XORA

Operand: # R,*@N,XR



The contents of register R are logically exclusive-ORed bitwise with the effective address. The result is placed in register R. Where bits in register R are equal to corresponding bits in the effective address zeros are placed in register R. Otherwise ones are placed in register R.

Execution Time: 6-2/3 microseconds

Overflow: No

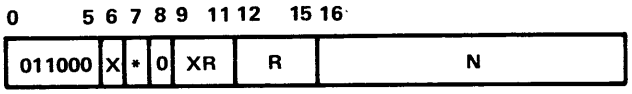
Mode Switching: No

4-5 SHIFT OPERATIONS

4-5.1 SHIFT MEMORY LEFT ARITHMETIC.

Mnemonic: MLA

Operand: R,*@N,XR



The effective operand is shifted left R places and stored in the effective address. R = 0 indicates to a shift of 16 places.

If the sign bit is changed during the shifting, the overflow indicator is set. Zeros fill vacated bit positions.

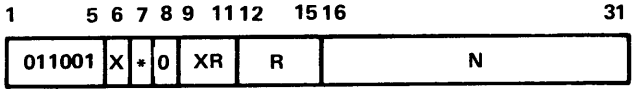
Overflow: Yes

Mode Switching: No

4-5.2 SHIFT MEMORY RIGHT ARITHMETIC.

Mnemonic: MRA

Operand: R,*@N,XR



The effective operand is shifted right R places and stored in the effective address. The sign bit is propagated during the shift.

Sixteen places are shifted if R = 0.

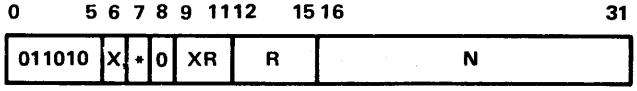
Overflow: No

Mode Switching: No

4-5.3 MEMORY ROTATE RIGHT.

Mnemonic: MRR

Operand: R,*@N,XR



The effective operand is shifted right R places. The bits shifted out of bit position 15 are placed in bit position 0. The result is stored in the effective address. If a shift count of 0 is specified, 16 places are shifted.

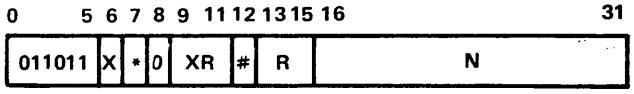
Overflow: No

Mode Switching: No

4-5.4 SHIFT AND ADD TALLY.

Mnemonic: SAT

Operand: #R,*@N,XR



The effective operand is shifted left until bit 0 is a one. If the effective operand is 0, 16 places will be shifted. The count of the number of positions shifted is added to register R. The shifted effective operand is stored in the effective address with bit position 0 forced to a 0.

Overflow: No

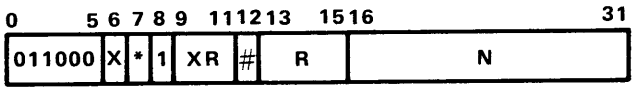
Mode Switching: No

Note: If bit 0 is initially a 1, no shifting is done.

4-5.5 SHIFT MEMORY LEFT ARITHMETIC, COUNT IN REGISTER R.

Mnemonic: MLAX

Operand: #R,*@N,XR



The effective operand is shifted left the number of places specified by the contents of bits 12-15 of register R. The shifted operand is stored in the effective address. If the sign bit is changed during shifting, the overflow indicator is set. Zeros fill vacated bit positions. If a shift count of zero is specified, 16 places are shifted.

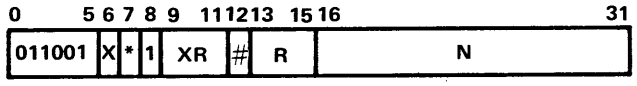
Overflow: Yes

Mode Switching: No

4-5.6 SHIFT MEMORY RIGHT ARITHMETIC, COUNT IN REGISTER R.

Mnemonic: MRAX

Operand: #R,*@N,XR



The effective operand is shifted right the number of places specified by the contents of bits 12-15 of register R. The shifted operand is stored in the effective address The sign bit is propagated during the shift. If a shift count of zero is specified, 16 places are shifted.

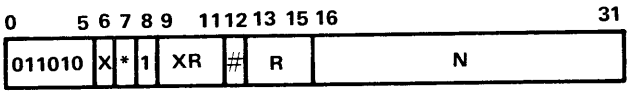
Overflow: No

Mode Switching: No

4-5.7 MEMORY ROTATE RIGHT, COUNT IN REGISTER R.

Mnemonic: MRRX

Operand: #R,*@N,XR



The effective operand is shifted right the number of places specified by the contents of bits 12-15 of register R. Bits shifted out of bit position 15 are entered in bit position 0. The shifted operand is stored in the effective address. If a shift count of zero is specified, 16 places are shifted.

Overflow: No

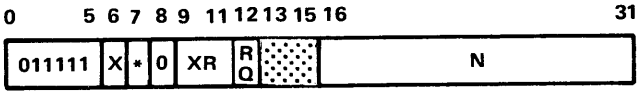
Mode Switching: No

4-6 LOAD-STORE STATUS OPERATIONS

4-6.1 LOAD STATUS BLOCK.

Mnemonic: LDS

Operand: *@N,XR,RQ



The effective operand is placed in the PC or EC and the contents of the effective address plus one are placed in the status register. The operation is performed with the PC if execution is in the supervisor mode and with the EC if execution is in worker mode. If RQ = 1, the contents of register 5 are added to the effective operand before it is placed in the PC or the EC.

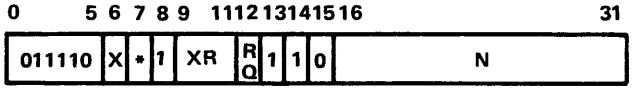
Overflow: Yes

Mode Switching: Yes

4-6.2 STORE STATUS BLOCK.

Mnemonic: SS

Operand: *@N,XR,RQ



The PC or the EC is stored at the effective address. The status register is stored at the next address.

If RQ = 1, the contents of register 5 are subtracted from the PC or EC before it is stored.

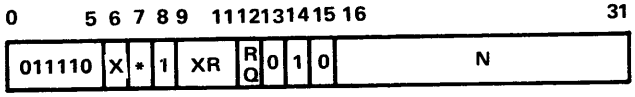
Overflow: No

Mode Switching: No

4-6.3 STORE STATUS BLOCK AND BRANCH.

Mnemonic: SSB

Operand: *@N,XR,RQ



The PC or EC is stored at the effective address. The status register is stored at the next address. The PC or EC is loaded with the contents of the effective address plus 2. If RQ = 1, the contents of register 5 are subtracted from the PC or EC before it is stored and the contents of register 5 are added to the contents of the effective address plus 2 before it is placed in the PC or EC.

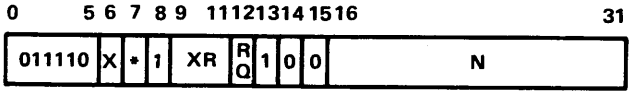
Overflow: No

Mode Switching: No

4-6.4 STORE STATUS BLOCK, TRANSFER TO SUPERVISOR MODE.

Mnemonic: SXS

Operand: *@N,XR,RQ



The PC is stored at the effective address. The status register is stored at the next address and a transfer to supervisor mode is forced. If RQ = 1, the contents of supervisor register 5 are subtracted from the PC before it is stored.

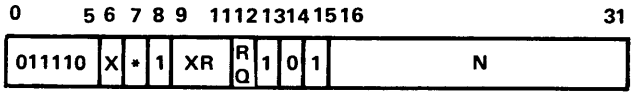
Overflow: No

Mode Switching: Yes

4-6.5 STORE STATUS, TRANSFER TO WORKER MODE.

Mnemonic: SXW

Operand: *@N,XR,RQ



The EC is stored at the effective address. The status register is stored at the next address and a transfer to worker mode is forced. If RQ = 1, the contents of worker mode register 5 are subtracted from the EC before it is stored.

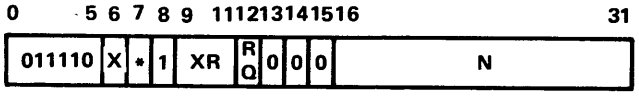
Overflow: No

Mode Switching: Yes

4-6.6 STORE STATUS, TRANSFER AND BRANCH IN SUPERVISOR.

Mnemonic: SXBS

Operand: *@N,XR,RQ



The PC is stored at the effective address and the status register is stored at the next address. A transfer to supervisor mode is forced and the PC is loaded with the contents of the effective address plus 2. If RQ=1, the contents of supervisor mode register 5 are subtracted from the PC before it is stored. The contents of the same register are added to the contents of the effective address plus 2 before it is loaded into the PC.

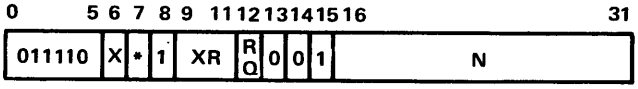
Overflow: No

Mode Switching: Yes

4-6.7 STORE STATUS, TRANSFER AND BRANCH IN WORKER.

Mnemonic: SXBW

Operand: *@N,XR,RQ



The EC is stored at the effective address and the status register is stored at the next address. A transfer to worker mode is forced and the EC is loaded with the contents of the effective address plus 2. If RQ = 1, the contents of worker mode register 5 are subtracted from the EC before it is stored and the contents of the same register are added to the contents of the effective address plus 2 before it is loaded into the EC.

Overflow: No

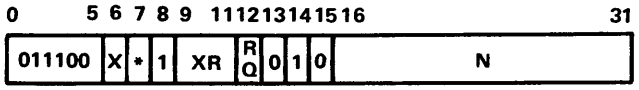
Mode Switching: Yes

4-7 BRANCH OPERATIONS

4-7.1 UNCONDITIONAL BRANCH.

Mnemonic: B

Operand: *@N,XR,RQ



If RQ = 0, the effective address is loaded into either the PC or the EC depending on the execution mode. If RQ = 1, the effective address plus the contents of the execution mode register 5 is loaded into either the PC or EC.

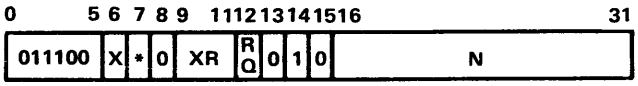
Overflow: No

Mode Switching: No

4-7.2 UNCONDITIONAL BRANCH INDIRECT.

Mnemonic: *B

Operand: *@N,XR,RQ



If RQ = 0, the effective operand is loaded into either the PC or EC depending on the execution mode. If RQ = 1, the effective operand plus the contents of execution mode register 5 is loaded into the PC or the EC.

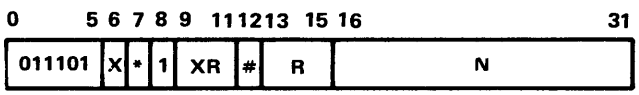
Overflow: No

Mode Switching: No

4-7.3 BRANCH AND LINK TO SUBROUTINE.

Mnemonic: BL

Operand: #R,*@N,XR



The contents of either the PC or the EC, depending on the mode of execution, are placed in register R. The effective address is loaded into either the PC or the EC.

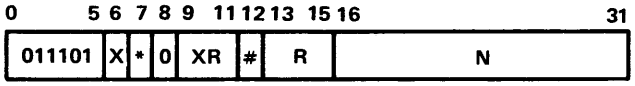
Overflow: No

Mode Switching: No

4-7.4 INDIRECT BRANCH AND LINK TO SUBROUTINE.

Mnemonic: *BL

Operand: #R,*@N,XR



The contents of either the PC or the EC, depending on execution mode, are placed in register R. The effective operand is loaded into either the PC or the EC.

Overflow: No

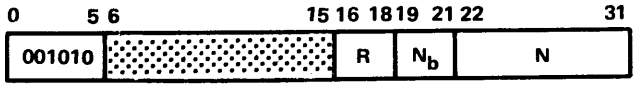
Mode Switching: No

4-7.5 BRANCH RELATIVE TO REGISTER AND LINK TO SUBROUTINE.

Mnemonic: BRRL

Operand: R,(N,n_b) — Option 1, Explicit base register definition.

R, NAMEN — Option 2, Base register determined by segment class in which symbol is defined.



The contents of the register specified by the n_b field are subtracted from the PC or EC, depending on the mode of

execution, and the result is stored in the register specified by the R field. The contents of the register specified by the n_b field are added to the value of the N field. This result is placed in the PC or the EC depending on the execution mode.

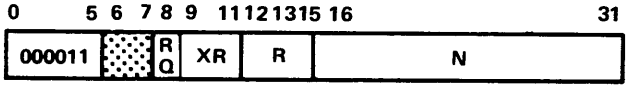
Overflow: No

Mode Switching: No

4-7.6 ADD TO REGISTER AND BRANCH.

Mnemonic: ARB

Operand: R,@N,XR,RQ



The 4 bit. two's complement value in the R field is added to the index register specified in the XR field. The result is placed in the index register. If the sign of the index register changes. the next instruction in sequence is executed. If the sign does not change, the next instruction will be executed at the address specified by N.

If RQ = 1, the effective address is the value from the N field plus the contents of execution mode register 5.

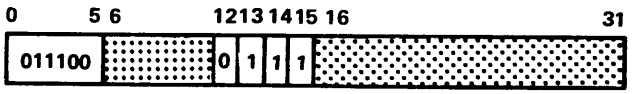
Overflow: No

Mode Switching: No

4-7.7 NO OPERATION.

Mnemonic: NOP

Operand: Not used



The PC or the EC, depending on the mode of execution is incremented by two.

Overflow: No

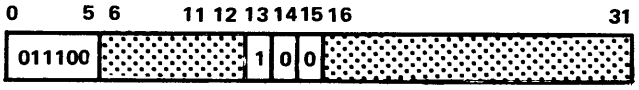
Mode Switching: No

4-8 MODE SWITCHING OPERATIONS

4-8.1 TRANSFER TO SUPERVISOR.

Mnemonic: XS

Operand: Not used



A transfer to supervisor mode is forced.

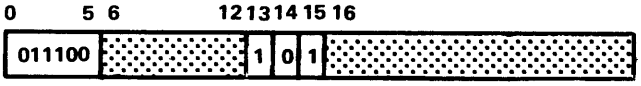
Overflow: No

Mode Switching: Yes

4-8.2 TRANSFER TO WORKER MODE.

Mnemonic: XW

Operand: Not used



A transfer to worker mode is forced.

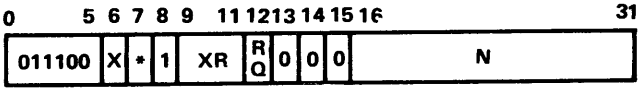
Overflow: No

Mode Switching: Yes

4-8.3 TRANSFER TO SUPERVISOR MODE AND BRANCH.

Mnemonic: XSB

Operand: *@N,XR,RQ



A transfer to supervisor mode is forced and the PC is loaded with the effective address. If RQ = 1, the contents of supervisor mode register 5 are added to the effective address before it is placed in the PC.

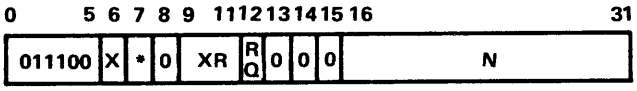
Overflow: No

Mode Switching: Yes

4-8.4 TRANSFER TO SUPERVISOR MODE AND BRANCH INDIRECT.

Mnemonic: *XSB

Operand: *@N,XR,RQ



A transfer to supervisor mode is forced, and the PC is loaded with the effective operand. If RQ = 1, the contents of supervisor mode register 5 are added to the effective operand before it is placed in the PC.

