NOV 25 1969

TEMPO I SYSTEM REFERENCE MANUAL

TEMPO

TEMPO COMPUTERS, INC. 1550 South State College Boulevard, Anaheim, California 92806 Phone: (714) 633-3660

TEMPO I

SYSTEM REFERENCE MANUAL

Tempo Computers, Inc. 1550 S. State College Blvd., Anaheim, Calif. 92806
714/633-3660
TA-1001-1069

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF ILLUSTRATIONS

CHAPTER ONE
INTRODUCTION

1.1 TEMPO I COMPUTER

TEMPO I is a systems-oriented, general purpose digital computer.

Many of the special features that distinguish TEMPO I are standard, others are optional. Both are organized so that the basic computer can operate efficiently on present-day applications of limited complexity, yet the same computer can be readily expanded to handle the most sophisticated applications now being considered.

TEMPO I can, with minimum memory, peripherals and operating options, be put to immediate use in process control, data communication, scientific computation and advanced instrumentation systems.

Through options and modular additions, however, the same basic unit can also serve as the center of a complete multi-program, multi-processor, time-sharing computer system for both scientific and commercial computing tasks.

The dynamic range of this expansion capability is indicated by the fact that a basic TEMPO I can be installed for less than $16,000 or expanded through options, peripherals and multiple processors to a very large system with extensive processing capability.

1.2 TEMPO COMPUTERS, INC.

TEMPO I is a product of Tempo Computers, Inc., a California corporation organized in mid-1968.

Tempo Computers is short on history but long on experience. The engineers, programmers and mathematicians who form the nucleus of the new company represent nearly a century of experience in the design of third-generation computers. They have individually participated in the concept, design and implementation of more than a dozen of the medium-to-small computers now being produced.

TEMPO I is a collaborative effort designed to retain the best features of these earlier units and to add significant new capabilities in anticipation of the expanding role of the computer in our society and technology.

The company occupies a new, 18,000 square foot facility in Anaheim, California, close to its major sources of supply and in the center of a large reservoir of trained workers, familiar with the techniques of computer fabrication.

The company organization today includes electronic engineers, mathematicians, mechanical engineers, programmers, logicians, production specialists, and an experienced office staff. It has the financial strength to move rapidly on its planned course of new-product introduction, national marketing, and establishmen of an in-depth field service and application organization.

1.3 SUMMARY OF SPECIFICATIONS, TEMPO I

MEMORY

Type	Ferrite core, destructive read
Word Size	16-bit, 18-bit with parity option
Basic Memory	4,096 words
Maximum Memory	65,536 words, 4,096 increments
Access Time	325 nanoseconds

Memory Cycle Time	900 nanoseconds
Interface with Control Processor	Asynchronous
Memory Disable	Automatic
Number of Words Directly Addressable	65,536
Address	9/16-bits

CONTROL PROCESSOR UNIT (CPU)

Circuitry	TTL integrated circuits and MSI (medium scale integrated) circuits	
Hardware Registers	Basic	With Options
Total	14	25
Programmable	10	19
General-Purpose	7	15
Interrupts		
Internal	0	6
External	4	256
Priority Levels	4	22
Enable/Disable Masks	0	16
Instruction Length	16/32-bits	
Address Modes		
Direct	1 mode	
Indirect	1 mode	
Index	4 modes	
Index/Indirect	5 modes	
Immediate	2 modes	
Relative	1 mode	
Table	1 mode	
Arithmetic	Parallel, two's complement, 16-bit	
Computation Time, Fixed Point		
Load & Store	1.8 microseconds	
Add without Memory Reference	0.9 microsecond	
Add with Memory Reference	1.8 microseconds	
Multiply, Hardware	6.0 microseconds	
Divide, Hardware	6.5 microseconds	
Multiply, Subroutine	50 microseconds	
Divide, Subroutine	100 microseconds	
Double Precision Computation		
Type	Software	
Range	$+2^{30}-1$ to -2^{30}	
Approximate Time (Add)	10 microseconds	
Floating Point Computation		
Type	Software	
Approximate Time (Add)	45 microseconds	
Interrupt Response Time		
High Rate I/O	0.5/5.4 microseconds, typically <1.0 microsecond	
Standard I/O	2.3/12.6 microseconds	
Console	Standard and remote	

SOFTWARE

- ASA Standard Fortran IV (8, 192 word memory required for compilation)
- Macro Assembler* (including relocating-linking loader, one or two pass)
- Real Time Monitor
- Utility Routines
- Debug
- Source Program Update
- Math Subroutine Library
- Test and Maintenance

PERIPHERALS

All devices include controllers.

Teletype	ASR 33, 35
Paper Tape	Readers, Punches, Systems
Cards	Readers, Punches
Magnetic Tape	7 or 9 Track, Cassettes
Line Printers	Various speeds
Discs	Various capacity and speeds
Interval Timer/Real-Time Clock	
Multiplexers	
A to D Converters	
D to A Converters	

ENVIRONMENTALIZATION

Temperature	
Storage	-25°C to +75°C
Operating	0°C to +55°C
Shock	
Operating	5G for a duration of 300 milliseconds in all three axes
Vibration	
Operating Range	0-30 cps with a displacement of .030 inches
Humidity	
Operating	0-90% relative humidity without condensation
Cooling	Ambient air only required. Each chassis contains fans and filters. Maximum temperature rise inside the chassis is 10°C.

PHYSICAL SPECIFICATIONS

Width	17 inches
Depth	24 inches
Height	12-1/4 inches
Weight	150 pounds
Power Supply	Bulk supply with distributed regulators 115 or 230 VAC ±10% 47 to 420 Hz

* The capability of combining programs generated by compiler and assembler has been incorporated into the software design.

RELIABILITY

The reliability of equipment is the function of many factors such as stress levels of components, materials employed, construction techniques, etc. In addition, failure rate figures for components vary from manufacturer to manufacturer. In general the most accepted source of failure rate data is MIL-STD-217A. Based upon MIL-STD-217A and testing data on components not covered by this standard, the mean-time-between-failures (MTBF) figure is estimated to be greater than 3,000 hours. Using manufacturer's data this figure would be more than doubled. In fact, it is possible to arrive at a figure of about 10,000 hours using manufacturer data. Tempo feels that this would be misleading and therefore will use the lower MTBF figure.

The real criteria of usefulness is, however, the availability of the system. Availability is defined as:

$$A = \frac{MTBF}{MTBF+MTTR}$$

where MTBF = Mean-Time-Between-Failure
MTTR = Mean-Time-To-Repair

With the modular construction techniques employed in the TEMPO I and the diagnostics available, the MTTR is estimated to be less than 10 minutes. Therefore,

$$A \approx 0.999945$$

CHAPTER TWO
COMPUTING CAPABILITIES

TEMPO I is a high-speed, program-controlled, general purpose computer.

The speed and versatility of the computer are the result of design concepts that take maximum advantage of the capabilities of memory technology and state-of-the-art integrated circuitry.

Details of this design can be found in Section V of this manual.

2.1 MEMORY CHARACTERISTICS

The TEMPO I memory is a high-speed, asynchronous, ferrite-core unit designed and assembled by Tempo. The memory is designed to respond asynchronously to a control processor "data-request" signal with a "data-available" return signal. The memory is organized into 4,096-word (4K) units, each a self-contained structure with its own control and logic circuits. The normal configuration is based on a 16-bit word; the parity option requires 18 bits per address location (16 bits for data, plus 1 parity bit for each of two 8-bit bytes).

The minimum memory capacity for a TEMPO I is a single 4K unit. A fully expanded system has a 65K capacity, or a total of 16 memory units. Two of these fit into the basic computer chassis with the Control Processor, the balance into expansion-chassis modules. An expansion chassis module can contain up to 4 (16K words) memory modules.

All 65K of memory may be directly addressed by the basic instruction repertoire. No special hardware or software options are required to expand the computer from the 4K threshold to the full 65K capacity.

The instructions which operate on the memory include a full complement of addressing modes, including direct, indirect, index, immediate and relative. In addition, there is a table mode that provides for highly efficient operations on tabular data.

The address structure allows the programmer to treat the memory as a series of 512-word "pages". The "pages" are relative to the program counter and "float" with the program counter as it changes.

Memory access time is 400 nanoseconds. The memory cycle time is 900 nanoseconds.

2.2 PROCESSING SPEED

Since all devices interfacing with the control processor operate in an asynchronous manner, the internal clock rate of the control processor need not be inhibited by the speeds of these devices but can operate at its maximum reliable rate. The result, when coupled with memory, is an effective 900 nanosecond cycle time.

Since the other major elements of the computer, the memory and the input/output (I/O) structure, are independent of the CPU clock they too are free to operate at their own inherent speed and timing. (As an aid in I/O controller design a 4-MHz square wave is carried by one of the lines in the I/O bus for use by the controllers, if desired.)

The asynchronous timing that results from this design approach allows the computer to process data as fast as it is available. At the same time, reliable data transfers are assured by sophisticated status-signalling schemes, such as the echo-response "handshaking" that takes place between CPU and I/O controllers.

Another gain on processing efficiency and speed is the availability of general-purpose, 16-bit registers in the CPU (denoted as X-Registers) plus a full complement of register-to-register instructions for their use. In the basic machine there are eight registers X_0 through X_7. X_0 and X_1 are used as the normal A and B Registers.

The "X" Registers are truly general-purpose; they are not dedicated to any specific function, but can serve as accumulators, index registers, or as quick-access memory cells. In the High-Rate I/O mode, for example, two are used to monitor and control block data transfers at rates up to 700,000 words per second. Register-to-register transfers are accomplished in a single cycle. The X-Registers are under program control at all times.

These features of TEMPO I allow many efficiencies to be gained in processor operations. For example, table operations can be performed much faster than they would be in a conventional machine and I/O data can be compared "on-the-fly" with values held in the X-Registers. The required number of memory accesses is therefore minimized. The high-rate I/O feature, utilizing the X-Registers, functions effectively as a Direct-Memory-Access and relieves the controller of many operations.

The processing time for an addition is 1.8 microseconds (with memory reference) and hardware multiply and hardware divide are 5.8 and 6.3 microseconds respectively.

2.3 PERIPHERAL INTERFACING

A full set of peripherals can be included in a TEMPO I system, including teletype, paper-tape, magnetic tape, punched cards, disc, line printer, and data-communication equipment.

All of the peripherals, which can number up to 256 devices, are connected to the computer via a single 49-line I/O bus and a series of I/O controllers. Each controller serves as the interface between the I/O bus and the device it serves, translating data and interpreting instructions.

The basic concept of the TEMPO I interface is one of simple, reliable asynchronous operation. All data is transmitted and received with "strobe" pulses and a positive response is required from the peripheral to the CPU in a specified period to indicate that the peripheral has responded. The "hand-shaking" operation results in more reliable data communication and instantaneous detection of malfunctions in the peripheral device.

A full description of the I/O bus, controllers, and peripheral interfacing may be found in the TEMPO I Interface Reference Manual (TA-1000-469).

2.4 SYSTEMS EXPANDABILITY

The TEMPO I computer has been designed so that it can operate efficient on problems and applications of limited scope, yet can be readily expanded to handle the highly sophisticated applications now developing.

This expansion takes four forms.

> Additions to existing capabilities;
> new capabilities added as options;
> new capabilities added as software;
> interfacing with compatible Tempo system elements.

Additions to existing capabilities would be, to cite typical examples, an increase in core memory, the addition of extra general-purpose "X" Registers, an increase in the number of interrupts available and the duplication of peripherals to increase the overall I/O capacity.

New and different capabilities could include such options as hardware multiply and hardware divide (to replace their time consuming software equivalents), high-rate I/O circuitry to enable block transfers to occur at up to 700,000 words per minute, and new controllers to handle a variety of new different peripherals.

With an internally expanded computer, further increases in its capabilities can be obtained through software innovations such as a Foreground/Background Monitor program that allows the computer to have several programs occupying memory and running concurrently. Such a multiprogramming capability is of essential importance in applications such as time sharing.

For ultimate systems expansion, the TEMPO I can be teamed with a variety of Tempo developed system components. These include (see Figure 2.1):

Memory Control Unit ... to enable a TEMPO I CPU to communicate simultaneously with several memory units, or several CPU's to communicate with each other through memory and operate simultaneously within multiple memory banks. The result would be an advanced multiprocessor system.

> NOTE: Typical of the plan-ahead features of the TEMPO I design is the inclusion of the WAT (Wait) instruction in the present instruction list. This is a special stand-by command, not found in any comparable computer, that is an essential instruction for efficient multiprocessor operations.

Input/Output Processor ... is a CPU dedicated to handling large volumes of I/O data transfers, such as those encountered in real-time data processing, process control, and many commercial applications. Dedication of units is possible due to their low unit cost and results in simplified software requirements.

Auxiliary Arithmetic Unit ... provides supplemental computational capabilities for scientific applications.

Peripheral Cross Bar ... serves as a means for physically reassigning peripherals to various processors without requiring recabling or system reconfiguration.

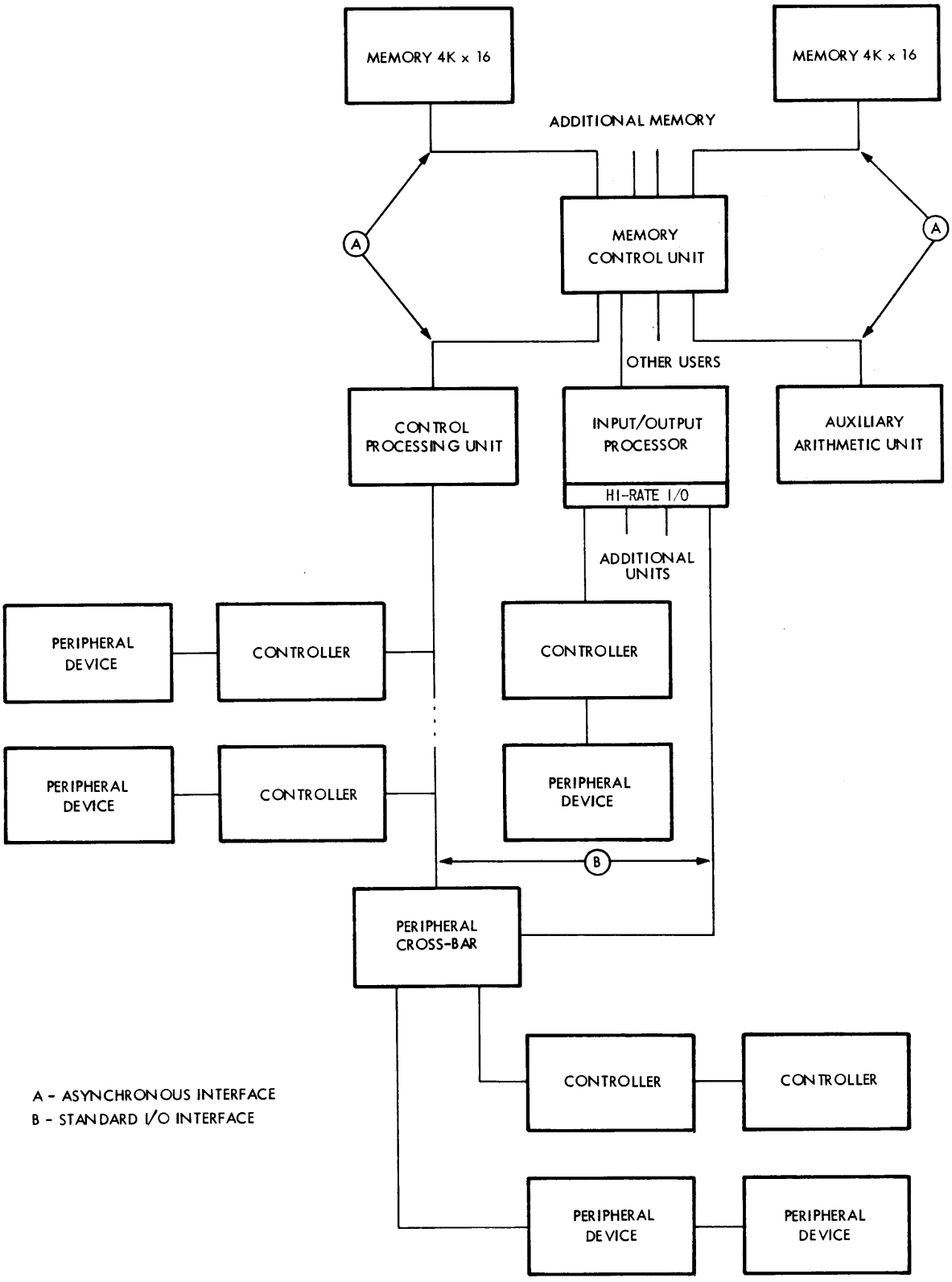


Figure 2-1. TEMPO I Block Diagram

2.5 SOFTWARE

A variety of standard and specialized software programs are available for use in programming the TEMPO I. These include:

TEMPO I Assembler ... an expandable symbolic assembler that translates source code programs written in the TEMPO I mnemonic form into binary object code capable of being loaded by the loader program. The basic assembler fits into 4K memory, and requires only a teletype peripheral. Features of the assembler include:

- One or two pass assembly
- Directives
- Nested macro capability
- Standard subroutine linkage
- User defined macros
- Relocatable object code
- Comprehensive error flags for ease of debugging
- Side-by-side source and object code listing.

TEMPO I FORTRAN IV Package ... consists of a compiler, loader, run-time executive and system library. The package operates in as little as 8192 words of memory. The compiler and loader automatically determine memory size and expand the applicable tables accordingly.

The Fortran IV Compiler language is a real-time version of that presented in the USA document, American Standard FORTRAN, X3.9-1966. This means that a reentrant capability is provided.

The compiler is a one pass compiler and will compile programs of up to approximately 300 statements and 200 identifiers. Compilation speed is approximately 2000 statements per minute exclusive of I/O time. The compiler has the capability of performing any number of consecutive compilations without having to reload the compiler. The capability of interfacing assembly language programs with FORTRAN IV programs is also provided.

