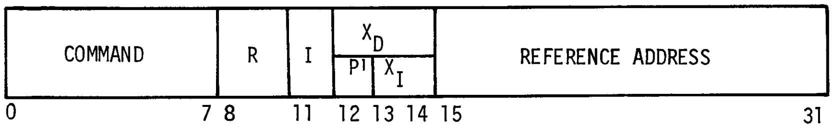
TENET REFERENCE MANUAL

210 COMPUTER

TENET 210 INSTRUCTION SET

Memory Reference



Code	Mnemonic	Name	Page
		WORD	
33	LW	Load Word	15
32	STW	Store Word	15
30	AW	Add Word	15
31	SW	Subtract Word	15
38	ISW	Inverse Subtract Word	15
3A	MW	Multiply Word	15
3B	DW	Divide Word	15
34	ANDW	AND Word	15
35	ORW	OR Word	16
36	EORW	Exclusive OR Word	16
42	XW	Exchange Word	16
3C	LWD	Load Word Doubled	16
37	CW	Compare Word	16
		DOUBLEWORD	
53	LD	Load Doubleword	16
52	STD	Store Doubleword	17
50	AD	Add Doubleword	17
51	SD	Subtract Doubleword	17
3E	MD	Multiply Doubleword	17
3F	DD	Divide Doubleword	18
57	CD	Compare Doubleword	18
		MEMORY	
40	AM	Add to Memory	18
41	SM	Subtract from Memory	18
44	ANDM	AND to Memory	19
45	ORM	OR to Memory	19
46	EORM	Exclusive OR to Memory	19
43	ILW	Increment and Load Word	19
64	CAIM	Complement and Add Immediate to Memory	19
60	AIM	Add Immediate to Memory	20
61	SIM	Subtract Immediate from Memory	20
62	STIM	Store Immediate to Memory	20
64	CIM	Compare Immediate with Memory	19
		SHIFT	
28	SAL	Shift Arithmetic Left	22
29	SALD	Shift Arithmetic Left Double	22
2A	SAR	Shift Arithmetic Right	23
2B	SARD	Shift Arithmetic Right Double	23
20	SLL	Shift Logical Left	23
21	SLLD	Shift Logical Left Double	24
22	SLR	Shift Logical Right	24
23	SLRD	Shift Logical Right Double	24
2C	SCL	Shift Circular Left	24
2D	SCLD	Shift Circular Left Double	24
2E	SCR	Shift Circular Right	24
2F	SCRD	Shift Circular Right Double	24
24	SUN	Shift Until Normalized	25
25	SUND	Shift Until Normalized Double	25
		FIELD	
5B	IFP	Increment Field Pointer	32
4B	DFP	Decrement Field Pointer	32
4C	LF	Load Field Logical	32
5C	LFI	Load Field Logical after Increment	32
5D	LFAI	Load Field Arithmetic after Increment	32
4A	STF	Store Field	32
5A	STFI	Store Field after Increment	33
6A	STIF	Store Immediate to Field	33
4E	CF	Compare Field Logical	33
5E	CFI	Compare Field Logical after Increment	33
4F	CFA	Compare Field Arithmetic	33
5F	CFAI	Compare Field Arithmetic after Increment	33
6E	CIF	Compare Immediate to Field	33
49	AFA	Add Field Arithmetic	34

Code	Mnemonic	Name	Page
		FIELD (cont.)	
59	SFA	Subtract Field Arithmetic	34
69	ISFA	Inverse Subtract Field Arithmetic	34
48	AF	Add Field Logical	34
58	SF	Subtract Field Logical	34
68	ISF	Inverse Subtract Field	34
		PRIVILEGED CONTROL	
99	LPS[3]	Load Program Status	37
98	XPS[3]	Exchange Program Status	37
9D	LM	Load Map Segment	38
9C	LMS	Load Map Status	38
9E	STMS	Store Map Status	38
91	ION	System Interrupts ON	38
90	IOFF	System Interrupts OFF	38
93	PAUS	Pause	38
97	LCR	Load Control Register	39
96	STCR	Store Control Register	39
94	PPS	Push Program Status	39
95	POPS	Pop Program Status	39
9F	LAA	Load Actual Address	39
		INPUT/OUTPUT	
9A	IOC	Input/Output Control	49
		UNPRIVILEGED CONTROL	
AA	STPS	Store Program Status	34
AD	LCI	Load Condition Indicators	34
AC	LAS	Load and Set	35
A9	LG	Load Register Group	35
A8	STG	Store Register Group	35
8F	EXEC	Execute	35
7C	MOVI[2]	Move with Incrementing Pointer	36
7D	MOVD[2]	Move with Decrementing Pointer	36
AE	STSR	Store Panel Switch Register	36
AF	LVA	Load Virtual Address	36
		FLOATING POINT	
B4	FAS	Floating Add Short	43
B0	FAL	Floating Add Long	43
B5	FSS	Floating Subtract Short	44
B1	FSL	Floating Subtract Long	44
B9	FISS	Floating Inverse Subtract Shift	44
B8	FISL	Floating Inverse Subtract Long	44
B6	FMS	Floating Multiply Short	44
B2	FML	Floating Multiply Long	44
B7	FDS	Floating Divide Short	44
B3	FDL	Floating Divide Long	44
BB	FIDS	Floating Inverse Divide Short	45
BA	FIDL	Floating Inverse Divide Long	45
		BRANCH (General Form)	
87	BAC[4]	Branch on Arithmetic Conditions	25
85	BCS[4]	Branch on Conditions Set	25
84	BCR[4]	Branch on Conditions Reset	27
81	BRCS[5]	Branch on Register Conditions Set	27
80	BRCR[5]	Branch on Register Conditions Reset	27
82	BIR[6]	Branch on Incrementing Register	28
83	BDR[6]	Branch on Decrementing Register	28
86	BAL	Branch and Link	25

[1] P = 1 post-indexing; = 0 pre-indexing
[2] R = 1 registers saved; = 0 registers not saved
[3] R5 = count; R6 = source; R7 = destination
[4] Condition code in R field
[5] Register in X field; test condition in R field
[6] Register R is incremented by value in IX field

TENET 210 COMPUTER
Reference Manual

OCTOBER 1970

TENET
927 Thompson Place
Sunnyvale, California 94086

Pub. No. 2004

Price: $4.50

PREFACE

This document is a reference manual which describes in detail the characteristics, organization, and instruction set of the TENET 210 Computer.

FIRST EDITION:

Specifications contained herein are subject to changes which will be reported in subsequent revisions. Copies of this and other TENET publications can be obtained through TENET branch offices. A form is provided at the back of this publication for readers' comments. If the form has been detached, please direct your comments to TENET, 927 Thompson Place, Sunnyvale, California 94086 (Attention: Software Publications Department).

CONTENTS

— APPENDIXES —

CONTENTS (Continued)

— TABLES —

— FIGURES —

1. SYSTEM DESCRIPTION

INTRODUCTION

The TENET 210 is a general-purpose computer designed for interactive time-sharing operations. The system architecture is centered around a data exchange which interconnects core memory modules, central processing units (CPU's), and input/output processors (IOP's). The system can be expanded by merely connecting additional memory modules, IOP's, or CPU's to the data exchange (see Figure 1).

The major elements of a TENET 210 system are:

- A memory consisting of either 8,192-word or 16,384-word modules. Up to a maximum of 131,972 words
- A central processing unit (CPU) equipped with eight general registers and eight control registers
- An input/output system (controlled by the data exchange) that provides 20 bidirectional direct memory access channels with access priority, and 20 levels of nested interrupts, each level expandable to 16 sublevels

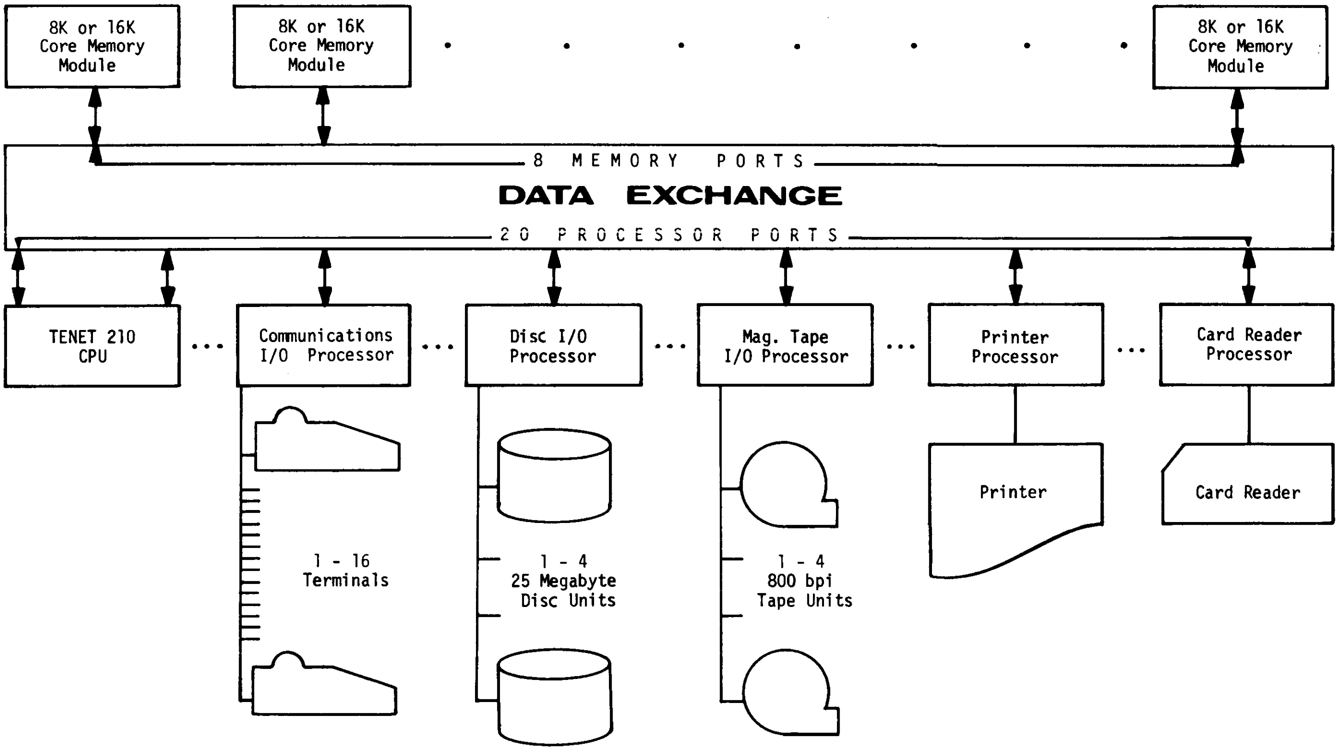


Figure 1. TENET 210 System Architecture

GENERAL CHARACTERISTICS

The TENET 210 System is uniquely qualified to handle a wide variety of applications in a multi-usage, real-time environment. Its outstanding price-performance characteristics make it the ideal general-purpose, time-sharing system. The features described in this section contribute to the overall versatility and capability of the computer.

Central Processing Unit (CPU)

The TENET 210 CPU (the 3210) is a general-purpose processor designed for time-sharing, but includes provisions for scientific data processing and real-time applications. Features of the 3210 include:

- Eight high-speed general registers, seven of which can be used for indexing
- Eight high-speed control registers
- Parity checking for all information transfers between the CPU, input/output processors, and memory
- Automatic traps to handle illegal internal conditions
- Priority interrupt system with 20 levels of nested interrupts, each level expandable to 16 sublevels, armed and disarmed under program control
- Privileged instruction logic (master/slave modes), by which basic operations affecting the state of the computer can be protected from the program
- Power fail-safe, for automatic, safe power shutdown and restart to guarantee the integrity of system operation when the computer is operated from an unreliable power source
- Interruptibility of long instructions to guarantee the fastest possible response to interrupts
- Powerful instruction set, including:
  - Programmed, user-defined instructions to enable the programmer to define additional computer instructions
  - High-speed memory block move instructions
  - Bit manipulation and testing
  - Ability to manipulate packed data fields of arbitrary length (up to 32 bits)
  - Word and doubleword operations
  - Comparison operations
  - Conditional branches
  - Program environment exchange and stacking
  - Immediate to register and immediate to memory
  - Use of all memory reference instructions with or without indirect addressing, pre-indexing, and post-indexing
  - Full complement of logical operations (AND, OR, exclusive OR)
  - Shift operations (left and right) of word and doubleword, logical or arithmetic
  - Fixed-point arithmetic operations in field, word, and doubleword modes
  - Optional floating-point instructions in short (single precision) and long (double precision) formats
- Fast instruction execution times:

Multiply:	3.4 μs
Divide:	7.8 μs
Branch:	0.8 μs
Register-to-register word:	1.0 μs
Memory-to-register word: (with interleaving)	1.2 μs

Core Memory

The TENET 210 memory system has the following characteristics:

- Modular, word-oriented memory (32-bit word) expandable from 8,192 to 131,072 words in 8,192 or 16,384 blocks
- Direct addressing of the entire core memory from the primary instructions
- Automatic interleaving of memory modules to increase processing speeds
- Memory mapping that insures efficient use of available core. (This feature permits programs to be broken up into "pages" which need not occupy contiguous memory, thus permitting multiple users in core without having to move programs; each page can be protected against reading or writing.)
- Multiple memory buses that provide greater bandwidth

Data Exchange

The TENET 210 input/output system is built around a data exchange which connects core memory to all peripheral subsystems (including CPU's), and is also the interface between the CPU and the peripheral processors. The data exchange contains 20 processor ports and 8 memory ports. Each CPU requires two processor ports, and each peripheral processor requires one processor port; thus, a system with one CPU can support up to 18 peripheral processors. Each memory module (8K or 16K) requires one memory port, thus allowing a maximum core memory of 128K. The data exchange resolves priority when several peripherals require access to memory, and when more than one subsystem attempts to interrupt the CPU at the same time. Features of the data exchange include:

- Twenty bidirectional direct memory access channels with access priority
- Twenty levels of nested interrupts, each level expandable to 16 levels by option
- Unique interrupt locations for each interrupt level
- Capability of supporting multiple CPU's
- Four simultaneous channels to memory

Time-Sharing Features

The TENET computer has been a time-share system from its inception. Every portion of the system architecture is designed to optimize time-sharing performance. Some of the key features include:

- Memory mapping, permitting programs that have been broken up into non-contiguous pages to appear contiguous (the result: better memory utilization)
- Automatic hardware detection of page alteration, which eliminates unnecessary page swapouts
- User-defined instructions, which reduce program size
- A rich machine instruction set, yielding more compact programs, less stringent memory requirements, and faster execution
- Program context switching in one instruction, allowing fast user switching

General-Purpose Features

The TENET 210 is designed to handle a variety of scientific and/or business applications in a multi-usage environment. The major general-purpose features are:

- A three-bit condition code that provides information during instruction execution to simplify the testing of results
- Floating-point instructions available in both short and long formats (7 significant digits and 16 significant digits, respectively)
- Over 140 standard instructions that satisfy the computational and data handling needs for a broad spectrum of applications
- Indirect addressing with both pre-indexing and post-indexing
- Trap system to detect illegal instructions, uninstalled options, machine malfunctions, and other warning indications

Real-Time Features

The following features of the TENET 210 enable the computer to respond quickly to a variety of external stimuli in a real-time environment:

- Multilevel interrupt system with 20 levels of nested interrupts, each level expandable to 16 levels by option. The system automatically identifies and responds to the interrupts according to their priority.
- Individual interrupt levels may be enabled/disabled, armed/disarmed, and program triggered.
- Rapid status switching for transferring control from one program to another

Supporting Software

Supporting software for the TENET 210 includes the following:

- An interactive time-sharing operating system supporting 32 to 64 simultaneous users
- A BASIC compiler with features that make TENET BASIC the most powerful time-sharing language in the industry
- TENET FORTRAN IV, an interactive level-H version of the language, with comprehensive editing and debugging facilities for on-line checkout of programs
- A high-level EDITOR language that is used to insert, copy, modify, or delete characters, portions of lines, or entire lines of text
- A one-pass Meta Assembler that accepts source language input and generates a relocatable binary object module and an assembly listing. In addition to conventional assembler functions, the Meta Assembler enables the user to define his own programming language and thus program his problem in a language more suitable to his needs
- An extensive mathematical library with routines for single-precision, double-precision, complex, and double complex calculations
- A relocatable linking loader that loads one or more binary object modules and links their external symbols
- Debug and Super Debug routines that give the programmer great flexibility in debugging and editing his program
- Comprehensive diagnostics for testing under a variety of conditions the full instruction set, the peripherals, the map, the interrupt system, and the memory system

Input/Output Processors

The TENET 210 input/output system consists of one or more input/output processors (IOP's) that operate simultaneously and independent of the CPU. The IOP's feature bidirectional access to all of core memory with data and order chaining. Each I/O control word in the chain includes interrupt control information. The four types of IOP's that can be connected to the TENET 210 are described below.

Disc IOP

Each disc IOP controls one to four single spindle disc units. One 11-high disc pack (25,000,000 bytes) can be mounted on each unit. Features include simultaneous seek operations on all disc units and data transfer to or from one disc unit with seeks in process on other units. Operating characteristics include:

- Byte capacity: 25,000,000 per unit
- Transfer rate: 320 kHz

- Rotational time: 25 milliseconds
- Average access time: 12.5 milliseconds
- Minimum positioning time (track to track): 10 milliseconds
- Average positioning time: 32 milliseconds

  Maximum positioning time (0-200): 60 milliseconds

Magnetic Tape IOP

One to four magnetic tape units can be connected to each IOP. Units are nine-channel, 800 bpi, IBM-compatible, with transfer rates of 36,000 bytes per second.

Card Reader IOP

The card reader IOP controls one card reader and reads binary, Hollerith, and in data-dependent mode at 600 cards per minute.

Line Printer IOP

The printer IOP supports one fully-buffered line printer with a speed of 400 lines per minute. Carraige width is 132 positions, including 63 print characters.

Communications Multiplexer IOP

The communications multiplexer IOP provides data interface and control for up to 16 full-duplex Model 33 Teletypewriter terminals. Options include direct wire connection or access via the public telephone network. Direct connected terminals have teletype On/Off power features under program control.

2. SYSTEM ORGANIZATION

CENTRAL PROCESSING UNIT

The TENET 3210 central processing unit (CPU) can address core memory, fetch and store information, perform arithmetic and logical operations, sequence and monitor instruction execution, and control the exchange of information between core memory and other processors of the TENET 210 system.

General Registers

The TENET 3210 CPU contains eight 32-bit general-purpose registers, R0 through R7. Any of the general registers can be used as fixed-point accumulators, floating-point accumulators, temporary storage, or can contain control information such as data addresses, counts, pointers, etc. Seven of the eight general registers (R1 through R7) can be used for indexing.

Control Registers

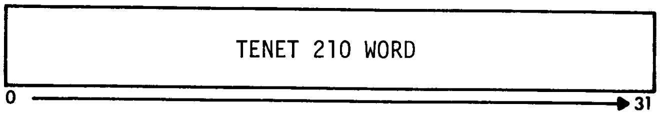
Eight internal control registers can be accessed under program control by the privileged instructions Load Control Register (LCR) and Store Control Register (STCR). The eight control registers are allocated as follows:

Control Register	Use
0	Used by the CPU to perform the divide instructions, and may be used in the future for other CPU functions.
1-4	Currently not used by the CPU, but may be used in the future for other CPU functions.
5	Defines conditions for trap to location 8. These conditions are defined in Section 5 under Traps.
6	Bits 15-31 contain the Current Stack Pointer (CSP). The CSP specifies the address at which the current program status word is stored. This address is used by the privileged control instructions Push Program Status (PPS) and Pop Program Status (POPS) described in Section 3.
7	Bits 15-31 contain the Stack Limit (SLM). The SLM specifies the maximum allowable value of the CSP, and is used by the privileged control instructions PPS and POPS.

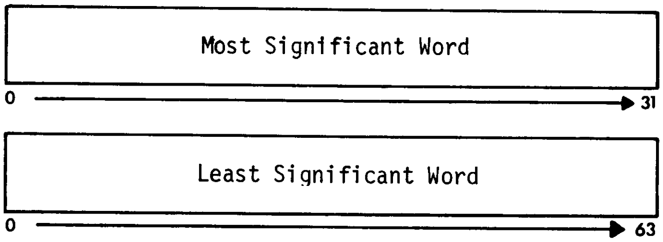
Information Formats

Data Formats

The basic element of TENET 210 addressable information is a 32-bit word. The bits in a word are numbered from left to right, 0 through 31, as follows:



Two consecutive TENET 210 words can be combined and processed as a doubleword. A doubleword contains 64 bits of information numbered from left to right, 0 through 63, as follows:



For fixed-point arithmetic operations, both word and doubleword data are represented as signed, two's complement quantities. Bit 0 is the most significant bit, and bit 31 of a word and bit 63 of a doubleword are the least significant bits.

A doubleword is located such that bits 0 through 31 of the doubleword are contained within the most significant word, and bits 32 through 63 of the doubleword are contained within the next consecutive (least-significant) word.

Four bits of information can be expressed by means of a single hexadecimal digit. Hexadecimal digits, and their binary and decimal equivalents, are expressed in the following notation:

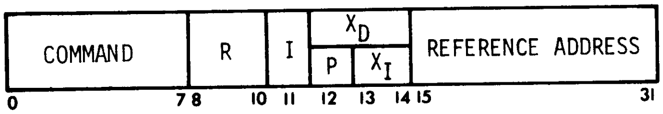
Hexadecimal	Binary	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5

Hexadecimal	Binary	Decimal
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Instruction Formats

Memory Reference Instruction Format

The standard TENET 210 memory-addressing instruction has the following format:



Command This 8-bit field contains the code that designates the operation to be performed.

R This 3-bit field designates one of the eight general registers to be used as an operand source or result destination.

I This bit position indicates whether or not indirect addressing is to be performed. One-level indirect addressing is performed if I = 1, and is not performed if I = 0.

P If indirect addressing is specified, this bit position indicates whether pre-indexing or post-indexing is to be performed. If P = 1, pre-indexing is performed, i.e., indexing first, then indirect addressing. If P = 0, post-indexing is performed, i.e., indirect addressing first, then indexing.

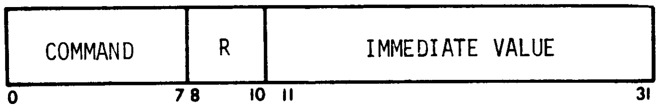
X_I This 2-bit field designates the register to be used for pre-indexing or post-indexing.

X_D If indirect addressing is not specified (I = 0), this 3-bit field permits any one of registers 1 through 7 to be designated as an index register. The contents of this register are then treated as a 32-bit displacement value, which is added to the 17-bit effective address.

Reference Address This 17-bit field contains the initial virtual address of the instruction operand. The reference address field allows any word or doubleword to be directly addressed. Reference addresses 0 - 7 refer to general registers (instead of memory).

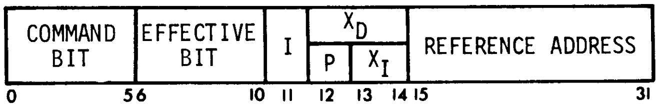
Immediate Instruction Format

Some TENET 210 instructions are immediate, i.e., the operand is an immediate value within the instruction word. The sign (bit 11) is extended to the left to form a 32-bit effective value:



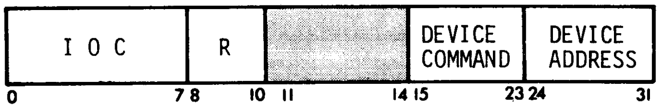
Test Bit Instruction Format

TENET 210 bit instructions are of two types: 1) register-designated bit instructions, which have the memory-reference format with the effective bit designated by the R field; and 2) test bit instructions with the effective bit defined in bits 6-10 of the instruction word. The latter type has the following format:



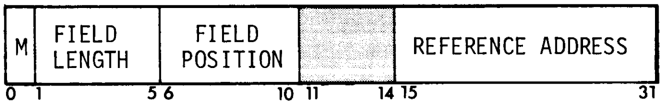
Input/Output Control (IOC) Instruction Format

The IOC instruction, which initiates an input/output operation, specifies a command and a particular input/output processor. The format is as follows:



Field Pointer Format

Field instructions reference a field pointer whose format is:



The field pointer is described in Section 3 under Field Instructions.

Machine Modes

The TENET 210 computer operates in either the master or slave mode; the mode of operation is determined by the state of the CPU mode bits in the program status word. In the master mode, all functions of the computer are available to the program; in the slave mode, any machine function relating to I/O or to changes in certain machine control states is prohibited to the program.

A master mode program can change the operating mode of the computer by executing either the instruction Load Program Status (LPS), Exchange Program Status (XPS), or Push Program Status (PPS). These privileged control instructions are described in Section 3. The resident executive program EXECUTIVE operates in the master mode and controls programs that may be operating in the slave mode.

A program operating in the slave mode cannot change the machine control from slave to master or from master to slave.

Master Mode

The master mode is the basic operating mode of the computer. In this mode, all legal TENET 210 operations are permissible. A program operating in the master mode can change the program status word and can execute any I/O function. Privileged control instructions, i.e., those relating to input/output and to changes in the basic control state of the computer, can be executed in the master mode only.

Slave Mode

In the slave mode, the basic operations affecting the state of the computer are protected from the program. A program operating in the slave mode cannot execute a privileged instruction; thus, a slave program cannot perform any I/O operations and cannot modify most of the control states that are indicated in the program status word.

Program Status Word (PSW)

The critical control conditions of the TENET 210 CPU are defined within 32 bits of information. These 32 bits are referred to as the program status word (PSW). The format is as follows:

CM	CC1	CC2	CC3	OF	CF	IE	PO	SIR	FO	FD	FU	RS	▨	PC
0 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 31

Designation | Meaning

CM CPU Mode.

These bits define the master/slave/map mode of machine operation as follows:

Bit 0	Bit 1	Mode
0	0	Master/Unmapped
0	1	Master/Selective Mapping
1	0	Master/Mapped
1	1	Slave/Mapped

Designation | Meaning

When in the selective mapping mode, addresses are unmapped except when indirect addressing, indexing, or field pointers are used, and bit 0 of the indirect address, index register, or field pointer is set (see Section 4).

CC1, CC2, CC3 Condition Codes. These bits are set/reset by certain instructions depending upon the results of the instruction just executed. In general, the following types of instructions modify the condition codes: compare, I/O test, shift, test bit, and instructions that modify memory (except store). The significance of the condition codes is explained in Section 3 in the instruction descriptions. In some operations, only a portion of the condition code is involved: CC1 refers to the first bit of the condition code, CC2 to the second bit, and CC3 to the third bit. The condition code is not affected by the first instruction of a trap or interrupt, unless that instruction is Exchange Program Status (XPS) or Load Program Status (LPS). The condition code is non-accumulative and is not altered by testing.

OF Overflow Flag. This flag is set under the following conditions:

(1) The signed, two's-complement result of any arithmetic fixed-point instruction requires more than 32 bits (for single-word instructions) or 64 bits (for doubleword instructions).

(2) A floating-point operation results in an exponent that is too large to be expressed in the exponent field (7 bits for a single-precision number and 9 bits for a double-precision number).

(3) If during a divide operation: $|\text{dividend}| * 2^{-31} \geq |\text{divisor}|$

(4) The sign of register R changes after execution of a Shift Arithmetic Left (SAL) or Shift Arithmetic Left Double (SALD) instruction.

(5) The instruction Load Condition Indicators (LCI) has set the overflow flag with the corresponding bit from the effective word.

(6) A Load Word Doubled (LWD) instruction sets the overflow flag to a 1 if bits 0 and 1 of the operand differ.

(7) The instruction Exchange Program Status (XPS) has set the overflow flag with bit 5 of EA + 1.

Designation Meaning

OF

(8) The instruction Push Program Status (PPS) or Load Program Status (LPS) has set the overflow flag with the corresponding bit from the effective word. The overflow flag is reset if the above conditions do not occur during the execution of these instructions. An instruction which has no possibility of setting the overflow flag will leave it unchanged. The overflow flag is non-accumulative and is not altered by testing.

CF

Carry Flag. This flag is set when:

(1) A carry is propagated past the most-significant bit of the result of an add or subtract instruction.

(2) A Divide Word (DW) or Divide Immediate (DI) instruction results in a non-zero remainder. If the overflow flag is set by a divide, the carry flag will not be changed.

(3) The carry flag is set to the contents of bit 0 of the Load Word Doubled (LWD) instruction operand.

(4) A zero divisor is detected in a floating-point operation.

(5) An unnormalized operand is detected in a floating-point operation.

(6) The instruction Exchange Program Status (XPS) has set the carry flag with bit 6 of EA + 1.

(7) The instruction Load Condition Indicators (LCI) has set the carry flag with the corresponding bit from the effective word.

(8) The instruction Push Program Status (PPS) or Load Program Status (LPS) has set the carry flag with the corresponding bit from the effective word. The carry flag is reset if the above conditions do not occur during the execution of these instructions. An instruction which has no possibility of setting the carry flag will leave it unchanged. The carry flag is non-accumulative and is not altered by testing.

IE

Interrupt Enable. IE = 1 allows the CPU to respond to requests for interrupt; IE = 0 disables CPU response to interrupts.

Designation Meaning

PO

Page Protect Override. PO = 1 causes the CPU to ignore the "read-only page" condition when the memory map is used, thereby permitting the CPU to write in write-protected pages.

SIR

Special Instruction Recovery Flag. This flag is set if an LFI, LFAI, STFI, CFI, or CFAI field instruction is trapped after the field pointer is incremented but before the instruction is completed. This case occurs when the field or an indirect word following the field pointer is read or write protected; in this event, the program counter (PSW bits 15-31) points to the offending field instruction, the SIR flag is set, and the trap occurs. If the SIR flag is set and an LFI, LFAI, STFI, CFI, or CFAI instruction is encountered for execution, the instruction will be executed as an LF, LFA, STF, CF, or CFA, respectively; thereby avoiding a second incrementing of the field pointer. The SIR flag is reset upon successful completion of any of the following instructions: Load Word (LW), Compare Word (CW), Store Field (STF, STFI, and STIF), Load Field (LF, LFI, LFA, and LFAI), and Compare Field (CF, CFI, CFA, CFAI, and CIF).

FO

Floating-Point Overflow. If FO = 1 and the overflow flag (OF) is set as the result of a floating-point operation, bit 10 of control register 5 is set and a trap to location 8 is executed.