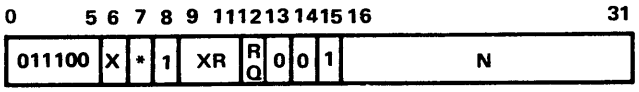
Overflow: No

Mode Switching: Yes

4-8.5 TRANSFER TO WORKER MODE AND BRANCH.

Mnemonic: XWB

Operand: *@N,XR,RQ



A transfer to worker mode is forced, and the EC is loaded with the effective address. If RQ = 1, the contents of worker mode register 5 are added to the effective address before it is placed in the EC.

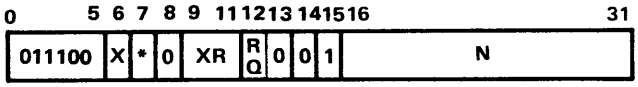
Overflow: No

Mode Switching: Yes

4-8.6 TRANSFER TO WORKER MODE AND BRANCH INDIRECT.

Mnemonic: *XWB

Operand: *@N,XR,RQ



A transfer to worker mode is forced. The EC is loaded with the effective operand. If RQ = 1, the contents of worker mode register 5 are added to the effective operand before it is placed in the EC.

Overflow: No

Mode Switching: Yes

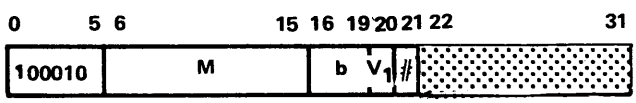
4-9 SOFTWARE FLAG OPERATIONS

4-9.1 SET FLAG

Mnemonic: SETF

Operand: #(M,b),V1 Explicit Definition

#NAMEF, V1 Definition by Name



The contents of the software Flag Base Register are added to M to obtain the effective address. Bit b of the effective operand is set equal to V1.

If the # attribute is used, bit 21 is 1 and the alternate mode Software Flag Base Register is used to calculate the effective address. If bit 21 is 0 the execution mode register is used.

Register 6 is the Software Flag Base Register.

Overflow: No

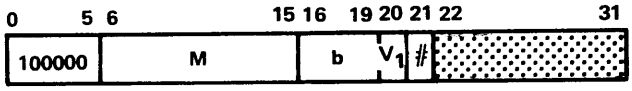
Mode Switching: No

4-9.2 SWITCH MODES IF FLAG NOT EQUAL.

Mnemonic: XFNE

Operand: #(M,b), V1 Explicit Definition

#NAMEF, V1 Definition by Name



The value V1 is compared with bit b of the memory location addressed by M plus the contents of the Software Flag Base Register. If the comparison fails a mode change is forced.

If the # attribute is used, bit 21 is a 1 and the alternate mode Software Flag Base Register is used to calculate the effective address. If bit 21 is a 0, the execution mode register is used.

When XFNE causes a mode change the EC or PC that addressed the instruction is not changed.

Register 6 is the Software Flag Base Register.

Overflow: No

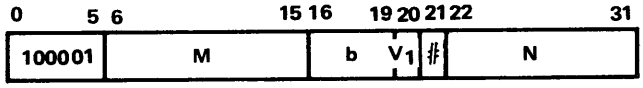
Mode Switching: Yes

4-9.3 BRANCH IF FLAG IS NOT EQUAL.

Mnemonic: BFNE

Operand: #(M,b), V1, N Explicit Definition

#NAMEF, V1, NAMEN Definition by Name



The value V1 is compared with bit b of the memory location addressed by M plus the contents of the Software Flag Base Register. If the comparison fails, the PC or EC, depending on execution mode, is loaded with N plus the contents of the Procedure Base Register.

If the # attribute is used, bit 21 is a 1, and the alternate mode Software Flag Base Register is used to calculate the effective address. If bit 21 is a 0, the execution mode register is used.

Register 6 is the Software Flag Base Register.

Overflow: No

Mode Switching: No

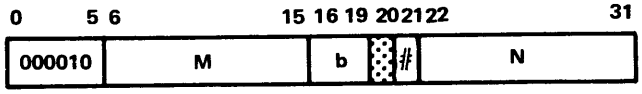
4-10 CRU OPERATIONS

4-10.1 LOAD COMMUNICATION REGISTER.

Mnemonic: LDCR

Operand: #(M,b), N Explicit Definition

#NAMEM, NAMEN Definition by Name



The right justified bit field in the memory location addressed by N plus the contents of the Data Base Register (4) is output to consecutive CRU output lines starting at CRU address M plus the contents of the CRU Base Register (7). The CRU bit addressed by M plus the contents of the CPU Base Register is loaded with the least significant bit of the memory word. The contents of the b field defines the width of the communication register and controls the number of bits sent to the CRU.

b = 0001	register width = 1
b = 0010	register width = 2
.	.
.	.
.	.
b = 1111	register width = 15
b = 0000	register width = 16

If the # attribute is used, bit 21 is a 1, and the alternate mode registers are used.

Overflow: No

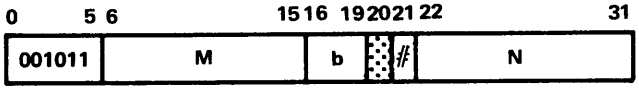
Mode Switching: No

4-10.2 STORE COMMUNICATIONS REGISTER.

Mnemonic: STCR

Operand: #(M,b), N Explicit Definition

#NAMEM,NAMEN Definition by Name



Sequential CRU input lines are read and stored as a right justified bit field in memory location N plus the contents of the Data Base Register. The CRU input line addressed by M plus the contents of the CRU Base Register is stored as the least significant bit of the field. The b field of the instruction word defines the number of bits read and stored in memory. (See LDCR.) The memory word bit positions to the left of the data stored are forced to agree with the most significant input bit position.

If the # attribute is used, bit 21 is a 1, and the alternate mode registers are used to calculate the effective address and the CRU address.

Overflow: No

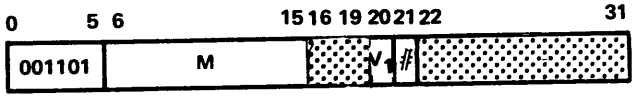
Mode Switching: No

4-10.3 SET CRU OUTPUT BIT.

Mnemonic: SETB

Operand: #M, V1 Explicit Definition

#NAMEM, V1 Definition by Name



The value in the V1 field is used to set the CRU output bit addressed by M plus the contents of the CRU Base Register (7).

The execution mode base register is used unless the # attribute has been invoked; then, bit 21 is a 1, and the alternate mode base register is used to calculate the CRU bit address

Overflow: No

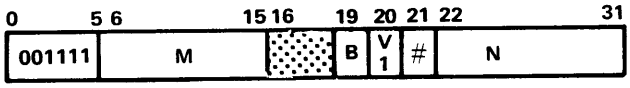
Mode Switching: No

4-10.4 TEST INPUT BIT AND SET OUTPUT BIT OR SWITCH MODES.

Mnemonic: TSBX

Operand: #M, V1, N, B Explicit Definition

#NAMEM, V1, NAMEN, B Definition by Name



The CRU input line addressed by M plus the contents of the CRU Base Register (7) is compared to V1. If the test fails, a mode change is forced; otherwise, the value of bit B is output to the CRU output line addressed by N plus the contents of the CRU Base Register.

The execution mode base register is used unless the # attribute has been invoked; then, bit 21 is a 1, and the alternate mode CRU Base Register is used to calculate the CRU addresses. Also, when bit 21 is a 1, mode switching is inhibited.

When TSBX causes a mode change the PC or EC which addressed the instruction is not changed.

Overflow: No

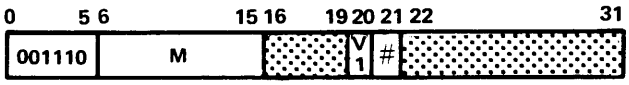
Mode Switching: Yes

4-10.5 SWITCH MODES IF BIT NOT EQUAL.

Mnemonic: XBNE

Operand: #M, V1 Explicit Definition

#NAMEM, V1 Definition by Name



The value V1 is compared with the CRU input line addressed by M plus the contents of the CRU Base Register (7). If the comparison fails a mode change is forced.

The execution mode base register is used unless the # attribute has been invoked; then, bit 21 is a 1, and the alternate mode CRU Base Register is used to calculate the CRU address.

When XBNE causes a mode change, the PC or EC that addressed the instruction is not changed.

Overflow: No

Mode Switching: Yes

4-10.6 BRANCH IF BIT NOT EQUAL.

Mnemonic: BBNE

Operand: #M, V1, N Explicit Definition

#NAMEM, V1, NAMEN Definition by Name

The value V1 is compared with the CRU input line addressed by M plus the contents of the CRU Base Register. If the comparison fails, the PC or EC is loaded with N plus the contents of the Procedure Base Register.

The execution mode base registers are used unless the # attribute has been invoked; then, bit 21 is a 1, and the alternate mode CRU Base Register is used in the CRU address calculation. The branch address is always calculated using the Procedure Base Register.

Overflow: No

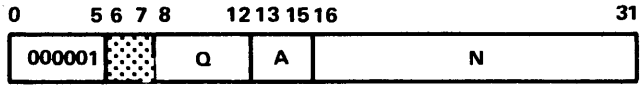
Mode Switching: No

4-11 DIRECT MEMORY ADDRESS INSTRUCTIONS

4-11.1 ACTIVATE DIRECT ACCESS CHANNEL.

Mnemonic: ADAC

Operand: A, N, Q



A command is generated to a device or controller on the direct memory access channel. The A field contains a device or controller address. The N field contains the memory address of an initialization list. The Q field contains information related to a specific device or controller.

SECTION V

INPUT/OUTPUT SYSTEM

5-1. SCOPE.

The computer provides the user with a wide variety of Input/Output operations for testing, monitoring, and controlling discrete events as well as processing operator messages and management reports.

To meet these varied requirements the computer is capable of I/O functions ranging from a single bit to a 16 bit word. This I/O field width flexibility is a unique feature of the Communications Register Unit (CRU), and is utilized for both medium speed message processing and high speed control device-oriented I/O.

High speed word or byte oriented I/O functions are provided with the Direct Memory Access Channel (DMAC). The DMAC is activated under program control and, once started, performs the designated I/O task independently of the program. An interrupt is generated where the I/O task is completed and the status of the task is automatically stored in memory.

The flexibility of the CRU and DMAC is portrayed in Figure 5-1.

5-2. COMMUNICATIONS REGISTER UNIT.

The Communications Register Unit (CRU) is capable of Input/Output with the following standard medium or slow speed devices:

Card Reader – up to 300 cards per minute

Teletype – 10 cps or optionally TI 720 printer 30 cps

Data Set – up to 2400 baud

Paper Tape Reader – up to 300 frames per second

Paper Tape Punch – up to 60 frames per second

In addition, the CRU presents an interface which can accommodate practically any specialized application-oriented device.

All CRU I/O operations are based on the set/reset state of individual I/O lines. Special bit oriented machine instructions are implemented in the computer instruction set to operate the CRU interfaces.

Mnemonic	Operation
LDCR	Load CRU Register (1-16 bits) from memory
STCR	Store CRU Register (1-16 bits) in memory
	Set CRU Output Bit
TSBX	Test Input Bit and Set Output Bit or Switch Mode
XBNE	Switch Mode if Bit not Equal
BBNE	Branch if Bit not Equal

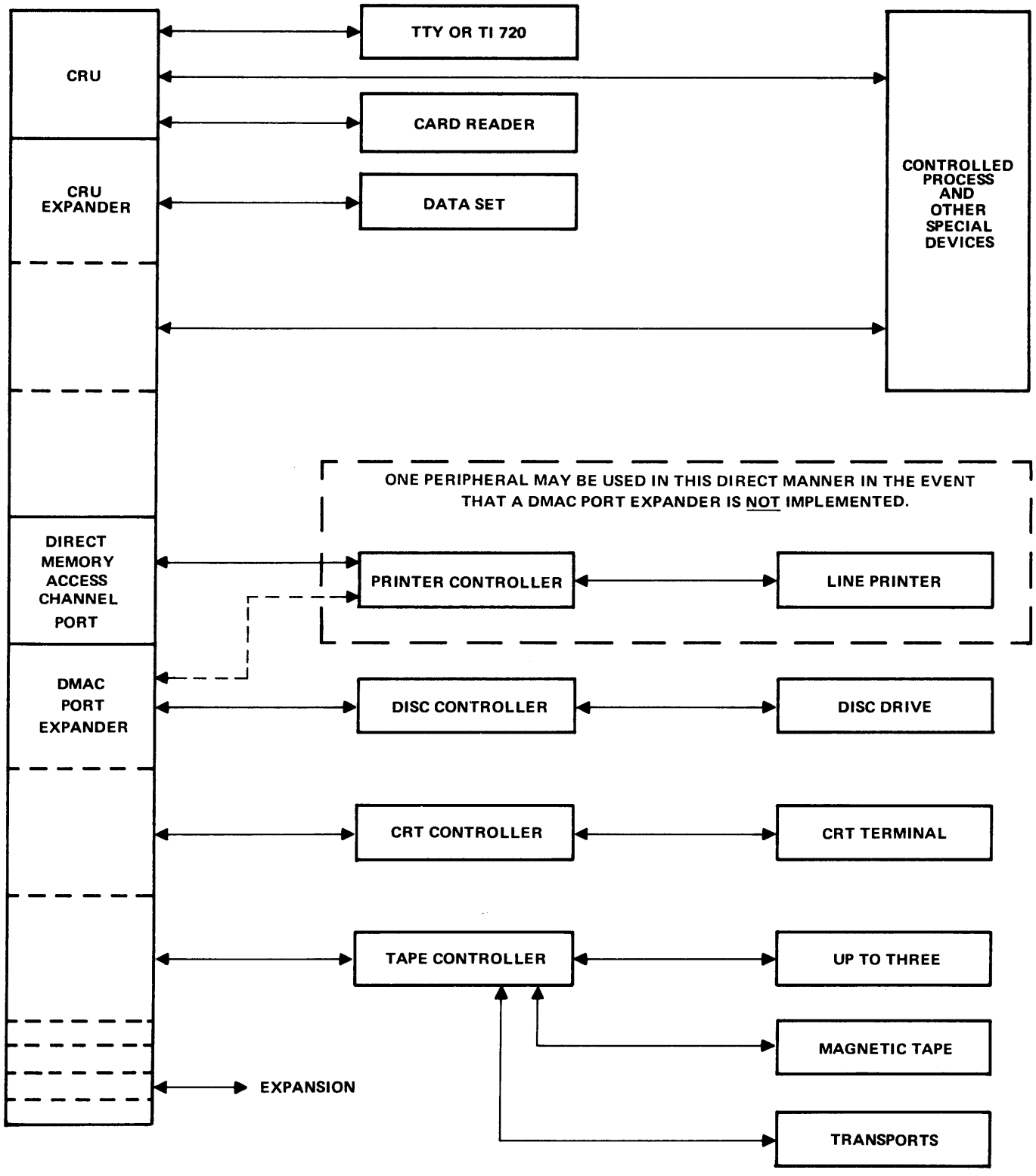
The CRU may be expanded to 4096 I/O lines. Each I/O line can be addressed independently or up to 16 lines can be addressed together as a register. The external function of each group of 16 lines is determined by the type of interface circuit board used. A group of 16 lines may act as a 16 bit data bus, an interval timer, external interrupts, or as address lines and input data for a multiplexer-A/D converter circuit board. Many other functions such as stepping motor control, teletypewriter interface, modem interface, D/A conversion, pulse accumulation, and pulse generation can be performed easily with the CRU I/O interface.