The compiler and loader are written to operate with the ASR 33 Teletype. However there is modularity of input/output such that these programs can be easily adapted to new peripherals as they are added to the system.

The FORTRAN IV System Loader allows selective library loading. At the completion of program loading a comprehensive memory map of all programs loaded, where they were loaded and their size can be printed out.

TEMPO I Loader ... can load and link programs into main memory in absolute or relocatable format. It will link-load the main program, subroutines called by the main program, and subroutines called by other subroutines.

TEMPO I Program Debug ... is relocatable and provides the ability to:

- Obtain memory dump to printer or teletype
- Enter corrections into memory
- Initiate program execution at a given location
- Abort present action and return control to user
- Zero selected memory
- Perform masked memory search
- Insert program fixes
- Obtain snapshot of selected memory locations during program execution
- Trace selected segments of program execution.

TEMPO I Test and Maintenance ... is a set of programs designed to aid in the diagnosing of system faults and aide in the maintenance of the system. It includes:

- Memory test
- CPU test
- Peripheral tests.

Trap Package ... allows the programmer to write software for the machine without regard for the options which are incorporated in the machine. This is accomplished by the machine recognizing (in hardware) that an unimplemented command has been requested and causing the computer to jump to a subroutine (the Trap operation). The subroutine can simulate the function or cause other action the user may require.

TEMPO I Operating System ... provides the basic environment for the standard software modules. It also gives the computer operator a way to exercise system control and to implement commands directing the loading and execution of a program.

Math Subroutine Library ... is composed of an extensive list of math subroutines. Some of the routines included in the library are:

Add	Sine	Exponential
Subtract	Cosine	Log Base E
Multiply	Arsine	Log Base K of X
Divide	Arcosine	
Square Root	Arctan	

CHAPTER THREE
APPLICATIONS

The TEMPO I is designed to serve in a variety of applications, from the solution of scientific problems to the processing of commercial data.

The speed and efficiency of the TEMPO I I/O structure makes it adaptable to such applications as time-sharing and data communications. Its processing speed and X-Registers are of value in real-time process control applications. The extensive instruction set is an important asset in the solution of complex mathematical problems.

In addition, the TEMPO I has been designed for versatile and economical expansion. In the most advanced configuration the TEMPO I system becomes a network of memory units and CPU's that form a multiprocessor system equivalent to a number of computers operating in parallel.

Typical arrangements of computer and peripheral elements are outlined on the following pages.

3.1 BASIC COMPUTER SYSTEM

The basic computer system is diagrammed on the opposite page (Figure 3.1). It consists of the Control Processor Unit (CPU) with minimum options, a minimum 4,096-word memory, a teletype unit to act as a method of loading program and data and possibly one or more other peripheral devices.

Yet even at this economical level, the TEMPO I offers exceptional performance features. It can be readily programmed, using available Tempo software packages. It has a fast, efficient processing speed. It contains general-purpose registers that increase efficiency and versatility. More than 100 instructions are available to the programmer, including the valuable table-mode instructions for processing tabular data.

Most important, it has the built-in potential for field expansion into one of the more extensive configurations shown on the pages that follow.

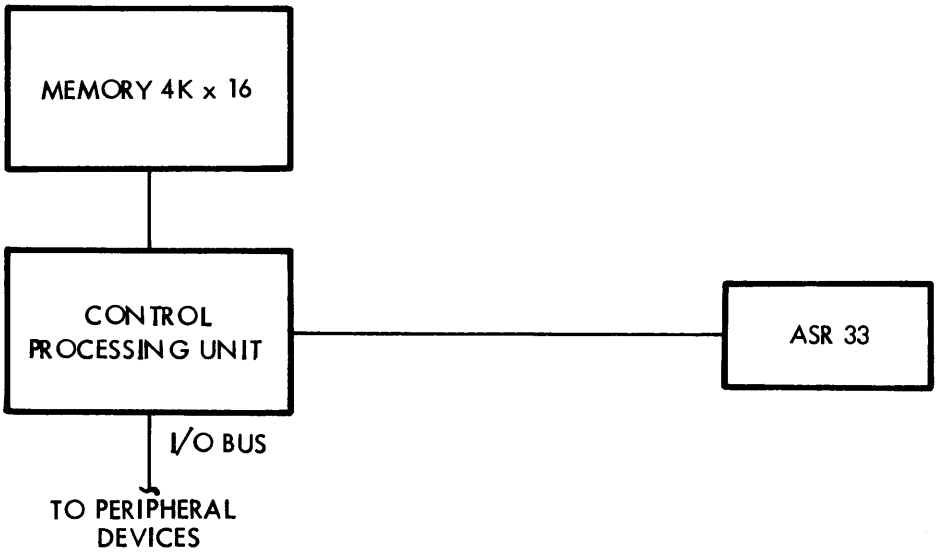


Figure 3-1. Basic TEMPO I Computer System

3.2 TIME SHARING

A basic TEMPO I can be expanded (see opposite page, Figure 3.2) into a time-sharing facility by the addition of bulk-storage memory devices, such as disk and magnetic-tape units, and provision for multiple teletype I/O channels.

To store and process the user programs, additional memory is added. Multiprogramming is accomplished by using a Foreground/Background Monitor with a time sharing language such as BASIC. This combines with a sophisticated interrupt structure to keep the computer operating at maximum efficiency with minimum delays for users requesting service.

An important option that would be added is the High Rate I/O, which allows bulk transfers of data at rates up to 700,000 words per second from the swapping store (the disk unit). With this feature the time lost in swapping users into and out of core is reduced. This, coupled with the high speed of internal processing, allows more users per system than any time share system of this class.

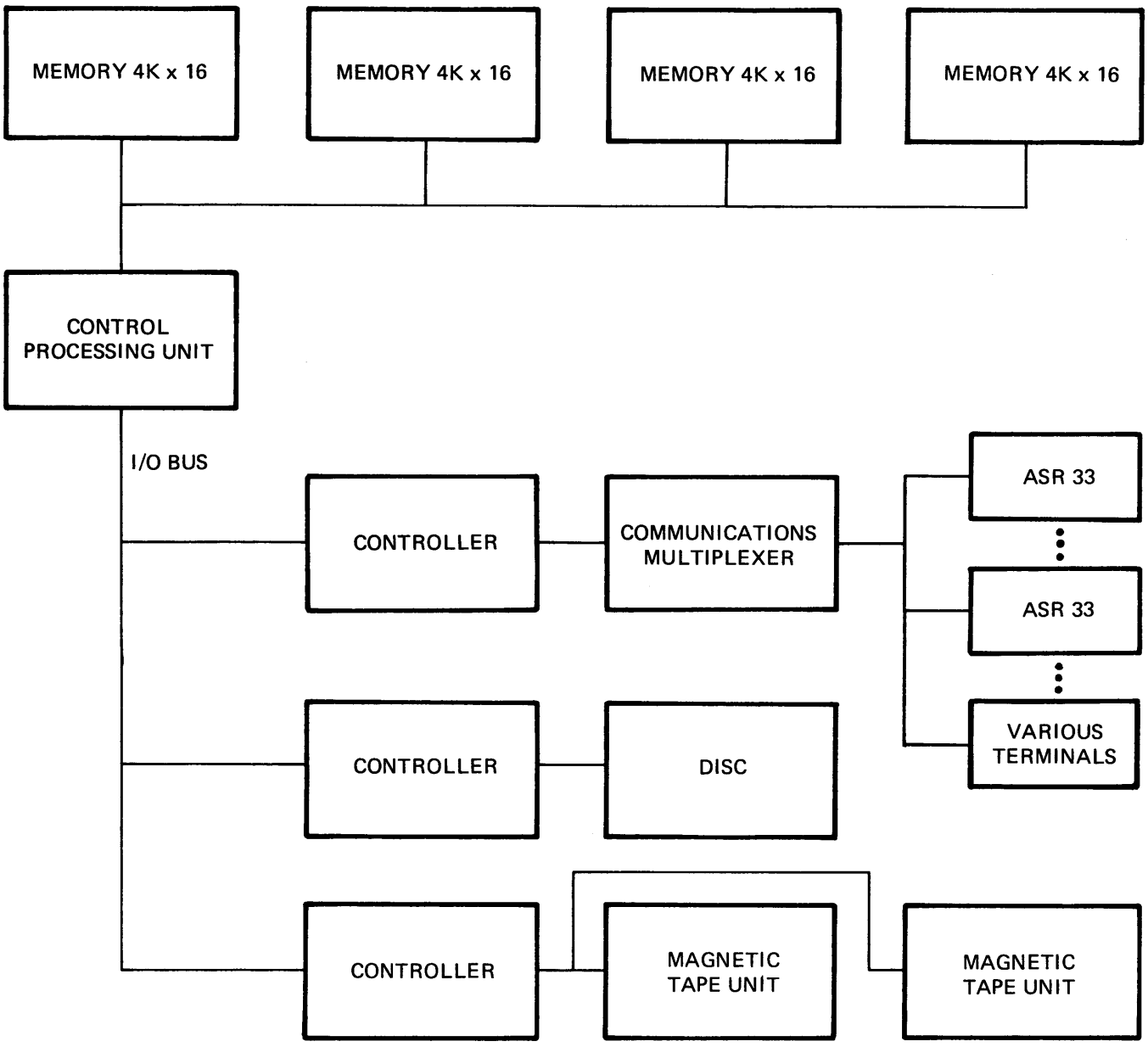


Figure 3-2 Timesharing System

3.3 DATA ACQUISITION

The data acquisition system shown in Figure 3.3 (opposite) consists of a standard CPU, a second CPU dedicated to I/O control, and a memory control unit that allows the two CPU's to communicate efficiently with each other and with the multiple memory banks.

The high speed of the TEMPO I design, combined with asynchronous I/O interfacing and priority interrupting makes the system ideal for real-time data processing, such as might be encountered in a chemical-plant process-control application.

The 16-bit format of the TEMPO I data word is a direct match for the four-digit BCD output that characterizes most analog-to-digital (A/D) converters used in data acquisition. The "handshaking" that takes place on the I/O bus assures the system that reliable data transfers are taking place.

A valuable feature for transferring processed data to bulk storage would be the High Rate I/O option.

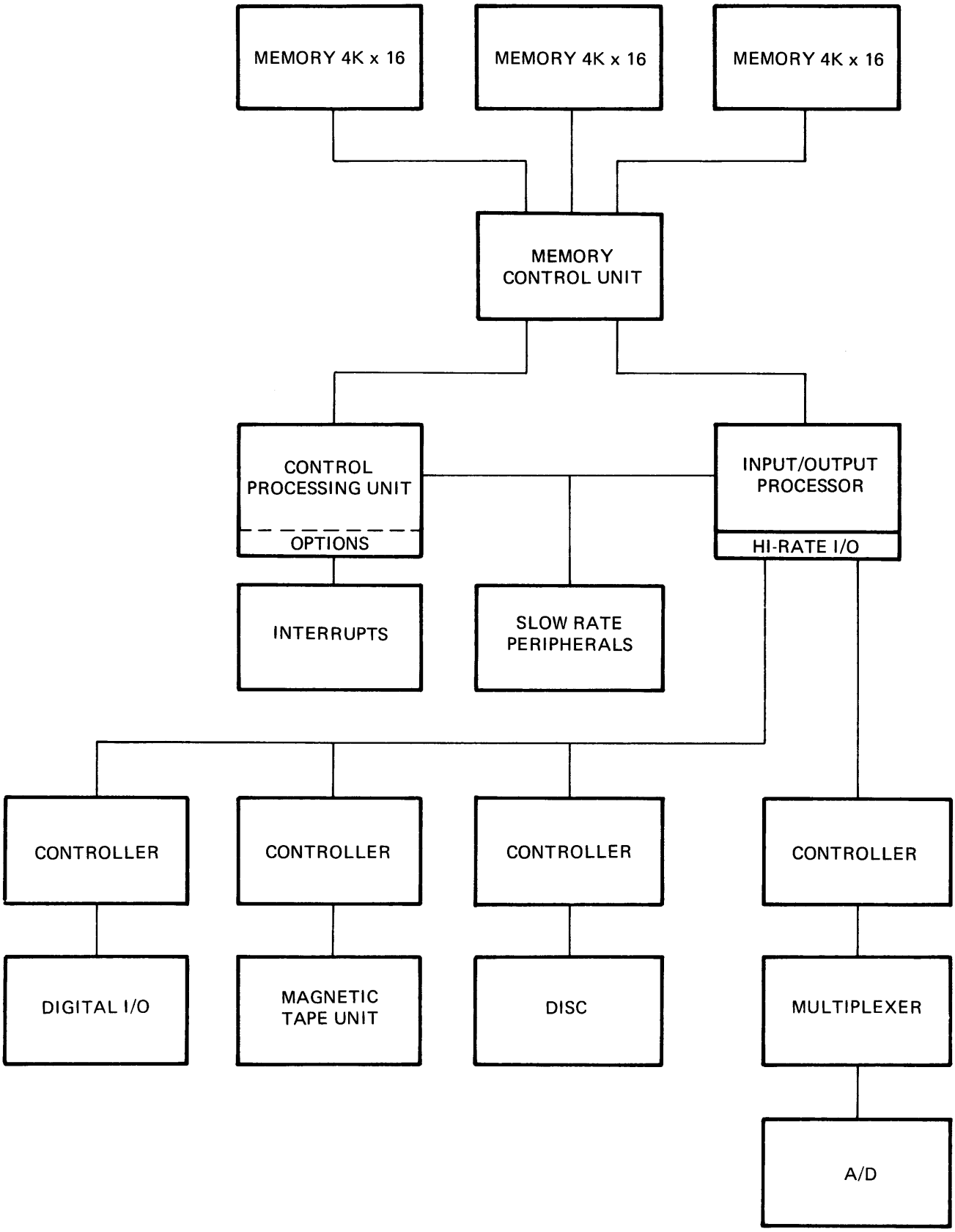


Figure 3-3. Data Acquisition System

3.4 SCIENTIFIC DATA PROCESSING

When internal processing rather than a high level of I/O activity is the principal requirement, additions take the form of multiple processors and memory units and an auxiliary arithmetic unit, as shown here (Figure 3.4, opposite).

Internal options would include additional general-purpose registers to serve as a high-speed scratch pad memory, hardware multiply and divide and a full set of high priority interrupts.

The basic design of the TEMPO I contains a number of features that would also be of value for computations of this type. The 900 nanosecond cycle time assures a high processing rate. The powerful instruction set, including the Table Mode for reference to tabular data, allows efficient programs to be written, and the capability for double-precision arithmetic (32 bits) means that nearly every normally encountered scientific problem can be accurately solved.

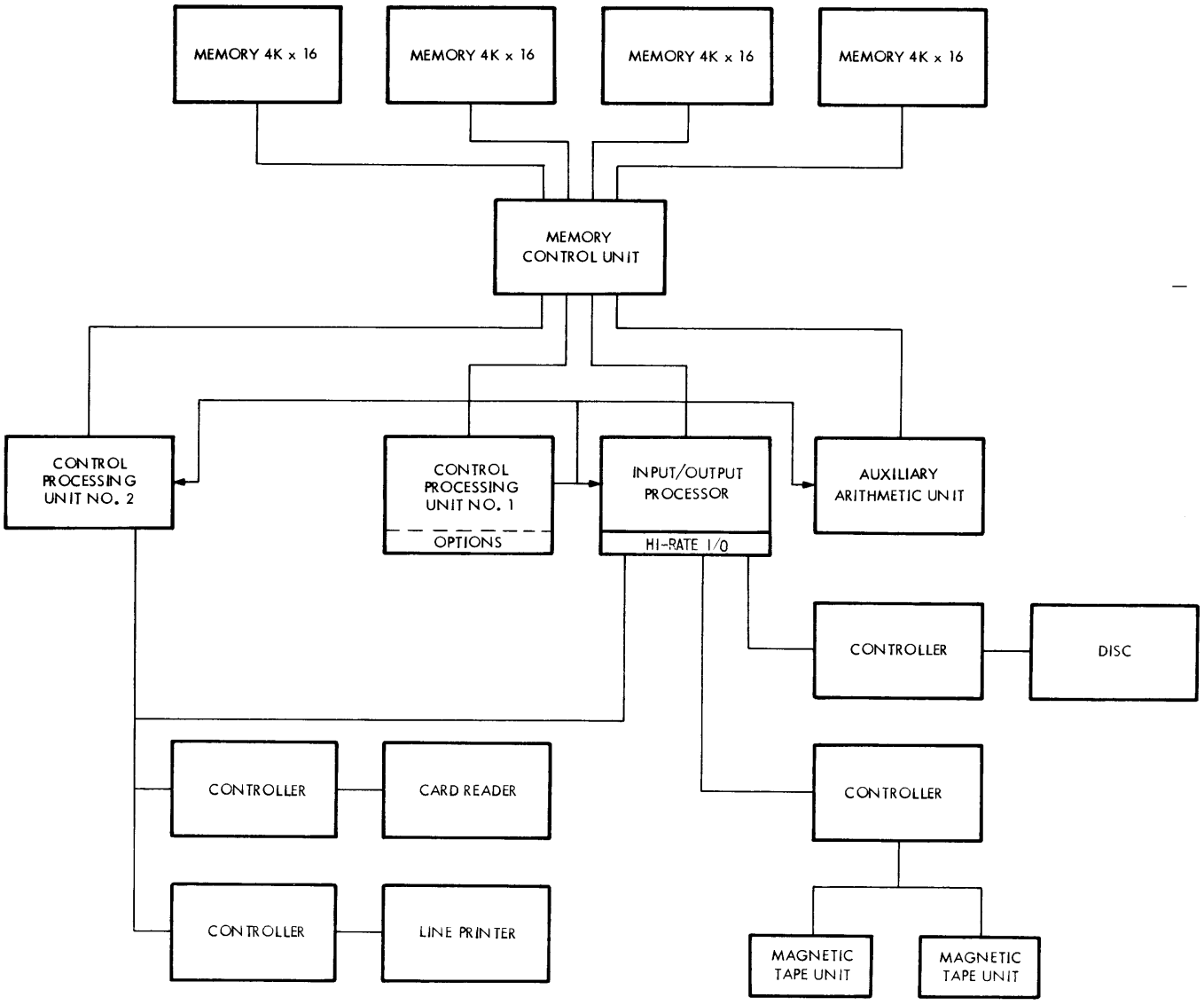


Figure 3-4. Scientific Data Processing System

3.5 GENERAL-PURPOSE COMPUTING AND TIMESHARING

The complex configuration on the opposite page (Figure 3.5) represents an ultimate expansion of a TEMPO I system, using all of the basic TEMPO I building blocks.