FD

Floating-Point Divide by Zero. If FD = 1 and the overflow flag (OF) and carry flag (CF) are set as the result of a floating-point zero divisor, bit 11 of control register 5 is set and a trap to location 8 is executed.

FU

Floating-Point Underflow. If FU = 1 and the carry flag (CF) is set as the result of an unnormalized floating-point operand, bit 12 of control register 5 is set and a trap to location 8 is executed.

RS

Registers-Stored Flag. This is a control flag used by the CPU to indicate that the eight general registers were stored with the current PSW (see Privileged Control Instructions PPS and POPS).

Designation	Meaning
PC	Program Counter that points to the core memory location of the next instruction to be executed in sequence. PC is incremented at the conclusion of a successful instruction fetch. The value in PC is mapped or not mapped depending on the state of CM (CPU Mode bits).

CORE MEMORY

The basic unit of information in the TENET 210 memory system is the 32-bit word. Memory is available in modules of 8,192 and 16,384 words, with a maximum memory size of 131,072 words. All memory is accessible to the CPU and can be directly referenced.

Allocated Memory Locations

Memory locations 0-143 are reserved by the system for special purposes (see Table 1).

Memory Map

The memory map (described in Section 4) is a high-speed integrated circuit memory module that permits programs to be broken into non-contiguous 512-word pages, thus ensuring efficient use of available core memory.

In the non-mapping mode, i.e., when the memory map is not in effect, all program addresses are used by the CPU as references to actual core memory locations. In the mapping mode, all program (virtual) addresses above 7 are transformed through the memory map before they become references to actual core memory locations.

Mapping is controlled by the CPU mode bits (bits 0 and 1) in the program status word, described in this section under Program Status Word.

Each map word (10 bits) corresponds to a virtual page and includes three fields:

R/W	A	ACTUAL PAGE
0	1	2 9

where

R/W = Read/Write bit: 0 = read only; 1 = read or write.

A = Altered/Unaltered page bit: 0 = page unaltered; 1 = page altered

Actual Page = Actual memory page address (0 = page unavailable)

Table 1. Allocated Memory Locations

Location		Function
Decimal	Hexadecimal	
0 7	0 7	If referenced by an I/O processor, core memory locations are accessed. If referenced by the CPU for operand fetch or store, general registers are accessed. If referenced by the CPU retrieving its next instruction, a trap is executed to location 8.
8	8	Trap location defined by control register 5
9	9	Power failure trap location
10	A	Parity failure trap location
11 . . 15	B . . F	Unused trap locations
16 . . . 47	10 . . . 2F	Bootstrap load area
48 . . . 63	30 . . . 3F	Floating-point pins (not used if floating-point hardware option is implemented)
64 . . . 79	40 . . . 4F	Global pins (operation codes C0 - CF)
80 . . . 127	50 . . . 7F	Local pins (operation codes D0 - FF)
128 . . . 143	80 . . . 143	Interrupt locations

Addressing

Addresses 0-7

Memory addresses 0-7 have a dual identity, depending on whether they are referenced by a CPU instruction or by an I/O processor.

If a CPU instruction produces a direct, indirect, or indexed program address in the range 0 through 7, the CPU does not read from or write into core memory. Instead, the program address is used as the address of a general register, and the general register corresponding to this address is used as the location for the instruction operand fetch or store operation. However, if addresses 0-7 are referenced by the CPU in attempting to retrieve its next instruction or as the result of a branch, a trap is executed to location 8.

If memory addresses 0-7 are referenced by an I/O processor, the locations in core memory are accessed.

Address Modification

Indirect Addressing

The reference address of most instructions[†] can be modified by indirect addressing, i.e., the 17-bit reference address field of the instruction is used to obtain a word, and the 17 low-order bits of the word thus obtained become the effective address. Indirect addressing is limited to one level, and does not proceed to further levels regardless of the contents of the word location pointed to by the current effective address.

Indexing

Indexing can occur with or without indirect addressing. With indexing, the contents of a general register are treated as a 32-bit value which is added to the 17-bit reference address to produce the 17-bit effective address.

If indirect addressing is not specified, the X_D field (bits 12-14) of the standard instruction format permits any one of registers 1 through 7 to be designated as an index register. A zero value in the X_D field designates no indexing.

If indirect addressing is specified, the X_I field (bits 13-14) of the standard instruction format designates a register (1 through 3) to be used for pre-indexing, i.e., indexing is performed first, and then indirect addressing is performed, or designates a register (1 through 4) to be used for post-indexing, i.e., indirect addressing is performed first, and then indexing is performed. A zero value in the X_I field designates no indirect addressing.

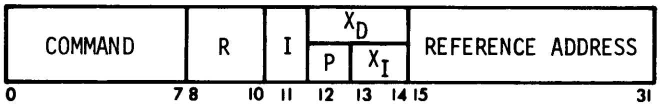
[†] Exceptions are the immediate instructions and branch instructions BIR and BDR.

3. COMPUTER INSTRUCTIONS

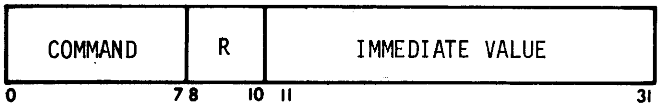
This section describes all TENET 210 instructions, grouped into the following functional classes:

TENET 210 instructions have three basic formats, described in Section 2 under Instruction Formats:

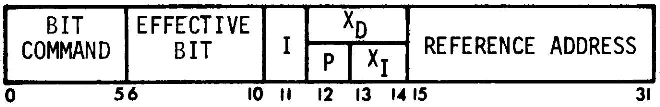
- Memory Reference



- Immediate



- Test Bit



With the exception of the immediate and test bit instructions, all other instructions have the memory reference format.

SYMBOLS

The symbols used in the instruction descriptions have the following meanings:

Symbol	Meaning
( )	The contents of. (The enclosed quantity is a general register or core memory location.)
←	Is replaced by, or is set to
→	Replaces
| |	The magnitude of
+	Addition
-	Subtraction
*	Multiplication
/	Division
:	Comparison
(overbar)	One's complement of
∪	Inclusive-OR, where 0 ∪ 0 = 0, 0 ∪ 1 = 1, 1 ∪ 0 = 1, 1 ∪ 1 = 1
∩	AND, where 0 ∩ 0 = 0, 0 ∩ 1 = 0, 1 ∩ 0 = 0, 1 ∩ 1 = 1
ⓤ	Exclusive-OR, where 0 ⓤ 0 = 0, 0 ⓤ 1 = 1, 1 ⓤ 0 = 1, 1 ⓤ 1 = 0
'(prime)	Denotes the quantity is made odd by ORing 1 to the least significant bit

GLOSSARY OF TERMS

Actual Memory Address

An actual memory location used by the CPU to access and retrieve information.

Analyzed Map Control Flag (AMC)

A one-bit control flag used by the instructions Load Virtual Address (LVA) and Load Actual Address (LAA).

Carry Flag (CF)

Bit 6 of the program status word. This bit is set at the end of an add or subtract instruction if a carry was propagated past the most significant bit of the result. If no carry was propagated, this bit is reset.

Condition Codes (CC1, CC2, CC3)

Bits 2, 3, and 4 of the program status word. These bits are set/reset during normal execution of any compare instruction or any to-memory instruction except store.

Control Register (CR)

One of the eight internal registers that can be accessed only by the privileged control instructions Load Control Register (LCR) and Store Control Register (STCR). These registers contain trap information and the program status stack pointer and limit.

CPU Mode Bits (CM)

Bits 0 and 1 of the program status word that define the master/slave/map mode of machine operation.

Current Stack Pointer (CSP)

A 17-bit memory address, contained in control register 6, that is used by the privileged control instructions Push Program Status (PPS) and Pop Program Status (POPS). The current program status word is stored at the memory location specified by the CSP.

Data Exchange

The TENET 210 unit that connects core memory to all peripheral subsystems (including CPU's). It is also the interface between the CPU and the peripheral processors.

Double Register (R, R')

The even-odd register pair used in multiply, divide, and doubleword instructions.

Effective Address (EA)

The reference address of an instruction after indirect addressing and indexing have been performed. If EA is 0-7, it references a general register; otherwise, EA is a virtual memory address if mapping is enabled, or an actual memory address if mapping is not enabled.

Effective Bit (EB)

In bit instructions, the particular bit in a word upon which the instruction is to operate.

Effective Doubleword (EDW)

For doubleword instructions, the 64-bit word formed by using (EA) for the most significant half and (EA + 1) for the least significant half.

Effective Doubleword Location (EDL)

The location in memory of the effective doubleword.

Effective Field Address (EFA)

For field instructions, the location of the data item.

Effective Field Word (EFW)

The 32-bit word generated for field instructions by right-justifying the required field and extending zeros (for logical operations) or the high-order bit of the field (for arithmetic operations) to bit 0 of the EFW.

Effective Value (EV)

For immediate instructions, the immediate value field of the instruction word (bits 11-31) expanded to a 32-bit word by extending the sign bit of the immediate value (bit 11) to the left.

Effective Word (EW)

The contents of the effective reference address: thus, EW is equivalent to (EA).

Effective Word Location (EWL)

The location in memory of the effective word.

Floating Point Divide by Zero (FD)

Bit 11 of the program status word (see Section 2).

Floating Point Overflow (FO)

Bit 10 of the program status word (see Section 2).

Floating Point Underflow (FU)

Bit 12 of the program status word (see Section 2).

General Register (R)

One of the eight 32-bit general-purpose registers (R0 through R7) in the TENET 210 CPU.

Indirect Map Control Flag (IMC)

Bit 0 of an indirect word that indicates conditional mapping control. When IMC = 1 and the CPU is in the selective mapping mode, the effective address generated from this address modification will be mapped.

Input/Output Control Doubleword (IOCD)

A 64-bit doubleword that contains input/output information used by an IOP, e.g., order to be performed, number of words to be transferred, memory address, and interrupt flags.

Input/Output Processor (IOP)

A unit that provides data transfer between core memory and peripheral input/output devices — card readers, printers, tape units, disc drives, and terminals.

Instruction Register (IR)

A 32-bit register in the CPU used to hold the current instruction obtained from memory while it is being decoded for execution.

Interrupt Enable (IE)

Bit 7 of the program status word that allows the CPU to respond to requests for interrupt.

Memory Map

A group of ten-bit registers, each of which contains an eight-bit actual memory page address code for a specific 512-word page of actual memory, a one-bit write protect flag, and a one-bit page altered flag. The map is available in segments of 64 registers each.

Overflow Flag (OF)

Bit 5 of the program status word that is set if the two's complement result of an arithmetic instruction cannot be unambiguously represented in a word (or doubleword), or sign of register R changes during execution of certain shift instructions.

Page

A 512-word section of virtual memory that is assigned an actual memory page address when memory mapping is in effect. For example, virtual page 1 is assigned the virtual address range 512 - 1023.

Page Protect Override (PO)

Bit 8 of the program status word that allows the CPU to write in write-protected pages when the memory map is used.

Program Counter (PC)

Bits 15-31 of the program status word that points to the location in core memory of the next instruction to be executed in sequence.

Program Status Word (PSW)

A 32-bit word that defines the critical control conditions of the TENET 210 CPU, e.g., mode of operation, condition code indicators, location of the next instruction.

Registers-Stored Flag (RS)

Bit 13 of the program status word used to indicate that the eight general registers were stored with the current PSW.

Special Instruction Recovery Flag (SIR)

Bit 9 of the program status word (see Section 2).

Stack Limit (SLM)

A 17-bit value, contained in control register 7, that is used by the privileged control instructions Push Program Status (PPS) and Pop Program Status (POPS). The SLM specifies the maximum allowable value of the current stack pointer.

Virtual Memory Address

The 17-bit effective address after address modification which is to be conditionally transformed via the memory map into an actual address.

ABBREVIATIONS

AMC	Analyzed Map Control Flag
CC	Condition Codes (bits 2, 3, and 4 of the program status word)
CF	Carry Flag (bit 6 of the program status word)
CM	CPU Mode (bits 0 and 1 of the program status word)
CR	Control Register
CSP	Current Stack Pointer (bits 15-31 of CR 6)
EA	Effective Address
EB	Effective Bit
EDL	Effective Doubleword Location
EDW	Effective Doubleword
EFA	Effective Field Address
EFW	Effective Field Word
EV	Effective Value
EW	Effective Word
EWL	Effective Word Location
FD	Floating Point Divide by Zero (bit 11 of the PSW)
FO	Floating-Point Overflow (bit 10 of the PSW)
FU	Floating-Point Underflow (bit 12 of the PSW)

ABBREVIATIONS (Cont'd)

IE Interrupt Enable (bit 7 of the program status word)

IMC Indirect Map Control Flag

IR Instruction Register

OF Overflow Flag (bit 5 of the program status word)

PC Program Counter (bits 15-31 of the program status word)

PO Page Protect Override (bit 8 of the program status word)

R The address of a general register or the register definition field (bits 8-10 of an instruction)

R, R' Double Register. For multiply, divide, and doubleword instructions, (R, R') is the even-odd register pair.

RS Registers-Stored Flag (bit 13 of the program status word)

SIR Special Instruction Recovery Flag (bit 9 of the program status word)

SLM Stack Limit (bits 15-31 of CR 7)

X The address of an index register or the index register definition field (bits 11-14 of an instruction)

INSTRUCTION DESCRIPTIONS

In the instruction descriptions, the instruction's descriptive name is followed by the mnemonic code used by the TENET 210 assembler to produce the instruction's basic operation code. This code is shown in hexadecimal in bits 0-7 of the instruction word. All registers and storage areas that can be affected by the instruction are listed symbolically after the word "Affected". The program counter (bits 15-31 of the program status word) is not listed since it is updated by 1 as part of every instruction execution.

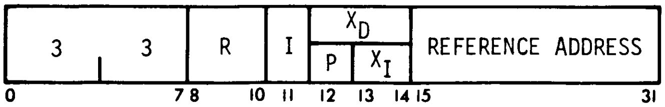
Word Instructions

TENET 210 word instructions (Table 2) operate with information in 32-bit word lengths. Load instructions load the contents of the effective address (EA) into a specified general register, but do not affect core memory. Store instructions store the contents of a general register (R) into a word specified by the effective address, but do not affect the contents of the general register. The fixed-point arithmetic instructions perform binary addition, subtraction, multiplication, and division with integer operands. One operand is specified by the effective address; the second operand is the contents of a general register. The general register contains the result of the operation.

Table 2. Word Instructions

Instruction Name	Mnemonic	Hex. Code	Description	Page
Load Word	LW	33	$EW \rightarrow (R)$	15
Store Word	STW	32	$(R) \rightarrow EW$	15
Add Word	AW	30	$(R) + EW \rightarrow (R)$	15
Subtract Word	SW	31	$(R) - EW \rightarrow (R)$	15
Inverse Subtract Word	ISW	38	$EW - (R) \rightarrow (R)$	15
Multiply Word	MW	3A	$(R) * EW \rightarrow (R)$	15
Divide Word	DW	3B	$(R) / EW \rightarrow (R)$	15
AND Word	ANDW	34	$(R) \cap EW \rightarrow (R)$	15
OR Word	ORW	35	$(R) \cup EW \rightarrow (R)$	16
Exclusive OR Word	EORW	36	$(R) \textcircled{\cup} EW \rightarrow (R)$	16
Exchange Word	XW	42	$(R) \rightarrow EWL; EW \rightarrow (R)$	16
Load Word Doubled	LWD	3C	$EW_{1-31} \rightarrow (R)_{0-30}$ $0 \rightarrow (R)_{31}$ $EW_0 \rightarrow CF$ $EW_0 \textcircled{\cup} EW_1 \rightarrow OF$	16
Compare Word	CW	37	$(R) : EW$	16

Load Word - LW

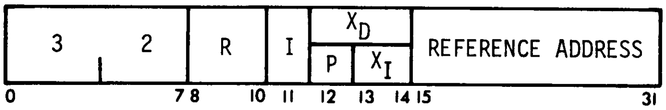


Description: EW → (R)

LW loads the effective word into register R.

Affected: (R)

Store Word - STW

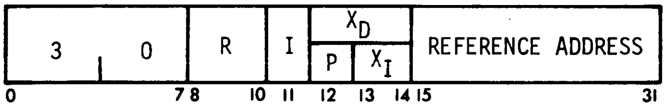


Description: (R) → (EW)

STW stores the contents of register R into the effective word.

Affected: (EA)

Add Word - AW

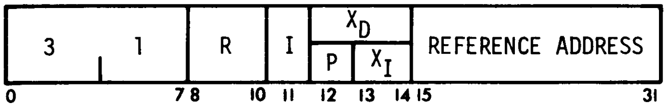


Description: (R) + EW → (R)

AW adds the effective word to the contents of register R and loads the sum in register R.

Affected: (R), OF, CF

Subtract Word - SW

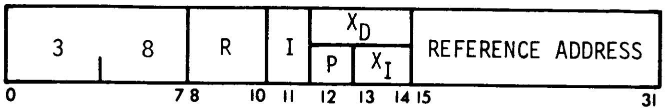


Description: (R) - EW → (R)

SW subtracts the effective word from the contents of register R and loads the difference in register R.

Affected: (R), OF, CF

Inverse Subtract Word - ISW

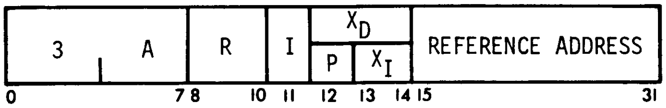


Description: EW - (R) → (R)

ISW subtracts the contents of register R from the effective word and loads the result in register R.

Affected: (R), OF, CF

Multiply Word - MW

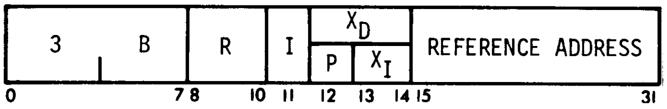


Description: (R) * EW → (R)

MW multiples the contents of register R by the effective word and loads the least significant 32 bits of the product in register R. If the result exceeds 31 significant bits, the overflow flag (OF) is set in the program status word.

Affected: (R), OF

Divide Word - DW

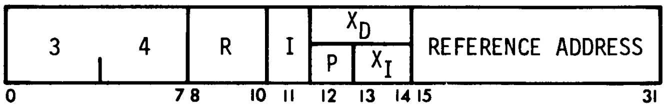


Description: (R)/EW → (R)

DW divides the contents of register R by the effective word and loads the quotient in register R. If overflow occurs (divisor = 0), register R is not disturbed but the overflow flag (OF) is set in the program status word. If the remainder is nonzero, the carry flag (CF) is set in the program status word.

Affected: (R), OF, CF, (CR 0)

AND Word - ANDW



Description: (R) ∩ EW → (R)

ANDW logically ANDs each bit in register R with the corresponding bit of the effective word and loads the result in register R.

Affected: (R)

OR Word - ORW

3 5	R	I	X_D		REFERENCE ADDRESS
			P	X_I	
0 7	8 10	11	12	13 14	15 31

Description: (R) ∪ EW → (R)

ORW logically ORs each bit in register R with the corresponding bit of the effective word and loads the result in register R.

Affected: (R)

Exclusive OR Word - EORW

3 6	R	I	X_D		REFERENCE ADDRESS
			P	X_I	
0 7	8 10	11	12	13 14	15 31

Description: (R) ⓤ EW → (R)

EORW exclusively ORs each bit in register R with the corresponding bit of the effective word and loads the result in register R.

Affected: (R)

Exchange Word - XW

4 2	R	I	X_D		REFERENCE ADDRESS
			P	X_I	
0 7	8 10	11	12	13 14	15 31

Description: (R) → EWL; EW → (R)

XW exchanges the contents of register R with the effective word.

Affected: (R), (EWL)

Load Word Doubled - LWD

3 C	R	I	X_D		REFERENCE ADDRESS
			P	X_I	
0 7	8 10	11	12	13 14	15 31

Description: $EW_{1-31} \rightarrow (R)_{0-30}$
$0 \rightarrow (R)_{31}$
$EW_0 \rightarrow CF$
EW_0 ⓤ $EW_1 \rightarrow OF$

LWD doubles the effective word and loads the result in register R. Bit 0 of the effective word is loaded in the carry flag (CF) of the program status word. If bit 0 and bit 1 of the effective word differ, the overflow flag (OF) in the program status word is set to 1. If bit 0 and 1 are the same, OF is set to 0.

Affected: (R), OF, CF

Compare Word - CW

3 7	R	I	X_D		REFERENCE ADDRESS
			P	X_I	
0 7	8 10	11	12	13 14	15 31

Description: (R) : EW

CW compares the contents of register R with the effective word and sets the condition code indicators as follows:

	Condition Code Indicators		
Result	1	2	3
(R) = EW	-	0	0
(R) < EW	-	0	1
(R) > EW	-	1	0
(R) ∩ EW = 0	0	-	-
(R) ∩ EW ≠ 0	1	-	-

Affected: CC1, CC2, CC3

Doubleword Instructions

With the exception of Multiply Doubleword and Divide Doubleword, which reference single-word operands, the doubleword instructions (Table 2) operate on data in doubleword format. A 64-bit effective doubleword (EDW) is formed by using the effective address (EA) for the most significant half and EA+1 for the least significant half.

If register R is an even register, an even-odd register pair (R, R') is operated upon. If register R is an odd register, the manner in which the instruction is executed varies with the instruction and is described accordingly.

Load Doubleword - LD

5 3	R	I	X_D		REFERENCE ADDRESS
			P	X_I	
0 7	8 10	11	12	13 14	15 31

Description:

If R is even (normal operation): EDW → (R, R')
LD loads the effective doubleword in registers R and R'.

If R is odd: $EDW_{32-63} \rightarrow (R')$
LD loads the least significant half of the effective doubleword in register R'.

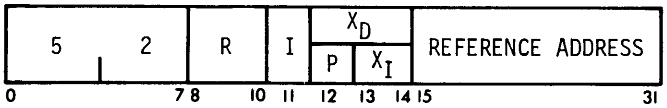
Affected: (R), (R')

Table 3. Doubleword Instructions

Instruction Name	Mnemonic	Hex. Code	Description†	Page
Load Doubleword	LD	53	EDW → (R, R')	16
Store Doubleword	STD	52	(R, R') → EDW	17
Add Doubleword	AD	50	(R, R') + EDW → (R, R')	17
Subtract Doubleword	SD	51	(R, R') - EDW → (R, R')	17
Multiply Doubleword	MD	3E	(R') * EW → (R, R')	17
Divide Doubleword	DD	3F	(R, R') / EW → (R')	18
Compare Doubleword	CD	57	(R, R') : EDW	18

† Normal operation (register R is even)

Store Doubleword - STD



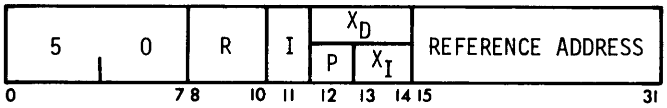
Description:

If R is even (normal operation): (R, R') → EDW
STD stores the double register contents in the effective doubleword.

If R is odd: (R') → EDW_{0-31}; (R') → EDW_{32-63}
STD stores the contents of register R' in both halves of the effective doubleword.

Affected: (EDL)

Add Doubleword - AD



Description:

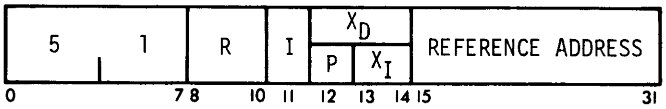
If R is even (normal operation): (R, R') + EDW → (R, R')
AD adds the double register contents to the effective doubleword, and loads the sum in registers R, R'.

If R is odd: EDW_{32-63} + (R') → (R')
AD adds the contents of register R' to the least significant half of the effective doubleword, and loads the sum in register R'.

In either of the above cases, the overflow and carry flags (OF and CF) in the program status word are defined by the overflow and carry from the operation EDW_{0-31} + (R) + carry from [EDW_{32-63} + (R')] .

Affected: (R), (R'), OF, CF

Subtract Doubleword - SD



Description:

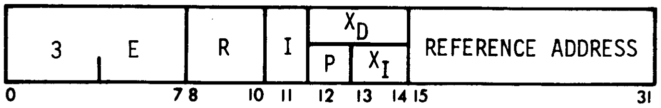
If R is even (normal operation): (R, R') - EDW → (R, R')
SD subtracts the effective doubleword from the double register contents, and loads the difference in registers R, R'.

If R is odd: (R') - EDW_{32-63} → (R')
SD subtracts the least significant half of the effective doubleword from register R', and loads the difference in register R'.

In either of the above cases, the overflow and carry flags (OF and CF) in the program status word are defined by the overflow and carry from the operation (R) + EDW_{32-63} + carry from [(R') + (EDW_{32-63} + 1)] .

Affected: (R), (R'), OF, CF

Multiply Doubleword - MD



Description:

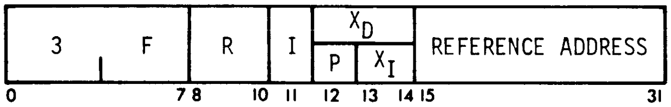
If R is even (normal operation): (R') * EW → (R, R')
MD multiplies the contents of register R' by the effective word and loads the 64-bit, signed two's-complement result in double register R, R'. The most significant half of the product is loaded in register R, the least significant half in register R'.

If R is odd: (R') * EW → (R')
MD multiplies the contents of register R' by the

effective word and replaces the contents of register R' with the least significant half of the product. The most significant half of the product is lost.

Affected: (R), (R'), OF

Divide Doubleword - DD



Description:

If R is even: (R, R')/EW → Quotient → (R');
Remainder → (R)

DD divides the contents of the double register R, R' by the effective word. The remainder is loaded in register R and the quotient in register R'. If overflow occurs, (R, R') are not disturbed; a non-zero remainder has the same sign as the dividend.

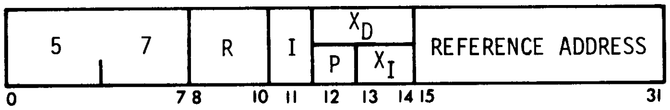
If R is odd: (R')/EW → Quotient → (R')

The sign of register R' is extended to provide the most significant half of the double-register dividend and the contents of register R' provide the least significant half of the double-register dividend. DD divides the contents of this double register by the effective word and loads the quotient into register R'; the remainder is lost. If overflow occurs, R' is not disburbed.

Overflow occurs if: $|\text{dividend}|*2^{-31} \geq |\text{divisor}|$

Affected: (R), (R'), OF, (CR0)

Compare Doubleword - CD



Description:

If R is even (normal operation): (R, R') : EDW
CD compares the contents of the double register R, R' with the effective doubleword, and sets the condition code indicators as shown below according to the result of the comparison.

If R is odd: (R', R') : EDW
The contents of register R' are used for both halves of the double word comparison. CD compares the contents of this double word with the effective doubleword, and sets the condition code indicators as shown below according to the result of the comparison.

Result	CC1	CC2	CC3
(R, R') = EDW	-	0	0
(R, R') < EDW	-	0	1
(R, R') > EDW	-	1	0
(R, R') ∩ EDW = 0	0	-	-
(R, R') ∩ EDW ≠ 0	1	-	-

Affected: CC1, CC2, CC3

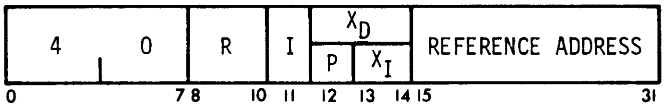
Memory Instructions

The results of all memory instructions (Table 3) except Store Immediate to Memory and Compare Immediate with Memory are tested for positive, negative, zero, even, and odd, and the condition code indicators are affected as follows:

	Condition Code Indicators		
Result	1	2	3
Zero	0	0	0
Positive	-	1	0
Negative	-	0	1
Even	0	-	-
Odd	1	-	-

Store Immediate to Memory never affects the condition code indicators; the results of a Compare Immediate with Memory are described under that instruction.

Add to Memory - AM

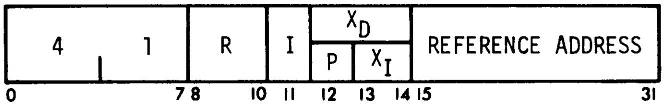


Description: EW + (R) → EWL

AM adds the contents of R to the effective word and stores the result in the effective word location.

Affected: (EWL), OF, CF, CC1, CC2, CC3

Subtract from Memory - SM



Description: EW - (R) → EWL

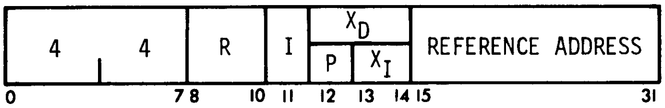
SM subtracts the contents of R from the effective word and stores the result in the effective word location.

Affected: (EWL), OF, CF, CC1, CC2, CC3

Table 4. Memory Instructions

Instruction Name	Mnemonic	Hex. Code	Description	Page
Add to Memory	AM	40	EW + (R) → EWL	18
Subtract from Memory	SM	41	EW - (R) → EWL	18
AND to Memory	ANDM	44	EW ∩ (R) → EWL	19
OR to Memory	ORM	45	EW ∪ (R) → EWL	19
Exclusive-OR to Memory	EORM	46	EW ⓤ (R) → EWL	19
Increment and Load Word	ILW	43	EW + 1 → R → EWL	19
Complement and Add Immediate to Memory	CAIM	64	$\overline{EW}$ + R → EWL	19
Add Immediate to Memory	AIM	60	EW + R → EWL	20
Subtract Immediate from Memory	SIM	61	EW - R → EWL	20
Store Immediate to Memory	STIM	62	R → EWL	20
Compare Immediate with Memory	CIM	67	R : EW	20

AND to Memory - ANDM

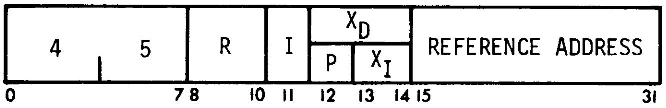


Description: EW ∩ (R) → EWL

ANDM logically ANDs each bit of (R) and the corresponding bit of the effective word and places the result in the effective word location.

Affected: (EWL), CC1, CC2, CC3

OR to Memory - ORM

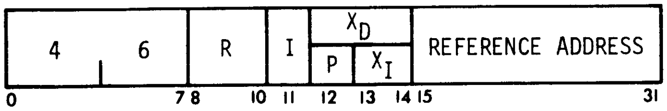


Description: EW ∪ (R) → EWL

ORM logically ORs each bit of (R) and the corresponding bit of the effective word and places the result in the effective word location.