5-3. DIRECT MEMORY ACCESS CHANNEL.

The computer provides high speed Input/Output through the Direct Memory Access Channel (DMAC). A single Direct Memory Access Port is included in the basic DMAC for I/O through one peripheral controller. A Direct Memory Access Port Expander may be added to the DMAC allowing up to eight high speed I/O peripheral devices.

The DMAC is capable of I/O through the following standard option device controllers.

1. Magnetic Tape Transports (3 per controller)
2. Magnetic Disc
3. Line Printer
4. Cathode Ray Tube (CRT) Display Terminal



ANY COMBINATION OF DMAC PERIPHERALS MAY BE ADDED OR SUBSTITUTED SO THAT IT IS POSSIBLE, FOR EXAMPLE, TO USE 24 TAPE TRANSPORTS.

FIGURE 5-1. POSSIBLE 960 I/O CONFIGURATION

All DMAC I/O is accomplished through the single command Activate Direct Memory Access Channel (ADAC) which is described in Section IV.

5-4. DMAC DATA HANDLING.

When transferring data between core memory and a single device the maximum transfer rate is 10^6 words per second.

When memory access is granted to more than one device, none requiring successive 1 us cycles, four system clock cycles must elapse between the servicing of the first device and the granting of access to the second. Maximum data transfer rate under these conditions is 6×10^5 words per second.

The DMAC will service access requests from multiple devices on a priority basis. Priority for data transfers from the devices is selectable by the user. However, an interrupt from any device has priority over access requests for data transfer. The DMAC has priority over the CPU when both require memory access at the same time.

5-5. DMAC/CONTROLLER INTERRUPTS.

The DMAC interrupt is taken whenever the DMAC interrupt line to the CPU is on and Status Register bit 8 is 0.

When interrupt lines from device controllers to the DMAC are turned on, they set bits in the DMAC interrupt status register to indicate the source of the interrupt. One or more bits set in the DMAC interrupt status register will cause the DMAC interrupt line from the DMAC to the CPU to turn on. If the CPU has the DMAC interrupt masked, further interrupts from other device controllers will only cause more bits to be set in the DMAC interrupt status register.

The device controllers keep their interrupt lines on until they receive the interrupt recognized (reset) signal.

As the CPU traps to the DMAC interrupt trap location (X'96'), it turns the interrupt recognized signal to the DMAC on. The DMAC responds with interrupt recognized to each controller that has set an interrupt bit in the DMAC status register.

DMAC and device controller's status words are then stored in dedicated memory. After status words are stored the DMAC trap is executed and control is transferred to the proper interrupt service routine.

DMAC interrupt is masked upon trapping to the trap location so that the interrupt routine is protected. Masking of individual device controller interrupts is controlled by a bit in the device initialization list.

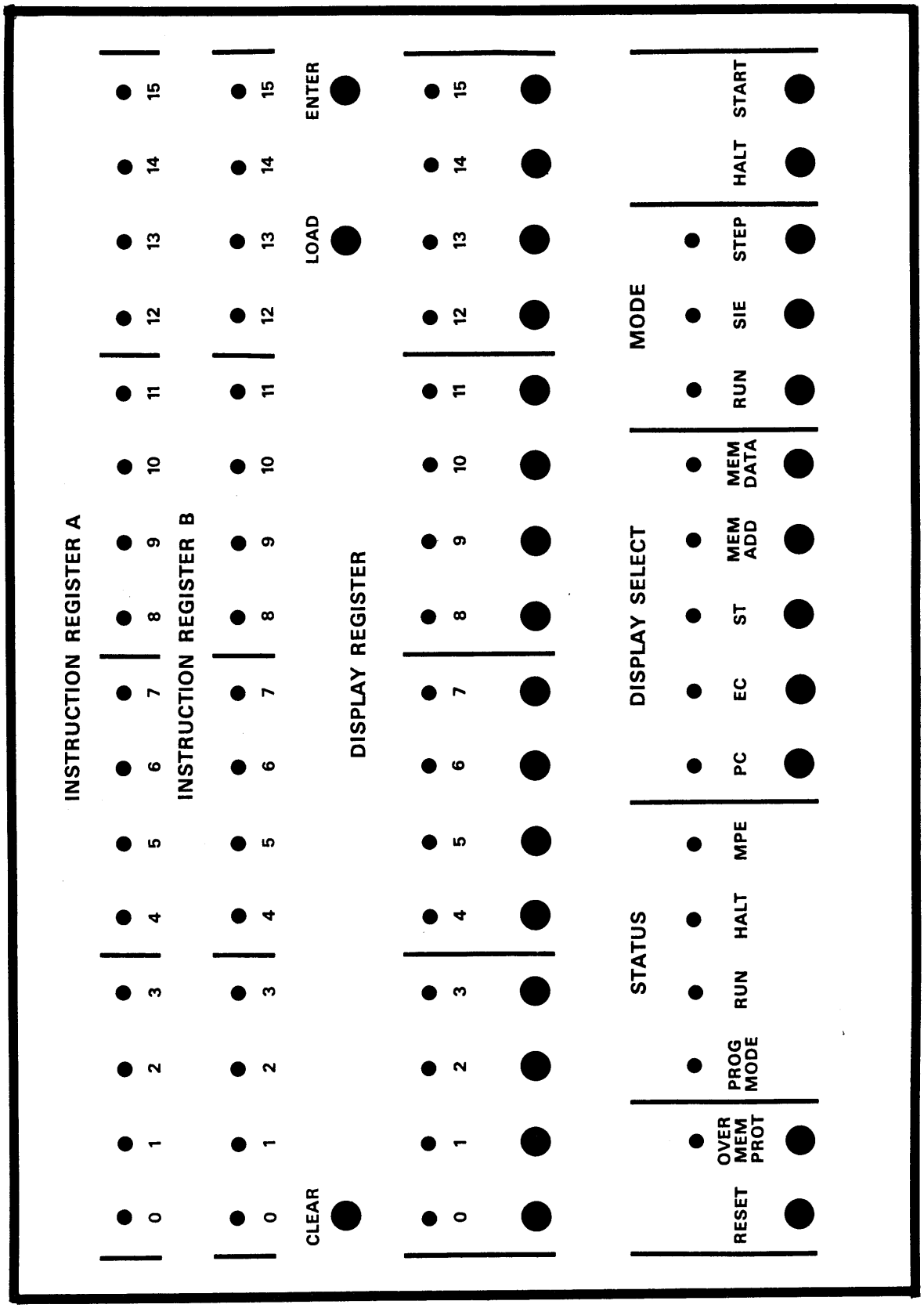


FIGURE 6-1. STANDARD OPERATOR CONSOLE CONTROL PANEL

SECTION VI

STANDARD OPERATOR CONSOLE

Figure 6-1 shows the computer display and control panel. Computer status and control capability is stated below.

INSTRUCTION REGISTER A – the first 16 bits of the instruction word

INSTRUCTION REGISTER B – the second 16 bits of the instruction word

DISPLAY REGISTER – the data selected by the DISPLAY SELECT switches

NOTE

Each bit of the DISPLAY REGISTER can be set by the pushbutton below it. The register can be cleared by depressing the CLEAR pushbutton. The register LOAD pushbutton causes the selected data to be loaded into the display register. The register ENTER pushbutton stores the contents of the register in the register selected by the DISPLAY SELECT switches.

STATUS – pertinent information about computer and program status

PROG MODE – Program Mode

RUN – The computer is running.

HALT – The computer is halted.

MPE – Memory Parity Error

DISPLAY SELECT – data to be used with the DISPLAY REGISTER

PC – Program Counter

EC – Event Counter

ST – Status Register

MEM ADD – Memory Address

MEM DATA – Memory Data

OVERide MEMory PROTect – Permits store cycles to be performed in the protected core area.

Operating MODE

RUN – Permits continuous instruction execution when the START pushbutton is depressed

SIE – Single Instruction Execute; permits only instruction to be executed when the START pushbutton is depressed

STEP – Permits only a single instruction microsequence operation to be executed when the START pushbutton is depressed

RESET – Resets all registers in the computer and halts program execution

HALT – Halts program execution after instruction being executed is finished

START – Depends on operating mode selected

An optional remote console can be plug-connected to the basic CPU. The remote console provides the following capabilities in addition to those of the standard console:

Breakpoint mode

Display of microsequence information

Memory inspect and change features

Display of effective operand

Display of effective address of each instruction

Display of the status register

Display of all 16 virtual registers.

APPENDIX A

MATHEMATICAL TABLES

HEXADECIMAL ARITHMETIC

ADDITION TABLE

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
2	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11
3	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12
4	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
5	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14
6	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
7	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16
8	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17
9	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18
A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19
B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A
C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

MULTIPLICATION TABLE

1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
3	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D
4	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

TABLE OF POWERS OF SIXTEEN$_{10}$

16^n	n	16^{-n}
1	0	0.10000 00000 00000 00000 x 10
16	1	0.62500 00000 00000 00000 x 10^{-1}
256	2	0.39062 50000 00000 00000 x 10^{-2}
4 096	3	0.24414 06250 00000 00000 x 10^{-3}
65 536	4	0.15258 78906 25000 00000 x 10^{-4}
1 048 576	5	0.95367 43164 06250 00000 x 10^{-6}
16 777 216	6	0.59604 64477 53906 25000 x 10^{-7}
268 435 456	7	0.37252 90298 46191 40625 x 10^{-8}
4 294 967 296	8	0.23283 06436 53869 62891 x 10^{-9}
68 719 476 736	9	0.14551 91522 83668 51807 x 10^{-10}
1 099 511 627 776	10	0.90949 47017 72928 23792 x 10^{-12}
17 592 186 044 416	11	0.56843 41886 08080 14870 x 10^{-13}
281 474 976 510 656	12	0.35527 13678 80050 09294 x 10^{-14}
4 503 599 627 370 496	13	0.22204 46049 25031 30808 x 10^{-15}
72 057 594 037 927 936	14	0.13877 78780 78144 56755 x 10^{-16}
1 152 921 504 606 846 976	15	0.86736 17379 88403 54721 x 10^{-18}

TABLE OF POWERS OF TEN$_{16}$

10^n	n	10^{-n}
1	0	1.0000 0000 0000 0000
A	1	0.1999 9999 9999 999A
64	2	0.28F5 C28F 5C28 F5C3 x 16^{-1}
3E8	3	0.4189 374B C6A7 EF9E x 16^{-2}
2710	4	0.68DB 8BAC 710C B296 x 16^{-3}
1 86A0	5	0.A7C5 AC47 1B47 8423 x 16^{-4}
F 4240	6	0.10C6 F7A0 B5ED 8D37 x 16^{-4}
98 9680	7	0.1AD7 F29A BCAF 4858 x 16^{-5}
5F5 E100	8	0.2AF3 1DC4 6118 73BF x 16^{-6}
3B9A CA00	9	0.44B8 2FA0 9B5A 52CC x 16^{-7}
2 540B E400	10	0.6DF3 7F67 5EF6 EADF x 16^{-8}
17 4876 E800	11	0.AFEB FF0B CB24 AAFF x 16^{-9}
E8 D4A5 1000	12	0.1197 9981 2DEA 1119 x 16^{-9}
918 4E72 A000	13	0.1C25 C268 4976 81C2 x 16^{-10}
5AF3 107A 4000	14	0.2D09 370D 4257 3604 x 16^{-11}
3 8D7E A4C6 8000	15	0.480E BE7B 9D58 566D x 16^{-12}
23 86F2 6FC1 0000	16	0.734A CA5F 6226 F0AE x 16^{-13}
163 4578 5D8A 0000	17	0.B877 AA32 36A4 B449 x 16^{-14}
DE0 B6B3 A764 0000	18	0.1272 5DD1 D243 ABA1 x 16^{-14}
8AC7 2304 89E8 0000	19	0.1D83 C94F B6D2 AC35 x 16^{-15}

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

HEXADECIMAL–DECIMAL INTEGER CONVERSION TABLE

The table appearing on the following pages provides a means for direct conversion of decimal integers in the range of 0 to 4095 and for hexadecimal integers in the range of 0 to FFF.

To convert numbers above those ranges, add table values to the figures below:

Hexadecimal	Decimal	Hexadecimal	Decimal
01 000	4 096	20 000	131 072
02 000	8 192	30 000	196 608
03 000	12 288	40 000	262 144
04 000	16 384	50 000	327 680
05 000	20 480	60 000	393 216
06 000	24 576	70 000	458 752
07 000	28 672	80 000	524 288
08 000	32 768	90 000	589 824
09 000	36 864	A0 000	655 360
0A 000	40 960	B0 000	720 896
0B 000	45 056	C0 000	786 432
0C 000	49 152	D0 000	851 968
0D 000	53 248	E0 000	917 504
0E 000	57 344	F0 000	983 040
0F 000	61 440	100 000	1 048 576
10 000	65 536	200 000	2 097 152
11 000	69 632	300 000	3 145 728
12 000	73 728	400 000	4 194 304
13 000	77 824	500 000	5 242 880
14 000	81 920	600 000	6 291 456
15 000	86 016	700 000	7 340 032
16 000	90 112	800 000	8 388 608
17 000	94 208	900 000	9 437 184
18 000	98 304	A00 000	10 485 760
19 000	102 400	B00 000	11 534 336
1A 000	106 496	C00 000	12 582 912
1B 000	110 592	D00 000	13 631 488
1C 000	114 688	E00 000	14 680 064
1D 000	118 784	F00 000	15 728 640
1E 000	122 880	1 000 000	16 777 216
1F 000	126 976	2 000 000	33 554 432

HEXADECIMAL–DECIMAL INTEGER CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
010	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
020	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
030	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
040	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
050	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
060	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
070	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
080	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
090	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A0	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B0	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C0	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D0	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E0	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F0	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255
100	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
110	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
120	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
130	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
140	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
150	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
160	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
170	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
180	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
190	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A0	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B0	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C0	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D0	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E0	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F0	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511
200	0512	0513	0514	0515	0516	0517	0518	0519	0529	0521	0522	0523	0524	0525	0526	0527
210	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
220	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
230	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
240	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
250	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
260	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
270	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
280	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
290	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A0	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B0	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C0	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D0	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E0	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F0	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767

HEXADECIMAL–DECIMAL INTEGER CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
300	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
310	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
320	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
330	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
340	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
350	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
360	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
370	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
380	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
390	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A0	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B0	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C0	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D0	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E0	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F0	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023
400	1024	1025	0126	0127	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
410	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
420	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
430	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
440	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
450	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
460	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
470	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
480	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
490	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A0	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B0	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C0	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D0	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E0	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F0	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
500	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1291	1293	1294	1295
510	1296	1297	1298	1299	1399	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
520	1312	1313	1314	1315	1316	1317	1318	1319	1329	1321	1322	1323	1324	1325	1326	1327
530	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
540	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1367	1358	1359
550	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
560	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
570	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
580	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1429	1421	1422	1423
590	1324	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A0	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
3B0	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C0	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D0	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E0	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F0	1520	1521	1522	1523	1524	1515	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535

HEXADECIMAL–DECIMAL INTEGER CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
600	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
610	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
620	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
630	1584	1585	1586	1587	1588	1589	1590	1591	1592	1592	1594	1595	1596	1597	1598	1599
640	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
650	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
660	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
670	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
680	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
690	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A0	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B0	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	17231	1724	1725	1726	1727
6C0	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D0	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E0	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F0	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791
700	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	8102	1803	1804	1805	1806	1807
710	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
720	1824	1825	1826	1827	1818	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
730	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
740	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
750	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
760	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1909	1902	1903
770	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
780	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
790	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A0	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B0	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C0	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D0	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E0	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F0	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047
800	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
810	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
820	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
830	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
840	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
850	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
860	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
870	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
880	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
890	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A0	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B0	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C0	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D0	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8C0	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F0	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303