The memory could be expanded to 256K (only 16K is shown). Multiple CPU's would serve as multiprocessors. Additional CPU's would be dedicated to an extensive I/O structure designed for high-volume timesharing. At the same time an auxiliary arithmetic unit gives the system the computational power it needs to solve the most complicated user programs. A peripheral crossbar switching module would effectively remove any limit on the number of peripheral units that could be incorporated into the system.

Systems of this complexity and versatility, combining both high-rate I/O and computational capabilities, would be of special value in an academic application, for example, where a number of university laboratories would be timesharing a single computer center.

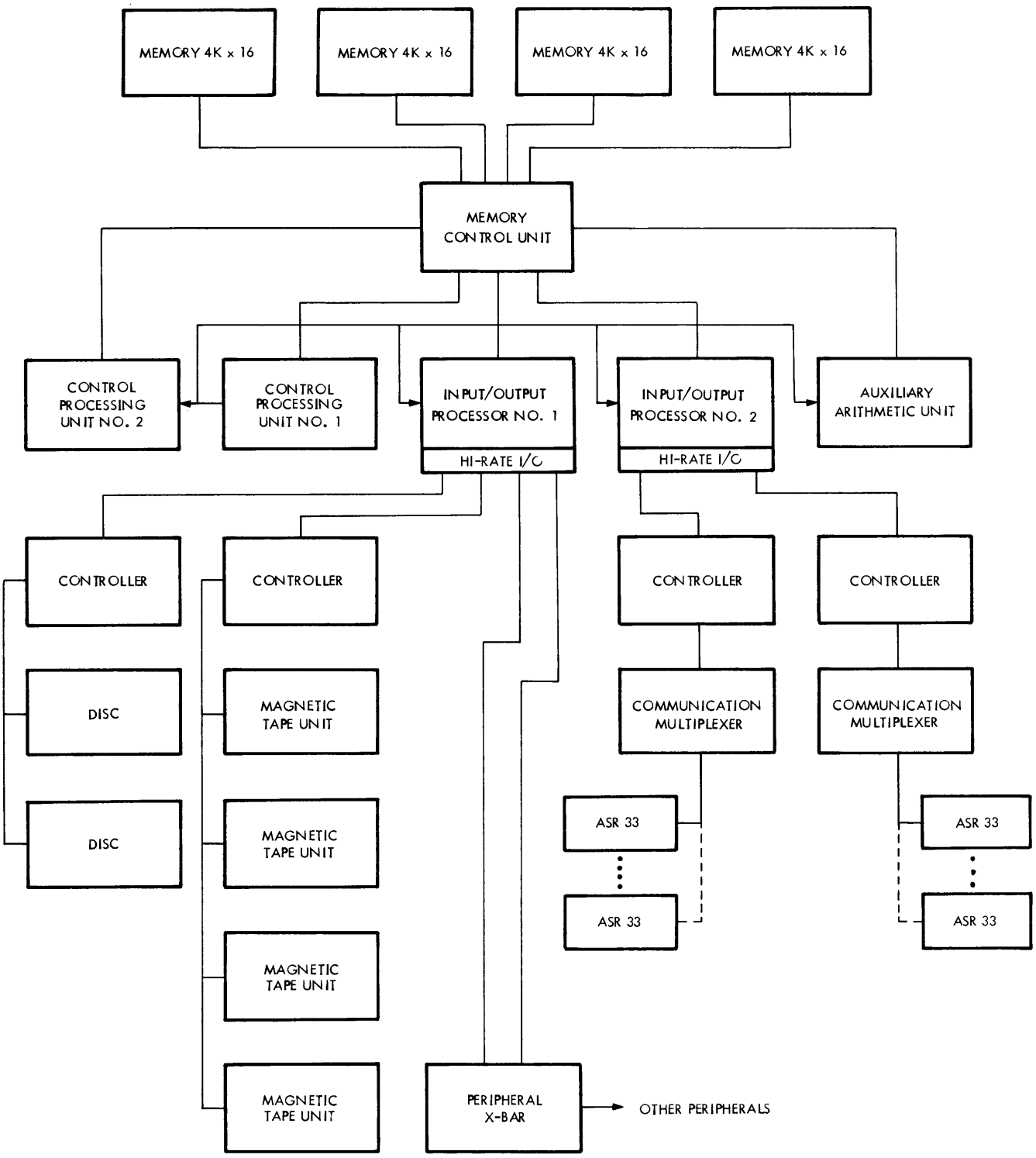


Figure 3-5. Large Scale TEMPO I System

3.6 DATA COMMUNICATION

The economy and versatility of the TEMPO I permit it to be used efficiently as a data-communication device, processing information for transfer to a larger computer, as shown in this example (Figure 3.6).

Two TEMPO I systems are included in the example. The first is a minimum system that accepts data from relatively slow peripheral devices, formats the information and packs it for efficient transfer across standard telephone lines.

The second TEMPO I acts as a multiplexer for a number of such data sources. It temporarily stores the data on a disc pack, then transmits the information at high speeds, and in the proper order and format, for processing by a major computer center. The system could, of course, be co-located with the larger computer.

By relieving the central computer of these data-communication chores, the TEMPO I units can increase the efficiency of the computer center by as much as 50%, rapidly paying for themselves in the savings realized.

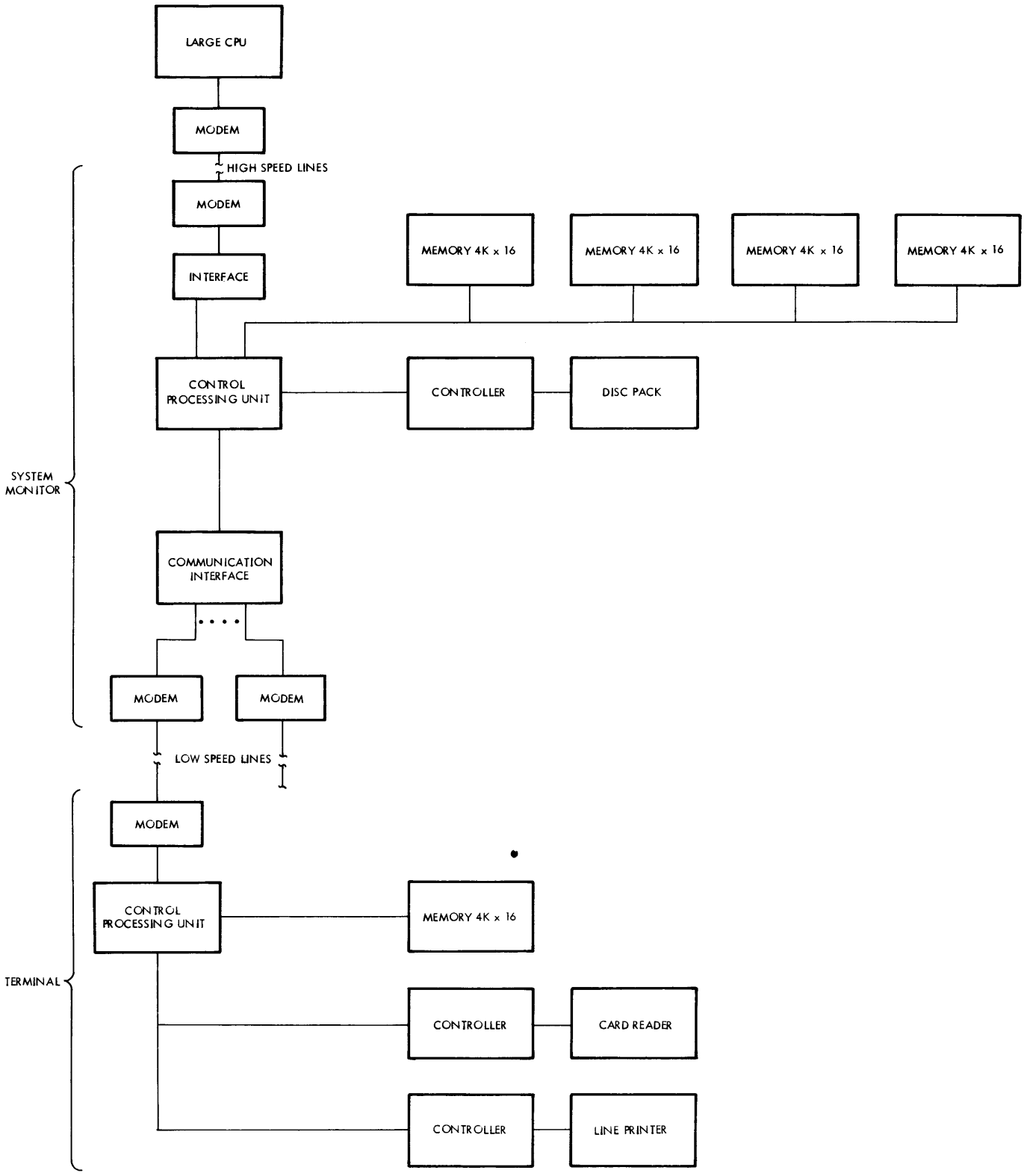


Figure 3-6. Data Communication System

CHAPTER FOUR
INSTALLATION, EXPANSION AND MAINTENANCE

The TEMPO I computer is designed for easy installation, reliable service and simplified maintenance.

The basic computer, including CPU, memory (up to 8K), and power supply, are all enclosed within a 12-1/4 inch high, rack-width chassis that is either contained in its own console stand or is rack mounted, using an adaptor kit supplied by Tempo.

Expanded systems use expansion chassis that are of the same overall dimensions as the basic chassis. Peripherals, such as teletype or magnetic tape transports, are free standing or rack mounted, depending on their design or on the installation requirements.

4.1 INSTALLATION

The installation procedure for a TEMPO I computer consists simply of receiving, unpackaging, inspection for shipping damage, connection to 120-volt power using a standard three-wire connector, and operation.

No special air conditioning or humidity controls are required. The unit operates within specifications at temperatures ranging from $0^{o}C$ to $+55^{o}C$ and at humidities up to 90%.

The user normally installs the computer himself on his own premises. Advice and consultation may be obtained from Tempo, if desired, at a nominal engineering charge.

4.2 EXPANSION

One of the most important built-in features of the TFMPO I computer is the ease with which it may be expanded in the field.

Connectors and mounting brackets are provided so that the user can expand the memory and the I/O structure (controller and peripherals) by simply plugging in assemblies shipped by Tempo. No special tools or checking out is required.

Other expansion options, such as Expanded Registers, High Priority Interrupts, and High Rate I/O, require modifications that can be accomplished in the field by Tempo field-service engineers. The downtime for these changes is normally only a few hours.

4.3 MAINTENANCE

Circuitry within the TEMPO I is designed for high reliability across a broad environmental range.

All components are derated to extend their statistical life expectancy. The variety of circuit components has been held to a minimum to increase the effectiveness of the Tempo quality assurance program and to simplify the user's spare parts inventory.

Most of the circuitry consists of integrated-circuit modules that are plugged into 14-lead and 16-lead sockets on a fiberglass-epoxy board. Power and ground are carried on the board; all other connections are wire wrapped. Rows of sockets may be individually decoupled.

The CPU and each 4K block of memory are self-contained on a single board. Arrangement has been made so that the CPU board may be removed from the rear of the chassis and reconnected at the front, alongside the console control panel. Diagnostic routines are then used to identify the source of any malfunction and repair generally takes the form of simple replacement of the faulty IC module. Since the module is plugged, not soldered, in place, no special tools or skills are required.

In applications where not even a brief downtime can be tolerated, a computer can be put back into service almost immediately by exchanging the complete CPU board or memory board.

CHAPTER FIVE
LOGIC AND HARDWARE DESCRIPTION

The TEMPO I computer can be functionally and physically divided into four basic elements for descriptive purposes (see Figure 5. 1).

- A core memory, which is organized into 4, 096-word memory banks.
- A control processor unit (CPU), which consists of the principal control and logic circuits, plus an expandable group of IC registers that process the data and serve as versatile memory elements for the data being processed.
- An input/output section, consisting of a general-purpose I/O bus and standard device controllers for the various peripheral devices included in the system.
- A power supply section, which converts ac power-line voltage to the dc voltage levels required by the computer circuits.

5.1 CORE MEMORY

The principal storage element for data and program instructions is a ferrite-core, destructive-read memory, organized into 4, 096 word (4K) memory banks.

Each memory bank contains its own data register and read/write control circuits. The TEMPO I configuration can therefore be expanded from a basic 4K memory up to a 65K memory (65, 536 words) without any changes elsewhere in the computer.

The memory word size is 16-bits (or 18-bits if the parity option is specified).

The memory consists of core matrix with its associated electronics. Within the Primary Memory the initial 192 locations represent fixed memory assignments that serve to simplify the programmer's task. These are:

Locations 0 to 9	System Programs Communication (required only when using systems/utility programs in the Standard Software Package).
Locations 10 to 15	Internal Interrupts
Locations 16 to 31	External Interrupts
Locations 32 to 127	Absolute Loader
Locations 128 to 191	System Call Pointers (required only if multi-program option installed).

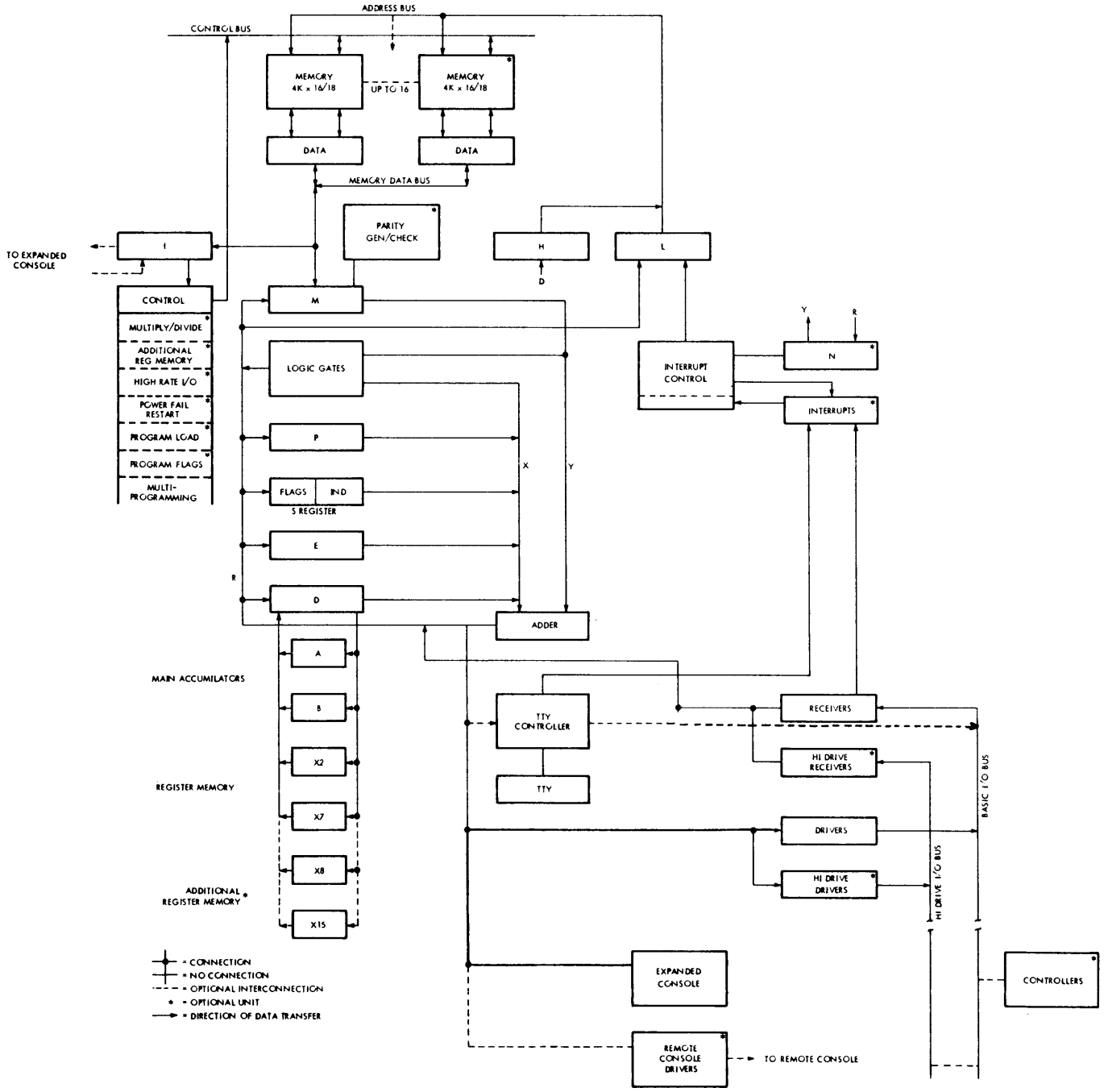


Figure 5-1. TEMPO I Control Processor Unit

5.2 CPU REGISTERS

The Control Processor Unit (CPU) incorporates most of the control and logic circuits contained in the computer, plus a group of 16-bit IC registers that hold and process the data in accordance with the program instructions contained in the core memory.

It is the versatility of the IC registers that gives the TEMPO I its unique processing capabilities. The various options that are available relate primarily to these registers and to their control circuits.

General Purpose Registers ... eight in number, expandable to 16, represent a significant innovation in computer design. One of these, designated as the A-Register, is held in reserve as the main accumulator register. A second unit, designated as the B-Register, serves either as an accumulator or as the basic index register. The remaining registers, designated as X-Registers, can serve as accumulator, index, or data-storage registers.