Affected: (EWL), CC1, CC2, CC3

Exclusive-OR to Memory - EORM

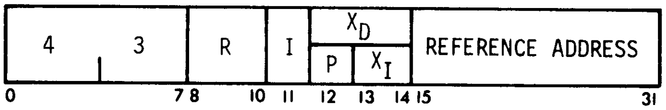


Description: EWⓤ(R) → EWL

EORM exclusively ORs each bit of (R) and the corresponding bit of the effective word and places the result in the effective word location.

Affected: (EWL), CC1, CC2, CC3

Increment and Load Word - ILW

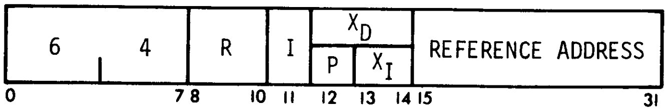


Description: EW + 1 → (R); EW + 1 → EWL

ILW adds 1 to the effective word and places the result in the effective word location and register R.

Affected: (EWL), (R), OF, CF, CC1, CC2, CC3

Complement and Add Immediate to Memory - CAIM

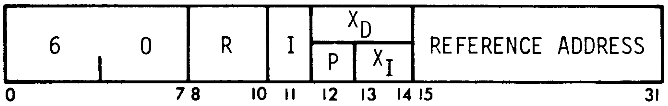


Description: $\overline{EW}$ + R → EWL

CAIM adds the absolute value of the three-bit R field to the one's complement of the effective word, and places the result in the effective word location. Thus, R=0 performs the one's complement of the effective word; R=1 performs the two's complement of the effective word.

Affected: (EWL), OF, CF, CC1, CC2, CC3

Add Immediate to Memory - AIM

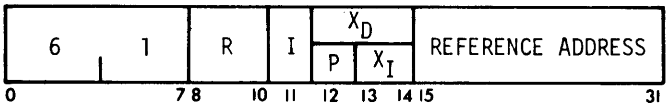


Description: EW + R → EWL

AIM adds the absolute value of the three-bit R field to the effective word, and places the result in the effective word location.

Affected: (EWL), OF, CF, CC1, CC2, CC3

Subtract Immediate from Memory - SIM

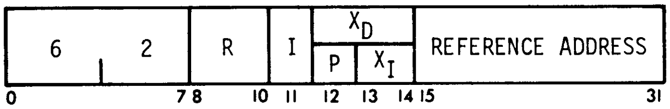


Description: EW - R → EWL

SIM subtracts the absolute value of the three-bit R field from the effective word, and places the result in the effective word location. When R=0, its absolute value is 8.

Affected: (EWL), OF, CF, CC1, CC2, CC3

Store Immediate to Memory - STIM

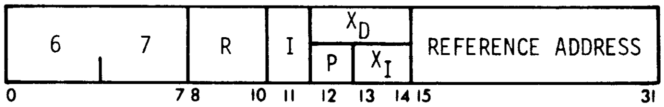


Description: R → EWL

STIM stores the absolute value of the three-bit R field, right-justified with leading zeros, in the effective word location.

Affected: (EWL)

Compare Immediate with Memory - CIM



Description: R : EW

CIM compares the absolute value of the three-bit R field with the effective word, and sets the condition code indicators as follows:

	Condition Code Indicators		
Result	1	2	3
R = EW	-	0	0
R < EW	-	0	1
R > EW	-	1	0
R ∩ EW = 0	0	-	-
R ∩ EW ≠ 0	1	-	-

Affected: CC1, CC2, CC3

Immediate Instructions

Immediate instructions (Table 5) provide for an operand within the instruction word itself and not in a general register or core memory location. Bit 11 is the sign of the immediate value and is extended to the left to form the effective value (EV) — a 32-bit, signed two's complement, integer value:

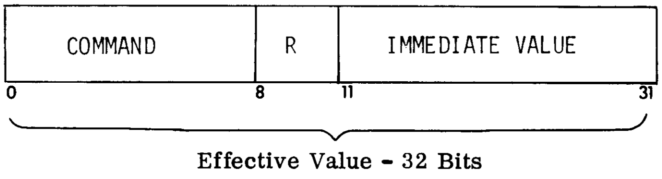


Table 5. Immediate Instructions

Instruction Name	Mnemonic	Hex. Code	Description	Page
Load Immediate	LI	73	EV → (R)	21
Add Immediate	AI	70	(R) + EV → (R)	21
Subtract Immediate	SUBI	71	(R) - EV → (R)	21
Multiply Immediate	MI	7A	(R) * EV → (R)	21
Divide Immediate	DI	7B	(R)/EV → (R)	21
AND Immediate	ANDI	74	(R) ∩ EV → (R)	21
OR Immediate	ORI	75	(R) ∪ EV → (R)	21
Exclusive-OR Immediate	EORI	76	(R)ⓤ EV → (R)	21
Compare Immediate	CI	77	(R) : EV	22
Inverse Subtract Immediate	ISI	78	EV - (R) → (R)	22

Load Immediate - LI

7 3	R	IMMEDIATE VALUE
0 7	8 10	11 31

Description: EV →(R)

LI loads the effective value into register R.

Affected: (R)

Add Immediate - AI

7 0	R	IMMEDIATE VALUE
0 7	8 10	11 31

Description: (R) + EV → (R)

AI adds the contents of register R to the effective value and places the result in register R.

Affected: (R), OF, CF

Subtract Immediate - SUBI

7 1	R	IMMEDIATE VALUE
0 7	8 10	11 31

Description: (R) − EV → (R)

SUBI subtracts the effective value from the contents of register R and places the result in register R.

Affected: (R), OF, CF

Multiply Immediate - MI

7 A	R	IMMEDIATE VALUE
0 7	8 10	11 31

Description: (R) * EV → (R)

MI multiplies the contents of register R by the effective value and places the result in register R. If the result exceeds 31 significant bits, the overflow flag (OF) is set in the program status word.

Affected: (R), OF

Divide Immediate - DI

7 B	R	IMMEDIATE VALUE
0 7	8 10	11 31

Description: (R)/EV → (R)

DI divides the contents of register R by the effective value and places the quotient in register R. If overflow occurs (divisor = 0), register R is not disturbed but the overflow flag (OF) is set in the program status word. If there is a non-zero remainder, the carry flag (CF) is set in the program status word.

Affected: (R), OF, CF, (CR 0)

AND Immediate - ANDI

7 4	R	IMMEDIATE VALUE
0 7	8 10	11 31

Description: (R) ∩ EV → (R)

ANDI logically ANDs each bit of register R and the corresponding bit of the effective value and places the result in register R.

Affected: (R)

OR Immediate - ORI

7 5	R	IMMEDIATE VALUE
0 7	8 10	11 31

Description: (R) ∪ EV → (R)

ORI logically ORs each bit of register R and the corresponding bit of the effective value and places the result in register R.

Affected: (R)

Exclusive-OR Immediate - EORI

7 6	R	IMMEDIATE VALUE
0 7	8 10	11 31

Description: $(R) \textcircled{\cup} EV \rightarrow (R)$

EORI exclusively-ORs each bit of register R and the corresponding bit of the effective value and places the result in register R.

Affected: (R)

Compare Immediate - CI

7 7	R	IMMEDIATE VALUE	
0	7 8	10 11	31

Description: (R) : EV

CI compares the contents of register R with the effective value and sets the condition code indicators as follows:

	Condition Code Indicators		
Result	1	2	3
$(R) = EV$	-	0	0
$(R) < EV$	-	0	1
$(R) > EV$	-	1	0
$(R) \cap EV = 0$	0	-	-
$(R) \cap EV \neq 0$	1	-	-

Affected: CC1, CC2, CC3

Inverse Subtract Immediate - ISI

7 8	R	IMMEDIATE VALUE	
0	7 8	10 11	31

Description: $EV - (R) \rightarrow (R)$

ISI subtracts the contents of register R from the effective value and places the result in register R.

Affected: (R), OF, CF

Shift Instructions

Shift instructions (Table 6) have the memory reference format. After the effective address is generated, bits 26-31 define the shift count — a non-negative value in the range $0\text{-}63_{10}$ †:

COMMAND	R	I	X_D			SHIFT COUNT	
			P	X_I			
0	8	11	12	14	15	26	31

All double register shift operations treat registers R and R' as a 64-bit register with the high-order bit (bit position 0 of register R) as the sign for the entire doubleword. When a doubleword shift instruction addresses an odd register, the contents of register R represent both the 32 high-order bits and the 32 low-order bits of the doubleword. The shift operation is then performed, and the 32 low-order bits of the result are loaded into register R.

Shift Arithmetic Left - SAL

2 8	R	I	X_D		REFERENCE ADDRESS	
			P	X_I		
0	7 8	10 11	12	13	14 15	31

Description:

SAL shifts the contents of register R the specified number of positions to the left. Vacated low-order bit positions are set to zero, and the overflow flag (OF) is set if the sign of R changes.

Affected: (R), OF

Shift Arithmetic Left Double - SALD

2 9	R	I	X_D		REFERENCE ADDRESS	
			P	X_I		
0	7 8	10 11	12	13	14 15	31

† Shift Until Normalized Double (SUND) does not use the shift count.

Table 6. Shift Instructions

Instruction Name	Mnemonic	Hex. Code	Page
Shift Arithmetic Left	SAL	28	22
Shift Arithmetic Left Double	SALD	29	22
Shift Arithmetic Right	SAR	2A	23
Shift Arithmetic Right Double	SARD	2B	23
Shift Logical Left	SLL	20	23
Shift Logical Left Double	SLLD	21	24
Shift Logical Right	SLR	22	24
Shift Logical Right Double	SLRD	23	24
Shift Circular Left	SCL	2C	24
Shift Circular Left Double	SCLD	2D	24
Shift Circular Right	SCR	2E	24
Shift Circular Right Double	SCRD	2F	24
Shift Until Normalized	SUN	24	25
Shift Until Normalized Double	SUND	25	25

Description:

SALD shifts the contents of registers R, R' the specified number of positions to the left. Vacated low-order bit positions are set to zero, and the overflow flag (OF) is set if the sign of R changes.

Affected: (R), (R'), OF

Shift Arithmetic Right - SAR

				X_D		
2	A	R	I	P	X_I	REFERENCE ADDRESS
0	7	8 10	11	12	13 14	15 31

Description:

SAR shifts the contents of register R the specified number of positions to the right. Vacated high-order bit positions are filled in with the sign of R (bit 0 of the word).

Affected: (R)

Shift Arithmetic Right Double - SARD

				X_D		
2	B	R	I	P	X_I	REFERENCE ADDRESS
0	7	8 10	11	12	13 14	15 31

Description:

SARD shifts the contents of registers R, R' the specified number of positions to the right. Vacated high-order bit positions are filled in with the sign of R (bit 0 of the doubleword).

Affected: (R), (R')

Shift Logical Left - SLL

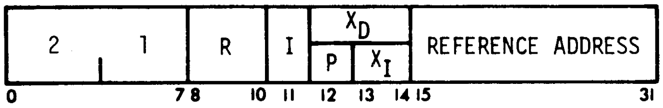
				X_D		
2	0	R	I	P	X_I	REFERENCE ADDRESS
0	7	8 10	11	12	13 14	15 31

Description:

SLL shifts the contents of register R the specified number of positions to the left. Vacated bit positions are set to zero and bits shifted off the end of the word are lost.

Affected: (R)

Shift Logical Left Double - SLLD

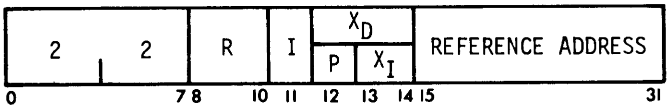


Description:

SLLD shifts the contents of registers R, R' the specified number of positions to the left. Vacated bit positions are set to zero and bits shifted off the end of the doubleword are lost.

Affected: (R), (R')

Shift Logical Right - SLR

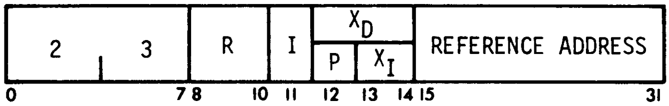


Description:

SLR shifts the contents of register R the specified number of positions to the right. Vacated bit positions are set to zero and bits shifted off the end of the word are lost.

Affected: (R)

Shift Logical Right Double - SLRD

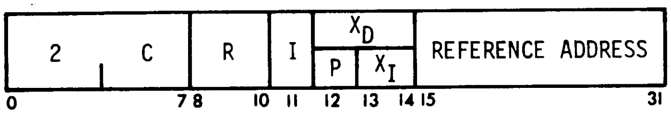


Description:

SLRD shifts the contents of registers R, R' the specified number of positions to the right. Vacated bit positions are set to zero and bits shifted off the end of the doubleword are lost.

Affected: (R), (R')

Shift Circular Left - SCL

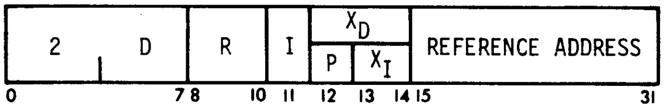


Description:

SCL shifts the contents of register R the specified number of positions to the left. The shift is performed in a circular manner such that bits shifted out of bit position 0 are shifted into bit position 31.

Affected: (R)

Shift Circular Left Double - SCLD

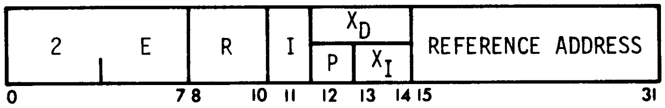


Description:

SCLD shifts the contents of registers R, R' the specified number of positions to the left. The shift is performed in a circular manner such that bits shifted out of bit position 0 are shifted into bit position 63.

Affected: (R), (R')

Shift Circular Right - SCR

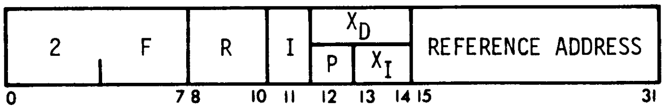


Description:

SCR shifts the contents of register R the specified number of positions to the right. The shift is performed in a circular manner such that bits shifted out of bit position 31 are shifted into bit position 0.

Affected: (R)

Shift Circular Right Double - SCRD



Description:

SCRD shifts the contents of registers R, R' the specified number of positions to the right. The shift is performed in a circular manner such that bits shifted out of bit position 63 are shifted into bit position 0.

Affected: (R), (R')

Shift Until Normalized - SUN

				X_D		
2	4	R	I	P	X_I	REFERENCE ADDRESS
0	7	8 10	11	12	13 14	15 31

Description:

SUN shifts the contents of register R either 31 positions to the left, or until the sign of the word differs from the state of the most significant bit of the magnitude. At the termination of the shift, the contents of X are decremented by the number of positions shifted, and vacated low-order bit positions are set to zero.

Affected: (R), (X)

Shift Until Normalized Double - SUND

				X_D		
2	5	R	I	P	X_I	REFERENCE ADDRESS
0	7	8 10	11	12	13 14	15 31

Description:

SUND shifts the contents of registers R, R' either 63 positions to the left, or until the sign of the doubleword differs from the state of the most significant bit of the magnitude. At the termination of the shift, the contents of X are decremented by the number of positions shifted, and vacated low-order bit positions are set to zero.

Affected: (R), (R'), (X)

Branch Instructions

TENET 210 branch instructions (Table 7) alter program sequence, either conditionally or unconditionally. If a branch is unconditional or if a branch is conditional and the condition is met, the instruction to which the effective reference address points becomes the next instruction executed. However, if a condition for a branch is not met, the next instruction executed is that in the next sequential location following the branch instruction.

A branch to locations 0-7 (index registers) is not allowed and will cause a trap to location 8 if the branch conditions are satisfied.

Branch on Arithmetic Conditions - BAC

				X_D		
8	7	R	I	P	X_I	REFERENCE ADDRESS
0	7	8 10	11	12	13 14	15 31

Description:

The R field specifies the conditions under which a branch to the effective address will occur. These conditions are based upon the status of the overflow and carry flags (OF and CF) in the program status word. Extensions of the BAC instruction, implemented in the Meta Assembler, will set the appropriate bits in the R field.

Ext'd Branch Mnemonic	R_8	R_9	R_{10}	Branch Conditions
B	0	0	0	Unconditional Branch
BNO	0	0	1	Branch if No Overflow
BNC	0	1	0	Branch if No Carry
BNCO	0	1	1	Branch if No Carry and No Overflow
NOP	1	0	0	No Operation
BOV	1	0	1	Branch if Overflow
BCRY	1	1	0	Branch if Carry
BCO	1	1	1	Branch if Carry or Overflow

Affected: PC

Branch and Link - BAL

				X_D		
8	6	R	I	P	X_I	REFERENCE ADDRESS
0	7	8 10	11	12	13 14	15 31

Description: $0 \rightarrow (R)_{0-14}$ $PC \rightarrow (R)_{15-31}$

BAL causes an unconditional branch to the effective address. The virtual address of the instruction immediately following the BAL instruction is stored in bits 15-31 of register R. Bits 0-14 of register R are reset to 0.

Affected: (R), PC

Branch on Conditions Set - BCS

				X_D		
8	5	R	I	P	X_I	REFERENCE ADDRESS
0	7	8 10	11	12	13 14	15 31

Description: Branch if and only if

$(R_8 \cap CC1) \cup (R_9 \cap CC2) \cup (R_{10} \cap CC3) = 1$

Branch if and only if the logical AND of the three-bit R field and the three-bit condition code is nonzero.

Table 7. Branch Instructions

Instruction Name	Mnemonic	Hex. Code - R Field	Description/Branch Condition	Page
Branch on Arithmetic Conditions	BAC	87		25
Extended BAC	B	87-0	Unconditional Branch	
	BNO	87-1	Branch if No Overflow	
	BNC	87-2	Branch if No Carry	
	BNCO	87-3	Branch if No Overflow and No Carry	
	NOP	87-4	No Operation	
	BOV	87-5	Branch if Overflow	
	BCRY	87-6	Branch if Carry	
	BCO	87-7	Branch if Carry or Overflow	
Branch and Link	BAL	86	$0 \rightarrow (R)_{0-14}$ PC $\rightarrow (R)_{15-31}$	25
Branch on Conditions Set	BCS	85	Branch if and only if: $(R_8 \cap CC1) \cup (R_9 \cap CC2) \cup (R_{10} \cap CC3) = 1$	25
Extended BCS	-	85-0	Never Branch	
	BL	85-1	(R) < EW = 1	
	BG	85-2	(R) > EW = 1	
	BNE	85-3	(P) ≠ EW = 1	
	BMO, BBS	85-4	Bit Compare = 1 (also set/reset by test bit instructions)	
	-	85-5	(R) < EW = 1 or Bit Compare = 1	
	-	85-6	(R) > EW = 1 or Bit Compare = 1	
	-	85-7	(R) ≠ EW = 1 or Bit Compare = 1	
Branch on Conditions Reset	BCR	84	Branch if and only if: $(R_8 \cap CC1) \cup (R_9 \cap CC2) \cup (R_{10} \cap CC3) = 0$	27
Extended BCR	-	84-0	Unconditional Branch	
	BGE	84-1	(R) ≥ EW = 0	
	BLE	84-2	(R) ≤ EW = 0	
	BE	84-3	(R) = EW = 0	
	BNMO, BBR	84-4	Bit Compare = 0 (also set/reset by test bit instructions)	
	-	84-5	(R) ≥ EW = 0 or Bit Compare = 0	
	-	84-6	(R) ≤ EW = 0 or Bit Compare = 0	
	-	84-7	(R) = EW = 0 or Bit Compare = 0	
Branch on Register Conditions Set	BRCS	81	Branch if and only if: $(V_8 \cap (R)_{31}) \cup (V_9 \cap (R) > 0) \cup (V_{10} \cap (R)_0) = 1$	27
Extended BRCS	-	81-0	Never Branch	
	BRL	81-1	Register < 0	
	BRG	81-2	Register > 0	
	BRNE	81-3	Register ≠ 0	
	BROD	81-4	Register Odd	
	-	81-5	Register Odd or Negative	
	-	81-6	Register Odd or Positive	
	-	81-7	Register ≠ 0 or Odd	
Branch on Register Conditions Reset	BRCR	80	Branch if and only if: $(V_8 \cap (R)_{31}) \cup (V_9 \cap (R) > 0) \cup (V_{10} \cap (R)_0) = 0$	27
Extended BRCR	-	80-0	Unconditional Branch	
	BRGE	80-1	Register ≥ 0	
	BRLE	80-2	Register ≤ 0	
	BRE	80-3	Register = 0	
	BREV	80-4	Register Even	
	-	80-5	Register Even and Non-negative	
	-	80-6	Register Even and Non-positive	
	-	80-7	Register = 0 and Even	
Branch on Incrementing Register	BIR	82	$(R) + V \rightarrow (R)$	28
Branch on Decrementing Register	BDR	83	$(R) - V \rightarrow (R)$	28

The following is a list of extended branch mnemonics which the TENET 210 Meta Assembler allows and generates a BCS instruction with the appropriate R field value.

Ext'd Branch Mnemonic	R_8	R_9	R_{10}	Result of Comparison	Condition Code
	0	0	0	Never branch	xxx
BL	0	0	1	(R) < EW	xx1
BG	0	1	0	(R) > EW	x1x
BNE	0	1	1	(R) ≠ EW	x1x or xx1
BMO BBS	1	0	0	Bit Compare	1xx
	1	0	1	(R) < EW or Bit Compare	1xx or xx1
	1	1	0	(R) > EW or Bit Compare	1xx or x1x
	1	1	1	(R) ≠ EW or Bit Compare	1xx or x1x or xx1

Affected: PC

Branch on Conditions Reset - BCR

8	4	R	I	X_D		REFERENCE ADDRESS
				P	X_I	

0 7 8 10 11 12 13 14 15 31

Description: Branch if and only if:

$(R_8 \cap CC1) \cup (R_9 \cap CC2) \cup (R_{10} \cap CC3) = 0$

Branch if and only if the logical AND of the three-bit R field and the three-bit condition code is zero.

The following is a list of extended branch mnemonics which the TENET 210 Meta Assembler allows and generates a BCR instruction with the appropriate R field value.

Ext'd Branch Mnemonic	R_8	R_9	R_{10}	Result of Comparison	Condition Code
	0	0	0	Unconditional Branch	xxx
BGE	0	0	1	(R) ≥ EW	x10
BLE	0	1	0	(R) ≤ EW	x01
BE	0	1	1	(R) = EW	x00
BNMO BBR	1	0	0	No Bit Compare	0xx
	1	0	1	(R) ≥ EW and No Bit Compare	010
	1	1	0	(R) ≤ EW and No Bit Compare	001
	1	1	1	(R) = EW and No Bit Compare	000

Affected: PC

Branch on Register Conditions Set - BRCS

8	1	R	I	X_D		REFERENCE ADDRESS
				P	X_I	

0 7 8 10 11 12 13 14 15 31

Description: Branch if and only if:

$(V_8 \cap (R)_{31}) \cup (V_9 \cap (R) > 0) \cup (V_{10} \cap (R)_0) = 1$

Branch if and only if the designated register meets the above condition. These conditions are best described by the extended branch mnemonics for this instruction.

This instruction format is unique in that the register is designated by bits 12-14. Indirect addressing is permitted; whereas, indexing is not allowed.

The following is a list of extended branch mnemonics which the TENET 210 Meta Assembler allows and generates a BRCS instruction with the appropriate V field value.

Ext'd Branch Mnemonic	R_8	R_9	R_{10}	Branch Conditions
	0	0	0	Never Branch
BRL	0	0	1	Branch if Register Less Than Zero
BRG	0	1	0	Branch if Register Greater Than Zero
BRNE	0	1	1	Branch if Register Not Equal Zero
BROD	1	0	0	Branch if Register Odd
	1	0	1	Branch if Register Negative or Odd
	1	1	0	Branch if Register Positive or Odd
	1	1	1	Branch if Register Not Equal Zero or Odd

Affected: PC

Branch on Register Conditions Reset - BRCR

8	9	R	I	X_D		REFERENCE ADDRESS
				P	X_I	

0 7 8 10 11 12 13 14 15 31

Description: Branch if and only if:

$$(V_8 \cap (R)_{31}) \cup (V_9 \cap (R) > 0) \cup (V_{10} \cap (R)_0) = 0$$

Branch if and only if the designated register meets the above condition. These conditions are best described by the extended branch mnemonics for this instruction.

The instruction format is unique in that the register is designated by bits 12-14. Indirect addressing is permitted; whereas, indexing is not allowed.

The following is a list of extended branch mnemonics which the TENET 210 Meta Assembler allows and generates a BRCR instruction with the appropriate V field value.

Ext'd Branch Mnemonic	R_8	R_9	R_{10}	Branch Conditions
	0	0	0	Unconditional Branch
BRGE	0	0	1	Branch if Register Greater Than or Equal Zero
BRLE	0	1	0	Branch if Register Less Than or Equal Zero
BRE	0	1	1	Branch if Register Equal Zero
BREV	1	0	0	Branch if Register Even
	1	0	1	Branch if Register Even and Non-negative
	1	1	0	Branch if Register Even and Non-positive
	1	1	1	Branch if Register Zero and even

Affected: PC

Branch on Incrementing Register - BIR

8 2	R	I	X_D		REFERENCE ADDRESS
			P	X_I	
0 7	8 10	11	12	13 14	15 31

Description: (R) + V → (R)

BIR adds the absolute value contained in bits 11-14 of the instruction register to the contents of register R. The sum replaces the contents of register R. The branch will not occur if $(R)_0$ changes state.

This instruction does not permit indirect addressing or indexing.

Affected: (R), PC

Branch on Decrementing Register - BDR

8 3	R	I	X_D		REFERENCE ADDRESS
			P	X_I	
0 7	8 10	11	12	13 14	15 31

Description: (R) - V → (R)

BDR subtracts the absolute value contained in bits 11-14 of the instruction register from the contents of register R. The difference replaces the contents of register R. The branch will not occur if $(R)_0$ changes state.

This instruction does not permit indirect addressing or indexing.

Affected: (R), PC

Bit Instructions

TENET 210 bit instructions (Table 8) operate upon a particular bit in the effective word, i.e., the word specified in the reference address field of the instruction. The bit position of interest, i.e., the effective bit (EB), is specified in one of two places, depending upon the instruction type. In test bit instructions, the effective bit is defined in bits 6-10 of the instruction word; in test register-designated bit instructions, the contents of the register specified in the R field of the instruction defines the effective bit.

Test Bit Instructions

The instruction word of test bit instructions has the following format:

COMMAND	EFFECTIVE BIT	I	X_D		REFERENCE ADDRESS
			P	X_I	
0 5	6 10	11	12	13 14	15 31

The Effective Bit (EB) is the bit position of interest (0-31) in the effective word. The remainder of the instruction word has the same format as memory reference instructions.

Test Bit - TB

0 4	EB	I	X_D		REFERENCE ADDRESS
			P	X_I	
0 5	6 10	11	12	13 14	15 31

Description: EW_{EB} → CC1

TB sets the condition code indicator CC1 to the state of the effective bit.

Affected: CC1

Test Bit and Reset - TBR

0 5	EB	I	X_D		REFERENCE ADDRESS
			P	X_I	
0 5	6 10	11	12	13 14	15 31

Table 8. Bit Instructions

TEST BIT

Instruction Name	Mnemonic	Hex. Code	Description	Page
Test Bit	TB	04	$EW_{EB} \rightarrow CC1$	28
Test Bit and Reset	TBR	05	$EW_{EB} \rightarrow CC1;$ $0 \rightarrow EW_{EB}$	28
Test Bit and Set	TBS	06	$EW_{EB} \rightarrow CC1;$ $1 \rightarrow EW_{EB}$	29
Store Bit	STB	07	$CC1 \rightarrow EW_{EB}$	29

REGISTER-DESIGNATED BIT

Instruction Name	Mnemonic	Hex. Code	Description	Page
Test Register-Designated Bit	TRB	03	$EW_{EB} \rightarrow CC1$	30
Test Register-Designated Bit and Reset	TRBR	07	$EW_{EB} \rightarrow CC1;$ $0 \rightarrow EW_{EB}$	30
Test Register-Designated Bit and Set	TRBS	0B	$EW_{EB} \rightarrow CC1;$ $1 \rightarrow EW_{EB}$	30
Store Register-Designated Bit	STRB	0F	$CC1 \rightarrow EW_{EB}$	30
Find and Reset Bit	FRB	26		30
Find and Reset Bit Double	FRBD	27		30

Description: $EW_{EB} \rightarrow CC1;\ 0 \rightarrow EW_{EB}$

TBR sets the condition code indicator CC1 to the state of the effective bit, then resets the effective bit to zero.

Affected: CC1, EW_{EB}

Test Bit and Set - TBS

<table>
<tr><td rowspan="2">0 6</td><td rowspan="2">EB</td><td rowspan="2">I</td><td colspan="2">X_D</td><td rowspan="2">REFERENCE ADDRESS</td></tr>
<tr><td>P</td><td>X_I</td></tr>
</table>

0 5 6 10 11 12 13 14 15 31

Description: $EW_{EB} \rightarrow CC1;\ 1 \rightarrow EW_{EB}$

TBS sets the condition code indicator CC1 to the state of the effective bit, then sets the effective bit to 1.

Affected: CC1, EW_{EB}

Store Bit - STB

<table>
<tr><td rowspan="2">0 7</td><td rowspan="2">EB</td><td rowspan="2">I</td><td colspan="2">X_D</td><td rowspan="2">REFERENCE ADDRESS</td></tr>
<tr><td>P</td><td>X_I</td></tr>
</table>

0 5 6 10 11 12 13 14 15 31

Description: $CC1 \rightarrow EW_{EB}$

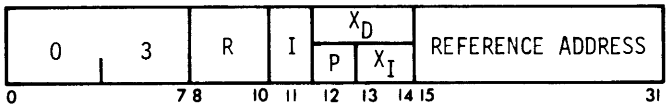
STB sets the effective bit to the state of the condition code indicator CC1.

Affected: EW_{EB}

Register-Designated Bit Instructions

Register-designated bit instructions have the memory reference instruction format. When reference address modification is complete, the effective address specifies the address of the word to be tested. Bits 27 through 31 of register R define the effective bit, i.e., $EB = (R)_{27-31}$.

Test Register-Designated Bit - TRB

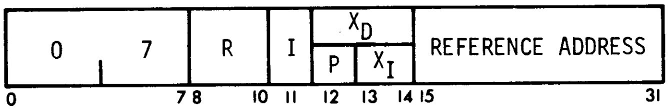


Description: $EW_{EB} \rightarrow CC1$

TRB sets the condition code indicator CC1 to the state of the effective bit.