HEXADECIMAL–DECIMAL INTEGER CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
900	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
910	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
920	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
930	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
940	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
950	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	3496	2397	2398	2399
960	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
970	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
980	2432	2433	2434	24351	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
990	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A0	2464	2465	2466	2467	2468	2469	2479	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B0	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C0	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D0	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E0	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F0	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559
A00	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A10	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A20	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A30	2608	2609	2610	2611	2612	2613	2614	2615	2626	2617	2618	2619	2620	2621	2622	2623
A40	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A50	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A60	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A70	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A80	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A90	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA0	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
Ab0	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC0	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD0	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE0	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF0	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B00	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B10	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B20	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B30	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B40	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B50	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B60	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B70	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B80	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B90	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA0	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB0	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC0	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD0	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE0	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF0	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071

HEXADECIMAL–DECIMAL INTEGER CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C00	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C10	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C20	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C30	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C40	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C50	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C60	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C70	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C80	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C90	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA0	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB0	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC0	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD0	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE0	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF0	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
D00	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D10	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D20	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D30	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D40	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D50	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D60	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D70	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D80	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D90	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA0	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB0	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC0	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD0	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE0	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF0	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583
E00	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E10	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E20	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E30	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E40	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E50	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E60	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E70	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E80	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E90	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA0	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB0	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775

HEXADECIMAL–DECIMAL INTEGER CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
EC0	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED0	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE0	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF0	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F00	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F10	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F20	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F30	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F40	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F50	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F60	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F70	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F80	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F90	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA0	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB0	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC0	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD0	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE0	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF0	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

HEXADECIMAL–DECIMAL FRACTION CONVERSION TABLE

Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal
.00 00 00 00	.00000 00000	.40 00 00 00	.25000 00000	.80 00 00 00	.50000 00000	.C0 00 00 00	.75000 00000
.01 00 00 00	.00390 62500	.41 00 00 00	.25390 62500	.81 00 00 00	.50390 62500	.C1 00 00 00	.75390 62500
.02 00 00 00	.00781 25000	.42 00 00 00	.25781 25000	.82 00 00 00	.50781 25000	.C2 00 00 00	.75781 25000
.03 00 00 00	.01171 87500	.43 00 00 00	.26171 87500	.83 00 00 00	.51171 87500	.C3 00 00 00	.76171 87500
.04 00 00 00	.01562 50000	.44 00 00 00	.26562 50000	.84 00 00 00	.51562 50000	.C4 00 00 00	.76562 50000
.05 00 00 00	.01953 12500	.45 00 00 00	.26953 12500	.85 00 00 00	.51953 12500	.C5 00 00 00	.76953 12500
.06 00 00 00	.02343 75000	.46 00 00 00	.27343 75000	.86 00 00 00	.52343 75000	.C6 00 00 00	.77343 75000
.07 00 00 00	.02734 37500	.47 00 00 00	.27734 37500	.87 00 00 00	.52734 37500	.C7 00 00 00	.77734 37500
.08 00 00 00	.03125 00000	.48 00 00 00	.28125 00000	.88 00 00 00	.53125 00000	.C8 00 00 00	.78125 00000
.09 00 00 00	.03515 62500	.49 00 00 00	.28515 62500	.89 00 00 00	.53515 62500	.C9 00 00 00	.78515 62500
.0A 00 00 00	.03906 25000	.4A 00 00 00	.28906 25000	.8A 00 00 00	.53906 25000	.CA 00 00 00	.78906 25000
.0B 00 00 00	.04296 87500	.4B 00 00 00	.29296 87500	.8B 00 00 00	.54296 87500	.CB 00 00 00	.79296 87500
.0C 00 00 00	.04687 50000	.4C 00 00 00	.29687 50000	.8C 00 00 00	.54687 50000	.CC 00 00 00	.79687 50000
.0D 00 00 00	.05078 12500	.4D 00 00 00	.30078 12500	.8D 00 00 00	.55078 12500	.CD 00 00 00	.80078 12500
.0E 00 00 00	.05468 75000	.4E 00 00 00	.30468 75000	.8E 00 00 00	.55468 75000	.CE 00 00 00	.80468 75000
.0F 00 00 00	.05859 37500	.4F 00 00 00	.30859 37500	.8F 00 00 00	.55859 37500	.CF 00 00 00	.80859 37500
.10 00 00 00	.06250 00000	.50 00 00 00	.31250 00000	.90 00 00 00	.56250 00000	.D0 00 00 00	.81250 00000
.11 00 00 00	.06640 62500	.51 00 00 00	.31640 62500	.91 00 00 00	.56640 62500	.D1 00 00 00	.81640 62500
.12 00 00 00	.07031 25000	.52 00 00 00	.32031 25000	.92 00 00 00	.57031 25000	.D2 00 00 00	.82031 25000
.13 00 00 00	.07421 87500	.53 00 00 00	.32421 87500	.93 00 00 00	.57421 87500	.D3 00 00 00	.82421 87500
.14 00 00 00	.07812 50000	.54 00 00 00	.32812 50000	.94 00 00 00	.57812 50000	.D4 00 00 00	.82812 50000
.15 00 00 00	.08203 12500	.55 00 00 00	.33203 12500	.95 00 00 00	.58203 12500	.D5 00 00 00	.83203 12500
.16 00 00 00	.08593 75000	.56 00 00 00	.33593 75000	.96 00 00 00	.58593 75000	.D6 00 00 00	.83593 75000
.17 00 00 00	.08984 37500	.57 00 00 00	.33984 37500	.97 00 00 00	.58984 37500	.D7 00 00 00	.83984 37500
.18 00 00 00	.09375 00000	.58 00 00 00	.34375 00000	.98 00 00 00	.59375 00000	.D8 00 00 00	.84375 00000
.19 00 00 00	.09765 62500	.59 00 00 00	.34765 62500	.99 00 00 00	.59765 62500	.D9 00 00 00	.84765 62500
.1A 00 00 00	.10156 25000	.5A 00 00 00	.35156 25000	.9A 00 00 00	.60156 25000	.DA 00 00 00	.85156 25000
.1B 00 00 00	.10546 87500	.5B 00 00 00	.35546 87500	.9B 00 00 00	.60546 87500	.DB 00 00 00	.85546 87500
.1C 00 00 00	.10937 50000	.5C 00 00 00	.35937 50000	.9C 00 00 00	.60937 50000	.DC 00 00 00	.85937 50000
.1D 00 00 00	.11328 12500	.5D 00 00 00	.36328 12500	.9D 00 00 00	.61328 12500	.DD 00 00 00	.86328 12500
.1E 00 00 00	.11718 75000	.5E 00 00 00	.36718 75000	.9E 00 00 00	.61718 75000	.DE 00 00 00	.86718 75000
.1F 00 00 00	.12109 37500	.5F 00 00 00	.37109 37500	.9F 00 00 00	.62109 37500	.DF 00 00 00	.87109 37500
.20 00 00 00	.12500 00000	.60 00 00 00	.37500 00000	.A0 00 00 00	.62500 00000	.E0 00 00 00	.87500 00000
.21 00 00 00	.12890 62500	.61 00 00 00	.37890 62500	.A1 00 00 00	.62890 62500	.E1 00 00 00	.87890 62500
.22 00 00 00	.13281 25000	.62 00 00 00	.38281 25000	.A2 00 00 00	.63281 25000	.E2 00 00 00	.88281 25000
.23 00 00 00	.13671 87500	.63 00 00 00	.38671 87500	.A3 00 00 00	.63671 87500	.E3 00 00 00	.88671 87500
.24 00 00 00	.14062 50000	.64 00 00 00	.39062 50000	.A4 00 00 00	.64062 50000	.E4 00 00 00	.89062 50000
.25 00 00 00	.14453 12500	.65 00 00 00	.39453 12500	.A5 00 00 00	.64453 12500	.E5 00 00 00	.89453 12500
.26 00 00 00	.14843 75000	.66 00 00 00	.39843 75000	.A6 00 00 00	.64843 75000	.E6 00 00 00	.89843 75000
.27 00 00 00	.15234 37500	.67 00 00 00	.40234 37500	.A7 00 00 00	.65234 37500	.E7 00 00 00	.90234 37500
.28 00 00 00	.15625 00000	.68 00 00 00	.40625 00000	.A8 00 00 00	.65625 00000	.E8 00 00 00	.90625 00000
.29 00 00 00	.16015 62500	.69 00 00 00	.41015 62500	.A9 00 00 00	.66015 62500	.E9 00 00 00	.91015 62500
.2A 00 00 00	.16406 25000	.6A 00 00 00	.41406 25000	.AA 00 00 00	.66406 25000	.EA 00 00 00	.91406 25000
.2B 00 00 00	.16796 87500	.6B 00 00 00	.41796 87500	.AB 00 00 00	.66796 87500	.EB 00 00 00	.91796 87500
.2C 00 00 00	.17187 50000	.6C 00 00 00	.42187 50000	.AC 00 00 00	.67187 50000	.EC 00 00 00	.92187 50000
.2D 00 00 00	.17578 12500	.6D 00 00 00	.42578 12500	.AD 00 00 00	.67578 12500	.ED 00 00 00	.92578 12500
.2E 00 00 00	.17968 75000	.6E 00 00 00	.42968 75000	.AE 00 00 00	.67968 75000	.EE 00 00 00	.92968 75000
.2F 00 00 00	.18359 37500	.6F 00 00 00	.43359 37500	.AF 00 00 00	.68359 37500	.EF 00 00 00	.93359 37500
.30 00 00 00	.18750 00000	.70 00 00 00	.43750 00000	.B0 00 00 00	.68750 00000	.F0 00 00 00	.93750 00000
.31 00 00 00	.19140 62500	.71 00 00 00	.44140 62500	.B1 00 00 00	.69140 62500	.F1 00 00 00	.94140 62500
.32 00 00 00	.19531 25000	.72 00 00 00	.44531 25000	.B2 00 00 00	.69531 25000	.F2 00 00 00	.94531 25000
.33 00 00 00	.19921 87500	.73 00 00 00	.44921 87500	.B3 00 00 00	.69921 87500	.F3 00 00 00	.94921 87500
.34 00 00 00	.20312 50000	.74 00 00 00	.45312 50000	.B4 00 00 00	.70312 50000	.F4 00 00 00	.95312 50000
.35 00 00 00	.20703 12500	.75 00 00 00	.45703 12500	.B5 00 00 00	.70703 12500	.F5 00 00 00	.95703 12500
.36 00 00 00	.21093 75000	.76 00 00 00	.46093 75000	.B6 00 00 00	.71093 75000	.F6 00 00 00	.96093 75000
.37 00 00 00	.21484 37500	.77 00 00 00	.46484 37500	.B7 00 00 00	.71484 37500	.F7 00 00 00	.96484 37500
.38 00 00 00	.21875 00000	.78 00 00 00	.46875 00000	.B8 00 00 00	.71875 00000	.F8 00 00 00	.96875 00000
.39 00 00 00	.22265 62500	.79 00 00 00	.47265 62500	.B9 00 00 00	.72265 62500	.F9 00 00 00	.97265 62500
.3A 00 00 00	.22656 25000	.7A 00 00 00	.47656 25000	.BA 00 00 00	.72656 25000	.FA 00 00 00	.97656 25000
.3B 00 00 00	.23046 87500	.7B 00 00 00	.48046 87500	.BB 00 00 00	.73046 87500	.FB 00 00 00	.98046 87500
.3C 00 00 00	.23437 50000	.7C 00 00 00	.48437 50000	.BC 00 00 00	.73437 50000	.FC 00 00 00	.98437 50000
.3D 00 00 00	.23828 12500	.7D 00 00 00	.48828 12500	.BD 00 00 00	.73828 12500	.FD 00 00 00	.98828 12500
.3E 00 00 00	.24218 75000	.7E 00 00 00	.49218 75000	.BE 00 00 00	.74218 75000	.FE 00 00 00	.99218 75000
.3F 00 00 00	.24609 37500	.7F 00 00 00	.49609 37500	.BF 00 00 00	.74609 37500	.FF 00 00 00	.99609 37500

HEXADECIMAL–DECIMAL FRACTION CONVERSION TABLE (cont.)

Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal
.00 00 00 00	.00000 00000	.00 40 00 00	.00097 65625	.00 80 00 00	.00195 31250	.00 C0 00 00	.00292 96875
.00 01 00 00	.00001 52587	.00 41 00 00	.00099 18212	.00 81 00 00	.00196 83837	.00 C1 00 00	.00294 49462
.00 02 00 00	.00003 05175	.00 42 00 00	.00100 70800	.00 82 00 00	.00198 36425	.00 C2 00 00	.00296 02050
.00 03 00 00	.00004 57763	.00 43 00 00	.00102 23388	.00 83 00 00	.00199 89013	.00 C3 00 00	.00297 54638
.00 04 00 00	.00006 10351	.00 44 00 00	.00103 75976	.00 84 00 00	.00201 41601	.00 C4 00 00	.00299 07226
.00 05 00 00	.00007 62939	.00 45 00 00	.00105 28564	.00 85 00 00	.00202 94189	.00 C5 00 00	.00300 59814
.00 06 00 00	.00009 15527	.00 46 00 00	.00106 81152	.00 86 00 00	.00204 46777	.00 C6 00 00	.00302 12402
.00 07 00 00	.00010 68115	.00 47 00 00	.00108 33740	.00 87 00 00	.00205 99365	.00 C7 00 00	.00303 64990
.00 08 00 00	.00012 20703	.00 48 00 00	.00109 86328	.00 88 00 00	.00207 51953	.00 C8 00 00	.00305 17578
.00 09 00 00	.00013 73291	.00 49 00 00	.00111 38916	.00 89 00 00	.00209 04541	.00 C9 00 00	.00306 70166
.00 0A 00 00	.00015 25878	.00 4A 00 00	.00112 91503	.00 8A 00 00	.00210 57128	.00 CA 00 00	.00308 22753
.00 0B 00 00	.00016 78466	.00 4B 00 00	.00114 44091	.00 8B 00 00	.00212 09716	.00 CB 00 00	.00309 75341
.00 0C 00 00	.00018 31054	.00 4C 00 00	.00115 96679	.00 8C 00 00	.00213 62304	.00 CC 00 00	.00311 27929
.00 0D 00 00	.00019 83642	.00 4D 00 00	.00117 49267	.00 8D 00 00	.00215 14892	.00 CD 00 00	.00312 80517
.00 0E 00 00	.00021 36230	.00 4E 00 00	.00119 01855	.00 8E 00 00	.00216 67480	.00 CE 00 00	.00314 33105
.00 0F 00 00	.00022 88818	.00 4F 00 00	.00120 54443	.00 8F 00 00	.00218 20068	.00 CF 00 00	.00315 85693
.00 10 00 00	.00024 41406	.00 50 00 00	.00122 07031	.00 90 00 00	.00219 72656	.00 D0 00 00	.00317 38281
.00 11 00 00	.00025 93994	.00 51 00 00	.00123 59619	.00 91 00 00	.00221 25244	.00 D1 00 00	.00318 90869
.00 12 00 00	.00027 46582	.00 52 00 00	.00125 12207	.00 92 00 00	.00222 77832	.00 D2 00 00	.00320 43457
.00 13 00 00	.00028 99169	.00 53 00 00	.00126 64794	.00 93 00 00	.00224 30419	.00 D3 00 00	.00321 96044
.00 14 00 00	.00030 51757	.00 54 00 00	.00128 17382	.00 94 00 00	.00225 83007	.00 D4 00 00	.00323 48632
.00 15 00 00	.00032 04345	.00 55 00 00	.00129 69970	.00 95 00 00	.00227 35595	.00 D5 00 00	.00325 01220
.00 16 00 00	.00033 56933	.00 56 00 00	.00131 22558	.00 96 00 00	.00228 88183	.00 D6 00 00	.00326 53808
.00 17 00 00	.00035 09521	.00 57 00 00	.00132 75146	.00 97 00 00	.00230 40771	.00 D7 00 00	.00328 06396
.00 18 00 00	.00036 62109	.00 58 00 00	.00134 27734	.00 98 00 00	.00231 93359	.00 D8 00 00	.00329 58984
.00 19 00 00	.00038 14697	.00 59 00 00	.00135 80322	.00 99 00 00	.00233 45947	.00 D9 00 00	.00331 11572
.00 1A 00 00	.00039 67285	.00 5A 00 00	.00137 32910	.00 9A 00 00	.00234 98535	.00 DA 00 00	.00332 64160
.00 1B 00 00	.00041 19873	.00 5B 00 00	.00138 85498	.00 9B 00 00	.00236 51123	.00 DB 00 00	.00334 16748
.00 1C 00 00	.00042 72460	.00 5C 00 00	.00140 38085	.00 9C 00 00	.00238 03710	.00 DC 00 00	.00335 69335
.00 1D 00 00	.00044 25048	.00 5D 00 00	.00141 90673	.00 9D 00 00	.00239 56298	.00 DD 00 00	.00337 21923
.00 1E 00 00	.00045 77636	.00 5E 00 00	.00143 43261	.00 9E 00 00	.00241 08886	.00 DE 00 00	.00338 74511
.00 1F 00 00	.00047 30224	.00 5F 00 00	.00144 95849	.00 9F 00 00	.00242 61474	.00 DF 00 00	.00340 27099
.00 20 00 00	.00048 82812	.00 60 00 00	.00146 48437	.00 A0 00 00	.00244 14062	.00 E0 00 00	.00341 79687
.00 21 00 00	.00050 35400	.00 61 00 00	.00148 01025	.00 A1 00 00	.00245 66650	.00 E1 00 00	.00343 32275
.00 22 00 00	.00051 87988	.00 62 00 00	.00149 53613	.00 A2 00 00	.00247 19238	.00 E2 00 00	.00344 84863
.00 23 00 00	.00053 40576	.00 63 00 00	.00151 06201	.00 A3 00 00	.00248 71826	.00 E3 00 00	.00346 37451
.00 24 00 00	.00054 93164	.00 64 00 00	.00152 58789	.00 A4 00 00	.00250 24414	.00 E4 00 00	.00347 90039
.00 25 00 00	.00056 45751	.00 65 00 00	.00154 11376	.00 A5 00 00	.00251 77001	.00 E5 00 00	.00349 42626
.00 26 00 00	.00057 98339	.00 66 00 00	.00155 63964	.00 A6 00 00	.00253 29589	.00 E6 00 00	.00350 95214
.00 27 00 00	.00059 50927	.00 67 00 00	.00157 16552	.00 A7 00 00	.00254 82177	.00 E7 00 00	.00352 47802
.00 28 00 00	.00061 03515	.00 68 00 00	.00158 69140	.00 A8 00 00	.00256 34765	.00 E8 00 00	.00354 00390
.00 29 00 00	.00062 56103	.00 69 00 00	.00160 21728	.00 A9 00 00	.00257 87353	.00 E9 00 00	.00355 52978
.00 2A 00 00	.00064 08691	.00 6A 00 00	.00161 74316	.00 AA 00 00	.00259 39941	.00 EA 00 00	.00357 05566
.00 2B 00 00	.00065 61279	.00 6B 00 00	.00163 26904	.00 AB 00 00	.00260 92529	.00 EB 00 00	.00358 58154
.00 2C 00 00	.00067 13867	.00 6C 00 00	.00164 79492	.00 AC 00 00	.00262 45117	.00 EC 00 00	.00360 10742
.00 2D 00 00	.00068 66455	.00 6D 00 00	.00166 32080	.00 AD 00 00	.00263 97705	.00 ED 00 00	.00361 63330
.00 2E 00 00	.00070 19042	.00 6E 00 00	.00167 84667	.00 AE 00 00	.00265 50292	.00 EE 00 00	.00363 15917
.00 2F 00 00	.00071 71630	.00 6F 00 00	.00169 37255	.00 AF 00 00	.00267 02880	.00 EF 00 00	.00364 68505
.00 30 00 00	.00073 24218	.00 70 00 00	.00170 89843	.00 B0 00 00	.00268 55468	.00 F0 00 00	.00366 21093
.00 31 00 00	.00074 76806	.00 71 00 00	.00172 42421	.00 B1 00 00	.00270 08056	.00 F1 00 00	.00367 73681
.00 32 00 00	.00076 29394	.00 72 00 00	.00173 95019	.00 B2 00 00	.00271 60644	.00 F2 00 00	.00369 26269
.00 33 00 00	.00077 81982	.00 73 00 00	.00175 47607	.00 B3 00 00	.00273 13232	.00 F3 00 00	.00370 78857
.00 34 00 00	.00079 34570	.00 74 00 00	.00177 00195	.00 B4 00 00	.00274 65820	.00 F4 00 00	.00372 31445
.00 35 00 00	.00080 87158	.00 75 00 00	.00178 52783	.00 B5 00 00	.00276 18408	.00 F5 00 00	.00373 84033
.00 36 00 00	.00082 39746	.00 76 00 00	.00180 05371	.00 B6 00 00	.00277 70996	.00 F6 00 00	.00375 36621
.00 37 00 00	.00083 92333	.00 77 00 00	.00181 57958	.00 B7 00 00	.00279 23583	.00 F7 00 00	.00376 89208
.00 38 00 00	.00085 44921	.00 78 00 00	.00183 10546	.00 B8 00 00	.00280 76171	.00 F8 00 00	.00378 41796
.00 39 00 00	.00086 97509	.00 79 00 00	.00184 63134	.00 B9 00 00	.00282 28759	.00 F9 00 00	.00379 94384
.00 3A 00 00	.00088 50097	.00 7A 00 00	.00186 15722	.00 BA 00 00	.00283 81347	.00 FA 00 00	.00381 46972
.00 38 00 00	.00090 02685	.00 7B 00 00	.00187 68310	.00 BB 00 00	.00285 33935	.00 FB 00 00	.00382 99560
.00 3C 00 00	.00091 55273	.00 7C 00 00	.00189 20898	.00 BC 00 00	.00286 86523	.00 FC 00 00	.00384 52148
.00 3D 00 00	.00093 07861	.00 7D 00 00	.00190 73486	.00 BD 00 00	.00288 39111	.00 FD 00 00	.00386 04736
.00 3E 00 00	.00094 60449	.00 7E 00 00	.00192 26074	.00 BE 00 00	.00289 91699	.00 FE 00 00	.00387 57324
.00 3F 00 00	.00096 13037	.00 7F 00 00	.00193 78662	.00 BF 00 00	.00291 44287	.00 FF 00 00	.00389 09912

HEXADECIMAL–DECIMAL FRACTION CONVERSION TABLE (cont.)

Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal
.00 00 00 00	.00000 00000	.00 00 40 00	.00000 38146	.00 00 80 00	.00000 76293	.00 00 C0 00	.00001 14440
.00 00 01 00	.00000 00596	.00 00 41 00	.00000 38743	.00 00 81 00	.00000 76889	.00 00 C1 00	.00001 15036
.00 00 02 00	.00000 01192	.00 00 42 00	.00000 39339	.00 00 82 00	.00000 77486	.00 00 C2 00	.00001 15633
.00 00 03 00	.00000 01788	.00 00 43 00	.00000 39935	.00 00 83 00	.00000 78082	.00 00 C3 00	.00001 16229
.00 00 04 00	.00000 02384	.00 00 44 00	.00000 40531	.00 00 84 00	.00000 78678	.00 00 C4 00	.00001 16825
.00 00 05 00	.00000 02980	.00 00 45 00	.00000 41127	.00 00 85 00	.00000 79274	.00 00 C5 00	.00001 17421
.00 00 06 00	.00000 03576	.00 00 46 00	.00000 41723	.00 00 86 00	.00000 79870	.00 00 C6 00	.00001 18017
.00 00 07 00	.00000 04172	.00 00 47 00	.00000 42319	.00 00 87 00	.00000 80466	.00 00 C7 00	.00001 18613
.00 00 08 00	.00000 04768	.00 00 48 00	.00000 42915	.00 00 88 00	.00000 81062	.00 00 C8 00	.00001 19209
.00 00 09 00	.00000 05364	.00 00 49 00	.00000 43511	.00 00 89 00	.00000 81658	.00 00 C9 00	.00001 19805
.00 00 0A 00	.00000 05960	.00 00 4A 00	.00000 44107	.00 00 8A 00	.00000 82254	.00 00 CA 00	.00001 20401
.00 00 0B 00	.00000 06556	.00 00 4B 00	.00000 44703	.00 00 8B 00	.00000 82850	.00 00 CB 00	.00001 20997
.00 00 0C 00	.00000 07152	.00 00 4C 00	.00000 45299	.00 00 8C 00	.00000 83446	.00 00 CC 00	.00001 21593
.00 00 0D 00	.00000 07748	.00 00 4D 00	.00000 45895	.00 00 8D 00	.00000 84042	.00 00 CD 00	.00001 22189
.00 00 0E 00	.00000 08344	.00 00 4E 00	.00000 46491	.00 00 8E 00	.00000 84638	.00 00 CE 00	.00001 22785
.00 00 0F 00	.00000 08940	.00 00 4F 00	.00000 47087	.00 00 8F 00	.00000 85234	.00 00 CF 00	.00001 23381
.00 00 10 00	.00000 09536	.00 00 50 00	.00000 47683	.00 00 90 00	.00000 85830	.00 00 D0 00	.00001 23977
.00 00 11 00	.00000 10132	.00 00 51 00	.00000 48279	.00 00 91 00	.00000 86426	.00 00 D1 00	.00001 24573
.00 00 12 00	.00000 10728	.00 00 52 00	.00000 48875	.00 00 92 00	.00000 87022	.00 00 D2 00	.00001 25169
.00 00 13 00	.00000 11324	.00 00 53 00	.00000 49471	.00 00 93 00	.00000 87618	.00 00 D3 00	.00001 25765
.00 00 14 00	.00000 11920	.00 00 54 00	.00000 50067	.00 00 94 00	.00000 88214	.00 00 D4 00	.00001 26361
.00 00 15 00	.00000 12516	.00 00 55 00	.00000 50663	.00 00 95 00	.00000 88810	.00 00 D5 00	.00001 26957
.00 00 16 00	.00000 13113	.00 00 56 00	.00000 51259	.00 00 96 00	.00000 89406	.00 00 D6 00	.00001 27553
.00 00 17 00	.00000 13709	.00 00 57 00	.00000 51856	.00 00 97 00	.00000 90003	.00 00 D7 00	.00001 28149
.00 00 18 00	.00000 14305	.00 00 58 00	.00000 52452	.00 00 98 00	.00000 90599	.00 00 D8 00	.00001 28746
.00 00 19 00	.00000 14901	.00 00 59 00	.00000 53048	.00 00 99 00	.00000 91195	.00 00 D9 00	.00001 29342
.00 00 1A 00	.00000 15497	.00 00 5A 00	.00000 53644	.00 00 9A 00	.00000 91791	.00 00 DA 00	.00001 29938
.00 00 1B 00	.00000 16093	.00 00 5B 00	.00000 54240	.00 00 9B 00	.00000 92387	.00 00 DB 00	.00001 30534
.00 00 1C 00	.00000 16689	.00 00 5C 00	.00000 54836	.00 00 9C 00	.00000 92983	.00 00 DC 00	.00001 31130
.00 00 1D 00	.00000 17285	.00 00 5D 00	.00000 55432	.00 00 9D 00	.00000 93579	.00 00 DD 00	.00001 31726
.00 00 1E 00	.00000 17881	.00 00 5E 00	.00000 56028	.00 00 9E 00	.00000 94175	.00 00 DE 00	.00001 32322
.00 00 1F 00	.00000 18477	.00 00 5F 00	.00000 56624	.00 00 9F 00	.00000 94771	.00 00 DF 00	.00001 32918
.00 00 20 00	.00000 19073	.00 00 60 00	.00000 57220	.00 00 A0 00	.00000 95367	.00 00 E0 00	.00001 33514
.00 00 21 00	.00000 19669	.00 00 61 00	.00000 57816	.00 00 A1 00	.00000 95963	.00 00 E1 00	.00001 34110
.00 00 22 00	.00000 20265	.00 00 62 00	.00000 58412	.00 00 A2 00	.00000 96559	.00 00 E2 00	.00001 34706
.00 00 23 00	.00000 20861	.00 00 63 00	.00000 59008	.00 00 A3 00	.00000 97155	.00 00 E3 00	.00001 35302
.00 00 24 00	.00000 21457	.00 00 64 00	.00000 59604	.00 00 A4 00	.00000 97751	.00 00 E4 00	.00001 35898
.00 00 25 00	.00000 22053	.00 00 65 00	.00000 60200	.00 00 A5 00	.00000 98347	.00 00 E5 00	.00001 36494
.00 00 26 00	.00000 22649	.00 00 66 00	.00000 60796	.00 00 A6 00	.00000 98943	.00 00 E6 00	.00001 37090
.00 00 27 00	.00000 23245	.00 00 67 00	.00000 61392	.00 00 A7 00	.00000 99539	.00 00 E7 00	.00001 37686
.00 00 28 00	.00000 23841	.00 00 68 00	.00000 61988	.00 00 A8 00	.00001 00135	.00 00 E8 00	.00001 38282
.00 00 29 00	.00000 24437	.00 00 69 00	.00000 62584	.00 00 A9 00	.00001 00731	.00 00 E9 00	.00001 38878
.00 00 2A 00	.00000 25033	.00 00 6A 00	.00000 63180	.00 00 AA 00	.00001 01327	.00 00 EA 00	.00001 39474
.00 00 2B 00	.00000 25629	.00 00 6B 00	.00000 63776	.00 00 AB 00	.00001 01923	.00 00 EB 00	.00001 40070
.00 00 2C 00	.00000 26226	.00 00 6C 00	.00000 64373	.00 00 AC 00	.00001 02519	.00 00 EC 00	.00001 40666
.00 00 2D 00	.00000 26822	.00 00 6D 00	.00000 64969	.00 00 AD 00	.00001 03116	.00 00 ED 00	.00001 41263
.00 00 2E 00	.00000 27418	.00 00 6E 00	.00000 65565	.00 00 AE 00	.00001 03712	.00 00 EE 00	.00001 41859
.00 00 2F 00	.00000 28014	.00 00 6F 00	.00000 61661	.00 00 AF 00	.00001 04308	.00 00 EF 00	.00001 42455
.00 00 30 00	.00000 28610	.00 00 70 00	.00000 66757	.00 00 B0 00	.00001 04904	.00 00 F0 00	.00001 43051
.00 00 31 00	.00000 29206	.00 00 71 00	.00000 67353	.00 00 B1 00	.00001 05500	.00 00 F1 00	.00001 43647
.00 00 32 00	.00000 29802	.00 00 72 00	.00000 67949	.00 00 B2 00	.00001 06096	.00 00 F2 00	.00001 44243
.00 00 33 00	.00000 30398	.00 00 73 00	.00000 68545	.00 00 B3 00	.00001 06692	.00 00 F3 00	.00001 44839
.00 00 34 00	.00000 30994	.00 00 74 00	.00000 69141	.00 00 B4 00	.00001 07228	.00 00 F4 00	.00001 45435
.00 00 35 00	.00000 31590	.00 00 75 00	.00000 69737	.00 00 B5 00	.00001 07884	.00 00 F5 00	.00001 46031
.00 00 36 00	.00000 32186	.00 00 76 00	.00000 70333	.00 00 B6 00	.00001 08480	.00 00 F6 00	.00001 46627
.00 00 37 00	.00000 32782	.00 00 77 00	.00000 70929	.00 00 B7 00	.00001 09076	.00 00 F7 00	.00001 47223
.00 00 38 00	.00000 33378	.00 00 78 00	.00000 71525	.00 00 B8 00	.00001 09672	.00 00 F8 00	.00001 47819
.00 00 39 00	.00000 33974	.00 00 79 00	.00000 75121	.00 00 B9 00	.00001 10268	.00 00 F9 00	.00001 48415
.00 00 3A 00	.00000 34570	.00 00 7A 00	.00000 72717	.00 00 BA 00	.00001 10864	.00 00 FA 00	.00001 49011
.00 00 3B 00	.00000 35166	.00 00 7B 00	.00000 73313	.00 00 BB 00	.00001 11460	.00 00 FB 00	.00001 49607
.00 00 3C 00	.00000 35762	.00 00 7C 00	.00000 73909	.00 00 BC 00	.00001 12056	.00 00 FC 00	.00001 50203
.00 00 3D 00	.00000 36358	.00 00 7D 00	.00000 74505	.00 00 BD 00	.00001 12652	.00 00 ED 00	.00001 50799
.00 00 3E 00	.00000 36954	.00 00 7E 00	.00000 75101	.00 00 BE 00	.00001 13248	.00 00 FE 00	.00001 51395
.00 00 3F 00	.00000 37550	.00 00 7F 00	.00000 75697	.00 00 BF 00	.00001 13844	.00 00 FF 00	.00001 51991