Since the A and B-Registers serve as the main accumulators, certain instructions, such as Multiply, Divide and Shifts, are effective only with these registers. During the Multiply instruction the B-Register contains the multiplier and A and B contain the product at the end of the operation. During the Divide instruction the A and B-Registers contain the dividend and upon completion, A and B contain the remainder and quotient respectively.

The remaining registers, X_2 to X_7 (or with the expanded-register option, X_2 to X_{15}), plus the B-Register itself, can be used as accumulators, index registers, or temporary memories. A full set of register-to-register instructions can be used by the programmer to process data without fetching individual data words from memory or storing intermediate results in memory.

X-Registers are used in pairs (under program control) to achieve the 700,000 words per second rate that characterizes the High Rate I/O option. One X-Register is used to count down the number of words that have been transferred while the other X-Register is incremented to indicate the address of the next word to be transferred.

P-Register ... is a programmable register that serves as the Program Counter. It contains the address of the next instruction word. Hardware control normally causes automatic increment of the P-Register, or the programmer may control it through the use of the Branch instructions.

S-Register ... is a programmable register that serves as the Status Register. It contains various flags set as the result of the execution of instructions. Standard flags are: Less Than; Greater Than; Equal; and Overflow. Optional flags include: Program Flags 1, 2, 4, and 8 and seven High Rate I/O Enable flags.

N-Register ... is a programmable register that serves as an interrupt mask, selectively enabling and disabling the external interrupts. It is used with both regular and High Rate I/O data transfers.

I-Register ... serves as the Instruction Register, receiving instruction words from memory during the instruction fetch cycle and holding them for interpretation by the control section during the execute cycle.

M-Register ... contains memory address and operand data and is used for information transfer to and from the memory. In addition, it is used to store intermediate results during the execution of instructions.

L-Register ... holds the effective address as calculated by the address modification sequence of the current instruction. All memory addresses pass through the L-Register since it drives the memory address bus.

D-Register ... serves as a buffer between the general-purpose accumulators (A, B, X_2 to X_7) and the rest of the CPU. It serves as a working register on which all arithmetic and shift operations are performed, as a communication link between the general-purpose accumulators and the memory, and as temporary storage for the general-purpose accumulators in register-to-register operations.

E-Register ... is coupled to the D-Register for performing double length operations such as double-length shifts and for containing the double length dividend in a divide operation and the double length product in a multiply operation.

H-Register ... is an optional register added as part of the High Rate I/O option. The H-Register receives memory-address data and directly drives the memory address bus, bypassing the L-Register.

5.3 CPU LOGIC AND CONTROL CIRCUITS

The balance of the Control Processor Unit (CPU) consists of a network of state counters, decode logic, flip-flops, gating, CPU clock, and 16-bit interconnecting busses.

The logic and control circuits sequence the operation of the CPU in accordance with the stored program, the console control panel, and interrupts occasioned by the exchange of data with peripheral devices.

Many of the most important options available with the computer are contained in this section. The following paragraphs describe both the standard and the optional features of the CPU logic and control circuitry.

Internal busses ... are formed by the "ORing" of the output of the various registers and other circuit elements that they interconnect. The three main 16-bit busses are X and Y, which provide inputs to the adder and logic gating, and R, which distributes the adder output.

Adder ... consists of a set of gates that provide for the parallel addition, subtraction, incrementing and comparison of 16-bit operands.

Logic gating ... consists of a set of gates that perform logic functions AND, OR and Exclusive OR. The gates form, in parallel to the adder, an alternative path between the X and Y busses and the R bus.

Interrupt control ... includes the control logic and flip-flops that establish interrupt recognition, priority selection, enable/disable (if the interrupt-mask option has been specified), and the generation of the memory address where the appropriate interrupt subroutine is located.

The interrupt source can be either internal or external. There are six internal interrupts, all associated with CPU options which must be separately specified:

Power Fail Restart
Parity Error
Instruction Trap
Memory Protect Error
Privileged Instruction Error
System

External interrupts fall into two groups. First is a group of four (standard in the basic machine), of which three are general-purpose and one is reserved for the teletypewriter. To these can be added 12 additional general-purpose interrupts, in optional groups of four. External interrupts associated with the High Rate I/O option are normally the highest priority interrupts in this group.

The "levels" of interrupt priority are generally established by the classes: a) internal, b) system, c) high-rate, and d) external. The priority logic of the system selects the appropriate interrupt for servicing when two or more interrupts are pending. The logic also permits higher level interrupts to interrupt lower level interrupts in process of being serviced without loss of the lower priority interrupt status.

Peripheral-device controllers each contain a plugboard that establishes the priority level and address for that device. The plugboards allow the priority levels and addresses to be reassigned at any time without rewiring.

When two or more devices share a single interrupt line, they can be considered as having a level of priority within a level. Their priority is determined first by the relative position of the shared interrupt and second by the priority established by the subroutine which services the shared interrupt.

The rate at which the external and internal interrupts may be processed depends on the length of the subroutine required to process each interrupt. Since High- Rate I/O interrupts are processed by a hardware generated sequence, they are not program dependent for rate. As a result, High-Rate I/O interrupts can be serviced at rates up to 700K words/second.

Both the regular and the High Rate I/O interrupts are asynchronous. They may be set by their respective controllers or control logic without regard to the CPU clock or the state of any other controller. Generally each controller has a flag which is made true when the controller requires service, as timed by the separate controller clock. The flag is reset when the CPU has serviced the I/O request.

The interrupt response time of a computer is a useful criteria for measuring system performance. The TEMPO I computer has a very fast interrupt response capability. In the high-rate mode the longest time for response to a requested interrupt occurs when the processor is executing a hardware divide operation and is equal to 5.4 microseconds. In all other cases if the interrupt request occurs a minimum of 250 nanoseconds before the end of an instruction cycle, the interrupt is accepted. Thus the minimum response time can approach 250 nanoseconds. Since interrupts are randomly occurring events with respect to the machine cycle (and the hardware multiply and divide occur infrequently) the typical response time for high-rate interrupts is approximately 1.0 microsecond.

For normal I/O operations, the response time is defined as the time which elapses from the occurrence of the request to the execution of the first useful I/O instruction (e.g., a Word Transfer Out). This is between 2.3 and 12.6 microseconds. Typically it is approximately 5.0 microseconds.

Interrupt masking ... is accomplished by the optional N-Register. These are optional, in groups of four, up to a maximum of 16 which correspond to the 16 external interrupts. Each interrupt may be selectively masked and unmasked under program control.

The masking operation does not reset the flag in the controller; it merely prevents the priority logic from considering the disabled interrupt at priority selection time. Disabled interrupts are "remembered" and serviced when the mask is removed.

Power Fail Restart option ... provides a continuous monitoring of the ac-power input. If the input voltage falls below a certain level, the main program is interrupted. A subroutine transfers the program count or contents, volatile flags and register contents to main memory. After approximately 200 μsec the memory is disabled and the dc power is allowed to decay. Upon restoration of ac power to an acceptable level for proper system operation, a restore sequence is initiated and the system is restarted and returns to the program state which it was in when power was interrupted.

Memory Parity option ... provides a check of the memory operation through the use of two redundant bits (one for each byte) in each word of memory. The control hardware also provides an interrupt to a fixed location to start an error subroutine if a parity error is detected.

A flip-flop is provided to disable the interrupt. Once the flip-flop is set, no parity error interrupts will take place until the flip-flop is reset.

The redundant bit (parity bit) of a correctly written byte in a memory word is set to a "1" if the number of "1's" in the other 8 bits are even. Correct parity is thus said to be "odd."

Multiply/Divide option ... consists of the necessary control logic to allow the MUL and DIV instructions to be used. MUL is a 16 x 16 bit binary, two's complement format multiply instruction which results in a 31-bit product. DIV is a 31-bit dividend/16-bit divisor instruction which results in a 16-bit quotient with a 16-bit remainder, also in two's complement format. No additional registers are required.

Program Flags option ... consists of four flip-flops and the associated control logic to set, reset, store and branch conditional on their state.

The flags, contained in the low order bits (12 through 15) of the S-Register, are for general-purpose use by the programmer in remembering various user specified program state changes.

Program Load option ... allows the user to cause a selected device to input a single record of variable length into a selected block of memory and cause the program thus loaded to be executed.

It consists of the necessary logic to execute the sequence using one of two versions of switches and indicators depending on whether the basic or the expanded console is installed. The High Rate I/O option is a prerequisite for the Program Load option since a block transfer is involved.

Instruction Trap option ... provides for the detection of instructions which cannot be executed because they are not implemented in the form of hardware options. The detection function, called a "trap," results in an interrupt to a subroutine which interprets and executes, in software, the attempted instruction. A standard set of interpretive subroutines is provided, but there is no restriction on how the instructions can be interpreted if the user wants to write his own subroutine.

Multiprogram option ... allows the CPU to operate a system monitor and one or more user programs in a time sharing mode. The CPU switches back and forth between a Master mode (when executing the monitor) and a Slave mode (when executing the user programs).

The Expanded-Register option is a prerequisite if the TEMPO I Monitor is used. Registers X_2 through X_{11} are for use by the monitor, primarily for I/O. User programs may use X_{12} through X_{15}.

The Multiprogram option includes the following:

a) Master Mode Flip-Flop - set so that only those functions which may be performed in Master mode are allowed.

b) Privileged Instructions - may be performed only when the machine is in Master mode. Attempts by user programs to perform these instructions will result in an internal interrupt.

c) Internal Interrupts - occur when operations are attempted by user programs which require the attention of the monitor.

d) Memory Protect - provided by the Memory Control Unit (MCU); is not included in the Multiprogram option but is a prerequisite for it.

e) System Instructions - provide for efficient communication between the monitor and user programs.

5.4 INPUT/OUTPUT

All communications between the Control Processor Unit (CPU) and the peripheral devices that complete the computer system are directed across a single 49-line I/O bus.

Between the I/O bus and the peripheral devices are a series of Controllers which decode the data and commands being transmitted on the bus and interpret them for operation by the devices which they service. Each Controller is specific for the device that it serves, but all have a common interface to the I/O bus and therefore to the computer itself.

Controllers are available for all standard peripheral devices (see Section 5.5 below). The simplicity of the I/O bus interface makes it possible to design, at reasonable cost, special controllers for other peripheral devices, as required.

Standard I/O bus ... consists of 49 lines compatible with DTL integrated circuits. These lines are grouped as follows:

Number of Lines	Type	Function
6	Address Lines	Transmit a binary code, from 0 to 63, for selection of device Controller; only the addressed Controller responds to signals on the other lines.
3	Op Code (o-field)Lines	Describes the order being sent.
3	Class Code (k-field) Lines	Describes the class of instruction to be performed.
1	Strobe Line	Signals to Controllers to examine the address, k-field, and data lines at a time when their condition has settled to a valid state.
1	Echo Line	Used by individual Controller to respond to valid address, instruction and strobe; also used to strobe input data into CPU.
1	Program Load Line	Used with the Program Load option.
1	System Reset Line	Returns all peripheral devices to an idle state, ready to accept a new instruction.
1	Peripheral Clock Line	Carries a free-running 4 MHz square wave for use by Controller logic, if desired; it is not synchronized with the CPU clock.
16	Data Lines	Transmit data words, in form of 16 parallel bits, between CPU registers and Controller that has been addressed.
16	Interrupt Lines	Transmit an interrupt signal from Controller needing servicing to CPU; up to 16 Controllers can be ORed to single line; basic system is equipped for four external interrupts.

All of these lines, except non-appropriate interrupt lines, are connected to all of the Controllers in parallel. However, only one Controller is addressed and activated at a time. When the CPU is ready to transmit data or an operating instruction to a Controller, it places that Controller's code on the address lines, the data is then placed on the line, and a strobe signal is transmitted; all other Controllers remain idle.

When a Controller needs servicing by the CPU, as when data is ready for input to the computer, an interrupt signal is transmitted to the CPU. This initiates an interrupt subroutine that again has the effect of placing that Controller's code on the address lines.

Asynchronous timing allows the I/O data flow to proceed as efficiently as possible. Thus, after a particular peripheral (e.g., a magnetic tape transport) has been started, no further attention is required of the CPU until the device has come up to speed and either has data ready for input or is ready to accept information. Upon reaching this ready state the associated Conroller activates an interrupt sequence without regard to the state of either the CPU or of the other Controllers. The priority interrupt logic of the CPU chooses the appropriate time to interrupt the CPU processing (generally at the end of the current instruction in process) and chooses the highest priority interrupt for service.

High-Drive I/O Bus option ... consists of driver and receiver circuits which allow an I/O bus to extend to distances of up to 30 feet with up to 16 Controllers attached. The High-Drive I/O Bus is functionally identical to the Standard I/O Bus.

Normal I/O data transfers and commands ... are between the A-Register and the Controller being addressed.

Three basic classes of I/O instructions are provided: a) Execute Device Function (examples: start reader, set read mode, stop write); b) Word Transfer (transfer one word into A-Register from peripheral, transfer one word from A-Register to peripheral); c) Status Examination (read a word of status information from peripheral to A-Register, interrogate interrupt status of common interrupt line).

High-Rate I/O data transfers ... require the High-Rate I/O option, and the installation of at least the four highest priority external interrupts.

This combination of options allows block transfers to occur at rates up to 700,000 16-bit words per second. More than one Controller can be operating in high-rate mode simultaneously. The transfers are direct from I/O bus to memory on a cycle stealing basis; consequently the contents of the programmable registers and CPU status flags are not affected.

Two X-Registers are used, under program control, to service each Controller provided with a High-Rate I/O interrupt capability. One X-Register contains the address for each word being transferred; the address is automatically incremented and transmitted to the memory address bus via the H-Register. The other X-Register is used to count down the number of word transfers and to terminate the High-Rate sequence when the entire data block has been processed.

The six X-Registers in a basic computer can accommodate up to three Controllers in the High-Rate mode. The Expanded Register option, which adds eight X-Registers, allows a total of seven Controllers to be serviced.

5.5 PERIPHERALS AND CONTROLLERS

Peripherals and associated Controllers are all designed to interface with the Standard or High-Drive I/O busses.

Standard peripheral options include:

Teletype	ASR 33 or ASR 35
Paper Tape	Readers, Punches
Punched Cards	Readers, Punches
Magnetic Tape	7 track, 9 track, Cassettes
Line Printers	
Discs	
Displays & Plotters	
Dataphone Couplers	

The basic peripheral for most systems is a Teletype Corporation Model ASR 33 TC. The option provides for paper tape input, paper tape output and typewriter output at 10 characters per second. A keyboard allows typewriter input at up to 10 characters per second.

The teletype is not part of the Console (see next section). But since it is part of the normal man/machine interface, along with the Console, they are treated as a unit in the programming and operating descriptions that follow:

5.6 CONSOLE

The console has all the features necessary to operate the system through the console alone. It consists of the control panel shown in Figure 5-2. Some of the indicators and switches may not be installed or may be inoperative if the associated option is not installed.

Indicators and switches on the panel have the following functions:

Data Display: Eighteen lights (16 data plus 2 parity) display the 16 bits of data selected by the Register Select Switch. The display of the parity bits (P_0, P_1) occurs when the Register Select Switch is in the M position. The display is effective regardless of the operation being performed.

Run, Halt, Wait: Three lights indicate the control state of the sequencing logic of the CPU.

In the Run state, instructions are fetched from memory and executed. The Run state may be entered from either the Halt or Wait states.

In the Wait state, no activity takes place, but interrupts are acknowledged and processed.

In the Halt state, no activity takes place and interrupts are not acknowledged. Data can be reliably entered through the console only when the CPU is in the Halt state.

Error Indicators (Parity, Instruction, Load): Three lights indicate that an error has occurred relative to the operation indicated by the individual light. (The instruction indicator applies to the operation of the Trap option.)

Data Insert and Clear: Seventeen pushbutton switches (16 plus clear) allow data to be inserted into the register selected by the Register Select switch. The insert pushbuttons cause "ones" to be set into the selected register. The Clear pushbutton is provided to reset all positions of the selected register to "zeros". The switches are effective only if the CPU is in the Halt state.

Register Select: Six toggle switches select certain of the programmable and non-programmable registers in the CPU for both display and data insert. If more than one switch is in select position, only the left-most switch is effective.

RM-Select: Four toggle switches select the Register Memory registers. Selection is obtained by setting a binary code into the RM-Select switches. Switches in the UP position signify a "one". The A-Register is selected by the binary code 0000; B-Register, by 0001; and the X-Registers, in ascending order, by 0010 to 1111.

CPU Reset and System Reset: Two pushbutton switches reset various flip-flops (indicators, flags and state counters) in the CPU and overall system.

Power: A two-position key switch that may be operated only when the proper key is inserted. In the UNLOCK position, the ac power is under the control of an on-off toggle switch (dc power is automatically applied subsequent to the application of ac power).

The Lock position leaves power on and disables the functions associated with all pushbutton switches.

Sense: Four toggle switches that are program sensed by the BST, BSF (Branch on Sense Switch True, False) commands. The Lock position of the Power switch disables these switches. Switches in the UP position signify a "one."

Program Load: A pushbutton that initiates the Program Load operation. The pushbutton is disabled when the Power switch is in the Lock position.

Run: A pushbutton that causes the computer sequence to leave the Halt state and commence executing the computer program.

Halt: A pushbutton that causes the CPU control to enter the Halt state. The switch is disabled when the Power switch is in the Lock position. When operable, additional depressions give a single-step operation.

Initiate: A pushbutton switch that initiates the operation selected by the Mode switch. It is disabled when the key switch is in the Lock position.

5.7 POWER SUPPLY

The Power Supply is a modular system component designed to supply either the CPU and memory or an assembly of I/O controllers and/or memory modules.

The Power Supply consists of rear panel, power distribution panel, 8V (unregulated), +5V, $\pm$15V and the inhibit regulator for the memories. The +8V supply is distributed to regulators located strategically in the processor and memories.