Affected: CC1

Test Register-Designated Bit and Reset - TRBR

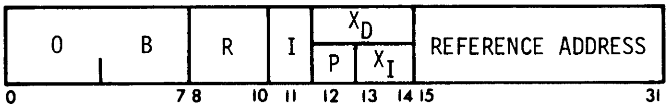


Description: $EW_{EB} \rightarrow CC1; 0 \rightarrow EW_{EB}$

TRBR sets the condition code indicator CC1 to the state of the effective bit, then resets the effective bit to zero.

Affected: CC1, EW_{EB}

Test Register-Designated Bit and Set - TRBS

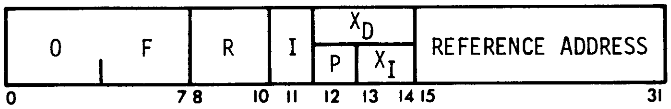


Description: $EW_{EB} \rightarrow CC1; 1 \rightarrow EW_{EB}$

TRBS sets the condition code indicator CC1 to the state of the effective bit, then sets the effective bit to 1.

Affected: CC1, EW_{EB}

Store Register-Designated Bit - STRB

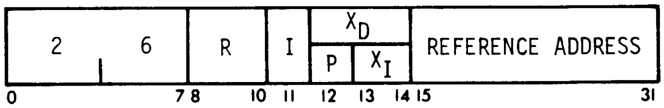


Description: $CC1 \rightarrow EW_{EB}$

STRB sets the effective bit to the state of the condition code indicator CC1.

Affected: EW_{EB}

Find and Reset Bit - FRB

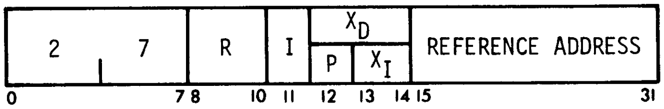


Description:

FRB searches the effective word beginning at the bit specified by bits 27-31 of register R, and proceeds to the right until either a 1 bit is found or the least significant bit has been tested. If a 1 bit is found, it is reset to 0, CC1 is set to 1, and the bit position is stored in register R. If no 1 bit is found, CC1 is reset to 0, and register R is not disturbed.

Affected: CC1, EW, (R)

Find and Reset Bit Double - FRBD



Description:

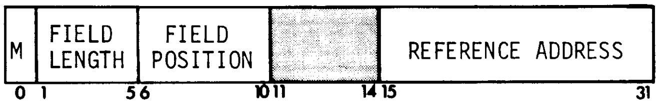
FRBD searches the effective doubleword (EDW) beginning at the bit specified by bits 26-31 or register R, and proceeds to the right until either a 1 bit is found or the least significant bit has been tested. If a 1 bit is found, it is reset to 0, CC1 is set to 1, and the bit position is stored in register R. If no 1 bit is found, CC1 is reset to 0, and register R is not disturbed.

Affected: CC1, EDW, (R)

Field Instructions

TENET 210 field instructions (Table 9) operate on fields packed within a word. Field instructions have a memory reference instruction format, but the reference address is always a direct address which references a field pointer rather than a data item. The data item, which contains the field to be operated upon, is located at the effective field address (EFA). The EFA is the effective address generated from the reference address of the field pointer, modified by the IX fields of the instruction word.

The field pointer has the following format:



where

M = Indirect mapping control flag, accumulative with the IMC bit from the pointer's effective address.

FL = Field length ($1-32_{10}$); FL = 0 implies 32_{10}.

FP = Field position (0-31); the position in the word of the least-significant bit of the field.

If FL exceeds FP by more than one, FP will be made equal to FL-1; i.e., the field will comprise bits 0 through bits FL-1.

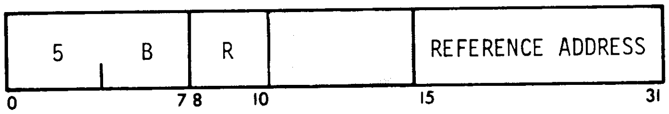
To execute the load, compare, add, and subtract field instructions, the field is retrieved from memory and a 32-bit word is generated with the field right-justified in the word. For the logical field operations, zeros are extended through the high-order bits of the word; for the arithmetic field operations, the high-order bit of the field is extended to bit 0 of the word. The value thus generated is called the effective field word (EFW).

Field instructions that are combined with the increment field pointer operation always increment the field pointer first. When the modified pointer is restored, the special instruction recovery flag (SIR) is set to 1 in the current program status word. When the field instruction is successfully completed, SIR is cleared to 0. If SIR = 1 when the CPU recognizes a field instruction, the increment pointer operation does not occur.

Table 9. Field Instructions

Instruction Name	Mnemonic	Hex. Code	Description	Page
Increment Field Pointer	IFP	5B		32
Decrement Field Pointer	DFP	4B		32
Load Field	LF	4C	EFW → (R)	32
Load Field After Increment	LFI	5C		32
Load Field Arithmetic	LFA	4D		32
Load Field Arithmetic After Increment	LFAI	5D		32
Store Field	STF	4A	(R) → EFW	32
Store Field After Increment	STFI	5A		33
Store Immediate to Field	STIF	6A		33
Compare Field	CF	4E	(R) : (EFA)	33
Compare Field After Increment	CFI	5E		33
Compare Field Arithmetic	CFA	4F	(R) : (EFA)	33
Compare Field Arithmetic After Increment	CFAI	5F		33
Compare Immediate to Field	CIF	6E		33
Add Field Arithmetic	AFA	49	(R) + EFW → (R)	34
Subtract Field Arithmetic	SFA	59	(R) - EFW → (R)	34
Inverse Subtract Field Arithmetic	ISFA	69	EFW - (R) → (R)	34
Add Field	AF	48	(R) + EFW → (R)	34
Subtract Field	SF	58	(R) - EFW → (R)	34
Inverse Subtract Field	ISF	68	EFW - (R) → (R)	34

Increment Field Pointer - IFP



Description:

IFP directly addresses a field pointer word in which the field length (FL) is added to the field position (FP) to produce a new field position (NFP). If NFP ≤ 31, it replaces the FP field in the pointer:

$$EWL_{6-10} \leftarrow EW_{6-10} + EW_{1-5}$$

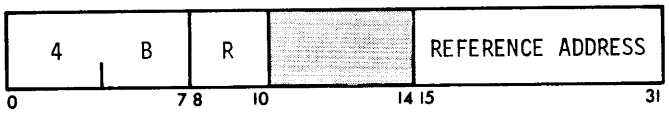
If NFP > 31, the FP field is replaced by FL-1 and the field pointer reference address is incremented by one; the modified field pointer then replaces the effective word.

$$EWL_{6-10} \leftarrow EW_{1-5} -1$$

$$EWL_{15-31} \leftarrow EW_{15-31} +1$$

Affected: (EWL)

Decrement Field Pointer - DFP



Description:

DFP directly addresses a field pointer word in which the field length (FL) is subtracted from the field position (FP) to produce a new field position (NFP). If NFP ≥ 0, it replaces the FP field in the pointer:

$$EWL_{6-10} \leftarrow EW_{6-10} - EW_{1-5}$$

If NFP < 0, the FP field is replaced by $1F_{16}$ and the field pointer reference address is decremented by one.

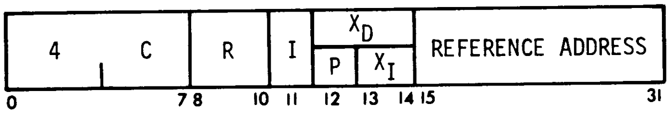
The modified field pointer then replaces the effective word.

$$EWL_{6-10} \leftarrow 1F_{16}$$

$$EWL_{15-31} \leftarrow EW_{15-31} -1$$

Affected: (EWL)

Load Field - LF

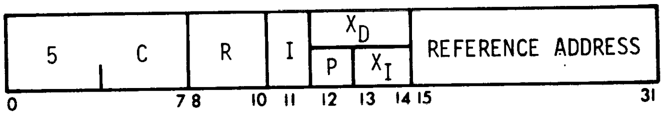


Description: EFW → (R)

LF loads the referenced field right-justified into register R, with zeros extended through the high-order bits of the register.

Affected: (R)

Load Field After Increment - LFI

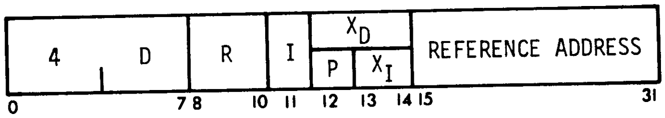


Description: Increment field pointer; EFW → (R)

LFI performs the Increment Field Pointer operation, then the Load Field operation.

Affected: (R), (FPL), SIR

Load Field Arithmetic - LFA

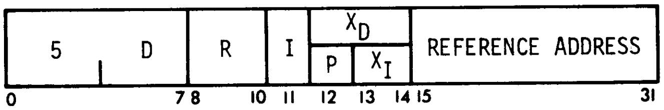


Description: EFW → (R)

LFA loads the referenced field right-justified into register R, with the high-order bit of the field extended to bit 0 of the register.

Affected: (R)

Load Field Arithmetic After Increment - LFAI

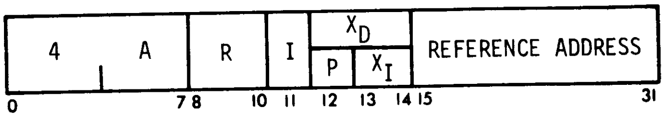


Description: Increment Field Pointer; EFW → (R)

LFAI performs the Increment Field Pointer operation, then the Load Field Arithmetic operation.

Affected: (R), (FPL), SIR

Store Field - STF

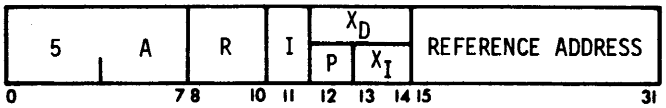


Description: (R) → EFL

STF stores the n least significant bits of register R left-justified into the referenced field where n is the field length. Bits adjacent to the field are left unchanged.

Affected: (EFL)

Store Field after Increment - STFI

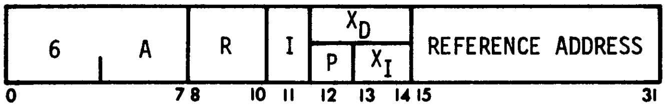


Description: Increment Field Pointer; (R) → EFL

STFI performs the Increment Field Pointer operation, then the Store Field operation.

Affected: (FPL), (EFL), SIR

Store Immediate to Field - STIF

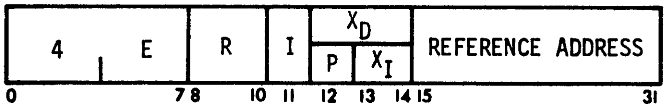


Description: R → EFL

STIF stores the R field of the instruction — a positive immediate value in the range 0-7 — into the contents of the referenced field.

Affected: (EFL)

Compare Field - CF



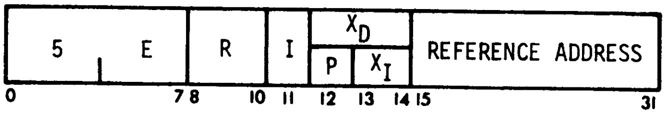
Description: (R) : (EFL)

CF compares the contents of register R with the contents of the referenced field, with the field treated as a 32-bit word by extending zeros through the high-order bits. The condition code indicators are set in the same manner as in the Compare Word (CW) instruction.

	Condition Code Indicators		
Result	1	2	3
(R) = EF	-	0	0
(R) < EF	-	0	1
(R) > EF	-	1	0
(R) ∩ EF = 0	0	-	-
(R) ∩ EF ≠ 0	1	-	-

Affected: CC1, CC2, CC3

Compare Field After Increment - CFI

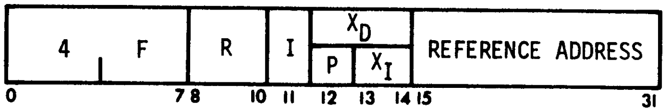


Description: Increment Field Pointer; (R) : (EFL)

CFI performs the Increment Field Pointer operation, then the Compare Field operation.

Affected: CC1, CC2, CC3, SIR

Compare Field Arithmetic - CFA

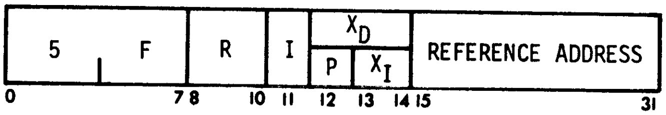


Description: (R) : (EFL)

CFA compares the contents of register R with the contents of the referenced field, with the field treated as a 32-bit word by extending the high-order bit of the field to bit 0 of the word. The condition code indicators are set in the same manner as in the Compare Field (CF) instruction.

Affected: CC1, CC2, CC3

Compare Field Arithmetic After Increment - CFAI

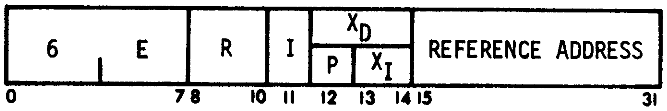


Description: Increment Field Pointer; (R) : (EFL)

CFAI performs the Increment Field Pointer operation, then the Compare Field Arithmetic operation.

Affected: CC1, CC2, CC3, SIR

Compare Immediate to Field - CIF

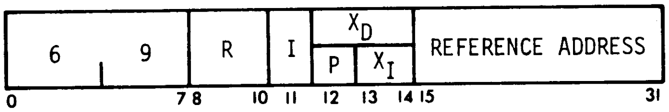


Description:

CIF compares the R field of the instruction — a positive immediate value in the range 0-7 — with the contents of the referenced field. The condition code indicators are set in the same manner as in the Compare Field (CF) instruction.

Affected: CC1, CC2, CC3

Inverse Subtract Field Arithmetic - ISFA

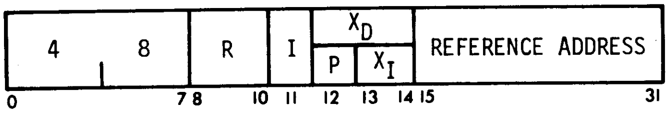


Description: EFW - (R) → (R)

ISFA obtains the arithmetic field and subtracts the contents of register R from the effective field word, then stores the result in register R.

Affected: (R), OF, CF

Add Field - AF

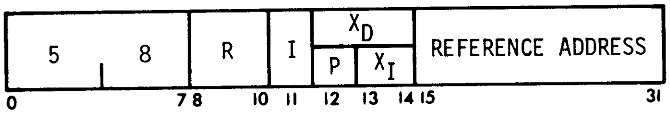


Description: (R) + EFW → (R)

AF obtains the field with preceding zeros and adds it to the contents of register R, then stores the result in register R.

Affected: (R), CF

Subtract Field - SF

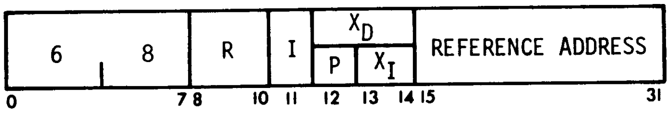


Description: (R) - EFW → (R)

SF obtains the field with preceding zeros and subtracts it from the contents of register R, then stores the result in register R.

Affected: (R), CF, OF

Inverse Subtract Field - ISF

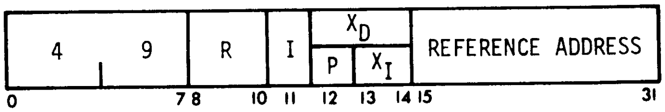


Description: EFW - (R) → (R)

ISF obtains the field with preceding zeros and subtracts the contents of register R from the effective field word, then stores the result in register R.

Affected: (R), CF, OF

Add Field Arithmetic - AFA

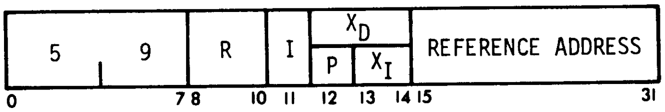


Description: (R) + EFW → (R)

AFA obtains the arithmetic field and adds the contents of register R to the effective field word, then stores the result in register R.

Affected: (R), OF, CF

Subtract Field Arithmetic - SFA



Description: (R) - EFW → (R)

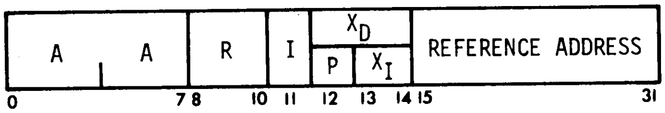
SFA obtains the arithmetic field and subtracts the effective field word from the contents of register R, then stores the result in register R.

Affected: (R), OF, CF

Unprivileged Control Instructions

Unprivileged control instructions (Table 10) can be executed when the computer is in either the master or the slave mode.

Store Program Status - STPS



Description: (PSW) → EWL

STPS stores the current program status word in the effective word location.

Affected: (EWL)

Load Condition Indicators - LCI

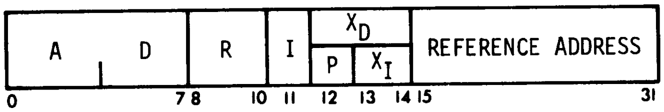


Table 10. Unprivileged Control Instructions

Instruction Name	Mnemonic	Hex. Code	Description	Page
Store Program Status	STPS	AA	(PSW) → EWL	34
Load Condition Indicators	LCI	AD	$EW_{2-6} \rightarrow PSW_{2-6}$	34
Load and Set	LAS	AC	$EW \rightarrow (R); EW_0 \leftarrow 1$	35
Load Register Group	LG	A9	EW thru EW + 7 → R0 thru R7	35
Store Register Group	STG	A8	R0 thru R7 → EW thru EW + 7	35
Execute	EXEC	8F	Execute effective instruction	35
Move with Incrementing Pointer	MOVI	7C		36
Move with Decrementing Pointer	MOVD	7D		36
Store Panel Switch Register	STSR	AE	(Switch Register) → EWL	36
Load Virtual Address	LVA	AF		36

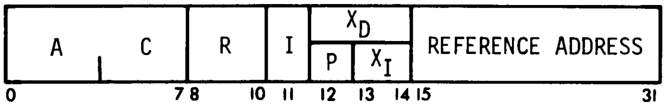
Description: $EW_2 \rightarrow CC1; EW_3 \rightarrow CC2; EW_4 \rightarrow CC3;$

$EW_5 \rightarrow OF; EW_6 \rightarrow CF$

LCI loads bits 2 through 6 of the effective word into bits 2 through 6 of the program status word: condition codes 1, 2, and 3, the overflow flag (OF), and the carry flag (CF).

Affected: CC1, CC2, CC3, OF, CF

Load and Set - LAS

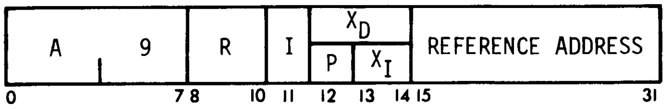


Description: $EW \rightarrow (R); EWL_0 \leftarrow 1$

LAS loads the effective word into register R; when the effective word is rewritten to memory, bit 0 is set to 1.

Affected: (R), (EWL_0)

Load Register Group - LG

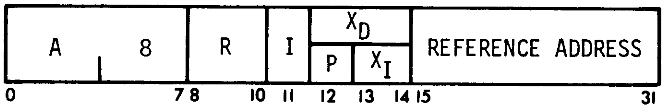


Description: (EWL+i) → (Ri) where i = 0, . . . , 7

LG loads the contents of the eight general registers from the eight consecutive locations beginning at EWL.

Affected: (R0), . . . , (R7)

Store Register Group - STG

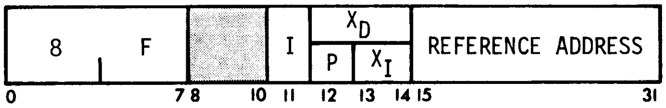


Description: (Ri) → (EWL+i) where i = 0, . . . , 7

STG stores the contents of the eight general registers at the eight consecutive locations beginning at EWL.

Affected: (EWL), . . . , (EWL+7)

Execute - EXEC



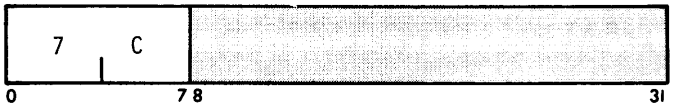
Description:

EXEC executes the instruction at the effective address. If the instruction is another EXEC, MOVI, or MOVD, an illegal instruction trap will occur.

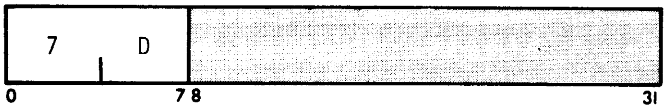
If the CPU is in the selective mapping mode and mapping of the EXEC effective address is designated, mapping is also implied for all accesses associated with the object instruction.

Affected: None

Move with Incrementing Pointer - MOVI



Move with Decrementing Pointer - MOVD



Description:

MOVI and MOVD perform a memory-to-memory transmission of a block of data. The contents of general registers 5, 6, and 7 specify the number of words, the source address, and the destination address respectively.

Register 5: The count register. A 32-bit value which is decremented after each word is moved. Upon instruction completion register 5 will contain 0. If register 5 contains 0 at the start of an instruction, the instruction will be ignored and the registers (including 6 and 7) will be left unchanged.

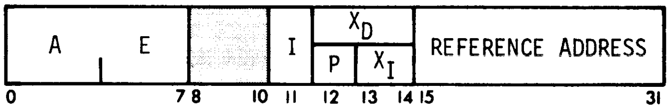
Register 6: The source register. Bits 15-31 of register 6 specify the source address. If the machine is in the mapped mode or if the machine is in the selective mapping mode and bit 0 of register 6 contains 1, the address is mapped. MOVI increments whereas MOVD decrements register 6 after each word is moved. Upon completion of the instruction register 6 will contain the next source address.

Register 7: The destination register. Bits 15-31 of register 7 specify the destination address. (Registers 5, 6, and 7 may not be used as the destination address.) If the machine is in the mapped mode or if the machine is in the selective mapping mode and bit 0 of register 7 contains 1, the destination address is mapped. MOVI increments whereas MOVD decrements register 7 after each word is moved. Upon completion of the instruction register 7 will contain the next destination address.

The move operation can be interrupted at the end of each move cycle. If the contents of registers 5, 6, or 7 are modified by the interrupt routine, the operation is indeterminant when control is returned. If an interrupt or trap occurs, the three registers will all reflect the end-of-cycle values and the program counter will contain the address of the move instruction.

Affected: (R5), (R6), (R7), (destination addresses)

Store Panel Switch Register - STSR

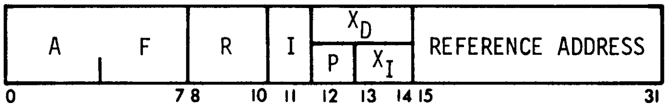


Description: (Switch Register) → EWL

STSR stores the value set in the switch register on the CPU operator's panel in the effective word location.

Affected (EWL)

Load Virtual Address - LVA



Description:

The unprivileged control instruction LVA computes the effective address from the object instruction and loads it into bits 15-31 of register R.

LVA performs standard address modification to retrieve the object instruction (effective word). The indirect, index, and address fields of the object instruction are then used to calculate the object instruction's effective address which is then loaded into bits 15-31 of register R; bits 11-14 of register R are set to 0; bits 1-10 of register R are loaded as follows:

- If the object instruction does not specify indirect addressing, bits 1-10 of the object instruction are loaded into bits 1-10 of register R.
- If the object instruction specifies indirect addressing, bits 1-10 of the indirect word are loaded into bits 1-10 of register R.

Bit 0 of register R is loaded with the map control bit which is calculated as the logical OR of bit 0 from each of the indirect words, index registers, and the object instruction. This includes any indexing or indirect addressing applied to retrieve the object instruction.

In summary, register R is loaded as follows:

$(R)_0$	map control bit
$(R)_{1-10}$	bits 1-10 of the object instruction (no indirect addressing), or bits 1-10 of the object instruction's indirect word
$(R)_{11-14}$	zero

$(R)_{15-31}$ effective address from the object instruction

Affected: (R)

Privileged Control Instructions

Privileged control instructions (Table 11) can be executed only when the computer is in the master mode. If execution is attempted while the computer is in the slave mode (i.e., bits 0 and 1 of the current program status word are both 1's), the computer unconditionally aborts execution of the instruction and traps to location 8.

Load Program Status - LPS

<table>
<tr><td rowspan="2">9 9</td><td rowspan="2">R</td><td rowspan="2">I</td><td colspan="2">X_D</td><td rowspan="2">REFERENCE ADDRESS</td></tr>
<tr><td>P</td><td>X_I</td></tr>
<tr><td>0 7</td><td>8 10</td><td>11</td><td>12</td><td>13 14</td><td>15 31</td></tr>
</table>

Description:

LPS places bits 0, 1, and 7-31 of the effective word into the corresponding bits of the program status word.

If bit 8 of LPS = 0: bits 2-6 of the current PSW (CC1, CC2, CC3, OF, CF) are also replaced by bits 2-6 of the effective word.

If bit 9 of LPS = 0: no mapping if in the selective mapping mode.

= 1: mapping

If bit 10 of LPS = 1: the contents of the eight locations preceding the effective address replace the contents of the eight general registers.

Affected: (PSW), (R0) through (R7)

Exchange Program Status - XPS

<table>
<tr><td rowspan="2">9 8</td><td rowspan="2">R</td><td rowspan="2">I</td><td colspan="2">X_D</td><td rowspan="2">REFERENCE ADDRESS</td></tr>
<tr><td>P</td><td>X_I</td></tr>
<tr><td>0 7</td><td>8 10</td><td>11</td><td>12</td><td>13 14</td><td>15 31</td></tr>
</table>

Description:

XPS stores the current PSW at the effective word location. Bits 0, 1, and 7-31 of (EA+1) are loaded into the corresponding bits of the program status word.

If bit 8 of XPS = 0: bits 2-6 of the current PSW (CC1, CC2, CC3, OF, CF) are also replaced by bits 2-6 of (EA+1).

If bit 9 of XPS = 0: no mapping if in the selective mapping mode

= 1: mapping

If bit 10 of XPS = 1: the contents of the eight general registers are stored in the eight locations preceding the effective address

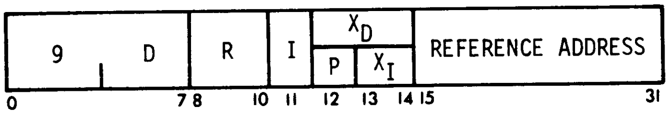
Table 11. Privileged Control Instructions

Instruction Name	Mnemonic	Hex. Code	Page
Load Program Status	LPS	99	37
Exchange Program Status	XPS	98	37
Load Map	LM	9D	38
Load Map Status	LMS	9C	38
Store Map Status	STMS	9E	38
Interrupt On	ION	91	38
Interrupt Off	IOFF	90	38
Pause	PAUS	93	38
Load Control Register	LCR	97	39
Store Control Register	STCR	96	39
Push Program Status	PPS	94	39
Pop Program Status	POPS	95	39
Load Actual Address	LAA	9F	39

The write protection flags in the memory map are ignored while XPS is being executed.

Affected: (PSW), (R0) through (R7), (EWL) ... (EWL-8)

Load Map - LM



Description:

LM loads one segment (32 pages) of the memory map with the actual page addresses. The R field specifies the segment (0-7) to be loaded. The eight word map segment is loaded from (EA) through (EA+7). The map status flags are not changed.

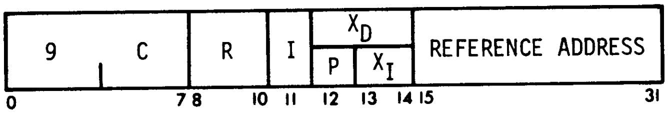
$EW_{0-7} \rightarrow$ Page 0, $EW_{8-15} \rightarrow$ Page 1,

$EW_{16-23} \rightarrow$ Page 2, $EW_{24-31} \rightarrow$ Page 3,...

... $EW+7_{24-31} \rightarrow$ Page 31

Affected: Memory map actual page addresses

Load Map Status - LMS



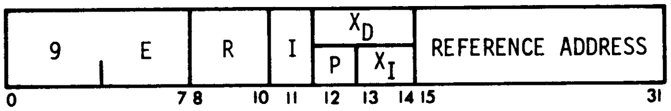
Description:

LMS loads one segment (32 pages) of the map with the altered and write protect status flags. The R field specifies the segment (0-7) to be loaded. Bits 0 to 31 of the effective word are loaded into the altered flags of pages 0 to 31 of the designated segment. Bits 0 to 31 of the word following the effective word are loaded into the write protect flags of pages 0 to 31 of the designated segment.

	0	1
Altered flag	Unaltered	Altered
Write protect flag	Read/Write	Read Only

Affected: Memory map status flags

Store Map Status - STMS

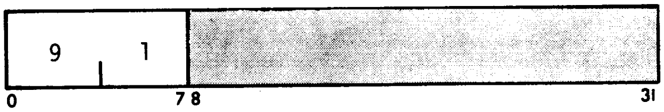


Description:

STMS replaces the contents of the effective doubleword referenced by the effective address with the altered and write protect status flags for the map segment specified by the R field (0-7). The first word will contain the altered flags for pages 0 through 31 of the designated segment whereas the second word will contain the write protect flags.

Affected: (EDW)

Interrupt On - ION

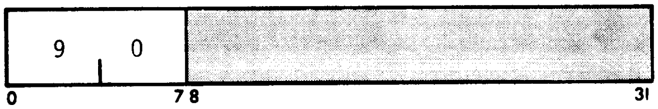


Description: IE ← 1

ION sets the interrupt enable flag (IE) in the program status word to 1, which allows the CPU to acknowledge interrupt requests after the next instruction. Bits 8-31 of the instruction word are ignored.

Affected: IE (Bit 7 of the PSW)

Interrupt Off - IOFF

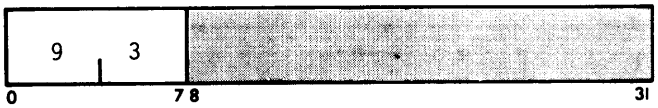


Description: IE ← 0

IOFF sets the interrupt enable flag (IE) in the program status word to 0, which immediately prevents the CPU from acknowledging interrupt requests. Bits 8-31 of the instruction word are ignored.

Affected: IE (Bit 7 of the PSW)

Pause - PAUS



Description:

PAUS causes the CPU to stop executing instructions until an interrupt occurs or the CPU is restarted from the operator's panel. If a PAUS is interrupted, control is returned to the next instruction after the PAUS. Bits 8 through 31 of the instruction word are ignored.