HEXADECIMAL–DECIMAL FRACTION CONVERSION TABLE

Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal
.00 00 00 00	.00000 00000	.00 00 00 40	.00000 00149	.00 00 00 80	.00000 00298	.00 00 00 C0	.00000 00447
.00 00 00 01	.00000 00002	.00 00 00 41	.00000 00151	.00 00 00 81	.00000 00300	.00 00 00 C1	.00000 00449
.00 00 00 02	.00000 00004	.00 00 00 42	.00000 00153	.00 00 00 82	.00000 00302	.00 00 00 C2	.00000 00451
.00 00 00 03	.00000 00006	.00 00 00 43	.00000 00155	.00 00 00 83	.00000 00305	.00 00 00 C3	.00000 00454
.00 00 00 04	.00000 00009	.00 00 00 44	.00000 00158	.00 00 00 84	.00000 00307	.00 00 00 C4	.00000 00456
.00 00 00 05	.00000 00011	.00 00 00 45	.00000 00160	.00 00 00 85	.00000 00309	.00 00 00 C5	.00000 00458
.00 00 00 06	.00000 00013	.00 00 00 46	.00000 00162	.00 00 00 86	.00000 00311	.00 00 00 C6	.00000 00461
.00 00 00 07	.00000 00016	.00 00 00 47	.00000 00165	.00 00 00 87	.00000 00314	.00 00 00 C7	.00000 00463
.00 00 00 08	.00000 00018	.00 00 00 48	.00000 00167	.00 00 00 88	.00000 00316	.00 00 00 C8	.00000 00465
.00 00 00 09	.00000 00020	.00 00 00 49	.00000 00169	.00 00 00 89	.00000 00318	.00 00 00 C9	.00000 00467
.00 00 00 0A	.00000 00023	.00 00 00 4A	.00000 00172	.00 00 00 8A	.00000 00321	.00 00 00 CA	.00000 00470
.00 00 00 0B	.00000 00025	.00 00 00 4B	.00000 00174	.00 00 00 8B	.00000 00323	.00 00 00 CB	.00000 00472
.00 00 00 0C	.00000 00027	.00 00 00 4C	.00000 00176	.00 00 00 8C	.00000 00325	.00 00 00 CC	.00000 00474
.00 00 00 0D	.00000 00030	.00 00 00 4D	.00000 00179	.00 00 00 8D	.00000 00328	.00 00 00 CD	.00000 00477
.00 00 00 0E	.00000 00032	.00 00 00 4E	.00000 00181	.00 00 00 8E	.00000 00330	.00 00 00 CE	.00000 00479
.00 00 00 0F	.00000 00034	.00 00 00 4F	.00000 00183	.00 00 00 8F	.00000 00332	.00 00 00 CF	.00000 00481
.00 00 00 10	.00000 00037	.00 00 00 50	.00000 00186	.00 00 00 90	.00000 00335	.00 00 00 D0	.00000 00484
.00 00 00 11	.00000 00039	.00 00 00 51	.00000 00188	.00 00 00 91	.00000 00337	.00 00 00 D1	.00000 00486
.00 00 00 12	.00000 00041	.00 00 00 52	.00000 00190	.00 00 00 92	.00000 00339	.00 00 00 D2	.00000 00488
.00 00 00 13	.00000 00044	.00 00 00 53	.00000 00193	.00 00 00 93	.00000 00342	.00 00 00 D3	.00000 00491
.00 00 00 14	.00000 00046	.00 00 00 54	.00000 00195	.00 00 00 94	.00000 00344	.00 00 00 D4	.00000 00493
.00 00 00 15	.00000 00048	.00 00 00 55	.00000 00197	.00 00 00 95	.00000 00346	.00 00 00 D5	.00000 00495
.00 00 00 16	.00000 00051	.00 00 00 56	.00000 00200	.00 00 00 96	.00000 00349	.00 00 00 D6	.00000 00498
.00 00 00 17	.00000 00053	.00 00 00 57	.00000 00202	.00 00 00 97	.00000 00351	.00 00 00 D7	.00000 00500
.00 00 00 18	.00000 00055	.00 00 00 58	.00000 00204	.00 00 00 98	.00000 00353	.00 00 00 D8	.00000 00502
.00 00 00 19	.00000 00058	.00 00 00 59	.00000 00207	.00 00 00 99	.00000 00356	.00 00 00 D9	.00000 00505
.00 00 00 1A	.00000 00060	.00 00 00 5A	.00000 00209	.00 00 00 9A	.00000 00358	.00 00 00 DA	.00000 00507
.00 00 00 1B	.00000 00062	.00 00 00 5B	.00000 00211	.00 00 00 9B	.00000 00360	.00 00 00 DB	.00000 00509
.00 00 00 1C	.00000 00065	.00 00 00 5C	.00000 00214	.00 00 00 9C	.00000 00363	.00 00 00 DC	.00000 00512
.00 00 00 1D	.00000 00067	.00 00 00 5D	.00000 00216	.00 00 00 9D	.00000 00365	.00 00 00 DD	.00000 00514
.00 00 00 1E	.00000 00069	.00 00 00 5E	.00000 00218	.00 00 00 9E	.00000 00367	.00 00 00 DE	.00000 00516
.00 00 00 1F	.00000 00072	.00 00 00 5F	.00000 00221	.00 00 00 9F	.00000 00370	.00 00 00 DF	.00000 00519
.00 00 00 20	.00000 00074	.00 00 00 60	.00000 00223	.00 00 00 A0	.00000 00372	.00 00 00 E0	.00000 00521
.00 00 00 21	.00000 00076	.00 00 00 61	.00000 00225	.00 00 00 A1	.00000 00374	.00 00 00 E1	.00000 00523
.00 00 00 22	.00000 00079	.00 00 00 62	.00000 00228	.00 00 00 A2	.00000 00377	.00 00 00 E2	.00000 00526
.00 00 00 23	.00000 00081	.00 00 00 63	.00000 00230	.00 00 00 A3	.00000 00379	.00 00 00 E3	.00000 00528
.00 00 00 24	.00000 00083	.00 00 00 64	.00000 00232	.00 00 00 A4	.00000 00381	.00 00 00 E4	.00000 00530
.00 00 00 25	.00000 00086	.00 00 00 65	.00000 00235	.00 00 00 A5	.00000 00384	.00 00 00 E5	.00000 00533
.00 00 00 26	.00000 00088	.00 00 00 66	.00000 00237	.00 00 00 A6	.00000 00386	.00 00 00 E6	.00000 00535
.00 00 00 27	.00000 00090	.00 00 00 67	.00000 00239	.00 00 00 A7	.00000 00388	.00 00 00 E7	.00000 00537
.00 00 00 28	.00000 00093	.00 00 00 68	.00000 00242	.00 00 00 A8	.00000 00391	.00 00 00 E8	.00000 00540
.00 00 00 29	.00000 00095	.00 00 00 69	.00000 00244	.00 00 00 A9	.00000 00393	.00 00 00 E9	.00000 00542
.00 00 00 2A	.00000 00097	.00 00 00 6A	.00000 00246	.00 00 00 AA	.00000 00395	.00 00 00 EA	.00000 00544
.00 00 00 2B	.00000 00100	.00 00 00 6B	.00000 00249	.00 00 00 AB	.00000 00398	.00 00 00 EB	.00000 00547
.00 00 00 2C	.00000 00102	.00 00 00 6C	.00000 00251	.00 00 00 AC	.00000 00400	.00 00 00 EC	.00000 00549
.00 00 00 2D	.00000 00104	.00 00 00 6D	.00000 00253	.00 00 00 AD	.00000 00402	.00 00 00 ED	.00000 00551
.00 00 00 2E	.00000 00107	.00 00 00 6E	.00000 00256	.00 00 00 AE	.00000 00405	.00 00 00 EE	.00000 00554
.00 00 00 2F	.00000 00109	.00 00 00 6F	.00000 00258	.00 00 00 AF	.00000 00407	.00 00 00 EF	.00000 00556
.00 00 00 30	.00000 00111	.00 00 00 70	.00000 00260	.00 00 00 B0	.00000 00409	.00 00 00 F0	.00000 00558
.00 00 00 31	.00000 00114	.00 00 00 71	.00000 00263	.00 00 00 B1	.00000 00412	.00 00 00 F1	.00000 00561
.00 00 00 32	.00000 00116	.00 00 00 72	.00000 00265	.00 00 00 B2	.00000 00414	.00 00 00 F2	.00000 00563
.00 00 00 33	.00000 00118	.00 00 00 73	.00000 00267	.00 00 00 B3	.00000 00416	.00 00 00 F3	.00000 00565
.00 00 00 34	.00000 00121	.00 00 00 74	.00000 00270	.00 00 00 B4	.00000 00419	.00 00 00 F4	.00000 00568
.00 00 00 35	.00000 00123	.00 00 00 75	.00000 00272	.00 00 00 B5	.00000 00421	.00 00 00 F5	.00000 00570
.00 00 00 36	.00000 00125	.00 00 00 76	.00000 00274	.00 00 00 B6	.00000 00423	.00 00 00 F6	.00000 00572
.00 00 00 37	.00000 00128	.00 00 00 77	.00000 00277	.00 00 00 B7	.00000 00426	.00 00 00 F7	.00000 00575
.00 00 00 38	.00000 00130	.00 00 00 78	.00000 00279	.00 00 00 B8	.00000 00428	.00 00 00 F8	.00000 00577
.00 00 00 39	.00000 00132	.00 00 00 79	.00000 00281	.00 00 00 B9	.00000 00430	.00 00 00 F9	.00000 00579
.00 00 00 3A	.00000 00135	.00 00 00 7A	.00000 00284	.00 00 00 BA	.00000 00433	.00 00 00 FA	.00000 00582
.00 00 00 3B	.00000 00137	.00 00 00 7B	.00000 00286	.00 00 00 BB	.00000 00435	.00 00 00 FB	.00000 00584
.00 00 00 3C	.00000 00139	.00 00 00 7C	.00000 00288	.00 00 00 BC	.00000 00437	.00 00 00 FC	.00000 00586
.00 00 00 3D	.00000 00142	.00 00 00 7D	.00000 00291	.00 00 00 BD	.00000 00440	.00 00 00 FD	.00000 00589
.00 00 00 3E	.00000 00144	.00 00 00 7E	.00000 00293	.00 00 00 BE	.00000 00442	.00 00 00 FE	.00000 00591
.00 00 00 3F	.00000 00146	.00 00 00 7F	.00000 00295	.00 00 00 BF	.00000 00444	.00 00 00 FF	.00000 00593

COMMON MATHEMATICAL CONSTANTS

Constant	Decimal Value			Hexadecimal Value	
π	3.14159	26535	89793	3.243F	6A89
$\pi-1$	0.31830	98861	83790	0.517C	C1B7
$\sqrt{\pi}$	1.77245	38509	05516	1.C5BF	891C
$\ln\pi$	1.14472	98858	49400	1.250D	048F
e	2.71828	18284	59045	2.B7E1	5163
e^{-1}	0.36787	94411	71442	0.5E2D	58D9
$\sqrt{e}$	1.64872	12707	00128	1.A612	98E2
$\log_{10}e$	0.43429	44819	03252	0.6F2D	EC55
$\log_{2}e$	1.44269	50408	88963	1.7154	7653
γ	0.57721	56649	01533	0.93C4	67E4
$\ln\gamma$	−0.54953	93129	81645	−0.8CAE	9BC1
$\sqrt{2}$	1.41421	35623	73095	1.6A09	E668
ln2	0.69314	71805	59945	0.B172	17F8
$\log_{10}2$	0.30102	99956	63981	0.4D10	4D42
$\sqrt{10}$	3.16227	76601	68379	3.298B	075C
ln 10	2.30258	40929	94046	2.4D76	3777

APPENDIX B

PROGRAMMING REFERENCES

SAL INSTRUCTION LIST BY OP CODE

HEXADECIMAL OP CODE	MNEMONIC	FORMAT	NAME	EXECUTION TIME IN MICROSECONDS
01	-	-	RESERVED	
02	LDCR	3	LOAD COMMUNICATION REGISTER	7–12
03	ARB	1	ADD TO REGISTER AND BRANCH ON NO SIGN CHANGE	5–6 2/3
04	CM	2	COMPARE MEMORY TO MEMORY	8 1/3-9 1/3
05	MOV	2	MOVE MEMORY WORD	8
06	CML	2	COMPARE MEMORY TO LIMITS IN MEMORY	9
07	CMI	2	COMPARE MEMORY IMMEDIATE	7
08	AMI	2	ADD TO MEMORY IMMEDIATE	7
09	ADAC	1	ACTIVATE DIRECT MEMORY ACCESS CHANNEL	5 1/3
0A	BRRL	1	BRANCH RELATIVE AND LINK	6
OB	STCR	3	STORE COMMUNICATION REGISTER	7-17
OC	BBNE	3	BRANCH IF BIT NOT EQUAL	5 2/3-6 2/3
OD	SETB	3	OUTPUT BIT TO CRU	4 2/3
OE	SBNE	3	SWITCH MODE IF BIT NOT EQUAL	5 2/3
OF	TSBX	3	TEST INPUT BIT AND OUTPUT BIT OR SWITCH MODE	6
10	XOR		EXCLUSIVE OR	9
10	XORA	1	EXCLUSIVE OR WITH EFFECTIVE ADDRESS	6-2/3
11	L	1	LOAD REGISTER	5
11	LA	1	LOAD REGISTER WITH ADDRESS	4
12	ST	1	STORE REGISTER	5 1/3
13	A	1	ADD TO REGISTER	6
13	AA	1	ADD ADDRESS TO REGISTER	5
14	S	1	SUBTRACT FROM REGISTER	6
14	SA	1	SUBTRACT ADDRESS FROM REGISTER	5
15	LOT	1	LOAD ONE'S TALLY	10 2/3
15	LOTA	1	LOAD ONE'S TALLY OF ADDRESS	9 2/3
16	N	1	LOGICAL AND	6 1/3
16	NA	1	LOGICAL AND WITH ADDRESS	5 1/3
17	OR	1	LOGICAL OR	6 1/3
17	ORA	1	LOGICAL OR WITH ADDRESS	5 1/3
18	MLA	1	SHIFT MEMORY LEFT ARITHMETIC	6-11
18	MLAX	1	SHIFT MEMORY LEFT ARITHMETIC, COUNT IN REGISTER R	7 1/3-12 1/3
19	MRA	1	SHIFT MEMORY RIGHT ARITHMETIC	6-11
19	MRAX	1	SHIFT MEMORY RIGHT ARITHMETIC, COUNT IN REGISTER R	7 1/3-12 1/3
1A	MRR	1	ROTATE MEMORY RIGHT LOGICAL	6-11
1A	MRRX	1	ROTATE MEMORY LOGICAL, COUNT IN REGISTER R	7 1/3-12 1/3
1B	SAT	1	SHIFT AND ADD TALLY OF LEADING ZEROS	8-13
1C	B	1	UNCONDITIONAL BRANCH	4 2/3-5 2/3
1C	*B	1	UNCONDITIONAL BRANCH INDIRECT	5 2/3-6 2/3
1C	NOP	1	NO OPERATION	5
1C	XS	1	SUPERVISOR MODE	4
1C	XSB	1	SUPERVISOR MODE AND BRANCH	4 2/3-5 2/3

SAL INSTRUCTION LIST BY OP CODE (cont.)