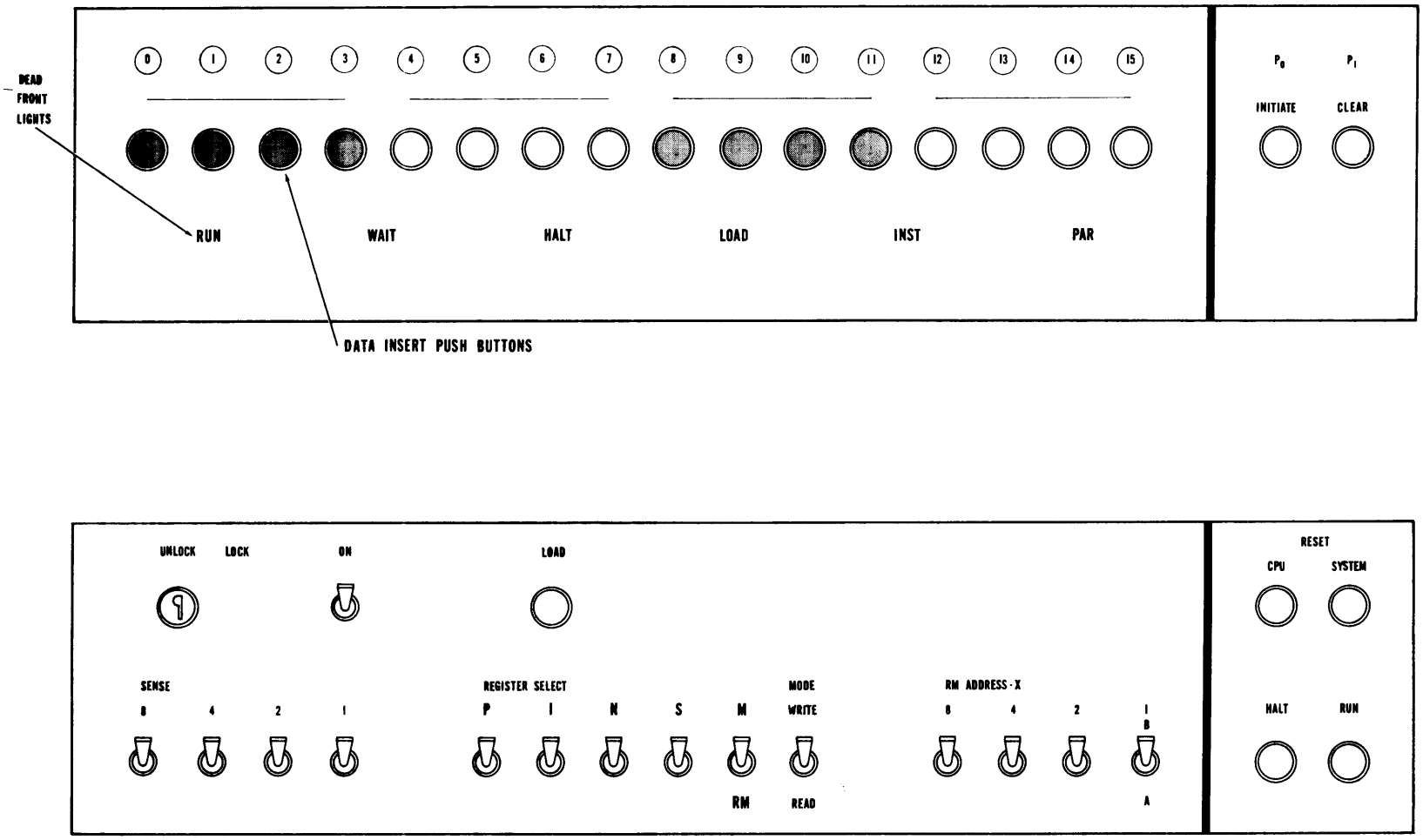


Figure 5-2. TEMPO I Console

CHAPTER SIX
INSTRUCTIONS

6.1 INTRODUCTION

The program processed by the computer consists of a sequence of individual instructions that specify transferring data to and from memory, to and from peripheral devices, and manipulation of data (e.g., add, subtract, compare) within the CPU.

Each program instruction consists of one or two 16-bit binary words, stored in memory and transferred in and out of memory exactly like data words.

The contents of the P-Register specify the address of the next instruction to be performed. On the completion of the current instruction, the contents of the memory location specified by the P-Register are transferred to the I-Register. Hardwire logic circuits decode the contents of the I-Register and initiate the instruction that is to be executed.

It is the number and variety of hardwired interpretive circuits associated with the I-Register that determine the number of instructions available to the programmer. The more there are, and the more powerful they are (i.e., one instruction to take the place of several in sequence), the simpler the programmer's task, the faster the computer's processing speed, the more efficient the system.

The basic TEMPO I computer is prewired to execute over a hundred different instructions. Many of these have multiple interpretations (especially those relating to the X-Registers). The result is an effective instruction repertoire in the hundreds.

6.2 INSTRUCTION FORMAT

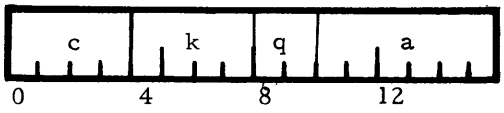
There are five basic classes of instructions, each with its own distinctive format or arrangement of bits and bit "fields". The latter are groups of bits within the instruction word that are used to code a specific element of the instruction, such as an address, device number, or arithmetic operation.

All instructions contain a 4-bit field known as the "c" field or Op Code (Operation Code) field. The Op Code defines the basic purpose of the instruction.

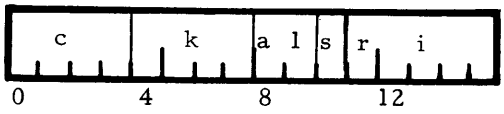
There are a total of 16 major Op Codes and these in turn establish the five basic classes of instructions:

Class I - One Word, Non-Memory Reference ... is identified by one major Op Code and contains three subclasses for control, shifts, and register-to-register operations.

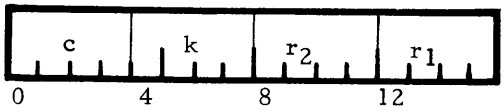
Control Operations



Shift Operations



Register Operations



The k-field is the same in all three subclass formats and is decoded to define the Op Codes for control, shift and register operations.

In the control format the q-field is decoded to define the four control operations. The "a" field is used to designate halt conditions for Halt and Wait instructions and to specify the CPU flags (program flags plus overflow) in the Set/Reset program flags instructions.

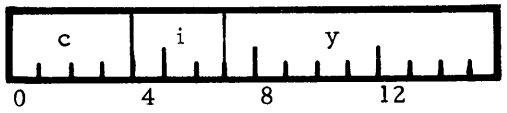
In the shift format the k-field determines whether the shifts further specified by the other fields will be left or right. The i-field specifies a binary count for the number of bits to be shifted. The a, l, and s-bits specify the other required shift information, i.e., arithmetic logical, single, double, etc. The r-bit designates whether the A or B-Register is to be used for single shifts and is the high order bit of the shift count for double length shifts.

In the register-to-register format the k-field is used to describe the operation to be performed using the two registers specified by the r_1 and r_2 fields.

Class II - One Word Relative Branches ... is composed of two major Op Codes each of which has its own format.

Branch A Condition

Branch Indicator Condition



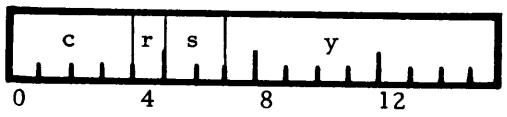
The y-field is the address which is added to the program counter to provide the relative branch address in all of the instructions in this class.

In the first format the l, g, e (less than, greater than, equal to), bits define the condition of the A-Register on which a branch is to take place. If l, e, g are all set true the instruction is interpreted as an unconditional branch.

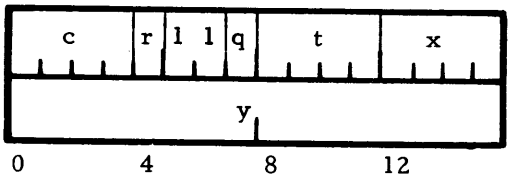
In the second format the i-field is binary decoded to define the branch conditions of the comparison indicators (overflow, less than, greater than, equal).

Class III - One and Two Word Memory Reference ... is composed of 11 major Op Codes and contains two subclasses: a one-word and a two-word format. (The second word of a two-word instruction is always located at the next higher memory location from the first word.)

Load and Store Operations,
Arithmetic Operations,
Logical Operations



Branch and Store P



All of these operations involve two storage locations: one the memory and the other an accumulator. The r and s-fields apply to both one and two-word formats. The r-field selects either the A or B-Register and the s-field provides the address modification specification. Three of the s-field codes provide relative, immediate, and indexing-with-B address modification for one-word instructions. The fourth s-field code is used to specify that the instruction is a two-word instruction.

The y-field, in both one and two-word formats, contains the memory address (before address modification), or, in the case of the immediate mode, the operand itself. In the one-word format (other than immediate mode) the 9-bit address is interpreted as a two's complement integer either between and including 0 and 511 for the one-word index B mode, or between and including -256 and 255 for the relative mode. When the one-word immediate mode is specified, the y-field is the operand and is interpreted as a two's complement integer between and including -256 and 255.

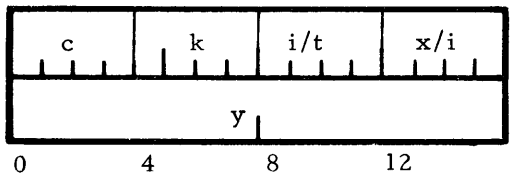
In the two-word format the y-field contains a positive binary number of value 0 to 65,535 inclusive. For the two-word immediate mode the y-field is interpreted as a two's complement number from $2^{15}-1$ to -2^{15} inclusive.

In the two-word format the t-field is used to designate an extensive set of address modification including the unique Table Mode of operation.

The q-bit is used to make effective either the r-bit (A/B selection) or the x-field (A, B, X_2 to X_{15}) for accumulator selection.

Class IV - Two Word Memory Reference ... is composed of one major Op Code and contains those instructions which are always two-word instructions referring to the memory.

Load and Store Operations
Branch Operations



The k-field is decoded to describe the operation to be performed. These operations generally involve transfers between memory and S, N, High Rate Interrupt Flags or branches on various flags, switches, and register conditions.

The x/i-field is used to select the X-Registers (A, B, X_2 to X_{15}), sense switches, program flags, compare indicators and interrupt line number (generally for branch commands).

Class V - Input/Output Instructions ... is composed of one major Op Code and includes all the instructions pertaining to communication between the CPU and the peripheral devices.

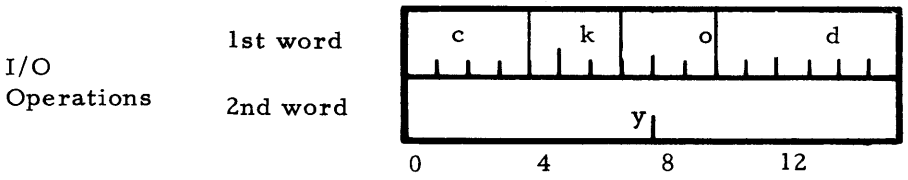
There are three basic types of I/O commands:

Execute Device Function (EDF) commands cause a controller and associated device to start or stop or go into a new mode. As with most I/O commands, failure to receive an EKO signal at the CPU within the timer limit will result in a reject branch.

Word transfer commands consist of Word Transfer In (WTI) and Word Transfer Out (WTO) commands. The WTO command causes 16-bit data to be transferred from the A-Register to the selected controller. The WTI command transfers 16-bits of data from the controller to the A-Register. Some controllers do not supply a full 16-bits (example: card readers supply only 12-bits) but this is a function of the gating of the controller. Generally, zeros will be placed in the unused bit positions.

Status Examination commands include Request Device Status (RDS), and Interrogate Common Interrupts (ICI). All of these commands are involved in examining the status of selected controllers. The execution of a Status Examination command does not reset the status flags in the controllers.

I/O Operations



The k-field is fully decoded to describe eight I/O instructions (three are reserved for special commands to system components). These signals are in the I/O bus.

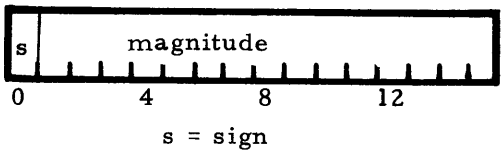
The o-field is sent to the peripheral controller via the I/O bus. This information is interpreted by the particular controller as an order to perform a specific operation. The A-Register is also presented on the I/O bus during Execute Device Function (EDF) commands to supplement the information in the o-field.

The d-field is placed on the I/O bus during most I/O commands to select the device controller (the Interrogate Common Interrupts is one exception). It is binary decoded providing up to 64 (0 through 63) device addresses.

The y-field is an address field used to provide a branch address in the event that an I/O command is rejected. It is located in the second word of the instruction, one memory location higher than the first word of the instruction.

6.3 DATA FORMATS

Binary Information ... is carried throughout the machine, in registers, display, I/O bus, and memory in the following format:

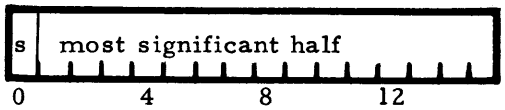


s = sign

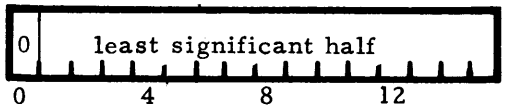
The format is two's complement wherein a one in the most significant position (bit 0) indicates a negative number and a zero indicates a positive number. The magnitude field (bits 1 through 15, most to least significant) is a binary code for positive numbers; for negative numbers the magnitude field is obtained by doing a "two's complement" (subtracting the desired negative numbers absolute magnitude from 2^{16}). The range of numbers which can be displayed is from $+2^{15}-1$ to -2^{15} inclusive.

Double Precision ... is not performed in hardware but by software subroutines. In these cases the following format is used:

Location a

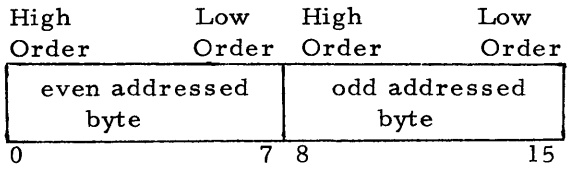


Location a+1



The range of numbers that can be represented in this format is from $+2^{30}-1$ to -2^{30}.

Byte Information ... is in ASCII format. Bytes are carried in the memory and CPU as shown below:



Within the memory bytes are loaded or stored starting with the high order byte and moving to the low order byte as the memory addresses increase; the bits being packed or unpacked as shown above (even numbered bytes in the high order position).

6.4 ADDRESSING AND ADDRESS MODIFICATION

The memory is addressed with 16-bit binary coded words providing addresses from 0 to 65, 535.

For systems having less than the full 64K of memory, the address will be caused to "roll over" when addresses higher than those available are used. As an example, if 4, 096 words are installed (address 0 - 4, 095) and address 4, 096 is sent to the memory, address 0000 will be selected.

Because of the relatively short word length of the system it is not possible to fully address the memory using a one-word instruction without some manipulation of the 9-bits (512-location capability) provided for address. This manipulation, along with other allied operations done for other reasons, is called address modification. Address modification can be specified by the programmer in the instructions of both one and two-word formats. The results of the various manipulations results in a final or Effective Address (EA) for the fetch of the operand or, in the case of a branch instruction, for placement in the program counter to indicate the next instruction to be executed.

One-Word Address Modifications ... use the two bits of the s-field to provide four binary combinations with the following meanings:

00	Relative Addressing*
01	Index with B
10	Immediate*
11	Two-Word Format

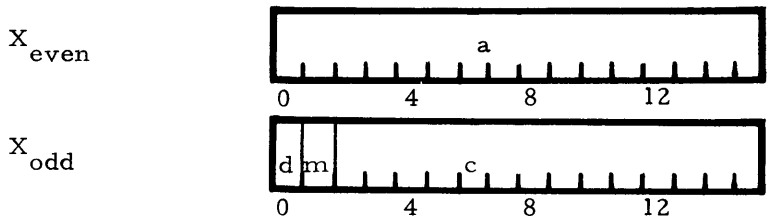
Two-Word Address Modifications ... use the four bits of the t-field to select the following address modification sequences:

0000	Direct
0001	Index with B
0010	Index with X_i
0011	Double Index with B and X_i
0100	Indirect
0101	Index B Indirect
0110	Index X_i Indirect
0111	Index B Indirect Index X_i
1000	Table Mode (see below)
1101	Indirect Index B
1110	Indirect Index X_i
1111	Immediate

*Note that the y-field is in two's complement form and can be either positive or negative.

Table Mode ... is a form of address modification but uses a pair of X-Registers to define a contiguous block of memory which is to be processed by the Table Mode instruction. The X_{even} Register is used to address memory and the X_{odd} Register maintains the count of addresses processed. Each time the Table Mode instruction is executed, the two X-Registers referenced are automatically incremented. When the count in the X_{odd} has been incremented to produce an overflow, the y-field of the second word of the instruction is used to load the program counter, causing an unconditional branch to y and ending the Table Mode operation. Note that while Table Mode is specified via the two-word format, its execution time is only that of a one-word instruction.

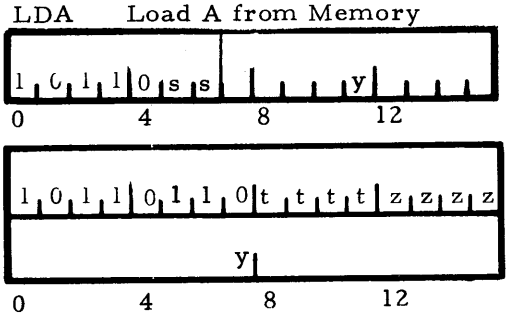
High-Rate I/O ... requires special mention as to work formats employed. The format held in the two X-Registers utilized for High-Rate I/O control are:



Field	Function
a	Address which I/O bus data is transferred to or from. Range: 0 to 65,535. The transfer is made and then the register is incremented. If m = 0 (byte mode) even addresses refer to the high order 8-bits (0 through 7) of the memory words and odd addresses refer to the low order bits (8 through 15) of the memory words.
d	Direction of transfer. "1" = input from I/O bus to memory; "0" = output from memory to I/O bus.
c	"Count" of words or bytes (depending on state of m). Since the count is updated by incrementing the count through the adder and the end of block transfer is detected by looking for an overflow, the "count" is set to the two's complement of the number of words or bytes required to be transferred, i.e., initial setting of $c = 2^{14}-n$, where n is the number of words to be transferred.
m	Mode of transfer, byte or word; if m = 0 the transfer takes place a byte per interrupt; if m = 1 a word per interrupt. Note that in byte mode automatic packing and unpacking of bytes occurs.