Load Control Register - LCR

			X_D		
9 7	R	I	P	X_I	REFERENCE ADDRESS
0 7	8 10	11	12	13 14	15 31

Description: EW → CR

LCR replaces the contents of a specified control register with the effective word.

Affected: (CR)

Store Control Register - STCR

			X_D		
9 6	R	I	P	X_I	REFERENCE ADDRESS
0 7	8 10	11	12	13 14	15 31

Description: (CR) → EWL

STCR stores the contents of a specified control register into the memory location defined by the effective address.

Affected: (EWL).

Push Program Status - PPS

			X_D		
9 4	R	I	P	X_I	REFERENCE ADDRESS
0 7	8 10	11	12	13 14	15 31

Description:

PPS pushes the current program status word and optionally the eight general registers into the program status stack, and then loads a new program status word from the effective address.

The logical OR of bit 9 of the PPS instruction and bit 0 of any indirect words or index registers used in the address modification determines mapping of the effective address where the new PSW is to be retrieved. A one designates mapping whereas zero designates no mapping. The CPU mode (bits 0, 1 of the PSW) does not affect whether mapping is to occur.

Bit 10 of the PPS instruction designates whether the registers are to be pushed. If set, the registers are pushed.

The Current Stack Pointer (CSP) is contained in control register 6. Bits 15-31 contain the address of the last stack word pushed. Bit 0 if set designates mapping, otherwise, no mapping of the stack pointer.

The Stack Limit (SLM) is contained in control register 7. Upon completion of a PPS instruction bits 15-31 of CSP are compared with bits 15-31 of SLM. If CSP is greater than SLM, a stack overflow trap to location 8 will occur.

Pushing is accomplished as follows:

With registers: The eight general registers and the current program status word with bit 13 set are stored in the nine memory locations beginning at (CSP)+1. CSP is incremented by nine.

Without registers: The current program status word with bit 13 reset is stored at the location following the location addressed by CSP. CSP is incremented by one.

Affected: (PSW), (CSP), 1 or 9 memory locations

Pop Program Status - POPS

			X_D		
9 5	R	I	P	X_I	REFERENCE ADDRESS
0 7	8 10	11	12	13 14	15 31

Description:

POPS loads the Program Status Word (PSW) and optionally registers R0 to R7 from the program status stack.

The Current Stack Pointer (CSP) contained in control register 6 addresses the last stack entry. The word at that address is loaded into the PSW. If bit 13 of this new PSW is set, registers R0 through R7 are loaded from the eight locations preceding the location addressed by the CSP. The CSP is decremented by nine if the registers are loaded, otherwise, CSP is decremented by one.

Bit 0 of CSP if set designates that the stack is in mapped memory, otherwise, it is unmapped.

Affected: (PSW), (R0)...(R7), (CSP)

Load Actual Address - LAA

			X_D		
9 F	R	I	P	X_I	REFERENCE ADDRESS
0 7	8 10	11	12	13 14	15 31

Description:

The privileged control instruction LAA computes the effective actual address from the object instruction and loads it into register R.

LAA performs the standard address modification to retrieve the object instruction. The indirect, index, and address fields of the object instruction are then used to calculate the object instruction's effective address which is then loaded into bits 15-31 of register R. This address is after mapping if and only if one of the following conditions is met:

- The CPU is in the unconditional mapping mode.
- The CPU is in the selective mapping mode and bit 0 of one of the indirect words, the index registers used, or the object instruction is set. This includes the address modification used to retrieve the object instruction.

Bits 0, 11-14 of register R are set to zero. Bits 1-10 of register R are loaded as follows:

- If the object instruction does not specify indirect addressing, bits 1-10 of the object instruction are loaded into bits 1-10 of register R.
- If the object instruction specifies indirect addressing, bits 1-10 of the indirect word are loaded into bits 1-10 of register R.

If mapping is in effect (unconditional or conditional), the virtual effective address is transformed by the memory map into an actual effective address, which is then stored in bits 15-31 of register R.

The map status bits of the page being accessed set the condition code indicators as follows if mapping is designated:

CC1 is set to the state of the page write protect status bit.

CC2 is set to the state of the altered/unaltered status bit.

CC3 is reset if an unassigned page is accessed or mapping did not occur. CC3 is set if successful mapping of the address occurred.

If an unassigned page is accessed while LAA is being executed, a trap does not occur, the virtual address accessing the unassigned page is stored in $(R)_{15-31}$, and CC1, CC2, and CC3 are reset to 0, 0, 1 (unprotected, unaltered, and unassigned).

If the memory map is not implemented or the mapping mode is not specified, CC1, CC2, and CC3 are reset to 0, 0, 0 (unprotected, unaltered, and assigned).

Affected: (R), CC1, CC2, CC3

Programmed Instruction

Programmed instructions (PINs) enable the programmer to define his own set of instructions. There are 16 global PINs typically used for calling a system service and 48 local PINs typically used for user defined instructions. Corresponding to each of the 64 PIN op codes are 64 memory locations. Execution of a PIN causes: register 0 to be loaded with the effective address from the PIN; and the instruction in the location corresponding to the PIN op code to be executed.

Global PINs

			X_D		
C0 - CF	R	I	P	X_I	REFERENCE ADDRESS
0 7	8 10	11	12	13 14	15 31

Description:

Register 0 is loaded as follows:

$(R0)_0$ 0 if CPU was in the unmapped mode or selective mapping mode and mapping not met; 1 if CPU was in the mapped mode or selective mapping mode and mapping is met.

$(R0)_{1-7}$ Zero.

$(R0)_{8-10}$ Bits 8-10 from the PIN instruction.

$(R0)_{11-14}$ Zero.

$(R0)_{15-31}$ Effective virtual address.

The instruction at the actual address associated with the specific PIN op code is then executed. Usually this is a PPS or XPS instruction but occasionally may be a BAL instruction.

Op codes C0-CF correspond to actual locations 40-4F.

Local PINs

			X_D		
D0 - FF	R	I	P	X_I	REFERENCE ADDRESS
0 7	8 10	11	12	13 14	15 31

Description:

Register 0 is loaded as follows:

$(R0)_{0-7}$ Zero.

$(R0)_{8-10}$ Bits 8-10 from the PIN instruction.

$(R0)_{11-14}$ Zero.

$(R0)_{15-31}$ Effective virtual address.

The instruction at the virtual address associated with the specific PIN op code is then executed. Usually this is a BAL instruction.

Op codes D0-FF correspond to virtual locations 50-7F.

Input/Output Instruction

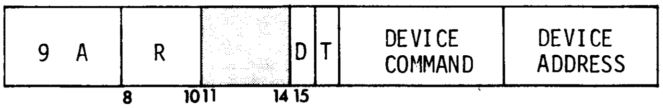
Input/Output Control - IOC

			X_D		
9 A	R	I	P	X_I	REFERENCE ADDRESS
0 7	8 10	11	12	13 14	15 31

Description:

IOC is a privileged instruction that selects and passes a command to an input/output processor (IOP) and, if the command requires a data transfer, transfers 32 bits of data or control information in either direction between the designated CPU general register and the selected IOP.

The reference address field of an IOC instruction can be modified by indexing and indirect addressing as if it were a memory reference instruction. When address modification is completed, the following information is loaded into the instruction register (IR):



where T = Register transfer flag:

1 = register transfer will occur
0 = register transfer will not occur

D = Register transfer direction flag; if T = 1:

1 = transfer from peripheral device to register
0 = transfer from register to peripheral device

The CPU informs all I/O processors that an IOC is being executed, and the processor that recognizes its device address proceeds to have the CPU verify the device command. If the command is invalid, an instruction trap is initiated; if the command is valid, the CPU examines (IR)$_{15,\ 16}$.

If (IR)$_{16}$ = 0, the IOC is terminated; if (IR)$_{16}$ = 1, the CPU transmits to or accepts from the processor 32 bits of data, depending upon the state of (IR)$_{15}$.

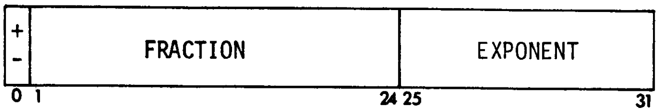
Floating-Point Instructions

The floating-point arithmetic instructions shown in Table 12 are included as an optional feature of the TENET 210 computer. When the option is not implemented, the floating-point op codes execute as an additional set of global PINs where op codes B0-BF correspond to actual locations 30-3F.

Both single- and double-precision floating-point numbers can be represented in arithmetic operations.

Single Precision

A single-precision number (short format) is represented in 32 bits — a normalized 25-bit two's complement fraction (24 bits plus sign), and a 7-bit two's complement power-of-two exponent. Single precision floating point represents slightly more than 7 digits of significance and the range 10^{-18} to 10^{+18}.



Double Precision

A double-precision number (long format) is represented in 64 bits — a normalized 55-bit two's complement fraction (54 bits plus sign), and a 9-bit two's complement power-of-two exponent. Double precision floating point represents slightly more than 16 digits of significance and the range 10^{-76} to 10^{+76}.

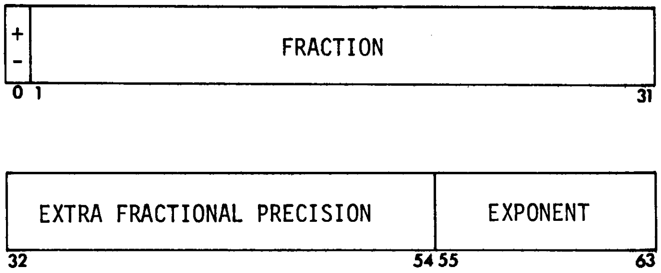


Table 13 contains examples of floating-point numbers.

Table 12. Floating - Point Instructions

Instruction Name	Mnemonic	Hex. Code	Description	PIN Location †	Page
Floating Add Short	FAS	B4	(R) + EW → (R)	34	43
Floating Add Long	FAL	B0	(R, R') + EDW→ (R, R')	30	43
Floating Subtract Short	FSS	B5	(R) - EW → (R)	35	44
Floating Subtract Long	FSL	B1	(R, R') - EDW → (R, R')	31	44
Floating Inverse Subtract Short	FISS	B9	EW - (R) → (R)	39	44
Floating Inverse Subtract Long	FISL	B8	EDW - (R, R') → (R, R')	38	44
Floating Multiply Short	FMS	B6	(R) * EW → (R)	36	44
Floating Multiply Long	FML	B2	(R, R') * EDW → (R, R')	32	44
Floating Divide Short	FDS	B7	(R)/EW → (R)	37	44
Floating Divide Long	FDL	B3	(R, R')/EDW → (R, R')	33	44
Floating Inverse Divide Short	FIDS	BB	EW/(R) → (R)	3B	45
Floating Inverse Divide Long	FIDL	BA	EDW/(R, R') → (R, R')	3A	45

† Floating-point PINs are not used if the hardware option is implemented.

Examples

The following examples show the step-by-step procedure used to convert decimal numbers to and from floating-point format.

Example 1: Convert 11.234375_{10} to floating-point.

a. Convert 11_{10} to binary: $11_{10} = 1011_2$

b. Convert 0.234375_{10} to binary: $0.234375_{10} = 001111_2$

c. Thus, $11.234375_{10} = 1011.001111_2$
$= 0.1011001111_2 \times 2^4$

d. The short-format floating-point representation is:

```
0101 1001 1110 0000 0000 0000 0 | 000 0100
0                            24   25    31
```

If the number being converted is negative (-11.234375), the following additional steps are necessary:

e. Compute the two's complement of the fraction:

```
1010 0110 0010 0000 0000 0000 0 | 000 0100
0                            24   25    31
```

f. Shift the fraction left until bits 0 and 1 differ, and subtract the amount shifted from the exponent. In this example, a shift was not necessary.

Example 2: Convert 0.125_{10} to floating-point.

a. Convert 0.125 to binary: $0.125_{10} = 0.001_2$
$= 0.1_2 \times 2^{-2}$

b. The short-format floating-point representation is:

```
0100 0000 0000 0000 0000 0000 0 | 111 1110
0                            24   25    31
```

c. Compute the two's complement of the fraction:

```
1100 0000 0000 0000 0000 0000 0 | 111 1110
0                            24   25    31
```

d. Shift the fraction left one position and subtract 1 from the exponent:

```
1000 0000 0000 0000 0000 0000 0 | 111 1101
0                            24   25    31
```

Example 3: Convert the following binary floating-point number to decimal:

```
1010 0110 0010 0000 0000 0000 0 | 000 0100
0                            24   25    31
```

a. Since bit 0 is set, shift the fraction right one position, extend the sign to the left, and add 1 to the exponent:

```
1101 0011 0001 0000 0000 0000 0 | 000 0101
0                            24   25    31
```

b. Compute the two's complement of the fraction:

```
0010 1100 1111 0000 0000 0000 0 | 000 0101
0                            24   25    31
```

c. Shift the fraction left one position and subtract 1 from the exponent:

```
0101 1001 1110 0000 0000 0000 0 | 000 0100
0                            24   25    31
```

d. The number can now be represented as:
$0.1011001111_2 \times 2^4 = 1011.001111_2$

e. Convert the integer and fractional portions to decimal:
$1011_2 = 11_{10}$
$0.001111_2 = 0.234375_{10}$

Three floating-point error conditions optionally cause a trap to location 8 (see Figure 2).

1. Exponent Overflow

If a floating-point operation results in an exponent that is too large to be expressed in the assigned field, the overflow flag (OF) is set in the program status word. The referenced register is loaded with zero if the resulting exponent is negative, or with the maximum magnitude if the resulting exponent is positive. The referenced register is complemented if the resulting fraction is negative. If the floating-point overflow bit (FO) is set in the program status word, a trap to location 8 is executed; otherwise, execution continues.

Table 13. Floating-Point Number Representation

Decimal Number	Single-Precision Floating-Point Format								Hexadecimal Value
0.125	0100	0000	0000	0000	0000	0000	0111	1110	4000007E
-0.125	1000	0000	0000	0000	0000	0000	0111	1101	8000007D
0.1	0110	0110	0110	0110	0110	0110	0111	1101	6666667D
-0.1	1001	1001	1001	1001	1001	1010	0111	1101	999999FD
π	0110	0100	1000	0111	1110	1101	1000	0010	6487ED82
$-\pi$	1001	1011	0111	1000	0001	0010	1000	0010	9B781202

2. Divide by Zero

   If a zero divisor is detected, the overflow flag (OF) and the carry flag (CF) are both set in the program status word. If the floating-point divide by zero bit (FD) is also set, a trap to location 8 is executed; otherwise, execution continues with the referenced register unchanged.

3. Illegal Operand

   If an unnormalized operand is detected, the carry flag (CF) is set in the program status word. If the unnormalized floating point (UF) is also set, a trap to location 8 is executed; otherwise, the improper operand is normalized, the exponent is adjusted, and execution continues.

Floating Add Short - FAS

<table>
<tr><td rowspan="2">B 4</td><td rowspan="2">R</td><td rowspan="2">I</td><td colspan="2">X_D</td><td rowspan="2">REFERENCE ADDRESS</td></tr>
<tr><td>P</td><td>X_I</td></tr>
</table>

0 7 8 10 11 12 13 14 15 31

Description: (R) + EW → (R)

FAS adds the effective word to the contents of register R and loads the sum into register R as a short-format floating-point number.

Affected: (R), OF, CF

Floating Add Long - FAL

<table>
<tr><td rowspan="2">B 0</td><td rowspan="2">R</td><td rowspan="2">I</td><td colspan="2">X_D</td><td rowspan="2">REFERENCE ADDRESS</td></tr>
<tr><td>P</td><td>X_I</td></tr>
</table>

0 7 8 10 11 12 13 14 15 31

Description: (R, R') + EDW → (R, R')

FAL adds the effective doubleword to the contents of registers R and R', and loads the sum into registers R and R' as a long-format floating-point number. R must be an even value for correct results.

Affected: (R), (R'), OF, CF

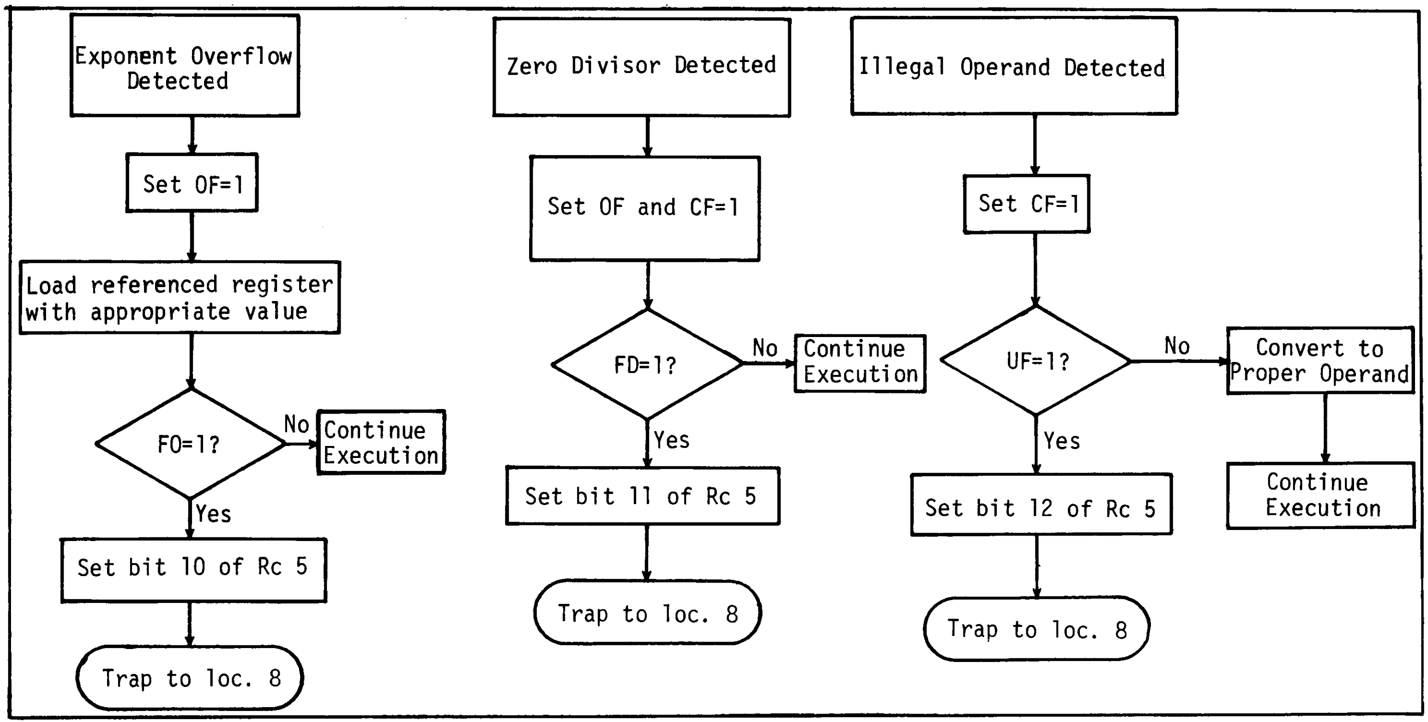


Figure 2. Floating Point Error Condition Recovery

Floating Subtract Short - FSS

			X_D		
B 5	R	I	P	X_I	REFERENCE ADDRESS

0 7 8 10 11 12 13 14 15 31

Description: (R) - EW → (R)

FSS subtracts the effective word from the contents of register R and loads the difference into register R as a short-format floating-point number.

Affected: (R), OF, CF

Floating Subtract Long - FSL

			X_D		
B 1	R	I	P	X_I	REFERENCE ADDRESS

0 7 8 10 11 12 13 14 15 31

Description: (R, R') - EDW → (R, R')

FSL subtracts the effective doubleword from the contents of registers R and R' and loads the difference into registers R and R' as a long-format floating-point number. R must be an even value for correct results.

Affected: (R), (R'), OF, CF

Floating Inverse Subtract Short - FISS

			X_D		
B 9	R	I	P	X_I	REFERENCE ADDRESS

0 7 8 10 11 12 13 14 15 31

Description: EW - (R) → (R)

FISS subtracts the contents of register R from the effective word and loads the difference into register R as a short-format floating-point number.

Affected: (R), OF, CF

Floating Inverse Subtract Long - FISL

			X_D		
B 8	R	I	P	X_I	REFERENCE ADDRESS

0 7 8 10 11 12 13 14 15 31

Description: EDW - (R, R') → (R, R')

FISL subtracts the contents of registers R and R' from the effective doubleword and loads the difference into registers R and R' as a long-format floating-point number. R must be an even value for correct results.

Affected: (R), (R'), OF, CF

Floating Multiply Short - FMS

			X_D		
B 6	R	I	P	X_I	REFERENCE ADDRESS

0 7 8 10 11 12 13 14 15 31

Description: (R) * EW → (R)

FMS multiplies the contents of register R by the effective word and loads the product into register R as a short-format floating-point number.

Affected: (R), OF, CF

Floating Multiply Long - FML

			X_D		
B 2	R	I	P	X_I	REFERENCE ADDRESS

0 7 8 10 11 12 13 14 15 31

Description: (R, R') * EDW → (R, R')

FML multiplies the contents of registers R and R' by the effective doubleword and loads the product into registers R and R' as a long-format floating-point number. R must be an even value for correct results.

Affected: (R), (R'), OF, CF

Floating Divide Short - FDS

			X_D		
B 7	R	I	P	X_I	REFERENCE ADDRESS

0 7 8 10 11 12 13 14 15 31

Description: (R)/EW → (R)

FDS divides the contents of register R by the effective word and loads the quotient into register R as a short-format floating-point number.

Affected: (R), OF, CF

Floating Divide Long - FDL

			X_D		
B 3	R	I	P	X_I	REFERENCE ADDRESS

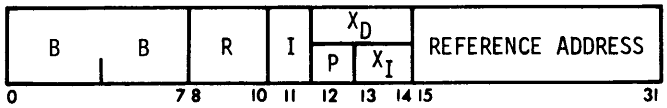
0 7 8 10 11 12 13 14 15 31

Description: (R, R')/EDW → (R, R')

FDL divides the contents of registers R and R' by the effective doubleword and loads the quotient into registers R and R' as a long-format floating-point number. R must be an even value for correct results.

Affected: (R), (R'), OF, CF

Floating Inverse Divide Short - FIDS

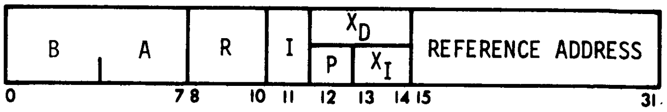


Description: EW/(R) → (R)

FIDS divides the effective word by the contents of register R and loads the quotient into register R as a short-format floating-point number.

Affected: (R), OF, CF

Floating Inverse Divide Long - FIDL



Description: EDW/(R, R') → (R, R')

FIDL divides the effective doubleword by the contents of registers R and R', and loads the quotient into registers R and R' as a long-format floating-point number. R must be an even value for correct results.

Affected: (R), (R'), OF, CF

4. MEMORY MAPPING

The TENET 210 memory map † provides for program fragmentation, access protection and altered page status. The memory map can be represented as a series of 256 high-speed registers, each register corresponding to a 512-word page of virtual memory. The map is divided into 4 segments of 64 pages each. The machine instructions which control the loading and testing of the memory map (Load Map, Load Map Status, Store Map Status) operate on only one segment at a time.

A virtual address referencing a map register containing an actual page address of zero or a map register in an unimplemented map segment causes a trap to location 8, with bit 6 set in control register 5 (unassigned virtual page). If the map register contains the value of an actual page which has not been implemented in real core, a trap to location 8 occurs with bit 4 set in control register 5 (unimplemented memory).

ADDRESS GENERATION

When the memory map is implemented and in effect, "virtual" addresses are transformed into "actual" addresses. A virtual address is the effective address before mapping. As depicted below, this value is transformed through the memory map into an actual address, i.e., the actual memory location used by the CPU to store and retrieve information. When the memory map is not in effect, the 17-bit virtual address is always an actual address.

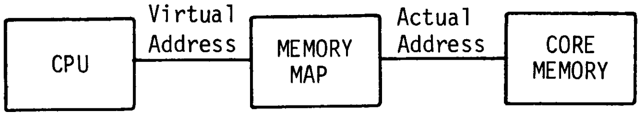


Figure 3 illustrates how a virtual address is transformed through the memory map into an actual memory address. In the mapping mode, all program reference addresses except 0-7 are virtual addresses which are transformed through the memory map before they become references to actual core memory locations. Addresses 0-7 are always actual addresses and reference the eight general registers.

The CPU uses the 8 high-order bits of a 17-bit virtual address as an index to access one of the 256 map registers. Each map register contains an 8-bit actual page address for a specific 512-word page of actual memory as well as 2 page status flags. The 9 low-order bits of the virtual address refer to the specific word within the actual memory page. Thus, the actual address is obtained by combining the appropriate 8 bit actual page address from the map with the lower 9 bits from the virtual address.

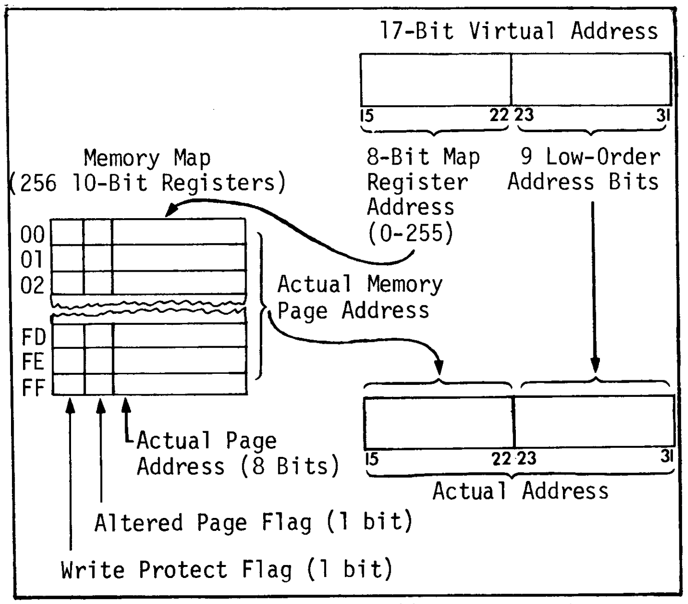

17-Bit Virtual Address
15
22 23
31
Memory Map
(256 10-Bit Registers)
8-Bit Map Register Address (0-255)
9 Low-Order Address Bits
00
01
02
FD
FE
FF
Actual Memory Page Address
Actual Page Address (8 Bits)
15
22 23
31
Actual Address
Altered Page Flag (1 bit)
Write Protect Flag (1 bit)

Figure 3. Memory Map Address Generation

MAPPING CONTROL

Mapping is controlled by the CPU mode bits (bits 0 and 1) in the program status word as follows:

Bit 0	Bit 1	Mode
0	0	Unmapped mode
0	1	Selective (conditional) mapping mode
1	0	Unconditional mapping
1	1	

Unmapped Mode

When in the unmapped mode, all program addresses except 0-7 are used by the CPU as actual core memory addresses.

Unconditional Mapping

When there is unconditional mapping, all virtual addresses are mapped to an actual address before a memory access occurs.

----

† Memory mapping applies only to addresses generated by the CPU. The I/O Processors have no capability for memory mapping.

Selective Mapping

When in the selective mapping mode, conditional mapping is controlled by bit 0 of the index registers, indirect words and field pointer words used during address modification. If any of these bits is a one, then mapping is designated; no mapping occurs only if all of these bits are zero.

WRITE PROTECT FLAGS

If a virtual address is mapped and a write-memory operation is to be performed, the Write Protect Flag (for the virtual page) is sensed. If the flag is zero and the Page Protect Override Bit in the current program status word is one, the write-memory operation is aborted and the CPU executes a trap to location 8, setting bit 5 of control register 5. The contents of the protected memory location are not altered. The privileged instructions Exchange Program Status (XPS) and Push Program Status (PPS) operate as if the Write Protect Flags are zero.

The Write Protect Flags are loaded via the Load Map Status (LMS) instruction and may be stored via the Store Map Status (STMS) instruction.

ALTERED PAGE FLAGS

The execution of an instruction which modifies memory will cause the Altered Page Flag corresponding to the referenced virtual page to be set to one.

The Altered Page Flags are loaded via the Load Map Status (LMS) instruction and may be stored via the Store Map Status (STMS) instruction.

5. INPUT/OUTPUT OPERATIONS

INTRODUCTION

In the TENET 210 system, input/output operations are performed under control of one or more input/output processors (IOPs). This allows the CPU to concentrate on program execution, free from the time-consuming details of I/O operations. Any I/O events that require CPU intervention are brought to its attention by means of the interrupt system. This input/output chapter frequently refers to interrupts which are not described until the next chapter. Therefore, it is suggested that the reader reread this chapter after reading the chapter on interrupts. The intent of this section is to describe the input/output system in general. For more specific information on programming for the individual input/output devices see the TENET 210 Input/Output Reference Manual.

In the following discussion, the terminology conventions used are that the CPU executes instructions, the IOP executes commands, and the I/O devices execute orders. To illustrate, the CPU will execute the Start Input/Output (SIO) instruction to initiate an IO operation. During the course of an I/O operation, the IOP might issue an order to an I/O device to perform a function such as rewind.

INPUT/OUTPUT PROCESSORS

An IOP may be either a multiplexer IOP or selector IOP and is further characterized according to the type of I/O device it may control. The following types of IOPs may be interfaced to the TENET 210 Data Exchange.

Communications Multiplexer IOP	1 - 16 communications lines per IOP
Disc Selector IOP	1 - 4 disc drives per IOP
Magnetic Tape Selector IOP	1 - 4 tape drives per IOP
Card Reader Selector IOP	1 card reader per IOP
Printer Selector IOP	1 printer per IOP

The major distinction between a multiplexer IOP and a selector IOP is the way data is transmitted.

Multiplexer IOP - A multiplexer IOP allows for all of the devices it controls to transfer data simultaneously. However, data is transferred between the multiplexer IOP and the CPU one byte at a time. Thus, CPU intervention is necessary for each byte transferred.

Selector IOP - Only one of the devices controlled by a selector IOP may transfer data at a time. However, an unlimited amount of data may be transferred between the selector IOP and memory without any CPU intervention.