HEXADECIMAL OP CODE	MNEMONIC	FORMAT	NAME	EXECUTION TIME IN MICROSECONDS
1C	XW	1	WORKER MODE	4
1C	XWB	1	WORKER MODE AND BRANCH	4 2/3-5 2/3
1C	*XSB	1	WORKER MODE AND BRANCH INDIRECT	5 2/3-6 2/3
1C	*XWB	1	WORKER MODE AND BRANCH INDIRECT	5 2/3-6 2/3
1D	BL	1	BRANCH AND LINK	4 2/3
1D	*BL	1	BRANCH INDIRECT AND LINK	5 2/3
1E	SS	1	STORE STATUS BLOCK	5-7
1E	SSB	1	STORE STATUS BLOCK AND BRANCH	7-9
1E	SXS	1	STORE STATUS BLOCK AND SWITCH TO SUPERVISOR MODE	5 1/3-7
1E	SXBS	1	STORE STATUS BLOCK, TRANSFER AND BRANCH IN SUPERVISOR MODE	7-9
1E	SXBW	1	STORE STATUS BLOCK, TRANSFER AND BRANCH IN WORKER MODE	7-9
1E	SXW	1	STORE STATUS BLOCK AND TRANSFER TO WORKER MODE	5-7
1F	LDS	1	LOAD STATUS BLOCK	6 1/3-7 1/3
20	XFNE	3	SWITCH MODE IF FLAG NOT EQUAL	6
21	BFNE	3	BRANCH IF FLAG NOT EQUAL	5 2/3-7
22	SETF	3	SET SOFTWARE FLAG	6

FOR INSTRUCTIONS INVOLVING SHIFTS, ADD 1/3 MICROSECOND PER BIT SHIFTED

*The asterisk symbol is a part of the mnemonic.

SAL INSTRUCTION LIST BY MNEMONIC

MNEMONIC	OP CODE	FORMAT	NAME	EXECUTION TIME IN MICROSECONDS
A	13	1	ADD TO REGISTER	6
AA	13	1	ADD ADDRESS TO REGISTER	5
ADAC	09	1	ACTIVATE DIRECT MEMORY ADDRESS CHANNEL	5 1/3
AMI	08	2	ADD TO MEMORY IMMEDIATE	5 2/3
ARB	03	1	ADD TO REGISTER AND BRANCH ON NO SIGN CHANGE	5-6 2/3
B	1C	1	UNCONDITIONAL BRANCH	4 2/3-5 2/3
BBNE	0C	3	BRANCH ON BIT NOT EQUAL	5 2/3-6 2/3
BFNE	21	3	BRANCH IF FLAG NOT EQUAL	5 2/3-6 2/3
BL	1D	1	BRANCH AND LINK	4 2/3
BRRL	0A	1	BRANCH RELATIVE AND LINK	6
CM	04	2	COMPARE MEMORY TO MEMORY	8 1/3-9 1/3
CMI	07	2	COMPARE MEMORY IMMEDIATE	6 1/3-7 1/3
CML	06	2	COMPARE MEMORY TO LIMITS IN MEMORY	8 1/3-10 2/3
L	11	1	LOAD REGISTER	5
LA	11	1	LOAD REGISTER WITH ADDRESS	4
LDCR	02	3	LOAD COMMUNICATION REGISTER	7-12
LDS	1F	1	LOAD STATUS BLOCK	6 1/3-7 1/3
LOT	15	1	LOAD ONE'S TALLY	10 2/3
LOTA	15	1	LOAD ONE'S TALLY OF ADDRESS	9 2/3
MLA	18	1	SHIFT MEMORY LEFT ARITHMETIC	6-11
MLAX	18	1	SHIFT MEMORY LEFT ARITHMETIC, COUNT IN REGISTER R	7 1/3-12 1/3
MOV	05	2	MOVE MEMORY WORD	7
MRA	19	1	SHIFT MEMORY RIGHT ARITHMETIC	6-11
MRAX	19	1	SHIFT MEMORY RIGHT ARITHMETIC, COUNT IN REGISTER R	7 1/3-13 1/3
MRR	1A	1	ROTATE MEMORY RIGHT LOGICAL	6-11
MRRX	1A	1	ROTATE MEMORY LOGICAL, COUNT IN REGISTER R	7 1/3-12 1/3
N	16	1	LOGICAL AND	6-11
NA	16	1	LOGICAL AND WITH ADDRESS	5 1/3
NOP	1C	1	NO OPERATION	5
OR	17	1	LOGICAL OR	6 1/3
ORA	17	1	LOGICAL OR WITH ADDRESS	5 1/3
S	14	1	SUBTRACT FROM REGISTER	6
SA	14	1	SUBTRACT ADDRESS FROM REGISTER	5
SAT	1B	1	SHIFT AND ADD TALLY OF LEADING ZEROS	8-13
SETB	0D	3	SET CRU OUTPUT BIT	4 2/3
SETF	22	3	SET SOFTWARE FLAG	6
SS	1E	1	STORE STATUS BLOCK	5-7
SSB	1E	1	STORE STATUS BLOCK AND BRANCH	7-9
ST	12	1	STORE REGISTER	5 1/3
STCR	0B	3	STORE COMMUNICATION REGISTER	7-17
SXBS	1E	1	STORE STATUS BLOCK, TRANSFER AND BRANCH IN SUPERVISOR MODE	7-9
SXBW	1E	1	STORE STATUS BLOCK, TRANSFER AND BRANCH IN WORKER MODE	7-9

SAL INSTRUCTION LIST BY MNEMONIC (cont.)

MNEMONIC	OP CODE	FORMAT	NAME	EXECUTION TIME IN MICROSECONDS
SXS	1E	1	STORE STATUS BLOCK AND TRANSFER TO SUPERVISOR MODE	5-7
SXW	1E	1	STORE STATUS BLOCK AND TRANSFER TO WORKER MODE	5-7
TSBX	0F	3	TEST INPUT BIT AND SWITCH MODE OR OUTPUT BIT	6
XBNE	0E	3	SWITCH MODE ON BIT NOT EQUAL	5 2/3
XFNE	20	3	SWITCH MODE IF FLAG NOT EQUAL	6
XOR	10	1	EXCLUSIVE OR	8
XORA	10	1	EXCLUSIVE OR WITH EFFECTIVE ADDRESS	6-2/3
XS	1C	1	TRANSFER TO SUPERVISOR MODE	4
XSB	1C	1	TRANSFER TO SUPERVISOR MODE AND BRANCH	4 2/3-5 2/3
XW	1C	1	TRANSFER TO WORKER MODE	5 2/3-6 2/3
XWB	1C	1	TRANSFER TO WORKER MODE AND BRANCH	4 2/3-5 2/3
*B	1C	1	UNCONDITIONAL BRANCH INDIRECT	5 2/3-6 2/3
*BL	1D	1	BRANCH INDIRECT AND LINK	5 2/3
*XSB	1C	1	SWITCH TO SUPERVISOR MODE AND BRANCH INDIRECT	5 2/3-6 2/3
*XWB	1C	1	SWITCH TO WORKER MODE AND BRANCH INDIRECT	5 2/3-6 2/3

FOR INSTRUCTIONS INVOLVING SHIFTING, ADD 1/3 MICROSECOND PER BIT SHIFTED

*The asterisk symbol is a part of the mnemonic.

SAL INSTRUCTION LIST BY FUNCTION

MACHINE INSTRUCTIONS

	MNEMONIC	NAME	CODE	EXECUTION TIME IN MICROSECONDS
		LOAD/STORE		
1.	L	LOAD REGISTER	11	5
2.	LA	LOAD REGISTER WITH ADDRESS	11	4
3.	ST	STORE REGISTER	12	5 1/3
4.	MOV	MOVE MEMORY WORD	05	7
5.	LOT	LOAD ONE'S TALLY	15	10 2/3
6.	LOTA	LOAD ONE' TALLY OF ADDRESS	15	9 2/3
		ARITHMETIC		
1.	A	ADD	13	6
2.	AA	ADD ADDRESS	13	5
3.	S	SUBTRACT	14	6
4.	SA	SUBTRACT ADDRESS	14	5
5.	AMI	ADD TO MEMORY IMMEDIATE	08	5 2/3
		COMPARE		
1.	CM	COMPARE MEMORY TO MEMORY	04	8 1/3-9 1/3
2.	CML	COMPARE MEMORY TO LIMITS IN MEMORY	06	8 1/3 - 102/3
3.	CMI	COMPARE MEMORY IMMEDIATE	07	6 1/3-7 1/3
		LOGICAL		
1;	N	LOGICAL AND	16	6 1/3
2.	NA	LOGICAL AND WITH ADDRESS	16	5 1/3
3.	OR	LOGICAL OR	17	6 1/3
4.	ORA	LOGICAL OR WITH ADDRESS	17	5 1/3
5.	XOR	EXCLUSIVE OR	10	8
6.	XORA	EXCLUSIVE OR WITH EFFECTIVE ADDRESS	10	6-2/3
		SHIFT		
1.	MLA	SHIFT MEMORY LEFT ARITHMETIC	18	6-11
2.	MLAX	SHIFT MEMORY LEFT ARITHMETIC COUNT IN REGISTER R	18	7 1/3 - 12 1/3
3.	MRA	SHIFT MEMORY RIGHT ARITHMERIC	19	6-11
4.	MRAX	SHIFT MEMORY RIGHT ARITHMETIC COUNT IN REGISTER R	19	7 1/3 - 12 1/3
5.	MRR	ROTATE MEMORY RIGHT LOGICAL	1A	6-11
6.	MRRX	ROTATE MEMORY LOGICAL, COUNT IN REGISTER R	1A	7 1/3 - 12 1/3
7.	SAT	SHIFT AND ADD TALLY OF LEADING ZEROS	1B	8-13

	MNEMONIC	NAME	CODE	EXECUTION TIME IN MICROSECONDS
		LOAD/STORE STATUS BLOCK		
1.	LDS	LOAD STATUS BLOCK	1F	6-7 1/3
2.	SS	STORE STATUS BLOCK	1E	5-7
3.	SSB	STORE STATUS BLOCK AND BRANCH	1E	7-9
4.	SXS	STORE STATUS BLOCK AND TRANSFER TO SUPERVISOR MODE	1E	5-7
5.	SXW	STORE STATUS BLOCK AND TRANSFER TO WORKER MODE	1E	5-7
6.	SXBS	STORE STATUS BLOCK' TRANSFER AND BRANCH IN SUPERVISOR MODE	1E	7-9
7.	SXBW	STORE STATUS BLOCK, TRANSFER AND BRANCH IN WORKER MODE	1E	7-9
		PROGRAM CONTROL		
		BRANCH		
1.	B	UNCONDITIONAL BRANCH	1C	4 2/3-5 2/3
2.	*B	UNCONDITIONAL BRANCH INDIRECT	1C	5 2/3-6 2/3
3.	BL	BRANCH AND LINK	1D	4 2/3
4.	*BL	BRANCH INDIRECT AND LINK	1D	5 2/3
5.	BRRL	BRANCH RELATIVE AND LINK	0A	6
6.	ARB	ADD TO REGISTER AND BRANCH	03	5-6 2/3
7.	NOP	NO OPERATION	1C	5
		MODE SWITCHING		
1.	XS	TRANSFER TO SUPERVISOR MODE	1C	4
2.	XW	TRANSFER TO WORKER MODE	1C	4
3.	XSB	TRANSFER TO SUPERVISOR MODE AND BRANCH	1C	5 2/3-6 2/3
4.	*XSB	TRANSFER TO SUPERVISOR MODE AND BRANCH INDIRECT	1C	5 2/3-6 2/3
5.	XWB	TRANSFER TO WORKER MODE AND BRANCH	1C	4 2/3-5 2/3
6.	*XWB	TRANSFER TO WORKER MODE AND BRANCH INDIRECT	1C	5 2/3-6 2/3
		SOFTWARE FLAG INSTRUCTIONS		
1.	SETF	SET SOFTWARE FLAG	22	6
2.	XFNE	SWITCH MODE IF FLAG NOT EQUAL	20	6
3.	BFNE	BRANCH IF FLAG NOT EQUAL	21	5 2/3-7

*The asterisk symbol is a part of the mnemonic.

INSTRUCTION LIST BY FUNCTION (cont.)