6.5 LOAD AND STORE INSTRUCTIONS

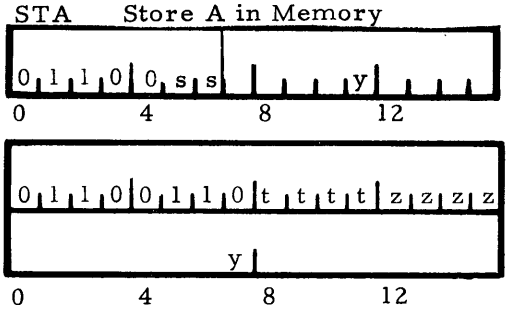
LDA Load A from Memory



Timing: 2/3 cycles

The contents of memory location y', which remain unchanged, are loaded into the A-Register.

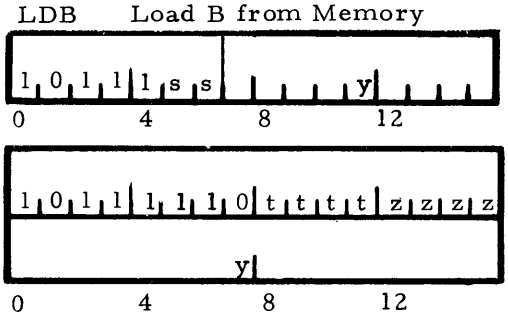
STA Store A in Memory



Timing: 2/3 cycles

The contents of A-Register, which remain unchanged, are stored into memory location y'.

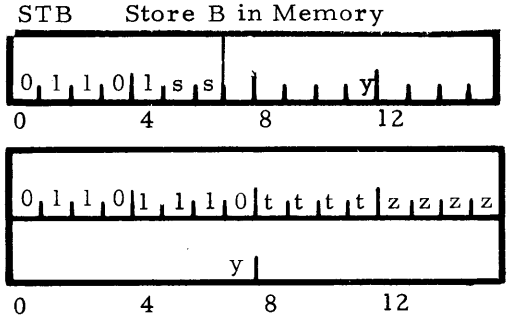
LDB Load B from Memory



Timing: 2/3 cycles

The contents of memory location y', which remain unchanged, are loaded into the B-Register.

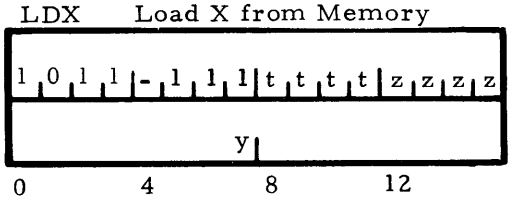
STB Store B in Memory



Timing: 2/3 cycles

The contents of the B-Register, which remain unchanged, are stored into memory location y'.

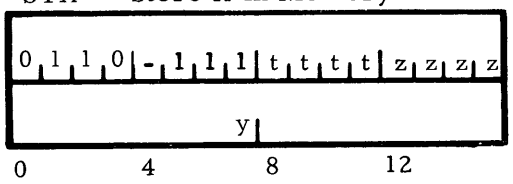
LDX Load X from Memory



Timing: 3 cycles

The contents of memory location y', which remain unchanged, are loaded into the X-Register specified by bits 12-15 of the instruction.

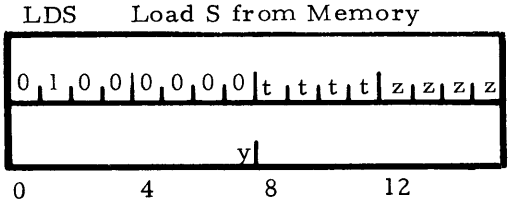
STX Store X in Memory



Timing: 3 cycles

The contents, which remain unchanged, of the X-Register specified by bits 12-15 of the instruction are stored into memory location y'.

LDS Load S from Memory



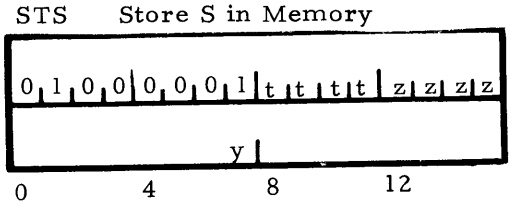
Timing: 3 cycles

The contents of memory location y' replace the contents of the S-Register. The contents of memory are unchanged. The contents of the various status flip-flops are replaced by the contents of the following bit positions of the memory location.

Bit	Indicator
0-7	Are not affected
8	Less
9	Greater
10	Equal
11	Overflow
12 to 15	Program Flags 8, 4, 2, 1

NOTE: Because this instruction will never be interpreted as an invalid instruction, programmers are cautioned against using LDS to change the status of the program flags. RPF and SPF are provided for this purpose.

STS Store S in Memory

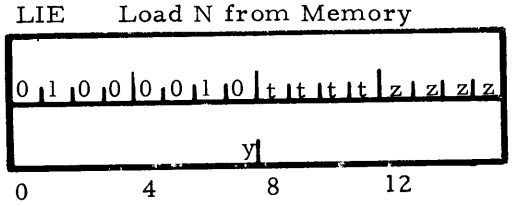


Timing: 3 cycles

The contents of the S-Register replace the contents of the memory location y'. The contents of the status flip-flops are unchanged by this instruction. The contents of the various status flip-flops will be stored into the specified memory location in the following bit positions:

Bit	Location
0 to 6	High-Rate Interrupt Flags 0 to 6
7	Not Used
8	Less
9	Greater
10	Equal
11	Overflow
12 to 15	Program Flags 8, 4, 2, 1

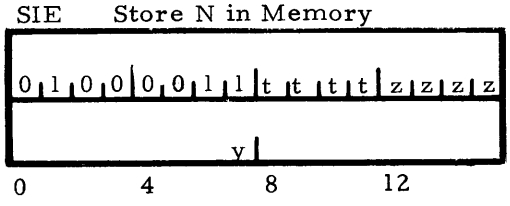
LIE Load N from Memory



Timing: 3 cycles

The contents of memory location y', which remain unchanged, are loaded into the N-Register. The bit position in the N-Register corresponds to the interrupt line number; i.e., bit position 1 corresponds to interrupt line 1.

SIE Store N in Memory



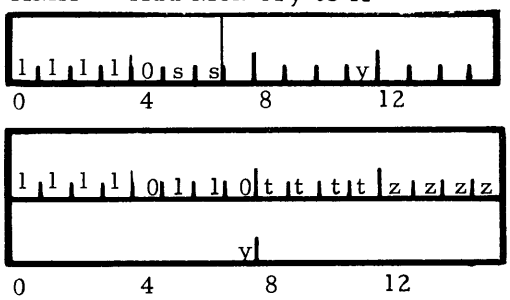
Timing: 3 cycles

The contents of the N-Register, which remain unchanged, are stored into memory location y'. The bit position in the N-Register corresponds to the interrupt line number; i.e., bit position 1 corresponds to interrupt line 1.

6.6 ARITHMETIC INSTRUCTIONS

AMA Add Memory to A

Timing: 2/3 cycles



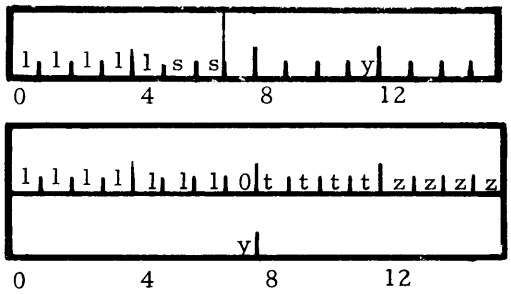
The contents of memory location y', which remain unchanged, are added to the contents of the A-Register and the result is placed in the A-Register. Negative values are in the two's complement form. The overflow indicator in the S-Register is set if either of the following two conditions exist:

(1) If the contents of memory and the contents of the A-Register are both positive, but the result is negative.

(2) If the contents of memory and the contents of the A-Register are both negative, but the result is positive.

AMB Add Memory to B

Timing: 2/3 cycles



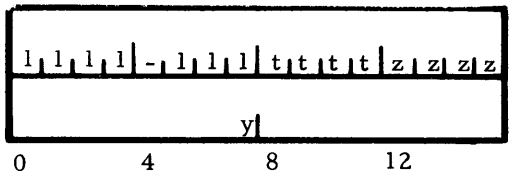
The contents of memory location y', which remain unchanged, are added to the contents of the B-Register and the result is placed in the B-Register. Negative values are in the two's complement form. The overflow indicator in the S-Register is set if either of the following two conditions exist.

(1) If the contents of memory and the contents of the B-Register are both positive, but the result is negative.

(2) If the contents of memory and the contents of the B-Register are both negative, but the result is positive.

AMX Add Memory to X

Timing: 3 cycles

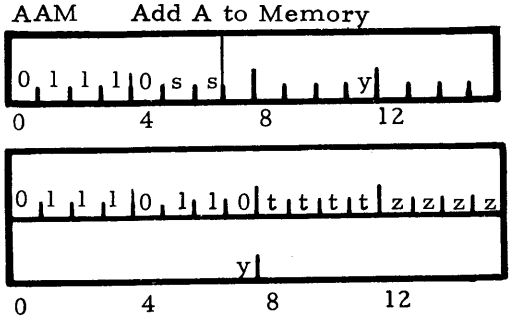


The contents of memory location y', which remain unchanged, are added to the contents of the X-Register specified by bits 12-15 of the instruction and the result is placed in the specified X-Register. Negative values are in the two's complement form. The overflow indicator in the S-Register is set if either of the following two conditions exist:

(1) If the contents of memory and the contents of the X-Register are both positive, but the result is negative.

(2) If the contents of memory and the contents of the X-Register are both negative, but the result is positive.

AAM Add A to Memory



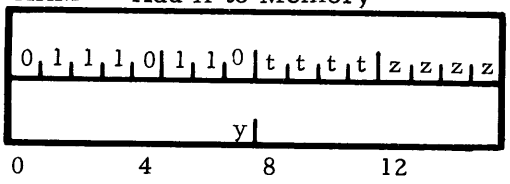
Timing: 3/4 cycles

The contents of the A-Register, which remain unchanged, are added to the contents of memory location y' and the result is placed in memory location y'. Negative values are in the two's complement form. The overflow indicator in the S-Register is set if either of the following two conditions exist:

(1) If the contents of memory and the contents of the A-Register are both positive, but the result is negative.

(2) If the contents of memory and the contents of the A-Register are both negative, but the result is positive.

AXM Add X to Memory



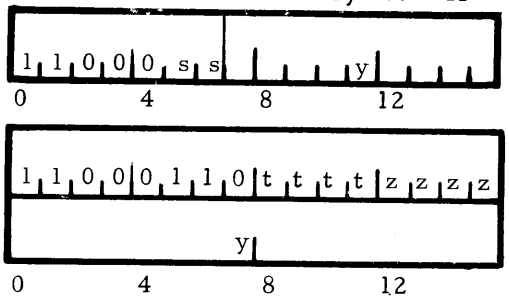
Timing: 4 cycles

The contents, which remain unchanged, of the X-Register specified by bits 12-15 of the instruction are added to the contents of memory location y' and the result is placed in memory location y'. Negative values are in the two's complement form. The overflow indicator in the S-Register is set if either of the following two conditions exist:

(1) If the contents of memory and the contents of the X-Register are both positive, but the result is negative.

(2) If the contents of memory and the contents of the X-Register are both negative, but the result is positive.

SMA Subtract Memory from A



Timing: 2/3 cycles

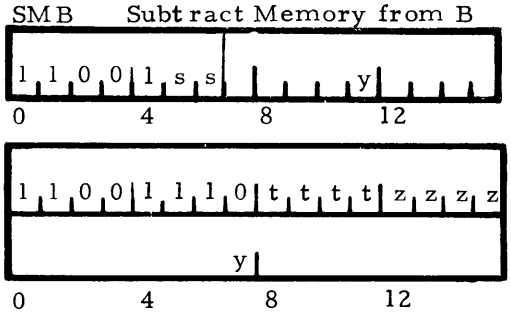
The contents of memory location y', which remain unchanged, are subtracted from the contents of the A-Register and the result is placed in the A-Register. Negative values are in the two's complement form. The overflow indicator in the S-Register is set if either of the following two conditions exist:

(1) If the contents of the A-Register are negative and the contents of memory are positive, but the result is positive.

(2) If the contents of the A-Register are positive and the contents of memory are negative, but the result is positive.

SMB Subtract Memory from B

Timing: 2/3 cycles



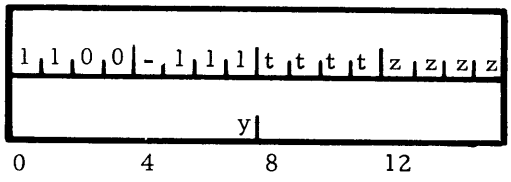
The contents of memory location y', which remain unchanged, are subtracted from the contents of the B-Register and the result is placed in the B-Register. Negative values are in the two's complement form. The overflow indicator in the S-Register is set if either of the following two conditions exist:

(1) If the contents of the B-Register are negative and the contents of memory are positive, but the result is positive.

(2) If the contents of the B-Register are positive and the contents of memory are negative, but the result is negative.

SMX Subtract Memory from X

Timing: 3 cycles



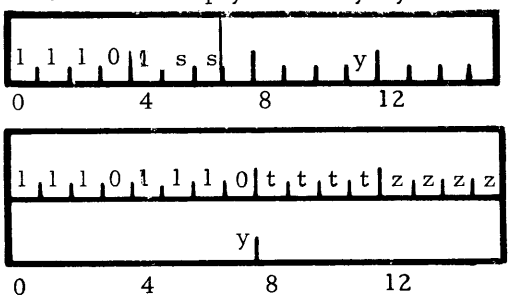
The contents of memory location y', which remain unchanged, are subtracted from the contents of the X-Register specified by bits 12-15 of the instruction and the result is placed in the specified X-Register. Negative values are in the the two's complement form. The overflow indicator in the S-Register is set if either of the following two conditions exist:

(1) If the contents of the X-Register are negative and the contents of memory are positive, but the result is positive.

(2) If the contents of the X-Register are positive and the contents of memory are negative, but the result is negative.

MUL Multiply Memory by B

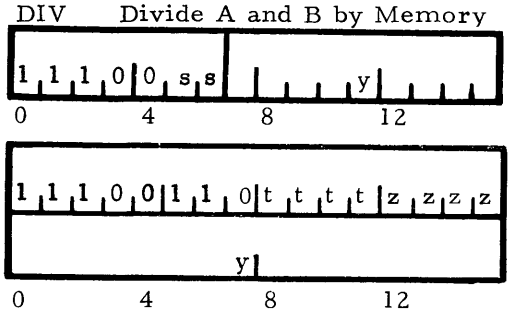
Timing: 6/6.9 μs



The contents of memory location y', which remain unchanged, are multiplied by the contents of the B-Register and the 31-bit product is placed in the A and B-Registers. The most significant bits of the result are in the A-Register (bit A_0 is the sign), and the least significant bits of the result are in the B-Register (bit B_0 is set to zero and is not part of the product). The original contents of the A-Register have no effect on the multiply operation. The overflow indicator in the S-Register is set if and only if the value 8000_{16} is multiplied by the value 8000_{16}.

DIV Divide A and B by Memory

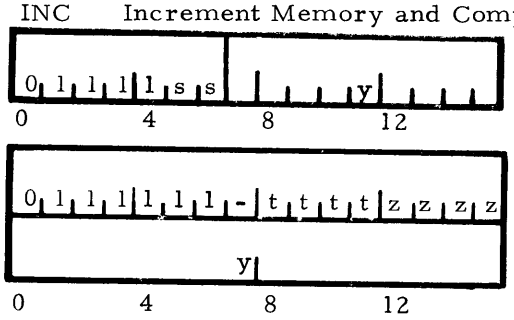
Timing: 6.5/7.4 μs



The contents of the A and B-Registers are divided by the contents of memory location y', which remain unchanged; the quotient is placed in the B-Register and the remainder is placed in the A-Register (unless it is zero, the sign of the remainder is the same as the sign of the dividend). Originally the A-Register must contain the most significant bits of the dividend (bit A_0 is the sign), and the B-Register must contain the least significant bits of the dividend (B_0 is not part of the dividend). The overflow indicator in the S-Register is set if and only if the absolute value of the dividend (A and B-Registers) is greater than or equal to the absolute value of the divisor (memory). In this case the result is not meaningful.

INC Increment Memory and Compare with Zero

Timing: 3/4 cycles

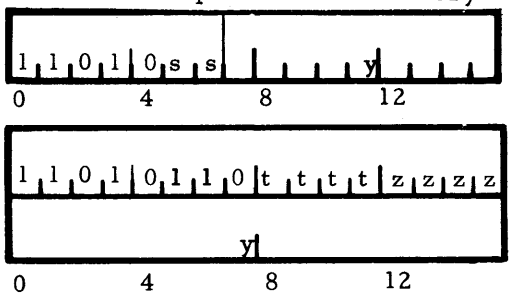


The contents of memory location y' are incremented by one (bit 15) and the result is placed in the same location. The result is compared with zero and the appropriate indicator in the S-Register is set. The indicators are reset by the logic as the first step in the execution of this instruction.

Condition	Indicator	Bit
Result < 0	L set to 1	8
Result > 0	G set to 1	9
Result = 0	E set to 1	10

The overflow indicator in the S-Register is set if and only if the original value in memory location y' is $7FFF_{16}$, since the result is 8000_{16} (L indicator will be set to 1).