Input/Output Control Instruction (IOC)

All communication between the CPU and external devices uses the Data Exchange and is controlled by the Input/Output Control (IOC) instruction. IOC is a privileged instruction that provides a parallel, 32-bit data transfer between a CPU general register and an input/output device. The reference address field of an IOC instruction can be modified by indexing and indirect addressing as if it were a memory reference instruction. When address modification is complete and the effective address is loaded in the instruction register, the instruction is defined as follows:

9 A	R	▨	D T COMMAND	DEVICE ADDRESS
0 7	8 10	11 14	15 16 23	24 31

where:

Device Command (bits 15-23) is the command that defines the function the device is to perform.

T is the register transfer flag:

T = 0 No register transfer is involved.

T = 1 The CPU will execute a register/device data transfer.

D is the transfer direction flag:

D = 0 The contents of register R will be transferred to the addressed device.

D = 1 The device will transfer 32 bits of information into register R.

The device address designates the IOP and device to which the command applies. Commands which apply to an IOP but not to any specific device may use the address of any of the devices attached to the IOP.

The execution of the IOC instruction is performed in two phases. During the first phase the CPU transmits bits 15-31 of the IOC instruction to the Data Exchange; at this time it is the responsibility of each device interface to examine the device address to determine if it is the one affected by the IOC. The affected device interface must then decode the command field (bits 15-23) in order to respond properly in the second phase.

During the second phase two items of information are transferred between the CPU and device interface via the Data Exchange. The first is the 32-bit transfer controlled by D and T. The second is three bits of status information which the device sends to the CPU to be loaded into the condition code field of the Program Status Word. Status can be used to convey information such as the state of the device, whether the command was accepted or rejected for execution, etc.

The execution of an IOC instruction can result in a trap to location eight under the following conditions.

- The device interface does not recognize the command field.
- A transmission error is detected during the data transfer controller by the D and T bits of the IOC.

The IOC is the only TENET 210 instruction which inhibits interrupts from the completion of the instruction until the initiation of the next instruction. This feature insures that no interrupts can occur between clearing the interrupt and the restoration of the interrupted routine's environment at the end of an I/O interrupt service routine.

MULTIPLEXER IOP

A multiplexer IOP can operate all of its attached devices simultaneously. It contains a one-character input buffer and a one-character output buffer for each device it controls. CPU intervention is required to transmit a byte of information between the CPU and one of the multiplexer IOP device buffers.

For a multiplexer IOP possible condition code responses are as follows:

Condition Code	Meaning
000	Rejected, device address not recognized
001	Rejected, device inoperative
010	Rejected, input buffer empty or output buffer full
100	Accepted

Multiplexer IOP Interrupts

A multiplexer IOP causes a CPU interrupt under any of the following conditions provided the system interrupts (PSW bit 7) and the processor interrupt are enabled and no higher level interrupts are active:

- a device input buffer is full
- a device output buffer is empty and the individual device's output buffer interrupt is armed
- a transition from on-line to off (or off-line) occurs for one of the devices.
- a transition from off (or off-line) to on-line occurs for one of the devices and the IOP transition interrupt is armed.

Multiplexer IOP Commands

The following forms of the Input/Output Control instruction apply to the multiplexer IOP designated by the IOP's device address:

PION - Processor Interrupts On

9 A	R	▨	003	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

PION enables the interrupt associated with the designated IOP. The condition codes will be set as follows:

000	Rejected, device address not recognized
100	Accepted

PIOFF - Processor Interrupts Off

9 A	R	▨	004	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

The PIOFF command disables the interrupt associated with the designated IOP. The condition codes will be set as follows:

000	Rejected, device address not recognized
100	Accepted

IXIT - Interrupt Exit

9 A	R	▨	005	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

IXIT signals the system that the processing of the interrupts associated with the designated IOP address has been completed (see Section 6: Interrupts and Traps for a complete description of exiting from interrupts). The condition codes will be set as follows:

000	Rejected, device address not recognized
100	Accepted

RDS - Read Device Status

9 A	R		1C4	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

RDS sets the low-order 16 bits of register R to indicate which of the respective 16 terminals attached to the designated IOP have changed terminal On/Off state since the previous RDS command.

0 = No state change

1 = Change in the terminal On/Off line switch

Bits 0-15 of register R are set to zero. The condition codes will be set as follows:

000	Rejected, device address not recognized
100	Accepted

RIS - Read Input Status

9 A	R	▨	1C2	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

RIS sets the low-order 16 bits of register R to indicate which of the respective 16 terminals attached to the designated IOP have full input buffers.

0 = Empty

1 = Full

Bits 0-15 of register R are set to zero. The condition codes will be set as follows:

000	Rejected, device address not recognized
100	Accepted

ROS - Read Output Status

9 A	R	▨	1C3	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

ROS sets the low-order 16 bits of register R to indicate which of the respective 16 terminals attached to the designated IOP have empty output buffers and the output interrupt enabled for that terminal (see the commands Reset Output Interrupt and Set Output Interrupt in the section Interrupts).

0 = Output buffer full or terminal output interrupt disabled

1 = Output buffer empty and terminal output interrupt enabled

Bits 0-15 of register R are set to zero. The condition codes will be set as follows:

000	Rejected, device address not recognized
100	Accepted

STI - Set Transition Interrupt

9 A	R	▨	022	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

STI arms the transition interrupt flag in the designated IOP. This control flag applies only to device transitions from off- to on-line. The condition codes will be set as follows:

000	Rejected, device address not recognized
100	Accepted

RTI - Reset Transition Interrupt

9 A	R	▨	023	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

RTI disarms the transition interrupt flag in the designated IOP. This control flag applies only to device transitions from off- to on-line. The condition codes will be set as follows:

000	Rejected, device address not recognized
100	Accepted

The following forms of the IOC instruction apply to the device designated by the device address.

PON - Power On

9 A	R	▨	020	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

PON turns on the motor for the designated device or terminal. (The OFF/LINE/LOCAL switch must be in LINE position to be effective.) The condition codes will be set as follows:

000	Rejected, device address not recognized
100	Accepted

POFF - Power Off

9 A	R	▨	021	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

POFF turns off the motor for the designated device or terminal. The condition codes will be set as follows:

000 Rejected, device address not recognized

100 Accepted

SI - Start Input

9 A	R		181	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

SI transmits the contents of the designated device input buffer to register R. The one-byte buffer is loaded into register R right-justified with leading zeros. The input buffer flag is set to empty.

If the input buffer was empty when SI was executed, register R is filled with zeros.

Characters keyed in while the buffer is full are lost.

The condition codes will be set as follows:

000 Rejected, device address not recognized

001 Rejected, device inoperative

010 Rejected, input buffer empty

100 Accepted

TI - Test Input

9 A	R		001	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

TI tests the status of the designated device input buffer and sets the condition codes as follows:

000 Device address not recognized

001 Device inoperative

010 Output buffer full

100 Device address recognized, device operative and output buffer empty

SOI - Set Output Interrupt

9 A	R		00C	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

SOI arms the designated output buffer interrupt for the designated device. An interrupt will be requested under the following conditions: the output buffer is empty, its output buffer interrupt is armed and the processor interrupt is enabled. The condition codes will be set as follows:

000 Rejected, device address not recognized

100 Accepted

ROI - Reset Output Interrupt

9 A	R		00D	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

ROI disarms the designated output buffer interrupt. The condition codes are set as follows:

000 Rejected, device address not recognized

100 Accepted

SO - Start Output

9 A	R		082	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

SO transmits the byte of information in bits 24-31 of register R to the designated device output buffer. The condition codes will be set as follows:

000 Rejected, device address not recognized

001 Rejected, device inoperative

010 Rejected, output buffer full

100 Accepted

TO - Test Output

9 A	R		002	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

TO tests the status of the designated device output buffer and sets the condition codes as follows:

000 Device address not recognized

001 Device inoperative

010 Output buffer full

100 Device address recognized, device operative and output buffer empty

SELECTOR IOP

A selector IOP operates simultaneously and independent of the CPU, allowing for only one of its attached devices to be transmitting data at a time. It automatically executes a chain of one or more I/O command double words

(IOCDs) from memory. These IOCD chains represent orders to the IOP and permit both order chaining (making possible multiple-record operations) and data chaining (making possible scatter-read and gather-write operations) without intervening CPU control. Order chaining refers to the execution of a sequence of I/O orders, under control of an IOP, on more than one physical record. Thus, a new order must be issued for each physical record even if the operation to be performed for a record is the same as that performed for the previous record. Data chaining refers to the execution of a sequence of IOCDs, under control of an IOP, that gather (or scatter) information within one physical record from (or to) more than one region of memory. Thus, a new IOCD must be issued for each portion of a physical record when the data associated with that physical record appears (or is to appear) in noncontiguous locations in memory. For example, if information in specific columns of two cards in a file are to be stored in specific regions of memory, the IOCD list might appear as follows:

1. Read card, store columns 1-12, data chain
2. Store columns 13-60, data chain
3. Store columns 61-80, order chain (end of data chain)
4. Read card, store columns 1-40, data chain
5. Store columns 41-80 (end of order chain, end of data chain)

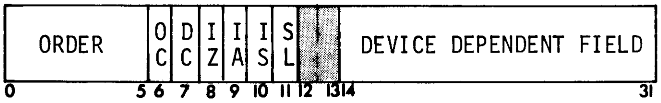
The TENET 210 CPU itself plays a minor role in the execution of a selector IOP input/output operation. The CPU-executed program is responsible for creating and storing the IOCD list (prepared prior to the initiation of any I/O operation) and for supplying the IOP with a pointer to the first IOCD in the I/O control list.

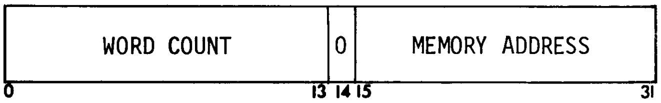
The following is an example of the sequence of events that occurs during an I/O operation:

1. A CPU-executed program writes a sequence of IOCD in core memory.
2. The CPU executes the instruction Start Input/Output (SIO) and furnishes the IOP with an 8-bit I/O address (designating the device to be started) and a 17-bit first IOCD address (designating the actual core memory location where the first IOCD for this device is located). At this point, either the device is started (if in the "ready" condition) or command reject occurs. The CPU is informed by condition code settings as to which of the two alternatives has occurred. If the Start I/O instruction is accepted, the designated IOP's IOCD counter is loaded with the address of the IOCD list. Assuming that the SIO instruction is accepted, from this time until the full sequence of IOCDs have been executed, the main program of the CPU need play no role in the I/O operation. At any time, however, it may obtain status information on the progress of the I/O operation without interfering with the operation.
3. The IOP and device is now in the "busy" condition. The IOP then fetches the IOCD from core memory, loads it into the IOP's IOCD register and transmits the first order (extracted from the IOCD) to the device.
4. The IOCD counter is incremented by two so that it will contain the memory address of the next IOCD.
5. If a data transmission order has been sent to a device, control of the transmission resides in the device. As each character is obtained by the I/O device, the IOP is signaled that data is available. The IOP uses the information stored in its own registers to control the information interchange between the I/O device and the memory on a word-by-word basis.
6. When all information exchanges called for by a single IOCD have been completed, the interrupt arming flags in the IOCD are examined to determine whether an I/O interrupt should be generated. If the appropriate conditions are met, the I/O interrupt is generated. Step 7 begins immediately, independent of whether or not an interrupt was generated.
7. If the IOCD list contains another IOCD, step 3 is repeated. This process continues until the IOCD list is exhausted. When the list is exhausted, the IOP and device enter the not busy state independent of whether any interrupts associated with that IOP remain to be serviced.

Input/Output Control Doublewords (IOCDs)

All IOCDs are represented by two words and have the following format:





ORDER

The Order code (represented in hexadecimal) is dependent upon the type of device.

<u>Printer</u>

20	Print and advance
21	Print and don't advance
30	Skip to channel X (channel number in bits 24-31 of device dependent field)
31	Skip n lines (count in bits 24-31 of device dependent field)

Card Reader

00	Read ANSI card
01	Read binary card
02	Read data dependent

Disc

01	Read
04	Read identification field from index
05	Read next identification field
21	Write
22	Check write
24	Write identification field
30	Seek

Tape

00	Read forward
02	Space record forward
03	Space file forward
10	Read backward
12	Space record backward
13	Space file backward
20	Write record
23	Write EOF
32	Transfer and continue

OC - Order Chain

If this flag is 1, order chaining is called for when the current IOCD is complete. The next IOCD is fetched and loaded into the IOCD register of the IOP and the new order is processed. If the OC flag is 0, no further order chaining is called for. When order chaining, an order will apply to the beginning of the next record.

DC - Data Chain

If this flag is 1, data chaining is called for when the current word count is reduced to 0. The next IOCD is fetched and loaded into the IOCD register associated with the IOP, but both the new order code and the device dependent field are ignored; thus, the operation called for by the previous order is continued. The new IOCD is used only to supply a new memory address, a new count, and new flags. If the data chain flag is 0, no further data chaining is required. Data chaining will continue reading or writing the device at the point where the previous IOCD left off, namely, it will not automatically advance to the next record.

IZ - Interrupt at Zero Word Count

If this flag is 1, the IOP requests an interrupt when the word count of this IOCD (as stored in the IOP register) is reduced to 0.

IA - Interrupt on Abnormal End

If this flag is 1, the IOP requests an interrupt when an error condition or unusual termination is encountered. Examples of an abnormal end are device inoperative, parity error, record length errors, end of file, etc. The IOP will continue with the current data transmission and any data chaining after an abnormal end until the next order is encountered. However, the interrupt will be requested as soon as the abnormal end condition occurs.

IS - Interrupt on Special Condition

The IS flag is totally device dependent. If set to a 1, the disc IOP will request an interrupt when the seek is complete. The other IOPs do not use this flag.

SL - Suppress Incorrect Length

If this flag is 1, an incorrect length indication by the device is not to be recorded as an error by the IOP. If the SL flag is 0, an incorrect length is considered an error and the IOP performs as specified by the IA flag. Incorrect length is caused by a record end condition occurring before the IOP word count reaches zero, or is caused by the word count reaching zero prior to the end of record, e.g., word count decremented to zero before 80 columns have been read from a card. The IOP is capable of suppressing an error condition on incorrect length since there are many situations in which incorrect length is a legitimate condition and not a true error.

Device Dependent Field

This field is device dependent, e.g., on the printer it indicates number of lines to skip or channel skip and on the disc it indicates a disc seek address.

Word Count

For all data transmission orders this field indicates the number of data words to be transmitted. After each data word is transmitted, this field is decremented in the IOP's IOCD register. If the initial value of the word count is zero, then 16384 (2^{14}) is implied. The TDV2 command allows the program to read the current word count value.

Memory Address

For all data transmission orders this field indicates the memory location for the next data word. For all but the

Read Backward order this field in the IOP register is incremented after each word is transmitted. For Read Backward this field is decremented after each word is transmitted. The Transfer and Continue order uses this field to designate the location of the next IOCD.

Selector IOP Commands

The following forms of the Input/Output Control instruction apply to selector IOPs and devices attached to a selector IOP. In all cases the condition codes will be set to indicate the acceptance or rejection of the IOC command and the state of the device. Rejection of an IOC instruction due to channel busy means that the command was issued to the IOP at the exact clock time when a word of data was being transferred between the IOP and the data exchange.

PION - Processor Interrupt On

9 A	R	▨	003	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

PION enables the interrupt associated with the designated IOP. The condition codes will be set as follows:

000	Rejected, device address not recognized
100	Accepted

PIOFF - Processor Interrupt Off

9 A	R	▨	004	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

PIOFF command disables the interrupt associated with the designated IOP. The condition codes will be set as follows:

000	Rejected, device address not recognized
100	Accepted

IXIT - Interrupt Exit

9 A	R	▨	005	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

IXIT signals the system that the processing of the interrupts associated with the designated IOP address has been completed. The condition codes will be set as follows:

000	Rejected, device address not recognized
100	Accepted

SIO - Start Input/Output

9 A	R		080	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

SIO is used to initiate an input or output operation with the device designated by the device address. The general register specified by the R field of the SIO instruction is assumed to contain the starting address of the IOCD list in bit positions 15-31. If the device is in the ready condition, then the SIO command is accepted and the IOP loads the first IOCD into its IOCD register and the device begins transmitting data according to the IOCD list. If the device was busy or for some reason the SIO command cannot be accepted, it will be rejected. The condition code settings will designate the acceptance or rejection of the SIO command as follows:

000	Rejected, device address not recognized
001	Rejected, device inoperative
010	Rejected, device busy
011	Rejected, processor busy
100	Accepted, transfer initiated

TIO - Test Input/Output

9 A	R	▨	000	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

TIO is used to test whether an SIO command would be accepted or rejected for the designated device if issued at this time. The condition codes are set to indicate the device state as follows:

000	Device address not recognized
001	Device inoperative
010	Device busy
011	Processor busy
100	IOP/device ready

HIO - Halt Input/Output

9 A	R	▨	016	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

HIO causes the designated IO device to halt its current operation after completing the transmission of the current record. All pending interrupts pertaining to this IOP are reset. The IOP registers may be then interrogated to determine the extent to which the halted IO operation had been completed. The condition codes will be set as a result of a HIO to designate the acceptance or rejection of the instruction as follows:

000	Rejected, device address not recognized
010	Rejected, device busy
011	Rejected, channel busy
100	Accepted, device ready
101	Accepted, device inoperative
110	Accepted, device busy

ISIO - Interrupt on Start Input/Output Acceptable

9 A	R	▨	006	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

ISIO sets the designated IOP in a mode such that an interrupt will be requested by that IOP when it is in a state where it is ready to accept a Start Input/Output instruction. The condition codes are set to indicate the acceptance or rejection as follows:

000	Rejected, device address not recognized
010	Rejected, device busy
011	Rejected, channel busy
100	Accepted, device ready
101	Accepted, device inoperative
110	Accepted, device busy

TISIO - Terminate Interrupt on Start Input/Output Acceptable

9 A	R	▨	007	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

TISIO resets the designated IOP's ISIO mode such that no interrupt will be requested by that IOP regarding its being in a state where it is ready to accept a Start Input/Output instruction. The condition codes are set to indicate the acceptance or rejection as follows:

000	Rejected, device address not recognized
010	Rejected, device busy
011	Rejected, channel busy
100	Accepted, device ready
101	Accepted, device inoperative
110	Accepted, device busy

The following commands (AIO, TDV1, TDV2 and TDV3) are used to read the status of a selector IOP. Each of these commands causes the designated general register to be loaded with the status information from the selected IOP. Bits 0-13 of the general register are loaded with the universal status by each of these instructions and bits 14-31 are loaded with other status information dependent upon which instruction is executed.

The universal status information is as follows:

Bit	Description
0	Abnormal end exists
1	Device inoperative
2	Device busy
3	Not used
4	Processor busy
5	Memory addressing violation
6	Memory error (parity error between IOP and memory)
7	Data error (transmission error between IOP and device)
8	Data overrun
9	Record length error
10	EOF
11	Illegal order
12	Not used
13	Interrupt pending

AIO - Acknowledge I/O Interrupt

9 A	R	▨	1CO	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

AIO loads the designated general register as follows:

UNIVERSAL STATUS	IZ	IA	IS	ISIO	DEVICE DEPENDENT FIELD
0 13	14	15	16	17	18 31

where:

IZ — Set if interrupt pending due to word count zero detected

IA — Set if abnormal end condition occurs such as incorrect length, parity errors, etc.

IS — Set if interrupt pending due to a special condition such as seek complete on disc

ISIO — Set if interrupt pending due to ISIO condition met (see ISIO and TISIO instructions)

Device Dependent Field	This field contains status information peculiar to the type of device (see TENET 210 Input/Output Reference Manual)

In addition to loading the above status information into the designated general register, the interrupt pending flags are reset in the designated IOP. This instruction must be executed, otherwise, the interrupt will be generated again when an IXIT instruction is executed.

The condition codes are set as follows to indicate the acceptance or rejection of the command:

000	Rejected, device address not recognized
010	Rejected, device busy
011	Rejected, channel busy
100	Accepted, device ready
101	Accepted, device inoperative
110	Accepted, device busy

TDV1 - Test Device Status 1

9 A	R	▨	1C1	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

TDV1 loads the designated general register as follows from the specified IOPs registers:

UNIVERSAL STATUS	▨	IOCD ADDRESS
0 13	14	15 31

The IOCD address is the address of the next IOCD in the chain.

The condition codes are set as follows to indicate the acceptance or rejection of the command:

000	Rejected, device address not recognized
010	Rejected, device busy
011	Rejected, channel busy
100	Accepted, device ready
101	Accepted, device inoperative
110	Accepted, device busy

TDV2 - Test Device Status 2

9 A	R	▨	1C2	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

TDV2 loads the designated general register as follows from the specified IOPs registers:

UNIVERSAL STATUS	▨	WORD COUNT
0 13	14	15 31

The word count represents the number of words which remain to be transferred.

The condition codes are set as follows to indicate the acceptance or rejection of the command:

000	Rejected, device address not recognized
010	Rejected, device busy
011	Rejected, channel busy
100	Accepted, device ready
101	Accepted, device inoperative
110	Accepted, device busy

6. INTERRUPTS AND TRAPS

This section describes the TENET 210 priority interrupt system and automatic traps. The distinction between interrupts and traps is as follows:

Interrupts occur as a result of conditions originating external to the CPU such as an I/O processor. Each interrupt level can be selectively armed/disarmed, enabled/disabled by the CPU. Traps are caused by illegal, internal conditions in the CPU (such as an illegal instruction or uninstalled option). Unlike interrupts, traps are always enabled and only four of them can be inhibited (described under Trap Masks). A trap may be considered the highest priority "interrupt" in the system, i.e., it results in the immediate execution of the instruction in the trap location.

Although the interrupt or trap location can contain any valid instruction, the instruction is normally an Exchange Program Status (XPS) or Push Program Status (PPS). These instructions save the current machine environment and cause control to be transferred to the appropriate interrupt or trap service routine.

INTERRUPT SYSTEM

The TENET 210 interrupt system provides 20 interrupt levels, each assigned a predetermined priority and each assigned a unique location in core memory. In addition, any 16 of the 20 interrupt levels can be assigned to an External Interrupt Group (EIG) and expanded to 16 sublevels. EIGs are described later in this section.

When an interrupt condition is sensed, a signal is sent to an interrupt level; however, the instruction in the interrupt location is not executed until all of the conditions have been met that permit the actual interrupt acknowledgment to occur.

Operational States

For an interrupt to occur, the following conditions must exist:

1. The interrupt level must be armed, i.e., the appropriate flag bit in the input/output control word (IOCW) must be set. These flag bits are described under Interrupt Types. When an interrupt level is armed, it is sensitive to and remembers the occurrence of an event. When an interrupt level is disarmed, i.e., the appropriate flag bit in the IOCW is not set, it will not be sensitive to nor remember the occurrence of the event for which it has been disarmed.
2. The interrupt level must be enabled. The I/O control command Processor Interrupts On (PION) enables an IOP to generate an interrupt signal to the CPU at the time the specified condition occurs. The I/O control command Processor Interrupts Off (PIOFF) is used to disable an interrupt level. In this state, it will not respond to the occurrence of an event; if armed, the event will be remembered and an interrupt signal will occur at the time the interrupt level is enabled.
3. The CPU must be in the interrupt-allowed mode, i.e., the privileged control instruction System Interrupts On (ION) has set the interrupt enable flag (IE) in the program status word. In this mode, the CPU will respond to the highest priority occurrence of an interrupt signal. When the CPU is in the interrupt-inhibit mode, i.e., the instruction System Interrupts Off (IOFF) has reset the IE bit in the PSW, the CPU will not respond to an interrupt signal from any device.

Interrupt Processing

When an interrupt level in the armed state receives an interrupt stimulus as thc result of an cvent, it advances to the waiting state, and remains in the waiting state until it is allowed to advance to the active state or is cleared by the command Interrupt Exit (IXIT) or through an I/O RESET at the control panel. IXIT is executed by interrupt subroutines to notify an IOP that an interrupt has been recognized and completed and another interrupt at that level can be initiated.

When an interrupt is in the waiting state, the following conditions must all exist simultaneously before the level advances to the active state:

1. The level must be enabled (a PION command has been executed).
2. The CPU must be at an interruptable point in the execution of an instruction (see Time of Occurrence).
3. The CPU must be in the interrupt-enabled mode (the IE bit in the program status word has been set by an ION instruction).
4. No higher-priority interrupt level is in the active state or in the waiting state and enabled.

When all the necessary conditions are met to allow an interrupt to move from the waiting to the active state, the computer executes the instruction in the assigned interrupt location.

An interrupt level remains in the active state until cleared by the execution of an IXIT command. An interrupt servicing routine may itself be interrupted whenever a higher-priority interrupt meets all the conditions necessary to become active. In such a case, the lower-priority interrupt remains in the active state although its execution is suspended. After the higher-priority interrupt is cleared, the lower-priority interrupt routine continues from the point of interruption. (An interrupt servicing routine may lock out all higher-priority interrupts by executing an IOFF instruction.)

Interrupt Types

There are four classes of I/O interrupts: abnormal end, zero word count, special condition, and start input/output (SIO) acceptable. The first three are armed from IOCWs while the latter is armed by the interrupt on start input/output (ISIO) command. A condition occurring within a class that is armed causes the IOP to remember that the affected class needs interrupt service. The request to the CPU is then determined by the IOP status, i.e., whether it has already interrupted and not received IXIT (Interrupt Exit), or is enabled or disabled. The causes of each interrupt type are described below:

Type	Causes
Abnormal End (IA flag = 1)	1. A physical record end is detected before the IOCW word count goes to 0. 2. The order chain flag (OC) in the IOCW is on when the word count goes to 0, and the physical record has not ended. 3. Both the order chain and data chain (OC and DC) flags are on in an IOCW. 4. Any disc read/write operation is stopped because of a failure detected by the IOP.
Zero Word Count (IZ flag = 1)	5. The IOCW word count goes to zero at the end of a physical record.
Special Condition (IS flag = 1)	6. Applies to disc operations only, and occurs when a seek to a specified address has been successfully completed.
SIO Acceptable	7. SIO command is accepted and an ISIO command has instructed the device to interrupt.

Time of Occurrence

Interrupts can occur during the following intervals of an instruction cycle:

1. Between Instructions

   An interrupt is permitted between the completion of an instruction and the initiation of the next instruction, with the following exception: interrupts are inhibited after the completion of an IOC instruction until after the completion of the next non-IOC instruction.

2. Between Iterations

   The move instructions maintain all iteration control information in the general registers and may be interrupted between the completion of one iteration and the beginning of the next. The address portion of the program status word remains pointing at the interrupted move instruction. The instruction continues from the point at which it was interrupted after the interrupt processing is completed.

3. At a Program Halt

   The PAUS instruction may be interrupted, but the address portion of the program status word will be incremented, so that at the completion of the interrupt processing, the instruction following the PAUS will be executed.

Interrupt Locations

The allocated interrupt locations and their associated device addresses are shown below.

Device Address (Hexadecimal)	Interrupt Location (Hexadecimal)	Description
00	-	Unassigned
01	81	Card Reader
02	82	Line Printer (Buffered)
0F	80	Clock
10-13	83	Disc Processor
14-17	84	Disc Processor
18-1B	85	Disc Processor
1C-1F	86	Disc Processor
20-23	87	Magnetic Tape Processor
30-3F	88	Teletype Processor
40-4F	8A	Teletype Processor
50-5F	8C	Teletype Processor
60-6F	8E	Teletype Processor

EXTERNAL INTERRUPT GROUP

The External Interrupt Group (EIG) permits any 16 of 20 interrupt levels to be expanded to 16 sublevels. Programs communicate with EIGs using the Input/Output Control (IOC) instruction. The instruction format is as follows:

9 A	R	I	X_D		OPERATION	1111	EIG NO.
			P	X_I			
0 7	8 10	11	12	13 14	15 23	24 27	28 31

The EIG number (0-15) selects an EIG that may be assigned to any one of the 20 interrupt levels. The 16 low-order bits of the specified general register select which of the 16 sublevels are to be affected: bits 16-31 of the register correspond to sublevels 0-15, respectively. (Sublevel 0 has the highest priority within the interrupt group.) A sublevel is selected by a one bit in its corresponding register position. The commands are defined in the following tables:

Set Interrupt Status

Command (Hex.)	Selected Sublevels	Other Sublevels
080	Armed	Disarmed
081	Enabled	Disabled
082	Armed and Enabled	Unchanged
083	Armed and Disabled	Unchanged
084	Disarmed	Unchanged
085	Enabled	Unchanged
086	Disabled	Unchanged
087	Trigger Interrupt	Unchanged

Test Interrupt Status

Command (Hex.) Description

180 The armed/disarmed states for each sublevel of the specified EIG are loaded into register R. Armed = 1 where disarmed = 0.

181 The enabled/disabled states for each sublevel of the specified EIG are loaded into register R. Enabled = 1 where disabled = 0.

182 The active/waiting/quiescent states for each sublevel of the specified EIG are loaded into register R. Bits 0-15 indicate:

Reset = not active (waiting or quiescent)
Set = active

Bits 16-31 indicate:

Reset = not waiting (active or quiescent)
Set = waiting

An interrupt in the active state may be in process or suspended due to a higher level active interrupt in process.

TRAPS

When the CPU detects an illegal internal condition (described below), the CPU transfers control to a trap location that is specifically assigned to handle the illegal condition. The instruction stored in the trap location provides for the preservation of the current machine state and for the branch to a routine that handles the trap condition. Unlike interrupts, when a trap condition is sensed, the instruction in the trap location is executed immediately. This instruction is executed before any interrupt acknowledgment can occur, i.e., a trap cannot be interrupted.

Three memory locations handle all of the trap conditions: 8, 9, and 10 (decimal). Power and parity failures are handled by locations 9 and 10, respectively. All other illegal conditions result in a trap to location 8, and control register 5 defines the trap condition. If the bit is set, the condition applies.