CRU INSTRUCTIONS

	MNEMONIC	NAME	CODE	EXECUTION TIME IN MICROSECONDS
1.	LDCR	LOAD COMMUNICATION REGISTER	02	7-12
2.	STCR	STORE COMMUNICATION REGISTER	0B	7-17
3.	BBNE	BRANCH IF BIT NOT EQUAL	0C	5-2/3-6-2/3
4.	XBNE	SWITCH MODE IF BIT NOT EQUAL	0E	5-2/3
5.	SETB	SET OUTPUT BIT	0D	4-2/3
6.	TSBX	TEST INPUT BIT AND SWITCH	0F	6

CHARACTERS RECOGNIZED BY SAL

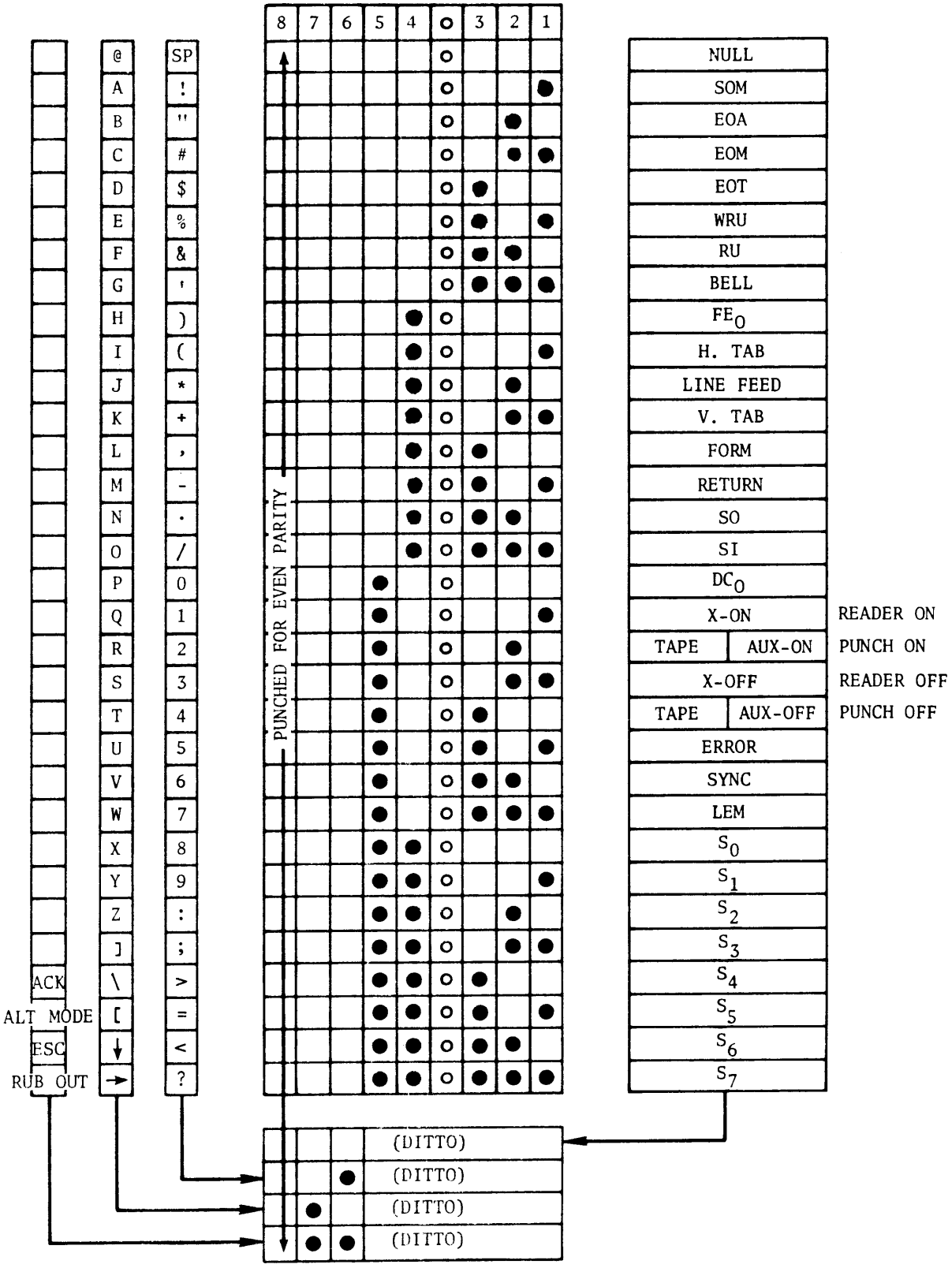
ASCII	H. CODE	CHAR	ASCII	H. CODE	CHAR	ASCII	H. CODE	CHAR	ASCII	H. CODE	CHAR
20	No Punches	SP	30	0	0	40	8 · 4	@	50	11 · 7	P
21	11 · 8 · 2	!	31	1	1	41	12 · 1	A	51	11 · 8	Q
22	UD		32	2	2	42	12 · 2	B	52	11 · 9	R
23	8 · 3	#	33	3	3	43	12 · 3	C	53	0 · 2	S
24	11 · 8 · 3	$	34	4	4	44	12 · 4	D	54	0 · 3	T
25	UD		35	5	5	45	12 · 5	E	55	0 · 4	U
26	12	&	36	6	6	46	12 · 6	F	56	0 · 5	V
27	8 · 5	'	37	7	7	47	12 · 7	G	57	0 · 6	W
28	12 · 8 · 5	(	38	8	8	48	12 · 8	H	58	0 · 7	X
29	11 · 8 · 5	)	39	9	9	49	12 · 9	I	59	0 · 8	Y
2A	11 · 8 · 4	*	3A	8 · 2	:	4A	11 · 1	J	5A	0 · 9	Z
2B	12 · 8 · 6	+	3B	11 · 8 · 6	;	4B	11 · 2	K	5B	UD	
2C	0 · 8 · 3	,	3C	UD		4C	11 · 3	L	5C	UD	
2D	11	–	3D	8 · 6	=	4D	11 · 4	M	5D	UD	
2E	12 · 8 · 3	.	3E	UD		4E	11 · 5	N	5E	UD	
2F	0 · 1	/	3F	0 · 8 · 7	?	4F	11 · 6	O	5F	UD	

CONTROL CHARACTER

ASCII	H. CODE	CHAR	MEANING
17	0 · 9 · 6	EOB	BLOCK END

UD = UNDEFINED

FOR INSTRUCTIONS INVOLVING SHIFTING, ADD 1/3 MICROSECOND PER BIT SHIFTED.



CHARACTER ARRANGEMENT (ASCII) FOR PAPER TAPE

APPENDIX C

TI 960 SAL DIRECTIVES

APPENDIX C

SYMBOLIC ASSEMBLY LANGUAGE DIRECTIVES

C-1 SCOPE

Symbolic Assembly Language (SAL) directives are included to:

identify symbols and program segments,

allocate memory within a segment,

define constants,

pass data to the loader, and

supplement the source language.

C-2 SEGMENT IDENTIFIER DIRECTIVE DEFINITIONS

Four directives are used to identify segment types.

Mnemonic	Function
PSEG	Procedure segment
DSEG	Data segment
FSEG	Flag segment
BSEG	CRU Symbolic Address segment

Note: Segments identified by FSEG and BSEG directives do not ordinarily reserve storage. These two segments are dummies, primarily allowing the programmer to assign symbolic addresses to a particular flag or bit. The programmer may optionally reserve storage in a flag segment by including a RES or DATA directive.

Proper use of a DATA statement within the data segment allows storage to be reserved and initialized.

Example use of a segment identification directive:

```
LABEL PSEG
```

Note: The LABEL entry will be passed to the loader as an external definition.

Unlike other segment identifier directives, the BSEG directive requires an operand. The BSEG operand value becomes the CRU base address for all symbolic CRU addresses defined within the segment.

Symbols defined within CRU, flag, and data segments will be passed to the loader as external definitions (see explanation of DEF directive).

The appropriate segment class attribute will be assigned automatically to symbols within a particular segment.

C-3 ASSEMBLER CONTROL DIRECTIVES

Directive: ABS

General Form: ABS OPERAND

This statement instructs SAL to assign an absolute attribute to symbols defined subsequently until an END card is encountered. A segment identifier must follow an ABS statement.

The label field is not required. The assembler ignores this field. The term in the operand field initializes the absolute assembly program counter.

Directive: MODE

General Form: MODE

This statement notifies SAL to permit reference to alternate mode registers using the # attribute. Note that any use of # when SAL is not provided a currently active MODE directive will cause an error.

Label and Operand field entries are ignored.

Directive: END

General Form: END OPERAND

This directive revokes an active ABS or MODE statement and/or terminates a segment.

A non-blank entry in the operand field will be passed to the loader identified as a transfer vector.

Entries in the label field are ignored.

Examples:

```
1.  D1      DSEG

    SBLOCK      DATA   P1,X'8000'
                .
                .
                .
            END SBLOCK


2.  P1      PSEG
                .
                .
                .
            END
    FS1     FSEG
```

In the first example the instruction labeled SBLOCK is the first instruction to be executed after loading is accomplished. The Label (SBLOCK) in the operand of END tells the loader where to transfer control. The loader uses a LDS to transfer control.

In the second example, END acts as a terminator for the procedure segment.

AN ABS, MODE, PAGE, TITL, or a Segment Identifier Directive must follow immediately an END when it is not used to mark the conclusion of a program. In that case, END must be followed by an "end-of-file" record.

Directive: REF

General Form: REF $OPERAND_1$,$OPERAND_2$,... $OPERAND_n$

This statement identifies symbols appearing in the operand list as external references. These externally referenced symbols are passed to the loader with appropriate data for subsequent assembly.

REF may be used only in Procedure and Data program segments.

Directive: DEF

General Form: DEF $OPERAND_1$,$OPERAND_2$,... $OPERAND_n$

The DEF statement identifies those symbols which will be defined within a segment and made available for reference or call from another segment. In order to REFerence a symbol from an external area, that symbol must appear as a DEF directive operand. DEF need not be used in a FSEG, BSEG or DSEG. Such usage would be redundant since symbols in those segments are external by convention.

Directive: PST

General Form: PST

This statement directs 960 SAL to list the symbol table on the system listing device and punch the table on the system card or tape punch.

Entries in the label and operand fields are ignored.

C-4 MEMORY ALLOCATION DIRECTIVE

Directive: RES

General Form: LABEL RES OPERAND

This statement is used to reserve word locations in memory, the term in the operand field entry is added algebraically to the contents of the current location counter. An entry in the label field is optional.

Example: XLABEL RES 10

In the above example, the operand (10) specifies that ten consecutive words will be reserved in memory. The label (XLABEL) will be the address of the first word reserved. Subsequent words in this reserved area may be addressed using XLABEL and indexing by the proper integer (1 to 9).

C-5 ASSIGNMENT DIRECTIVES

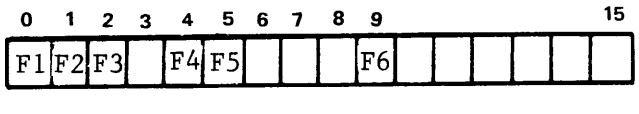
Directive: FLAG

General Form: LABEL FLAG $OPERAND_1$, $OPERAND_2$,...$OPERAND_n$

This directive allows naming of flag bits within the Flag Segment (FSEG).

Flag addresses are assigned sequentially to the undefined symbols appearing in the operand list. The Current Bit Counter is started at bit 0 of the word specified in a FSEG statement and is maintained modulo 16. Each time the counter passes through zero, the Current Location Counter is incremented by one. For example:

```
XLABEL  FSEG
        FLAG  F1,F2,F3,1,F4,F5,3,F6
```



Terms encountered in the operand list are added to the current location counter appropriately to advance the next bit address.

A constant or a symbol representing a constant appearing in the operand list specifies the number of bits to be skipped.

The FLAG directive may be used only in the Flag Segment.

FLAG directives do not allocate memory.

Directive: CON

General Form: LABEL CON OPERAND,MODIFIER

The CONnect directive is used in the CRU Symbolic Address Segment (BSEG) to name the address of a CRU bit or the address of a series of CRU bits.

The modifier is optional and has a default value of 1.

Example

	COMM	BSEG	0
1.	TAPE1	CON	18
2.	TAPE2	CON	19,3
3.	TAPE3	CON	20
4.	TREG	CON	18

Line 1 of the example assigns the name TAPE1 to bit location 18 of the bit program segment. Line 4 assigns a different name to the same location.

Line 2 assigns a name to bits 19, 20 and 21. These three bits may be referenced as a communication register. Line 3 assigns a label to bit 20. This bit is contained within the TAPE2 register but may also be addressed as TAPE3.

Note that the CON directive does not initialize any bits.

CON is used only in a Bit segment.

Programming Note: A sequence of bits defined as a communication register with the CON directive may be addressed as a register by the LDCR and STCR instructions.

Directive: EQU

General Form: LABEL EQU OPERAND

The EQU assignment directive assigns the value entered in the operand field to the symbol appearing in the label field. Among other uses, this statement may be used to assign a name to a register.

EXAMPLE: REGONE EQU 1

This directive would allow General Register 1 to be referred to as REGONE in subsequent instructions.

C-6 DATA DEFINITION

Directive: DATA

General Form: LABEL DATA $OPERAND_1$, $OPERAND_2$,...$OPERAND_n$

This directive is used to place specific values in memory.

Values specified in the operand list are entered in successive core locations. The current location counter is advanced in increment levels suitable for storing the specific type of data being used.

Four types of data are permitted. They are: decimal integer, hexadecimal, ASCII (plain language or Hollerith) and address.

A decimal integer list might appear as:

DATA 529,-3,65

A hexadecimal list:

DATA X'ABCO',X'A',X'3F10'

Note that for hexadecimal numbers, each entry is preceded by X and is enclosed by apostrophes. Neither hexadecimal nor decimal numbers may require more than 16 binary bits for internal representation. If a number (such as X'A') requires fewer than 16 bits, the number is right justified internally and the field is padded with leading zeroes.

In whichever type a number is entered, its value (V) is limited by:

$-2^{15} < V < 2^{15}-1$.

An example of an ASCII list is:

DATA C'TEXAS INSTRUMENTS',C'960 SAL'

Note again that the Data is enclosed in apostrophes and is preceded by C.

One memory word can store 2 ASCII characters so that the first constant, TEXAS INSTRUMENTS, requires 9 storage words. There are 17 characters in the constant including the space between words. 960 SAL left justifies ASCII characters and fills the right most half word with a blank if an odd number of characters is specified.

The DATA directive may be used only in PSEG's, DESG's, and FSEG's.

The SAL has been so designed as to allow for additional types of data to be implemented as a future date. New data types and their parameters will be delineated as they are developed.

C-7 SOURCE LANGUAGE EXTENSION DIRECTIVE

Directive: FRM

General Form: LABEL FRM $OPERAND_1$, $OPERAND_2$,...$OPERAND_{32}$

New instructions and data structures can be designed using the format directive.

Field widths in bits are listed in the operand field. For example:

XLAB FRM 4,2,10,5,5,6

WORD XLAB 4,1,103,26,9,15

The programmer has the option to include a two entry, parenthetic sublist in the FRM operand.

XLABEL FRM 6,3,7,(X'FFFF',0)

XLABEL 15,4,23

Should this option be exercised, a logical "and" will be performed between the first sublist entry and the final version of the FoRMatted word (or double word) and a logical "or" performed between the second sublist entry and the FoRMatted word. When the sublist is used, the number of binary bits (in the example, 16) required to represent each number of the sublist must not exceed the number of bits specified by the field width operands.

Should the sublist number require less than the specified number of bits, the number will be right justified in a field with leading zeroes added.

The number of bits specified in the operand list must equal either 16 or 32.

An entry in the label field is required. After FoRMat has been defined it is referenced by entering this label symbol in the Operation Code field. The Label symbol may not exceed four characters in in length. For example:

XLAB FRM 4,4,8

ALPHA XLAB 12,6,21

BRAVO XLAB 13,5,20

C-8 OUTPUT CONTROL DIRECTIVES

Directive: PAGE

General Form: PAGE

The PAGE directive causes the listing output device to be advanced to "top of form". The actual directive, "PAGE" will not be printed.

Directive: TITL

General Form: TITL OPERAND

This directive is used to specify the "plain language" (ASCII) characters to be used in program identification. These characters will be printed on the first line of each page of the list generated by the 960 SAL assembler.

Note that this does not refer to application output headings. For example:

TITL TI 960 MONITOR SYSTEM

No further output instructions are required. The presence of TITL is sufficient to cause printing. Storage method and requirements are identical with ASCII DATA.