CAM Compare A with Memory

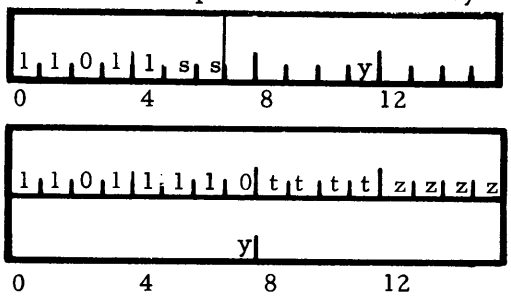


Timing: 2/3 cycles

The contents of the A-Register, which remain unchanged, are compared algebraically with the contents of memory location y', which remain unchanged, and the appropriate indicator in the S-Register is set. The indicators are reset by the logic as the first step in the execution of this instruction.

Condition	Indicator	Bit
(A) < (y')	L set to 1	8
(A) > (y')	G set to 1	9
(A) = (y')	E set to 1	10

CBM Compare B with Memory

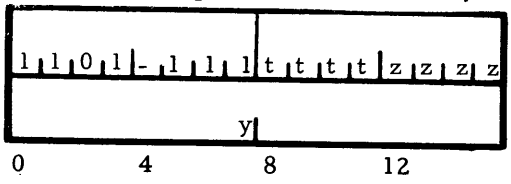


Timing: 2/3 cycles

The contents of the B-Register, which remain unchanged, are compared algebraically with the contents of memory location y', which remain unchanged, and the appropriate indicator in the S-Register is set. The indicators are reset by the logic as the first step in the execution of this instruction.

Condition	Indicator	Bit
(B) < (y')	L set to 1	8
(B) > (y')	G set to 1	9
(B) = (y')	E set to 1	10

CXM Compare X with Memory



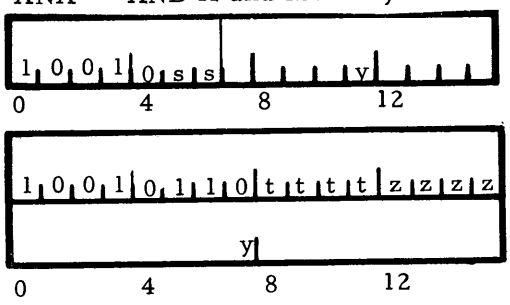
Timing: 3 cycles

The contents of the X-Register specified by bits 12-15 of the instruction, which remain unchanged, are compared algebraically with the contents of memory location y', which remain unchanged, and the appropriate indicator in the S-Register is set. The indicators are reset by the logic as the first step in the execution of this instruction.

Condition	Indicator	Bit
(X) < (y')	L set to 1	8
(X) > (y')	G set to 1	9
(X) = (y')	E set to 1	10

6.7 LOGICAL INSTRUCTIONS

ANA AND A and Memory



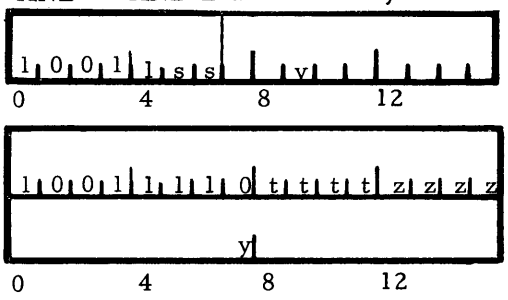
Timing: 2/3 cycles

The contents of memory location y', which remain unchanged, and the contents of the A-Register form a logical product. The result is placed in the A-Register.

Example:

Memory	101101
A-Register	011001
A-Register (Result)	001001

ANB AND B and Memory



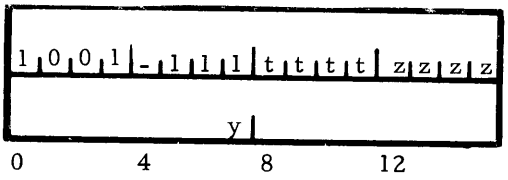
Timing: 2/3 cycles

The contents of memory location y', which remain unchanged, and the contents of the B-Register form a logical product. The result is placed in the B-Register.

Example:

Memory	001110
B-Register	111111
B-Register (Result)	001110

ANX AND X and Memory



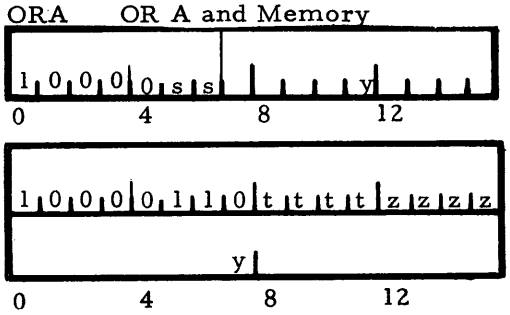
Timing: 3 cycles

The contents of memory location y', which remain unchanged, and the contents of the X-Register specified by bits 12-15 of the instruction form a logical product. The result is placed in the specified X-Register.

Example:

Memory	100011
X-Register	111001
X-Register (Result)	100001

ORA OR A and Memory



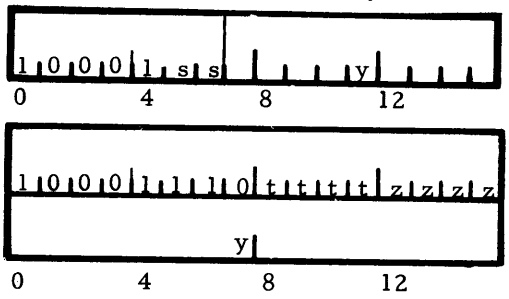
Timing: 2/3 cycles

The contents of memory location y', which remain unchanged, and the contents of the A-Register form a logical sum. The result is placed in the A-Register.

Example:

Memory	100110
A-Register	001100
A-Register (Result)	101110

ORB OR B and Memory



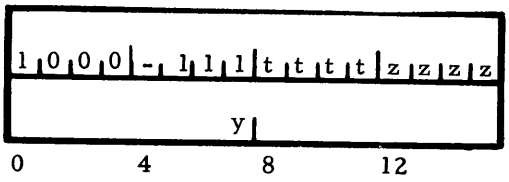
Timing: 2/3 cycles

The contents of memory location y', which remain unchanged, and the contents of the B-Register form a logical sum. The result is placed in the B-Register.

Example:

Memory	000111
B-Register	111010
B-Register (Result	111111

ORX OR X and Memory



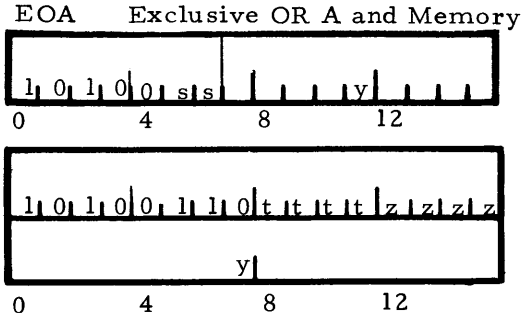
Timing: 3 cycles

The contents of memory location y', which remain unchanged, and the contents of the X-Register specified by bits 12-15 of the instruction form a logical sum. The result is placed in the specified X-Register.

Example:

Memory	110001
X-Register	010000
X-Register (Result)	110001

EOA Exclusive OR A and Memory



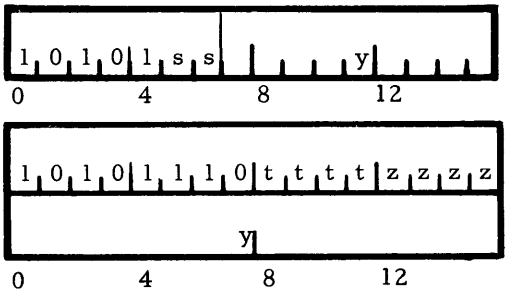
Timing: 2/3 cycles

The contents of memory location y', which remain unchanged, and the contents of the A-Register form an exclusive OR. The result is placed in the A-Register.

Example:

Memory	110011
A-Register	010101
A-Register (Result)	100110

EOB Exclusive OR B and Memory



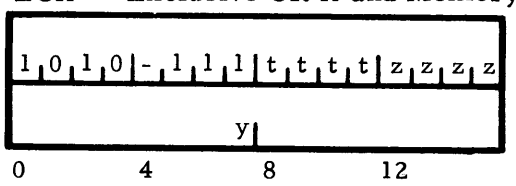
Timing: 2/3 cycles

The contents of memory location y', which remain unchanged, and the contents of the B-Register form an exclusive OR. The result is placed in the B-Register.

Example:

Memory	001011
B-Register	111001
B-Register (Result)	110010

EOX Exclusive OR X and Memory



Timing: 3 cycles

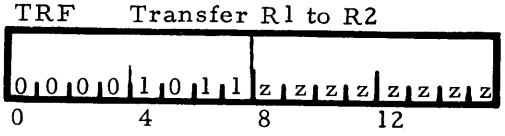
The contents of memory location y', which remain unchanged, and the contents of the X-Register specified by bits 12-15 of the instruction form an exclusive OR. The result is placed in the specified X-Register.

Example:

Memory	001011
X-Register	100011
X-Register (Result)	101000

6.8 REGISTER INSTRUCTIONS

TRF Transfer R1 to R2

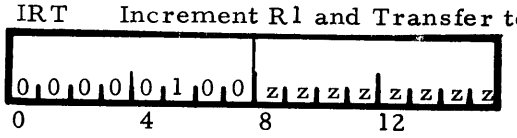


Timing: 1 cycle

Transfer the contents, which remain unchanged, of register R1, specified by bits 12-15 of the instruction, to register R2, specified by bits 8-11 of the instruction.

R1/R2 Code	Register
0	A
1	B
2 to 15	X2 to X15

IRT Increment R1 and Transfer to R2

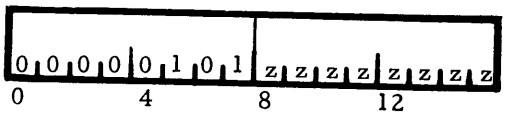


Timing: 1 cycle

The contents, which remain unchanged, of register R1, specified by bits 12-15 of the instruction, are incremented by one (bit 15) and the result is transferred to register R2, specified by bits 8-11 of the instruction. The overflow indicator in the S-Register is set if the contents of register R1 is positive and the result is negative.

R1/R2 Code	Register
0	A
1	B
2 to 15	X2 to X15

DRT Decrement R1 and Transfer to R2

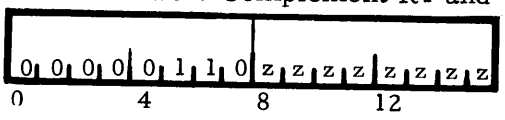


Timing: 1 cycle

The contents, which remain unchanged, of register R1, specified by bits 12-15 of the instruction, are decremented by one (bit 15) and the result is transferred to register R2, specified by bits 8-11 of the instruction. The overflow indicator in the S-Register is set if the contents of register R1 is negative and the result is positive.

R1/R2 Code	Register
0	A
1	B
2 to 15	X2 to X15

TRT Two's Complement R1 and Transfer to R2 Timing: 1 cycle

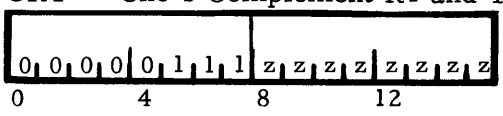


The contents, which remain unchanged, of register R1, specified by bits 12-15 of the instruction, are two's complemented and the result is transferred to register R2, specified by bits 8-11 of the instruction.

R1/R2 Code	Register
0	A
1	B
2 to 15	X2 to X15

The overflow indicator in the S-Register is set if an attempt is made to form the two's complement of the contents of a register that contains the value 8000_{16}.

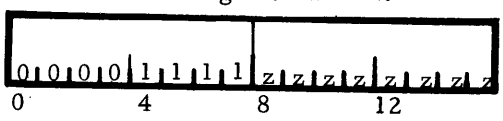
ORT One's Complement R1 and Transfer to R2 Timing: 1 cycle



The contents, which remain unchanged, of register R1 specified by bits 12-15 of the instruction, are one's complemented and the result is transferred to register R2, specified by bits 8-11 of the instruction.

R1/R2 Code	Register
0	A
1	B
2 to 15	X2 to X15

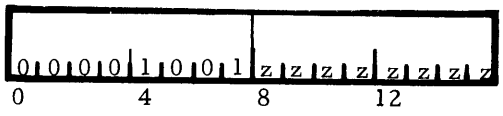
EXG Exchange R1 with R2 Timing: 1 cycle



The contents of register R1, specified by bits 12-15 of the instruction are exchanged with the contents of register R2, specified by bits 8-11 of the instruction.

R1/R2 Code	Register
0	A
1	B
2 to 15	X2 to X15

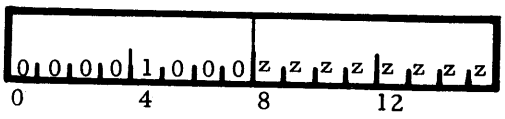
CRZ Clear R1 and R2 to Zero Timing: 1 cycle



Register R1, specified by bits 12-15, and register R2, specified by bits 8-11 of the instruction, are both cleared to zero.

R1/R2 Code	Register
0	A
1	B
2 to 15	X2 to X15

ARR Add R1 to R2



Timing: 1 cycle

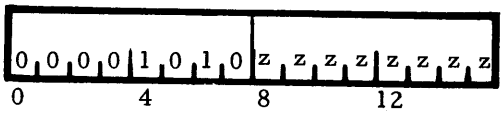
The contents, which remain unchanged, of register R1, specified by bits 12-15 of the instruction, are added to the contents of register R2, specified by bits 8-11 of the instruction and the result is placed in register R2. Negative values are in two's complement form. The overflow indicator in the S-Register is set if either of the following two conditions exist:

(1) If the contents of registers R1 and R2 are both positive, but the result is negative.

(2) If the contents of registers R1 and R2 are both negative, but the result is positive.

R1/R2 Code	Register
0	A
1	B
2 to 15	X2 to X15

SRR Subtract R1 from R2



Timing: 1 cycle

The contents, which remain unchanged, of register R1, specified by bits 12-15 of the instruction, are subtracted from the contents of register R2, specified by bits 8-11 of the instruction, and the result is placed in register R2. Negative values are in two's complement form. The overflow indicator in the S-Register is set if either of the following two conditions exist:

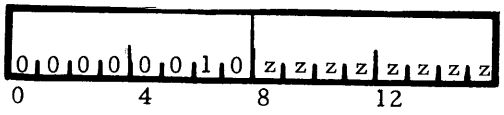
(1) If the contents of register R1 are positive and the contents of register R2 are negative, but the result is positive.

(2) If the contents of register R1 are negative and the contents of register R2 are positive, but the result is negative.

R1/R2 Code	Register
0	A
1	B
2 to 15	X2 to X15

CRR Compare R1 with R2

Timing: 1 cycle



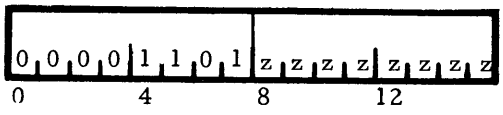
The contents, which remain unchanged, of register R1, specified by bits 12-15 of the instruction, are compare algebraically with the contents of register R2, specified by bits 8-11 of the instruction, which remain unchanged, and the appropriate indicator in the S-Register is set. The indicators are reset by the logic as the first step in the execution of this instruction.

Condition	Indicator	Bit
(R1) < (R2)	L set to 1	8
(R1) > (R2)	G set to 1	9
(R1) = (R2)	E set to 1	10

R1/R2 Code	Register
0	A
1	B
2 to 15	X2 to X15

ANR AND R1 and R2

Timing: 1 cycle



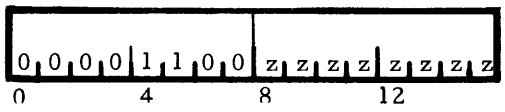
The contents, which remain unchanged, of register R1, specified by bits 12-15 of the instruction, and the contents of register R2, specified by bits 8-11 of the instruction, form a logical product. The result is placed in register R2.

R1/R2 Code	Register
0	A
1	B
2 to 15	X2 to X15

Example:

R1	110110
R2	001110
R2 (Result)	000110

ORR OR R1 and R2



Timing: 1 cycle

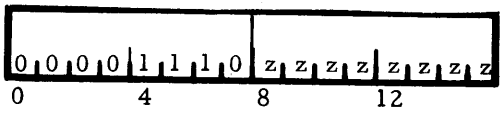
The contents, which remain unchanged, of register R1, specified by bits 12-15 of the instruction, and the contents of register R2, specified by bits 8-11 of the instruction, form a logical sum. The result is placed in register R2.

R1/R2 Code	Register
0	A
1	B
2 to 15	X2 to X15

Example:

R1	010101
R2	101100
R2 (Result)	111101

EOR Exclusive OR R1 and R2



Timing: 1 cycle

The contents, which remain unchanged, of register R1, specified by bits 12-15 of the instruction, and the contents of register R2, specified by bits 8-11 of the instruction, form an exclusive OR. The result is placed in register R2.

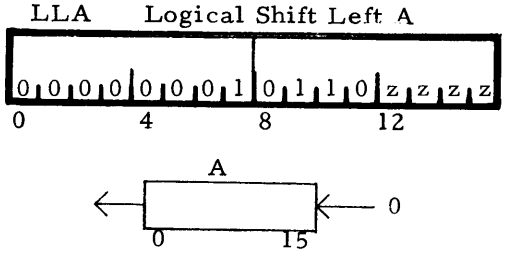
R1/R2 Code	Register
0	A
1	B
2 to 15	X2 to X15

Example:

R1	000110
R2	101100
R2 (Result)	101010

6.9 SHIFT INSTRUCTIONS

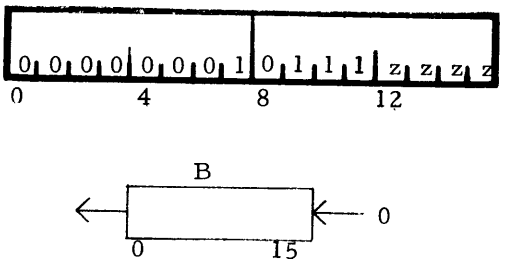
LLA Logical Shift Left A



Timing: 1.16 + 0.13 nμs

The contents of the A-Register are shifted left the number of bit positions specified by bits 12-15 of the instruction. Bits shifted out of the A-Register from bit position 0 are lost, and zeros are shifted into the vacated bit positions.