Bit Position in CR 5	Condition
0	Unused (always 0)
1	Unimplemented instruction, i.e., an optional instruction that has been assigned an operation code was not installed in this machine.
2	An attempt was made to execute a privileged instruction while the CPU was in the slave mode.
3	The instruction at the effective address of an EXEC instruction was another EXEC, MOVI, or MOVD.
4	An attempt was made to access an unimplemented memory address.
5	A write operation was attempted on a virtual page with its write protect flag set (see Section 4: Memory Mapping).
6	Unassigned virtual page, i.e., a virtual address referenced a map register containing a zero or a map register in an unimplemented map segment (see Section 4: Memory Mapping).
7	The CPU referenced general registers 0-7 in attempting to retrieve its next instruction or as the result of a branch.
8	Unassigned operation code, i.e., an attempt was made to execute a nonexistent instruction.
9	The command selected by an IOC instruction was invalid, i.e., not recognized by the CPU.
10	Floating-point overflow/underflow

Bit Position in CR 5	Condition
	was detected with bit 10 (floating overflow/underflow armed) set in the program status word.
11	Floating-point divide by zero was detected with bit 11 (floating divide by zero armed) set in the program status word.
12	Unnormalized floating-point value was referenced with bit 12 (unnormalized floating value armed) set in the program status word.
13	Program status stack overflow, i.e., the privileged control instruction Push Program Status (PPS) has stored the current program status word at a location that exceeds the stack limit (bits 15-31 of control register 7).
14	This bit is set if conditions 4 or 6 were caused by an instruction fetch, reset if the trap resulted from a data fetch.
15-31	Offending address for conditions 4, 5, and 6.

Control register 5 must be reset by the instruction Load Control Register (LCR).

The address portion of the program status word is incremented prior to a trap and points to the instruction following the one which caused the trap (this includes the move instruction). The only exception is for an instruction fetch error (conditions 4 or 6 where bit 14 would also be set).

Recovery for Addressing Error

The unassigned virtual page trap may occur either during an instruction fetch or during a data, indirect word, or field pointer fetch. Bit 14 is set for an addressing error due to an instruction fetch, indicating that the address portion of the program status word has not been incremented.

The program may be continued by assigning the virtual page specified by the address in bits 15-31 of control register 5, decrementing by one the address portion of the PSW if bit 14 of control register 5 is reset, and restoring the program environment by executing an Exchange Program Status (XPS) or Pop Program Status (POPS) instruction.

If the trap resulted from a data fetch (bit 14 reset), care must be taken when assigning a new virtual page not to release the adjoining page.

Doubleword instructions do not modify the general register until they fetch both addresses. As a result, they may require both pages when they are reexecuted.

Special Instruction Cases

The CPU automatically handles the special cases of instructions which may modify memory prior to a memory reference error, namely, the class of field instructions which increment the field pointer prior to a subsequent memory reference for the same instruction. In these cases where memory is modified, the CPU sets the Special Instruction Recovery Flag (SIR) in the program status word. If a trap occurs and the instruction is reexecuted, the CPU resets the SIR bit, skips over the portion of the instruction which modifies memory, and executes the remainder of the instruction.

Trap Masks

Four of the traps summarized in Table 14 can be masked (not allowed to occur) by setting the appropriate bit in the program status word:

1. Page Protect Override

   When PO (bit 8) is set to 1, the CPU ignores the read-only page condition when the memory map is used so that a write operation will not cause a trap.

2. Floating-Point Overflow/Underflow

   When FO (bit 10) is set to 0, a trap will not occur as the result of floating-point overflow/underflow.

3. Floating-Point Divide by Zero

   When FD (bit 11) is set to 0, a trap will not occur as the result of a floating-point zero divisor.

4. Floating-Point Unnormalized Value

   When FU (bit 12) is set to 0, a trap will not occur as the result of an unnormalized floating-point operand.

Table 14. Summary of TENET 210 Trap System

Trap Condition	Bit in CR 5	PSW Mask Bit	Time of Occurrence
1. Unimplemented instruction	1	None	Instruction decoding
2. Nonallowed operation			
a. Privileged instruction in slave mode	2	None	Instruction decoding
b. Attempt to EXEC an EXEC	3	None	Instruction decoding of second EXEC
c. Unimplemented memory address	4	None	Prior to memory access
d. Memory protection violation	5	8 (PO)	Prior to memory access
e. Unassigned virtual page	6	None	Prior to memory access
f. Branch to general register	7	None	Instruction decoding
g. Nonexistent instruction	8	None	Instruction decoding
h. Invalid IOC command	9	None	Instruction decoding
3. Floating-Point Arithmetic Fault			
a. Overflow/underflow	10	10 (FO)	At time of fault detection
b. Divide by zero	11	11 (FD)	At time of fault detection
c. Unnormalized value	12	12 (FU)	After operand access
4. Stack Overflow	13	None	After instruction completion

7. OPERATOR CONTROLS

CPU CONTROL PANEL

The control panel for the TENET 210 CPU is illustrated in Figure 4. The controls and indicators are described below.

Register Display A

Register display A is used to display the contents of the CPU instruction register or the A register. It is composed of 32 indicator lights (numbered from 0 through 31 in groups of 4) with the memory reference instruction format delineated. The rotary switch to the right of the register display selects the register to be displayed. The LOAD switch loads the contents of the switch register into the selected register.

The A register is an intermediate buffer register that contains the argument value at the beginning of an instruction and the resultant value after the instruction is executed.

Register Display B

Register display B, consisting of 32 indicators, is used to display the contents of the PSW, the E register, any of the eight general registers, or any of the eight control registers. The rotary switch to the right of the register display selects the register to be displayed. The GENERAL/CONTROL switch selects between the general and control registers. The LOAD switch loads the contents of the switch register into the selected register.

The E register is an extension of the A register used when the operand is double precision. The A register represents bits 0-31; whereas, the E register represents bits 32-63.

Switch Register

The switch register, consisting of 32 two-position switches, is used to:

1. Store the contents of the switch register in the effective word specified by the instruction Store Panel Switch Register (STSR).
2. Load the contents of the switch register into a register selected by one of the LOAD switches associated with the register displays.
3. Halt the central processor when it references the location selected by switches 15 through 31. Each time a memory access is initiated by the CPU, the ADDRESS STOP switches — ACTUAL and VIRTUAL — determine which memory address — actual or virtual — is compared with the address specified by the switch register. If either the ACTUAL or VIRTUAL switch is on and the corresponding address is equal to the address specified by the register, the CPU goes into the idle state. If both the ACTUAL and VIRTUAL switches are on, the CPU enters the idle state if either an actual or virtual address is equal to the address specified by the switch register. The ADDRESS STOP indicators OPERAND and INST show whether the central processor halted as a result of an operand or instruction fetch cycle.

If an address stop is caused by an instruction fetch, the program counter will contain the address corresponding to the switch register, the instruction register will contain the instruction at the location designated by switch register and the instruction will not have been executed.

If an address stop is caused by an operand fetch, the program counter will contain the address of the next instruction to be executed; the instruction register will contain the next instruction to be executed; and the instruction causing the address stop will have completed execution.

Indirect words or register references from the address field of an instruction will cause an address stop if the switch register matches such an operand reference.

If the CPU branches to a location which matches the switch register, an operand address stop will occur; whereas, an instruction fetch address stop will occur when the CPU retrieves the next sequential instruction from a location which matches the switch register.

To continue execution after an address stop, move the COMPUTE switch to IDLE and then back to RUN.

Status Indicators

Status indicators on the CPU control panel reflect the status of the labeled conditions that are controlled by the various switches. If the indicator is lit, the function is in an "on" state; if the indicator is not lit, the function is in an "off" state.

Power

The POWER indicator is lit whenever AC power is applied to the TENET 210 System.

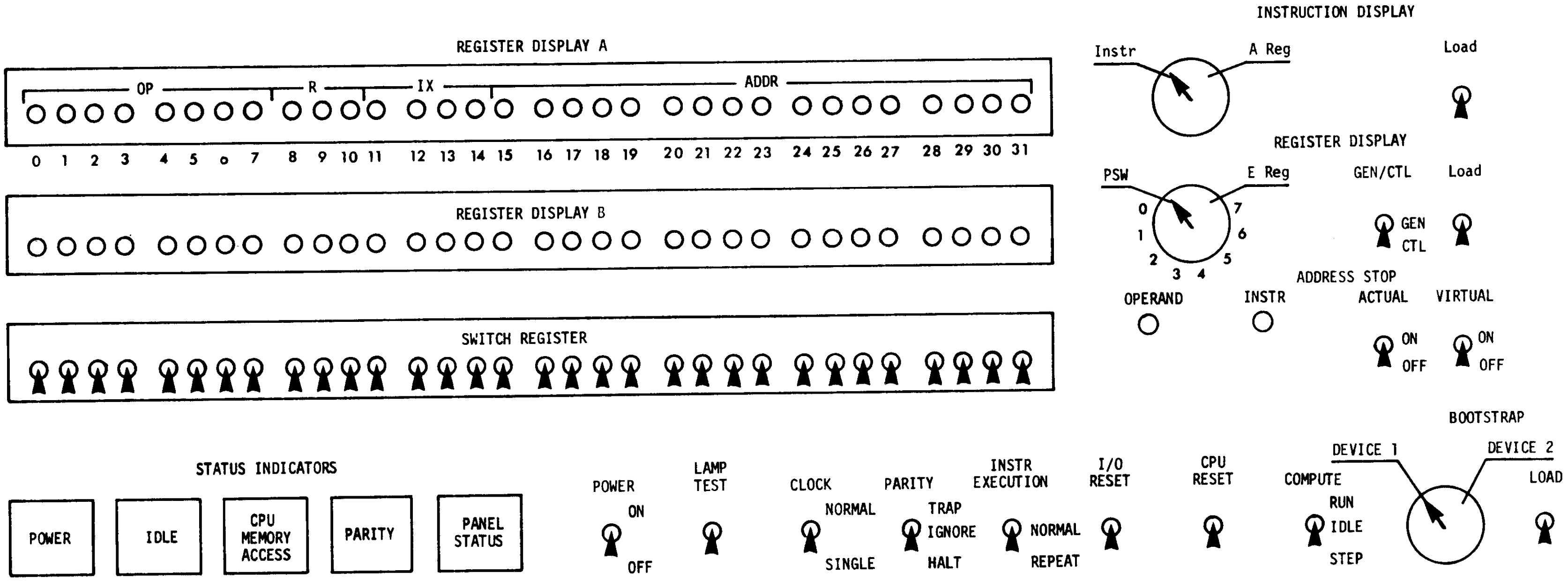


Figure 4. TENET 210 CPU Control Panel

Idle

The IDLE indicator is lit while the computer is in the idle state as a result of the COMPUTE switch set to IDLE, an address halt or parity halt. The PAUS instruction will not cause this indicator to light.

Panel Status

The PANEL STATUS indicator, if lit, indicates one or more of the panel switches are not set in their normal running position; e.g., the CLOCK switch is set to SINGLE or INSTR EXECUTION is set to REPEAT.

Parity

The PARITY indicator is lit whenever a memory parity error is detected. (If the PARITY ERROR switch is in the IGNORE or TRAP position, this light may not be noticeable when a parity error occurs.) If a parity halt occurs, the instruction causing the parity error (instruction fetch or operand fetch) will have completed execution and the program counter and the instruction register will represent the next instruction to be executed.

CPU Memory Access

The CPU MEMORY ACCESS indicator is lit whenever a machine malfunction is detected and the CPU is unable to gain access to memory.

Switches

Power

The POWER switch controls all AC power to the CPU and to all units under its direct control.

Lamp Test

The LAMP TEST switch tests the working condition of the register indicators and status indicators. When this switch is on, all indicators that are in working order will light.

Clock

The CLOCK switch is used in maintenance and machine debugging to change the stepping mode of the computer. When this switch is in the NORMAL position, changing the mode from IDLE to STEP results in the execution of one instruction; in the SINGLE position, going from IDLE to STEP advances the computer one clock pulse.

Parity

The PARITY switch controls the action of the computer when a memory parity error occurs. In the TRAP position, a parity error results in a trap to memory location 8; in the IGNORE position, parity errors are ignored; in the HALT position, the CPU halts upon detection of a parity error and the PARITY indicator is illuminated.

Instruction Repeat

When the INSTR REPEAT switch is on, the instruction in the instruction register is repeatedly executed, and the address field of the instruction is incremented by 1 after each instruction execution.

IO Reset

The IO RESET switch is used to initialize the input/output system. When the switch is triggered, all peripheral processors are reset to the ready condition, and all status, interrupt, and control indicators are reset. The current state of the central processor is not affected by this switch.

CPU Reset

The CPU RESET switch is used to initialize the central processor. When the switch is triggered, a NOP instruction will be loaded into the instruction register, bits 0-14 of the program status word will be set to all zeros, and the PSW's program counter (bits 15-31) will be set to 10_{16}. The CPU RESET switch does not reset the general registers, control registers, or map status register.

Compute

The COMPUTE switch is used to control the execution of instructions. The RUN and IDLE positions are latching, and the STEP position is momentary. In the RUN position, instructions are executed at full machine speed. In the IDLE position, the CPU halts with the instruction register containing the next instruction to be executed and the program counter containing the location of the current instruction. After going from RUN to IDLE, if the instruction register contains an execute instruction pointing to a trap location, a trap condition occurred during the last instruction execution, and the program counter will point to the memory location of the instruction causing the trap. Each time the switch is moved to the STEP position, one instruction is executed or the CPU is advanced one clock pulse, depending on the setting of the CLOCK switch.

Bootstrap

BOOTSTRAP LOAD is a momentary switch that is used to load one binary card or one record from disc into core memory, starting at location 10_{16}. The two-position rotary switch to the left of the LOAD switch is used to select one of two devices from which to load the record. DEVICE 1 generally is assigned to a card reader and DEVICE 2 to a disc.

The bootstrap sequence is initiated from the card reader processor's or disc processor's initialized state. For the card reader, the word counter is set to 30 for a 30-word transfer, and the address counter is set to memory location 10_{16}, the starting location of the bootstrap load area. For the disc processor, the word counter and memory address counter are set to 32 words starting at location 10_{16}, and a seek is initiated to disc address 00 (cylinder, head, and sector). An interrupt is given on zero word count or abnormal end.

NORMAL LOAD PROCEDURE

The following procedure is used to initially load pro-programs into core memory:

a. Set COMPUTE switch to IDLE

b. Trigger the CPU RESET switch (to set the PSW's program counter to 10_{16}).

c. Trigger the I/O RESET switch.

d. Set the BOOTSTRAP LOAD switch to the appropriate device.

e. Trigger the LOAD switch.

f. Set COMPUTE switch to RUN (to begin execution.

APPENDIX A. CONVERSION TABLES

This appendix contains the following 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}
281 474 976 710 656	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}

HEXADECIMAL—DECIMAL INTEGER CONVERSION

The table below provides for direct conversions between hexadecimal integers in the range 0 – FFF and decimal integers in the range 0 – 4095. For conversion of larger integers, the table values may be added to the following figures:

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 fractions may be converted to decimal fractions as follows:

1. Express the hexadecimal fraction as an integer times 16^{-n}, where n is the number of significant hexadecimal places to the right of the hexadecimal point.

$$0.\mathrm{CA9BF3}_{16} = \mathrm{CA9\ BF3}_{16} \times 16^{-6}$$

2. Find the decimal equivalent of the hexadecimal integer

$$\mathrm{CA9\ BF3}_{16} = 13\ 278\ 195_{10}$$

3. Multiply the decimal equivalent by 16^{-n}.

$$\begin{array}{r} 13\ 278\ 195 \\ \underline{\times\ 596\ 046\ 448 \times 10^{-16}} \\ 0.791\ 442\ 096_{10} \end{array}$$

Decimal fractions may be converted to hexadecimal fractions by successively multiplying the decimal fraction by 16_{10}. After each multiplication, the integer portion is removed to form a hexadecimal fraction by building to the right of the hexadecimal point. However, since decimal arithmetic is used in this conversion, the integer portion of each product must be converted to hexadecimal numbers.

Example: Convert 0.895_{10} to its hexadecimal equivalent

$$\begin{array}{r} 0.895 \\ \underline{16} \\ ⑭.320 \\ \underline{16} \\ ⑤.120 \\ \underline{16} \\ ①.920 \\ \underline{16} \\ ⑭.720 \end{array}$$

$0.\mathrm{E51E}_{16}$

	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

HEXADECIMAL - DECIMAL INTEGER CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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	0520	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
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

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
400	1024	1025	1026	1027	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	1292	1293	1294	1295
510	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
520	1312	1313	1314	1315	1316	1317	1318	1319	1320	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	1357	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	1420	1421	1422	1423
590	1424	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
5B0	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	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535
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	1593	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	1723	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

HEXADECIMAL - DECIMAL INTEGER CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
700	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	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	1828	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	1901	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
8E0	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
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	2396	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	2435	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	2470	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

HEXADECIMAL - DECIMAL INTEGER CONVERSION TABLE (cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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	2616	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
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

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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
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

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 42431	.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 3B 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 66161	.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 07288	.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 72121	.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 FD 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 (cont.)

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

POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719,476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625

MATHEMATICAL CONSTANTS

Constant	Decimal Value	Hexadecimal Value
π	3.14159 26535 89793	3.243F 6A89
π^{-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
$\ln 2$	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. ANSI CHARACTER SET

ANSI Hex Code	Character	Teletype Key	Teletype/ Printer Graphic	Hollerith Card Code
00	NUL	P^{cs}		12-0-9-8-1
01	SOH or DEL	A^c		12-9-1
02	STX	B^c		12-9-2
03	ETX	C^c		12-9-3
04	EOT	D^c		9-7
05	ENQ	E^c		0-9-8-5
06	ACK	F^c		0-9-8-6
07	BEL	G^c		0-9-8-7
08	BS	H^c		11-9-6
09	HT	I^c		12-9-5
0A	LF	Line Feed		0-9-5
0B	VT	K^c		12-9-8-3
0C	FF	L^c		12-9-8-4
0D	CR	Return		12-9-8-5
0E	SO	N^c		12-9-8-6
0F	SI	O^c		12-9-8-7
10	DLE	P^c		12-11-9-8-1
11	DC1	Q^c		11-9-1
12	DC2	R^c		11-9-2
13	DC3	S^c		11-9-3
14	DC4	T^c		9-8-4
15	NAK	U^c		9-8-5
16	SYN	V^c		9-2
17	ETB	W^c		0-9-6
18	CAN	X^c		11-9-8
19	EM	Y^c		11-9-8-1
1A	SUB	Z^c		9-8-7
1B	ESC	K^{cs}		0-9-7
1C	FS	L^{cs}		11-9-8-4
1D	GS	M^{cs}		11-9-8-5
1E	RS	N^{cs}		11-9-8-6
1F	US	O^{cs}		11-9-8-7
20	Blank	Space Bar		
21	!	1^s	!	12-8-7
22	"	2^s	"	8-7
23	#	3^s	#	8-3
24	$	4^s	$	11-8-3
25	%	5^s	%	0-8-4
26	&	6^s	&	12
27	'	7^s	'	8-5
28	(	8^s	(	12-8-5
29	)	9s	)	11-8-5
2A	*	$:^s$	*	11-8-4
2B	+	$;^s$	+	12-8-6
2C	,	,	,	0-8-3
2D	-	-	-	11
2E	.	.	.	12-8-3
2F	/	/	/	0-1
30	0	0	0	0
31	1	1	1	1
32	2	2	2	2
33	3	3	3	3
34	4	4	4	4
35	5	5	5	5
36	6	6	6	6
37	7	7	7	7
38	8	8	8	8
39	9	9	9	9
3A	:	:	:	8-2
3B	;	;	;	11-8-6
3C	<	$,^s$	<	12-8-4
3D	=	$\_^s$	=	8-6
3E	>	$.^s$	>	0-8-6
3F	?	$/^s$	?	0-8-7

ANSI Hex Code	Character	Teletype Key	Teletype/ Printer Graphic	Hollerith Card Code	ANSI Hex Code	Character	Teletype Key	Teletype/ Printer Graphic	Hollerith Card Code
40	@	P^S	@	8-4	60	` or ¢		@	8-1
41	A	A	A	12-1	61	a		A	12-0-1
42	B	B	B	12-2	62	b		B	12-0-2
43	C	C	C	12-3	63	c		C	12-0-3
44	D	D	D	12-4	64	d		D	12-0-4
45	E	E	E	12-5	65	e		E	12-0-5
46	F	F	F	12-6	66	f		F	12-0-6
47	G	G	G	12-7	67	g		G	12-0-7
48	H	H	H	12-8	68	h		H	12-0-8
49	I	I	I	12-9	69	i		I	12-0-9
4A	J	J	J	11-1	6A	j		J	12-11-1
4B	K	K	K	11-2	6B	k		K	12-11-2
4C	L	L	L	11-3	6C	l		L	12-11-3
4D	M	M	M	11-4	6D	m		M	12-11-4
4E	N	N	N	11-5	6E	n		N	12-11-5
4F	O	O	O	11-6	6F	o		O	12-11-6
50	P	P	P	11-7	70	p		P	12-11-7
51	Q	Q	Q	11-8	71	q		Q	12-11-8
52	R	R	R	11-9	72	r		R	12-11-9
53	S	S	S	0-2	73	s		S	11-0-2
54	T	T	T	0-3	74	t		T	11-0-3
55	U	U	U	0-4	75	u		U	11-0-4
56	V	V	V	0-5	76	v		V	11-0-5
57	W	W	W	0-6	77	w		W	11-0-6
58	X	X	X	0-7	78	x		X	11-0-7
59	Y	Y	Y	0-8	79	y		Y	11-0-8
5A	Z	Z	Z	0-9	7A	z		Z	11-0-9
5B	[	K^S	[	12-8-2	7B	{			12-0
5C	\	L^S	\	0-8-2	7C	|			12-11
5D	]	M^S	]	11-8-2	7D	}			11-0
5E	↑	N^S	↑	11-8-7	7E	¬ or ~			11-0-1
5F	←	O^S	←	0-8-5	7F	RUBOUT			12-9-7

APPENDIX C. OPERATION CODE ASSIGNMENTS

<table>
<tr><th></th><th></th><th colspan="16">LEAST SIGNIFICANT DIGIT</th></tr>
<tr><th></th><th>Hex</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>A</th><th>B</th><th>C</th><th>D</th><th>E</th><th>F</th></tr>
<tr><td rowspan="16">MOST SIGNIFICANT DIGIT</td><td>0</td><td></td><td></td><td></td><td>TRB</td><td></td><td></td><td></td><td>TRBR</td><td></td><td></td><td></td><td>TRBS</td><td></td><td></td><td></td><td>STRB</td></tr>
<tr><td>1</td><td colspan="4">◄— TB —►</td><td colspan="4">◄— TBR —►</td><td colspan="4">◄— TBS —►</td><td colspan="4">◄— STB —►</td></tr>
<tr><td>2</td><td>SLL</td><td>SLLD</td><td>SLR</td><td>SLRD</td><td>SUN</td><td>SUND</td><td>FRB</td><td>FRBD</td><td>SAL</td><td>SALD</td><td>SAR</td><td>SARD</td><td>SCL</td><td>SCLD</td><td>SCR</td><td>SCRD</td></tr>
<tr><td>3</td><td>AW</td><td>SW</td><td>STW</td><td>LW</td><td>ANDW</td><td>ORW</td><td>EORW</td><td>CW</td><td>ISW</td><td></td><td>MW</td><td>DW</td><td>LWD</td><td></td><td>MD</td><td>DD</td></tr>
<tr><td>4</td><td>AM</td><td>SM</td><td>XW</td><td>ILW</td><td>ANDM</td><td>ORM</td><td>EORM</td><td></td><td>AF</td><td>AFA</td><td>STF</td><td>DFP</td><td>LF</td><td>LFA</td><td>CF</td><td>CFA</td></tr>
<tr><td>5</td><td>AD</td><td>SD</td><td>STD</td><td>LD</td><td></td><td></td><td></td><td>CD</td><td>SF</td><td>SFA</td><td>STFI</td><td>IFP</td><td>LFI</td><td>LFAI</td><td>CFI</td><td>CFAI</td></tr>
<tr><td>6</td><td>AIM</td><td>SIM</td><td>STIM</td><td></td><td>CAIM</td><td></td><td></td><td>CIM</td><td>ISF</td><td>ISFA</td><td>STIF</td><td></td><td></td><td></td><td>CIF</td><td></td></tr>
<tr><td>7</td><td>AI</td><td>SUBI</td><td></td><td>LI</td><td>ANDI</td><td>ORI</td><td>EORI</td><td>CI</td><td>ISI</td><td></td><td>MI</td><td>DI</td><td>MOVI</td><td>MOVD</td><td></td><td></td></tr>
<tr><td>8</td><td>BRCR</td><td>BRCS</td><td>BIR</td><td>BDR</td><td>BCR</td><td>BCS</td><td>BAL</td><td>BAC</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>EXEC</td></tr>
<tr><td>9</td><td>IOFF</td><td>ION</td><td></td><td>PAUS</td><td>PPS</td><td>POPS</td><td>STCR</td><td>LCR</td><td>XPS</td><td>LPS</td><td>IOC</td><td></td><td>LMS</td><td>LM</td><td>STMS</td><td>LAA</td></tr>
<tr><td>A</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>STG</td><td>LG</td><td>STPS</td><td></td><td>LAS</td><td>LCI</td><td>STSR</td><td>LVA</td></tr>
<tr><td>B</td><td>FAL</td><td>FSL</td><td>FML</td><td>FDL</td><td>FAS</td><td>FSS</td><td>FMS</td><td>FDS</td><td>FISL</td><td>FISS</td><td>FIDL</td><td>FIDS</td><td></td><td></td><td></td><td></td></tr>
<tr><td>C</td><td colspan="16">◄— GLOBAL PINS —►</td></tr>
<tr><td>D</td><td colspan="16">◄— LOCAL PINS —►</td></tr>
<tr><td>E</td><td colspan="16">◄— LOCAL PINS —►</td></tr>
<tr><td>F</td><td colspan="16">◄— LOCAL PINS —►</td></tr>
</table>

APPENDIX D. BASIC OPERATION CODES– NUMERIC LISTING

Code	Mnemonic	Instruction Name	Type	Page
00				
01				
02				
03	TRB	Test Register Designated Bit	Register Designated Bit	30
04				
05				
06				
07	TRBR	Test Register Designated Bit and Reset	Register Designated Bit	30
08				
09				
0A				
0B	TRBS	Test Register Designated Bit and Set	Register Designated Bit	30
0C				
0D				
0E				
0F	STRB	Store Register Designated Bit	Register Designated Bit	30
10-13	TB	Test Bit	Test Bit	28
14-17	TBR	Test Bit and Reset	Test Bit	28
18-1B	TBS	Test Bit and Set	Test Bit	28
1C-1F	STB	Store Bit	Test Bit	29
20	SLL	Shift Logical Left	Shift	23
21	SLLD	Shift Logical Left Double	Shift	24
22	SLR	Shift Logical Right	Shift	24
23	SLRD	Shift Logical Right Double	Shift	24
24	SUN	Shift Until Normalized	Shift	25
25	SUND	Shift Until Normalized Double	Shift	25
26	FRB	Find and Reset Bit	Register Designated Bit	30
27	FRBD	Find and Reset Bit Double	Register Designated Bit	30
28	SAL	Shift Arithmetic Left	Shift	22
29	SALD	Shift Arithmetic Left Double	Shift	22
2A	SAR	Shift Arithmetic Right	Shift	23
2B	SARD	Shift Arithmetic Right Double	Shift	23
2C	SCL	Shift Circular Left	Shift	24
2D	SCLD	Shift Circular Left Double	Shift	24
2E	SCR	Shift Circular Right	Shift	24
2F	SCRD	Shift Circular Right Double	Shift	24
30	AW	Add Word	Word	15
31	SW	Subtract Word	Word	15
32	STW	Store Word	Word	15
33	LW	Load Word	Word	15
34	ANDW	AND Word	Word	15
35	ORW	OR Word	Word	16
36	EORW	Exclusive OR Word	Word	16
37	CW	Compare Word	Word	16
38	ISW	Inverse Subtract Word	Word	15
39				
3A	MW	Multiply Word	Word	15
3B	DW	Divide Word	Word	15
3C	LWD	Load Word Doubled	Word	16
3D				
3E	MD	Multiply Doubleword	Doubleword	17
3F	DD	Divide Doubleword	Doubleword	18
40	AM	Add to Memory	Memory	18
41	SM	Subtract from Memory	Memory	18
42	XW	Exchange Word	Word	16
43	ILW	Increment and Load Word	Memory	19

Code	Mnemonic	Instruction Name	Type	Page
44	ANDM	AND to Memory	Memory	19
45	ORM	OR to Memory	Memory	19
46	EORM	Exclusive OR to Memory	Memory	19
47				
48	AF	Add Field Logical	Field	34
49	AFA	Add Field Arithmetic	Field	34
4A	STF	Store Field	Field	32
4B	DFP	Decrement Field Pointer	Field	32
4C	LF	Load Field Logical	Field	32
4D	LFA	Load Field Arithmetic	Field	32
4E	CF	Compare Field Logical	Field	33
4F	CFA	Compare Field Arithmetic	Field	33
50	AD	Add Doubleword	Doubleword	17
51	SD	Subtract Doubleword	Doubleword	17
52	STD	Store Doubleword	Doubleword	39
53	LD	Load Doubleword	Doubleword	16
54				
55				
56				
57	CD	Compare Doubleword	Doubleword	18
58	SF	Subtract Field Logical	Field	34
59	SFA	Subtract Field Arithmetic	Field	34
5A	STFI	Store Field after Increment	Field	33
5B	IFP	Increment Field Pointer	Field	32
5C	LFI	Load Field Logical after Increment	Field	32
5D	LFAI	Load Field Arithmetic after Increment	Field	32
5E	CFI	Compare Field Logical after Increment	Field	33
5F	CFAI	Compare Field Arithmetic after Increment	Field	33
60	AIM	Add Immediate to Memory	Memory	20
61	SIM	Subtract Immediate from Memory	Memory	20
62	STIM	Store Immediate to Memory	Memory	20
63				
64	CAIM	Complement and Add Immediate to Memory	Memory	19
65				
66				
67	CIM	Compare Immediate with Memory	Memory	20
68	ISF	Inverse Subtract Field Logical	Field	34
69	ISFA	Inverse Subtract Field Arithmetic	Field	34
6A	STIF	Store Immediate to Field	Field	33
6B				
6C				
6D				
6E	CIF	Compare Immediate to Field	Field	33
6F				
70	AI	Add Immediate	Immediate	21
71	SUBI	Subtract Immediate	Immediate	21
72				
73	LI	Load Immediate	Immediate	21
74	ANDI	AND Immediate	Immediate	21
75	ORI	OR Immediate	Immediate	21
76	EORI	Exclusive OR Immediate	Immediate	21
77	CI	Compare Immediate	Immediate	22
78	ISI	Inverse Subtract Immediate	Immediate	22
79				
7A	MI	Multiply Immediate	Immediate	21
7B	DI	Divide Immediate	Immediate	21
7C	MOVI	Move with Incrementing Pointer	Unprivileged Control	36
7D	MOVD	Move with Decrementing Pointer	Unprivileged Control	36
7E				
7F				
80	BRCR	Branch on Register Conditions Reset	Branch	27
81	BRCS	Branch on Register Conditions Set	Branch	25
82	BIR	Branch on Incrementing Register	Branch	28
83	BDR	Branch on Decrementing Register	Branch	28
84	BCR	Branch on Conditions Reset	Branch	27
85	BCS	Branch on Conditions Set	Branch	25

Code	Mnemonic	Instruction Name	Type	Page
86	BAL	Branch and Link	Branch	25
87	BAC	Branch on Arithmetic Conditions	Branch	25
88				
89				
8A				
8B				
8C				
8D				
8E				
8F	EXEC	Execute	Unprivileged Control	35
90	IOFF	System Interrupts OFF	Privileged Control	38
91	ION	System Interrupts ON	Privileged Control	38
92				
93	PAUS	Pause	Privileged Control	38
94	PPS	Push Program Status	Privileged Control	39
95	POPS	Pop Program Status	Privileged Control	39
96	STCR	Store Control Register	Privileged Control	39
97	LCR	Load Control Register	Privileged Control	39
98	XPS	Exchange Program Status	Privileged Control	37
99	LPS	Load Program Status	Privileged Control	37
9A	IOC	Input/Output Control	Input/Output	49
9B				
9C	LMS	Load Map Status	Privileged Control	38
9D	LM	Load Map Segment	Privileged Control	38
9E	STMS	Store Map Status	Privileged Control	38
9F	LAA	Load Actual Address	Privileged Control	39
A0				
A1				
A2				
A3				
A4				
A5				
A6				
A7				
A8	STG	Store Register Group	Unprivileged Control	35
A9	LG	Load Register Group	Unprivileged Control	35
AA	STPS	Store Program Status	Unprivileged Control	34
AB				
AC	LAS	Load and Set	Unprivileged Control	35
AD	LCI	Load Condition Indicators	Unprivileged Control	34
AE	STSR	Store Switch Register	Unprivileged Control	36
AF	LVA	Load Virtual Address	Unprivileged Control	36
B0	FAL	Floating Add Long	Floating Point	43
B1	FSL	Floating Subtract Long	Floating Point	44
B2	FML	Floating Multiply Long	Floating Point	44
B3	FDL	Floating Divide Long	Floating Point	44
B4	FAS	Floating Add Short	Floating Point	43
B5	FSS	Floating Subtract Short	Floating Point	44
B6	FMS	Floating Multiply Short	Floating Point	44
B7	FDS	Floating Divide Short	Floating Point	44
B8	FISL	Floating Inverse Subtract Long	Floating Point	44
B9	FISS	Floating Inverse Subtract Short	Floating Point	44
BA	FIDL	Floating Inverse Divide Long	Floating Point	45
BB	FIDS	Floating Inverse Divide Short	Floating Point	45
BC				
BD				
BE				
BF				
C0-CF	Global Pins			40
D0-FF	Local Pins			40

APPENDIX E. INSTRUCTION MNEMONICS – ALPHABETIC LISTING

Mnemonic	Code-R field	Instruction Name	Type	Page
AD	50	Add Doubleword	Doubleword	17
AF	48	Add Field Logical	Field	34
AFA	49	Add Field Arithmetic	Field	34
AI	70	Add Immediate	Immediate	21
AIM	60	Add Immediate to Memory	Memory	20
AM	40	Add to Memory	Memory	18
ANDI	74	AND Immediate	Immediate	21
ANDM	44	AND to Memory	Memory	19
ANDW	34	AND Word	Word	15
AW	30	Add Word	Word	15
BAC	87	Branch on Arithmetic Conditions	Branch	25
B	87-0		Branch	25
BNO	87-1		Branch	25
BNC	87-2		Branch	25
BNCO	87-3	Extended BAC	Branch	25
NOP	87-4		Branch	25
BOV	87-5		Branch	25
BCRY	87-6		Branch	25
BCO	87-7		Branch	25
BAL	86	Branch and Link	Branch	25
BCR	84	Branch on Conditions Reset	Branch	27
—	84-0		Branch	
BGE	84-1		Branch	27
BLE	84-2		Branch	27
BE	84-3	Extended BCR	Branch	27
BNMO, BBR	84-4		Branch	27
—	84-5		Branch	
—	84-6		Branch	
—	84-7		Branch	
BCS	85	Branch on Conditions Set	Branch	25
—	85-0		Branch	
BL	85-1		Branch	27
BG	85-2		Branch	27
BNE	85-3	Extended BCS	Branch	27
BMO, BBS	85-4		Branch	27
—	85-5		Branch	
—	85-6		Branch	
—	85-7		Branch	
BDR	83	Branch on Decrementing Register	Branch	28
BIR	82	Branch on Incrementing Register	Branch	28
BRCR	80	Branch on Register Conditions Reset	Branch	27
—	80-0		Branch	
BRGE	80-1		Branch	28
BRLE	80-2		Branch	28
BRE	80-3	Extended BRCR	Branch	28
BREV	80-4		Branch	28
—	80-5		Branch	
—	80-6		Branch	
—	80-7		Branch	
BRCS	81	Branch on Register Conditions Set	Branch	27
—	81-0		Branch	
BRL	81-1		Branch	27
BRG	81-2	Extended BRCS	Branch	27
BRNE	81-3		Branch	27
BROD	81-4		Branch	27

Mnemonic	Code-R field	Instruction Name	Type	Page
—	81-5	↑	Branch	
—	81-6	Extended BRCS cont.	Branch	
—	81-7	↓	Branch	
CAIM	64	Complement and Add Immediate to Memory	Memory	19
CD	57	Compare Doubleword	Doubleword	18
CF	4E	Compare Field Logical	Field	33
CFA	4F	Compare Field Arithmetic	Field	33
CFAI	5F	Compare Field Arithmetic after Increment	Field	33
CFI	5E	Compare Field Logical after Increment	Field	33
CI	77	Compare Immediate	Immediate	22
CIF	6E	Compare Immediate to Field	Field	33
CIM	67	Compare Immediate with Memory	Memory	20
CW	37	Compare Word	Word	16
DD	3F	Divide Doubleword	Doubleword	18
DFP	4B	Decrement Field Pointer	Field	32
DI	7B	Divide Immediate	Immediate	21
DW	3B	Divide Word	Word	15
EORI	76	Exclusive OR Immediate	Immediate	21
EORM	46	Exclusive OR to Memory	Memory	19
EORW	36	Exclusive OR Word	Word	16
EXEC	8F	Execute	Unprivileged Control	35
FAL	B0	Floating Add Long	Floating Point	43
FAS	B4	Floating Add Short	Floating Point	43
FDL	B3	Floating Divide Long	Floating Point	44
FDS	B7	Floating Divide Short	Floating Point	44
FIDL	BA	Floating Inverse Divide Long	Floating Point	45
FIDS	BB	Floating Inverse Divide Short	Floating Point	45
FISL	B8	Floating Inverse Subtract Long	Floating Point	44
FISS	B9	Floating Inverse Subtract Short	Floating Point	44
FML	B2	Floating Muptiply Long	Floating Point	44
FMS	B6	Floating Multiply Short	Floating Point	44
FRB	26	Find and Reset Bit	Register-Designated Bit	30
FRBD	27	Find and Reset Bit Double	Register-Designated Bit	30
FSL	B1	Floating Subtract Long	Floating Point	44
FSS	B5	Floating Subtract Short	Floating Point	44
IFP	5B	Increment Field Pointer	Field	32
ILW	43	Increment and Load Word	Memory	19
IOC	9A	Input/Output Control	Input/Output	40,49
IOFF	90	System Interrupt OFF	Privileged Control	38
ION	91	System Imterrupts ON	Privileged Control	38
ISF	68	Inverse Subtract Field Logical	Field	34
ISFA	69	Inverse Subtract Field Arithmetic	Field	34
ISI	78	Inverse Subtract Immediate	Immediate	22
ISW	38	Inverse Subtract Word	Word	15
LAA	9F	Load Actual Address	Privileged Control	39
LAS	AC	Load and Set	Unpriveleged Control	35
LCI	AD	Load Condition Indicators	Unprivileged Control	34
LCR	97	Load Control Register	Privileged Control	39
LD	53	Load Doubleword	Doubleword	16
LF	4C	Load Field Logical	Field	32
LFA	4D	Load Field Arithmetic	Field	32
LFAI	5D	Load Field Arithmetic after Increment	Field	32
LFI	5C	Load Field Logical after Increment	Field	32
LG	A9	Load Register Group	Unprivileged Control	35
LI	73	Load Immediate	Immediate	21
LM	9D	Load Map Segment	Privileged Control	38
LMS	9C	Load Map Status	Privileged Control	38
LPS	99	Load Program Status	Privileged Control	37
LVA	AF	Load Virtual Address	Unprivileged Control	36
LW	33	Load Word	Word	15

Mnemonic	Code	Instruction Name	Type	Page
LWD	3C	Load Word Doubled	Word	16
MD	3E	Multiply Doubleword	Doubleword	17
MI	7A	Multiple Immediate	Immediate	21
MOVD	7D	Move with Decrementing Pointer	Unprivileged Control	36
MOVI	7C	Move with Incrementing Pointer	Unprivileged Control	36
MW	3A	Multiply Word	Word	15
ORI	75	OR Immediate	Immediate	21
ORM	45	OR to Memory	Memory	19
ORW	35	OR Word	Word	16
PAUS	93	Pause	Privileged Control	38
PIN	C0-FF	Programmed Instruction	Unprivileged Control	40
POPS	95	Pop Program Status	Privileged Control	39
PPS	94	Push Program Status	Privileged Control	39
SAL	28	Shift Arithmetic Left	Shift	22
SALD	29	Shift Arithmetic Left Double	Shift	22
SAR	2A	Shift Arithmetic Right	Shift	23
SARD	2B	Shift Arithmetic Right Double	Shift	23
SCL	2C	Shift Circular Left	Shift	24
SCLD	2D	Shift Circular Left Double	Shift	24
SCR	2E	Shift Circular Right	Shift	24
SCRD	2F	Shift Circular Right Double	Shift	24
SD	51	Subtract Doubleword	Doubleword	17
SF	58	Subtract Field Logical	Field	34
SFA	59	Subtract Field Arithmetic	Field	34
SIM	61	Subtract Immediate from Memory	Memory	20
SLL	20	Shift Logical Left	Shift	23
SLLD	21	Shift Logical Left Double	Shift	24
SLR	22	Shift Logical Right	Shift	24
SLRD	23	Shift Logical Right Double	Shift	24
SM	41	Subtract from Memory	Memory	18
STB	1C	Store Bit	Test Bit	29
STCR	96	Store Control Register	Privileged Control	29
STD	52	Store Doubleword	Doubleword	39
STF	4A	Store Field	Field	32
STFI	5A	Store Field after Increment	Field	33
STG	A8	Store Register Group	Unprivileged Control	35
STIF	6A	Store Immediate to Field	Field	33
STIM	62	Store Immediate to Memory	Memory	20
STMS	9E	Store Map Status	Privileged Control	38
STPS	AA	Store Program Status	Unprivileged Control	34
STRB	0F	Store Register-Designated Bit	Register-Designated Bit	30
STSR	AE	Store Switch Register	Unprivileged Control	36
STW	32	Store Word	Word	15
SUBI	71	Subtract Immediate	Immediate	21
SUN	24	Shift until Normalized	Shift	25
SUND	25	Shift until Normalized Double	Shift	25
SW	31	Subtract Word	Word	15
TB	14	Test Bit	Test Bit	28
TBR	16	Test Bit and Reset	Test Bit	28
TBS	18	Test Bit and Set	Test Bit	28
TRB	03	Test Register-Designated Bit	Register-Designated Bit	30
TRBR	07	Test Register-Designated Bit and Reset	Register-Designated Bit	30
TRBS	0B	Test Register-Designated Bit and Set	Register-Designated Bit	30
XPS	98	Exchange Program Status	Privileged Control	37
XW	42	Exchange Word	Word	16

APPENDIX F. INSTRUCTION TIMING

This appendix shows the timing (in microseconds) for executing individual TENET 210 computer instructions. All of the times are based on the assumption that whenever the CPU requests a service cycle from a particular memory bank, it never has to wait for such service due to other devices (such as IOPs) that may be currently accessing that memory bank.

Execution times depend not only on the nature of the specific instructions, but also on the configuration of memory banks.

The Basic Instruction Timing Table assumes complete memory overlap, no indexing, no indirect addressing, and no mapping. The instruction time for the basic instruction should be increased according to the following conditions:

Each map reference	.2 μs
Indexing	.4 μs
Indirect from register	.2 μs
Indirect from memory	.8 μs
Operand and instruction in the same memory bank (no memory overlap)	.4 μs

BASIC INSTRUCTION TIMING

Instruction	Direct from Register	Direct from Memory	Comments
WORD			
LW	1.0	1.4	
STW	1.0	1.4	
AW	1.0	1.4	
SW	1.0	1.4	
ISW	1.0	1.4	
MW	1.6 + n * .2	2.0 + n * .2	n = number of shifts required to perform multiplication where $4 \leq n \leq 16$
DW	9.4	9.8	
ANDW	1.0	1.4	
ORW	1.0	1.4	
EORW	1.0	1.4	
CW	1.0	1.4	
XW	1.4	2.2	
LWD	1.0	1.4	
DOUBLEWORD			
LD	1.6	2.4	
STD	2.0	2.4	
AD	2.0	2.8	
SD	2.0	2.8	
CD	1.8	2.6	
DD	9.4	9.8	
MD	1.6 + n * .2	2.0 + n * .2	n = number of shifts required to perform multiplication where $4 \leq n \leq 16$
MEMORY			
AM	1.2	2.0	
SM	1.2	2.0	
ANDM	1.2	2.0	
ORM	1.2	2.0	
ILW	1.2	2.0	
EORM	1.2	2.0	
CAIM	1.2	2.0	
AIM	1.2	2.0	
SIM	1.2	2.0	
STIM	1.0	1.4	
CIM	1.0	1.4	
SHIFT			
SAL	1.6 to 3.0	NA	
SAR	1.6 to 3.0	NA	
SALD	2.0 to 4.2	NA	
SARD	2.0 to 4.2	NA	

BASIC INSTRUCTION TIMING (Continued)

Instruction	Direct from Register	Direct from Memory	Comments
SHIFT (Continued)			
SLL	1.6 to 3.0	NA	
SLR	1.6 to 3.0	NA	
SLLD	2.0 to 4.2	NA	
SLRD	2.0 to 4.2	NA	
SCL	1.6 to 2.6	NA	
SCR	1.6 to 2.6	NA	
SCLD	2.0 to 3.4	NA	
SCRD	2.0 to 3.4	NA	
SUN	1.8 to 3.2	NA	
SUND	2.2 to 4.4	NA	
FIELD			
DFP	1.8 2.2	2.2 2.6	Decrements within word Decrements to next word
IFP	1.8 2.2	2.2 2.6	Increments within word Increments to next word
AFA	2.4 to 3.4 2.8 to 3.8	2.6 to 3.6 3.0 to 4.0	Pointer in register Pointer in memory
AF	2.2 to 3.2 2.6 to 3.6	2.4 to 3.4 2.8 to 3.8	Pointer in register Pointer in memory
SFA	2.4 to 3.4 2.8 to 3.8	2.6 to 3.6 3.0 to 4.0	Pointer in register Pointer in memory
SF	2.2 to 3.2 2.6 to 3.6	2.4 to 3.4 2.8 to 3.8	Pointer in register Pointer in memory
ISFA	2.4 to 3.4 2.8 to 3.8	2.6 to 3.6 3.0 to 4.0	Pointer in register Pointer in memory
ISF	2.2 to 3.2 2.6 to 3.6	2.4 to 3.4 2.8 to 3.8	Pointer in register Pointer in memory
LF	2.2 to 3.2 2.6 to 3.6	2.4 to 3.4 2.8 to 3.8	Pointer in register Pointer in memory
LFI	2.6 to 3.6 3.0 to 4.0 3.0 to 4.0 3.4 to 4.4	3.2 to 4.2 3.6 to 4.6 3.6 to 4.6 4.0 to 5.0	Pointer in register; increments within word Pointer in register; increments to next word Pointer in memory; increments within word Pointer in memory; increments to next word
LFA	2.4 to 3.4 2.8 to 3.8	2.6 to 3.6 3.0 to 4.0	Pointer in register Pointer in memory
LFAI	2.8 to 3.8 3.2 to 4.2 3.2 to 4.2 3.6 to 4.6	3.4 to 4.4 3.8 to 4.8 3.8 to 4.8 4.2 to 5.2	Pointer in register; increments within word Pointer in register; increments to next word Pointer in memory; increments within word Pointer in memory; increments to next word
STF	3.2 to 4.2	4.0 to 5.0	
STFI	3.6 to 4.6 4.0 to 5.0	4.8 to 5.8 5.2 to 6.2	Increments within word Increments to next word
STIF	3.2 to 4.2	4.0 to 5.0	
CF	2.2 to 3.2 2.6 to 3.6	2.4 to 3.4 2.8 to 3.8	Pointer in register Pointer in memory

BASIC INSTRUCTION TIMING (Continued)

Instruction	Direct from Register	Direct from Memory	Comments
FIELD (Continued)			
CFI	2.8 to 3.6 3.0 to 4.0 3.0 to 4.0 3.4 to 4.4	3.2 to 4.2 3.6 to 4.6 3.6 to 4.6 4.0 to 5.0	Pointer in register; increments within word Pointer in register; increments to next word Pointer in memory; increments within word Pointer in memory; increments to next word
CFA	2.4 to 3.4 2.8 to 3.8	2.6 to 3.6 3.0 to 4.0	Pointer in register Pointer in memory
CFAI	2.8 to 3.8 3.2 to 4.2 3.2 to 4.2 3.6 to 4.6	3.4 to 4.4 3.8 to 4.8 3.8 to 4.8 4.2 to 5.2	Pointer in register; increments within word Pointer in register; increments to next word Pointer in memory; increments within word Pointer in memory; increments to next word
CIF	2.2 to 3.2 2.6 to 3.6	2.4 to 3.4 2.8 to 3.8	Pointer in register Pointer in memory
PRIVILEGED CONTROL			
LPS	8.6 1.6	9.4 2.0	Load registers No load registers
XPS	9.4 3.0	9.8 3.4	Registers No registers
LCR	1.0	1.4	
PPS	9.4 3.0	10.2 3.8	Registers No registers
STCR	1.0	1.4	
POPS	9.2 2.6	9.6 3.0	Registers No registers
PAUS	NA	NA	
ION	1.0	1.0	
IOFF	1.0	1.0	
LM	8.0	8.4	
LMS	2.6	3.4	
LAA	2.2 + x	2.6 + x	x = additional time for address modification in calculating actual address
STMS	3.6	4.6	
INPUT/OUTPUT			
IOC	NA	NA	
UNPRIVILEGED CONTROL			
STSR	1.0	1.4	
STPS	1.0	1.4	
LCI	1.0	1.2	
EXEC	0.4	0.8	
LG	6.4	7.2	
STG	7.0	7.0	
MOVI	1.4 + n * 2.8	1.4 + n * 2.8	n = number of words moved
MOVD	1.4 + n * 2.8	1.4 + n * 2.8	n = number of words moved
LAS	1.2	1.2	
LVA	2.2 + x	2.2 + x	x = additional time for address modification in calculating virtual address

BASIC INSTRUCTION TIMING (Continued)

Instruction	Direct from Register	Direct from Memory	Comments
UNPRIVILEGED CONTROL (Continued)			
PIN	NA	2.0	
BRANCH			
BAC	NA NA	0.8 1.0	Branch No branch
BCS	NA NA	0.8 1.0	Branch No branch
BCR	NA NA	0.8 1.0	Branch No branch
BRCS	NA	1.2	
BRCR	NA	1.2	
BIR	NA	1.6	
BDR	NA	1.6	
BAL	NA	0.8	
TEST REGISTER DESIGNATED BIT			
TRB	1.4	1.8	
STRB	1.4 1.4	2.0 2.4	Bit tested for is already correct state Bit tested for must be changed
TRBS	1.4 1.4	2.0 2.4	Bit tested for is already correct state Bit tested for must be changed
TRBR	1.4 1.4	2.0 2.4	Bit tested for is already correct state Bit tested for must be changed
FRB	1.8 to 8.0	2.4 to 8.6	
FRBD	1.8 to 15.4	2.4 to 15.8	
TEST BIT			
TB	1.4	1.8	
STB	1.4 1.4	2.0 2.4	Bit tested for is already correct state Bit tested for must be changed
TBS	1.4 1.4	2.0 2.4	Bit tested for is already correct state Bit tested for must be changed
TBR	1.4 1.4	2.0 2.4	Bit tested for is already correct state Bit tested for must be changed
IMMEDIATE			
LI	1.0	NA	
AI	1.0	NA	
SUBI	1.0	NA	
MI	1.6 + n * .2	NA	n = number of shifts required to perform multiplication where $4 \leq n \leq 16$
DI	9.4	NA	
ANDI	1.0	NA	
ORI	1.0	NA	
EORI	1.0	NA	
CI	1.0	NA	
ISI	1.0	NA	

APPENDIX G. INTERVAL TIMER

The Interval Timer is an optional feature of the TENET 210 System which consists of two parts: a time out counter (TOC) and a real time clock (RTC).

The TOC is a 24-bit register (counter) into which a value is loaded and subsequently decremented every 40 system clocks. When the value of the TOC is decremented to zero, an interrupt to memory location 80_{16} will occur. The TOC may only be loaded with an initial value and cause an interrupt when it is counted down to zero; it may not be read. If initially loaded with zero, it will count to its maximum count (2^{24}) back to zero.

The RTC is a 32-bit register (counter) which is continually incremented every 40 system clocks counting through 2^{32} and starting over again. The RTC may be read only; it may not be loaded, nor will it cause any interrupt.

For a 5 MHz system clock, there is a resolution of 8μs and a timer capacity of approximately 9.5 hours for the RTC and 2 minutes for the TOC.

The Interval Timer is addressed as an IO device and the instruction and commands which reference it are the same as for input/output. The TOC and RTC share the same device address.

INTERVAL TIMER INSTRUCTION

The Interval Timer instruction is the same as the standard IOC instruction used for all input/output operations. After indexing and/or indirect address modification, its format is:

9 A	R	▨	COMMAND	DEVICE ADDRESS
0 7	8 10	11 14	15 23	24 31

Only bit positions 15 through 31 are used. Interval Timer Command (bits 15-23) is the command to be executed; Interval Timer Address (bits 24-31) is the address of the Interval Timer.

I/O Data Transfer Commands

SI - Start Input

9 A	R	▨	181	TIMER ADDRESS
0 7	8 10	11 14	15 23	24 31

SI copies the contents of the Interval Timer's RTC into CPU general register R. The condition codes will be set as follows:

000 Device not implemented

010 Command rejected. This will occur only if the command was issued at exactly the same system-clock time when the RTC was being decremented - 1 chance in 40.

100 Command accepted

SO - Start Output

9 A	R	▨	82	TIMER ADDRESS
0 7	8 10	11 14	15 23	24 31

SO transfers the least significant 24 bits of CPU general register R into the Interval Timer's TOC. Bit 7 of register R designates the following:

0 The TOC is continually decremented independent of the interrupt enable/disable state.

1 The TOC is continually decremented only while its interrupt level is enabled. Thus, the TOC may be stopped/started by disabling/enabling its interrupt level. Any partial count of the 40 system clocks is lost when the TOC is stopped.

The condition codes will be set as follows:

000 Device not implemented

100 Command accepted

I/O Control Commands

PION - Processor Interrupts On

9 A	R		003	TIMER ADDRESS
0 7	8 10	11 14	15 23	24 31

PION enables the TOC interrupt when it counts down from one to zero.

PIOFF - Processor Interrupts Off

9 A	R		004	TIMER ADDRESS
0 7	8 10	11 14	15 23	24 31

PIOFF disables TOC interrupts.

IXIT - Interrupt Routine Exit

9 A	R		005	TIMER ADDRESS
0 7	8 10	11 14	15 23	24 31

IXIT signals the interrupt system that interrupt processing for the TOC has been completed.

INDEX

DOCUMENT REVIEW FORM

TENET 210 COMPUTER Reference Manual

Your comments concerning this document help us produce better documentation for you.

General Comments

	Yes	No
Is the material easy to read ?	□	□
well organized ?	□	□
accurate ?	□	□
complete ?	□	□
well illustrated ?	□	□
suitable for your needs ?	□	□

How do you use this document ?

□ As an introduction to the subject
□ For additional knowledge
□ For continual reference
□ Other

Specific Clarifications and/or Corrections

Reference	Page No.

This form should not be used as an order blank. Requests for copies of publications should be directed to the TENET sales office serving your locality.

FOLD

FIRST CLASS
PERMIT NO. 480
SUNNYVALE, CA.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A

POSTAGE WILL BE PAID BY

TENET
927 THOMPSON PLACE
SUNNYVALE, CA. 94086

ATTENTION: SOFTWARE PUBLICATIONS

FOLD

FROM: NAME________________________________

POSITION________________________________

ADDRESS________________________________

TENET 210 INSTRUCTION SET (Continued)

Code	Mnemonic	Name	Page
		BRANCH (extended Mnemonics)	
87-0	B	Branch	25
87-1	BNO	Branch if No Overflow	25
87-2	BNC	Branch if No Carry	25
87-3	BNCO	Branch if No Carry and No Overflow	25
87-4	NOP	No Operation	25
87-5	BOV	Branch if Overflow	25
87-6	BCRY	Branch if Carry	25
87-7	BCO	Branch if Carry or Overflow	25
85-1	BL	Branch if Less	27
85-2	BG	Branch if Greater	27
85-3	BNE	Branch if Not Equal	27
85-4	BMO	Branch if Any Matching Ones	27
	BBS	Branch if Bit Set	27
84-1	BGE	Branch if Greater or Equal	27
84-2	BLE	Branch if Less or Equal	27
84-3	BE	Branch if Equal	27
84-4	BNMO	Branch if No Matching Ones	27
	BBR	Branch if Bit Reset	27
81-1	BRL	Branch if Register Less Than Zero	27
81-2	BRG	Branch if Register Greater Than Zero	27
81-3	BRNE	Branch if Register Not Equal Zero	27
81-4	BROD	Branch if Register Odd	27
80-1	BRGE	Branch if Register Greater Than or Equal Zero	28
80-2	BRLE	Branch if Register Less Than or Equal Zero	28
80-3	BRE	Branch if Register Equal Zero	28
80-4	BREV	Branch if Register Even	28
		TEST REGISTER-DESIGNATED BIT	
03	TRB	Test Register-Designated Bit	30
07	TRBR	Test Register-Designated Bit and Reset	30
0B	TRBS	Test Register-Designated Bit and Set	30
0F	STRB	Store Register-Designated Bit	30
26	FRB	Find and Reset Bit	30
27	FRBD	Find and Reset Bit Double	30

Test Bit

COMMAND	BIT POSITION	I	X_D		REFERENCE ADDRESS
			P	X_I	
0 5	6 10	11	12 13	14 15	31

Code	Mnemonic	Name	Page
04[1]	TB	Test Bit	28
05[1]	TBR	Test Bit and Reset	28
06[1]	TBS	Test Bit and Set	29
07[1]	STB	Store Bit	29

[1] Value in bits 0-5

Immediate

COMMAND	R	SIGNED, TWO'S COMPLEMENT IMMEDIATE VALUE
0 7	8 10	11 31

Code	Mnemonic	Name	Page
73	LI	Load Immediate	21
70	AI	Add Immediate	21
71	SUBI	Subtract Immediate	21
7A	MI	Multiply Immediate	21
7B	DI	Divide Immediate	21
74	ANDI	AND Immediate	21
75	ORI	OR Immediate	21
76	EORI	Exclusive OR Immediate	21
77	CI	Compare Immediate	22
78	ISI	Inverse Subtract Immediate	22

FIELD POINTER FORMAT

M	FIELD LENGTH	FIELD POSITION	▨	REFERENCE ADDRESS
0	1 5	6 10	11 14	15 31

ALLOCATED MEMORY LOCATIONS

LOCATION Decimal	LOCATION Hexadecimal	FUNCTION
0-7	0-7	General Registers
8	8	Trap location defined by control register 5
9	9	Power failure trap location
10	A	Parity failure trap location
11-15	B-F	Unused trap locations
16-47	10-2F	Bootstrap load area
48-63	30-3F	Unimplemented instruction PINs (floating-point)
64-79	40-4F	Global PINs (op codes C0-CF)
80-127	50-7F	Local PINs (op codes D0-FF)
128-143	80-8F	Interrupt locations

UNIVERSAL TRAP

Trap to location 8 sets control register 5

Bit Position	Condition
0	Not used.
1	Unimplemented instruction.
2	Privileged instruction in slave mode.
3	Attempt to execute another EXEC instruction.
4	Unimplemented memory.
5	Write protect violation.
6	Unassigned virtual page.
7	Branch to locations 0-7 (register address).
8	Unassigned op code.
9	Unassigned IOC command field.
10	Floating-point overflow.
11	Floating-point divide by zero.
12	Floating-point underflow.
13	Stack overflow.
14	Set if conditions 4 and 6 are caused by an illegal instruction fetch.
15-31	Offending virtual address for conditions 4, 5, and 6.

Control Register	Use
0-3	Reserved by the system for use during instruction execution.
4	Reserved for trap information.
5	Defines conditions for trap to location 8.
6	Bits 15-31 = Current Stack Pointer (CSP).
7	Bits 15-31 = Stack Limit (SLM).

PROGRAM STATUS WORD

CM	CC1	CC2	CC3	OF	CF	IE	PO	SIR	FO	FD	FU	RS	▨	PC
0 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 31

Bit		Meaning
0-1	CM	CPU Mode: 00 Master/Unmapped 01 Master/Selective 10 Master/Mapped 11 Slave/Mapped
2-4	CC1-3	Condition Code
5	OF	Overflow Flag
6	CF	Carry Flag
7	IE	Interrupt Enable
8	PO	Page Protect Override
9	SIR	Special Instruction Recovery Flag
10	FO	Floating-point Overflow/Underflow
11	FD	Floating-point Divide by Zero
12	FU	Floating-point Unnormalized Value
13	RS	Registers Stacked Flag
14		Not Assigned
15-31	PC	Program Counter (17 Bit Address)

TENET, Inc. / 927 Thompson Place / Sunnyvale, California 94086 / (408) 245-8751