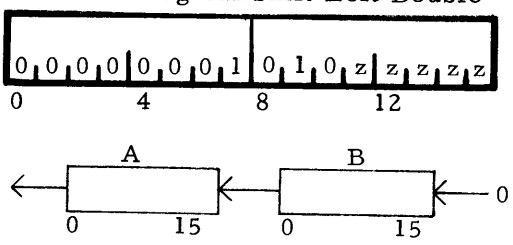
LLB Logical Shift Left B



Timing: 1.15 + 0.125nμs

The contents of the B-Register are shifted left the number of bit positions specified by bits 12-15 of the instruction. Bits shifted out of the B-Register from bit position zero are lost, and zeros are shifted into the vacated bit positions.

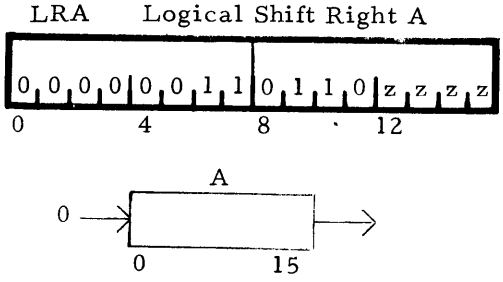
LLD Logical Shift Left Double



Timing: 1.15 + 0.125nμs

The contents of the A and B-Registers are shifted left the number of bit positions specified by bits 11-15 of the instruction. Bits shifted out of the A-Register from bit position zero are lost, bits are shifted from bit position zero of the B-Register into bit position 15 of the A-Register, and zeros are shifted into the vacated bit positions.

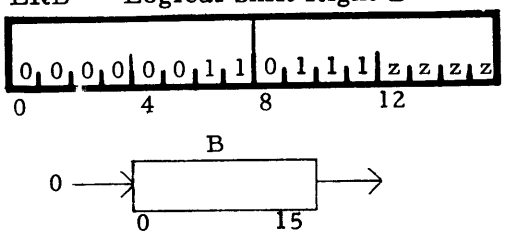
LRA Logical Shift Right A



Timing: 1.15 + 0.125nμs

The contents of the A-Register are shifted right the number of bit positions specified by bits 12-15 of the instruction. Bits shifted out of the A-Register from bit position 15 are lost, and zeros are shifted into the vacated bit positions.

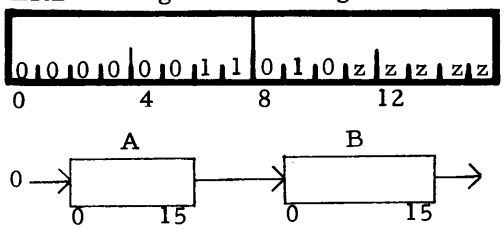
LRB Logical Shift Right B



Timing: 1.15 + 0.125nμs

The contents of the B-Register are shifted right the number of bit positions specified by bits 12-15 of the instruction. Bits shifted out of the B-Register from bit position 15 are lost, and zeros are shifted into the vacated bit positions.

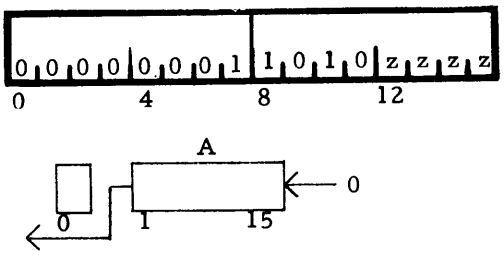
LRD Logical Shift Right Double



Timing: 1.15 + 0.125nμs

The contents of the A and B-Registers are shifted right the number of bit positions specified by bits 11-15 of the instruction. Bits shifted out of the B-Register from bit position 15 are lost, bits are shifted from bit position 15 of the A-Register into bit position zero of the B-Register, and zeros are shifted into the vacated bit positions.

ALA Arithmetic Shift Left A



Timing: 1.15 + 0.125nμs

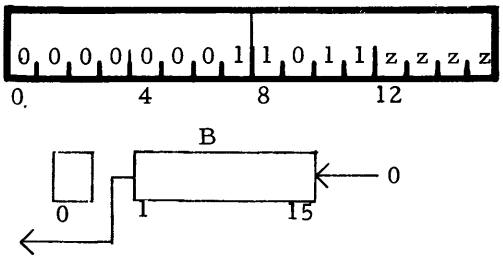
The contents of the A-Register are shifted left the number of bit positions specified by bits 12-15 of the instruction. Bits shifted out of the A-Register from bit position 1 are lost, zeros are shifted into the vacated bit positions, and bit position zero (sign bit) does not change. The overflow indicator in the S-Register is set if either of the following two conditions exist:

(1) If the sign of the A-Register is negative and a zero is shifted from bit position 1.

(2) If the sign of the A-Register is positive and a one is shifted from bit position 1.

ALB Arithmetic Shift Left B

Timing: 1.15 + 0.125nμs



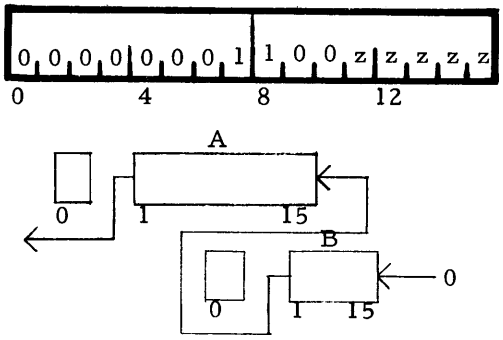
The contents of the B-Register are shifted left the number of bit positions specified by bits 12-15 of the instruction. Bits shifted out of the B-Register from bit position 1 are lost, zeros are shifted into the vacated bit positions, and bit position zero (sign bit) does not change. The overflow indicator in the S-Register is set if either of the following two conditions exist:

(1) If the sign of the B-Register is negative and a zero is shifted from bit position 1.

(2) If the sign of the B-Register is positive and a one is shifted from bit position 1.

ALD Arithmetic Shift Left Double

Timing: 1.15 + 0.125nμs



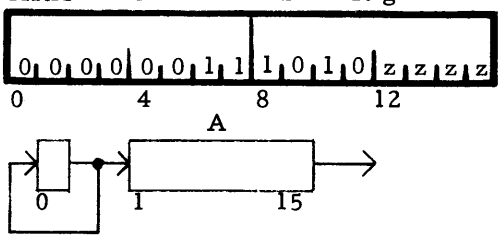
The contents of the A and B-Registers are shifted left the number of bit positions specified by bits 11-15 of the instruction. Bits shifted out of the A-Register from bit position 1 are lost, bits are shifted from bit position 1 of the B-Register into bit position 15 of the A-Register, zeros are shifted into the vacated bit positions, and bit position zero of both the A and B-Registers (sign bits) do not change. The overflow indicator in the S-Register is set if either of the following two conditions exist:

(1) If the sign of the A-Register is negative and a zero is shifted from bit position 1.

(2) If the sign of the A-Register is positive and a one is shifted from bit position 1.

ARA Arithmetic Shift Right A

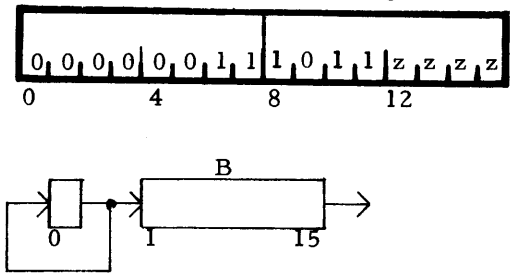
Timing: 1.15 + 0.125nμs



The contents of the A-Register are shifted right the number of bit positions specified by bits 12-15 of the instruction. Bits shifted out of the A-Register from bit position 15 are lost and bit position zero (sign bit), which remains unchanged, is shifted into the vacated bit positions.

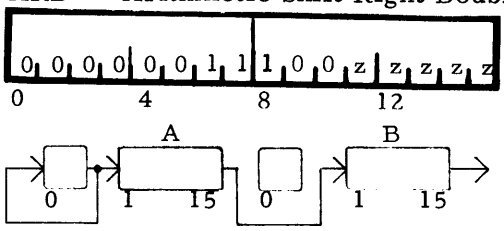
ARB Arithmetic Shift Right B

Timing: 1.15 + 0.125nμs



The contents of the B-Register are shifted right the number of bit positions specified by bits 12-15 of the instruction. Bits shifted out of the B-Register from bit position 15 are lost and bit position zero (sign bit), which remains unchanged, is shifted into the vacated bit positions.

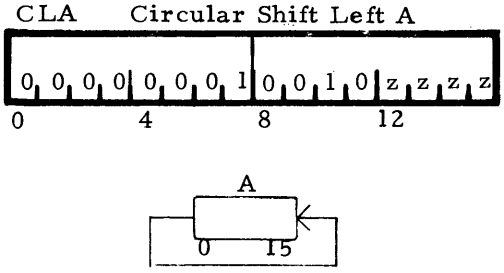
ARD Arithmetic Shift Right Double



Timing: 1.15 + 0.125n μs

The contents of the A and B-Registers are shifted right the number of bit positions specified by bits 11-15 of the instruction. Bits shifted out of the B-Register from bit position 15 are lost, bits are shifted from bit position 15 of the A-Register into bit position 1 of the B-Register, bit position zero of the B-Register remains unchanged and bit position zero of the A-Register (sign bit), which remains unchanged, is shifted into the vacated bit positions.

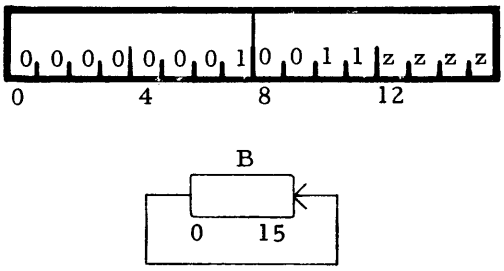
CLA Circular Shift Left A



Timing: 1.15 + 0.125n μs

The contents of the A-Register are circular shifted left the number of bit positions specified by bits 12-15 of the instruction. Bits shifted out of the A-Register from bit position zero are re-entered at bit position 15.

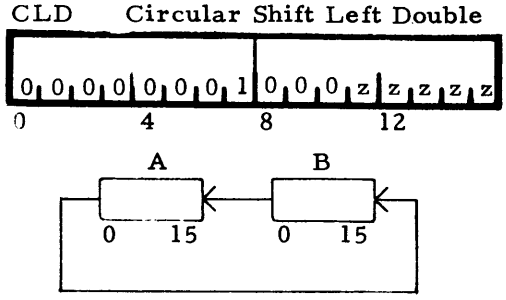
CLB Circular Shift Left B



Timing: 1.15 + 0.125n μs

The contents of the B-Register are circular shifted left the number of bit positions specified by bits 12-15 of the instruction. Bits shifted out of the B-Register from bit position zero are re-entered at bit position 15.

CLD Circular Shift Left Double

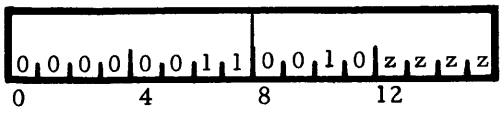


Timing: 1.15 + 0.125n μs

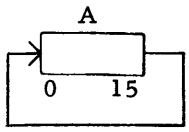
The contents of the A and B-Registers are circular shifted left the number of bit positions specified by bits 11-15 of the instruction. Bits shifted out of the A-Register from bit position zero are re-entered at bit position 15 of the B-Register, and bits are shifted from bit position zero of the B-Register into bit position 15 of the A-Register.

CRA Circular Shift Right A

Timing: 1.15 + 0.125n μs

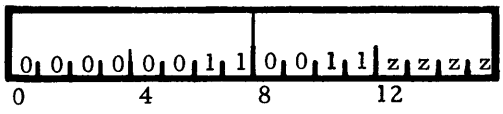


The contents of the A-Register are circular shifted right the number of bit positions specified by bits 12-15 of the instruction. Bits shifted out of the A-Register from bit position 15 are re-entered at bit position zero.

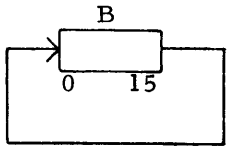


CRB Circular Shift Right B

Timing: 1.15 + 0.125n μs

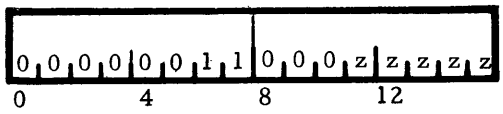


The contents of the B-Register are circular shifted right the number of bit positions specified by bits 12-15 of the instruction. Bits shifted out of the B-Register from bit position 15 are re-entered at bit position zero.

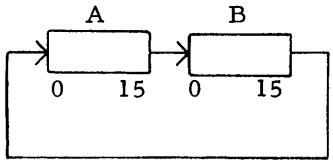


CRD Circular Shift Right Double

Timing: 1.15 + 0.125n μs

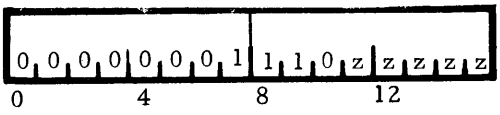


The contents of the A and B-Registers are circular shifted right the number of bit positions specified by bits 11-15 of the instruction. Bits shifted out of the B-Register from bit position 15 are re-entered at bit position zero of the A-Register, and bits are shifted from bit position 15 of the A-Register into bit position zero of the B-Register.

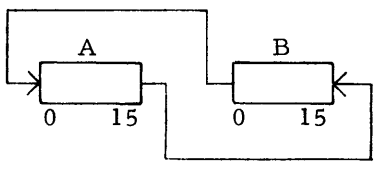


BRV Bit Reversal

Timing: 1.15 + 0.125n μs

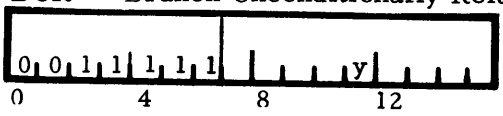


The contents of the A-Register are shifted right and the contents of the B-Register are shifted left the number of bit positions specified by bits 11-15 of the instruction. Bits shifted out of the A-Register from bit position 15 are re-entered into the B-Register at bit position 15, and bits shifted out of the B-Register from bit position zero are re-entered into the A-Register at bit position zero.



6.10 BRANCH INSTRUCTIONS

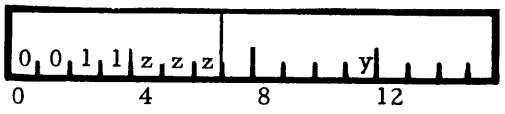
BUR Branch Unconditionally Relative



Timing: 1 cycle

The signed value y, specified by bits 7-15 of the instruction (bit 7 is sign bit), where $-256 \leq y < 256$, is added to the memory address in the P-Register and the result is placed in the P-Register. This allows an unconditional branch relative to the existing P-Register address specification.

BAR Branch A Condition Relative



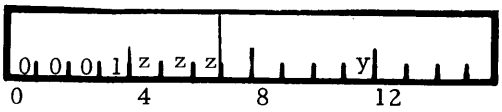
Timing: 1 cycle

Any single test or combination of tests may be specified in the instruction. The contents of the A-Register are compared algebraically with zero and the result (less, greater or equal) is compared with the corresponding indicator bit in the instruction (bits 4-6). If a correspondence exists, the memory address from which the next instruction is to be obtained is derived by adding the signed value y specified by bits 7-15 of the instruction (bit 7 is the sign bit), where $-256 \leq y < 256$, to the memory address in the P-Register. The result is placed in the P-Register and the branch is effected. If no correspondence exists, the next sequential instruction is executed. The execution of the BAR instruction does not change either the A-Register or S-Register. The indicator bit assignments for the instruction are the following:

Branch Condition	Indicator	Bit
(A) < 0	L, Less than	4
(A) > 0	G, Greater than	5
(A) = 0	E, Equal	6

BIR Branch Indicator Condition Relative

Timing: 1 cycle

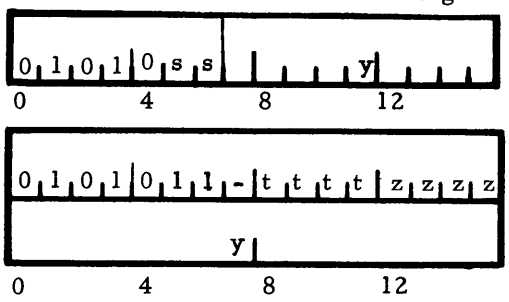


Any single test or combination of tests may be specified in the instruction after the indicator bits in the S-Register have been set/reset by the execution of a CAM, CBM, CXM, CRR, INC or LDS instruction. If a correspondence exists between any of the indicators specified by bits 4-6 of the instruction and the associated indicators set in the S-Register, then the memory address from which the next instruction is to be obtained is derived by adding the signed value y specified by bits 7-15 of the instruction (bit 7 is sign bit), where $-256 \leq y < 256$, to the memory address in the P-Register. The result is placed in the P-Register and the branch is effected. If no correspondence exists, the next sequential instruction is executed. When a branch on the overflow condition is executed, the overflow indicator in the S-Register is reset. The other indicators in the S-Register are not affected by the execution of the BIR instruction. The indicator bit assignments for the instruction are the following:

Indicator	Bit Assignment 4	5	6
O, Overflow	0	0	0
L, Less	1	0	0
G, Greater	0	1	0
LG, Less or Greater	1	1	0
E, Equal	0	0	1
LE, Less or Equal	1	0	1
GE, Greater or Equal	0	1	1
N, No Overflow	1	1	1

BSP Branch and Store P-Register

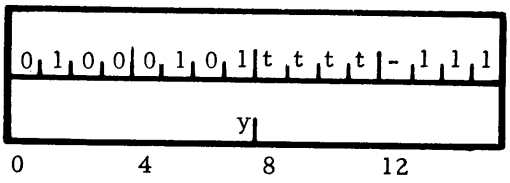
Timing: 2/3 cycles



The contents of the P-Register, which represents the address of the instruction following the BSP, are stored in memory location y'. The next instruction to be executed is fetched from memory location y' plus one.

BRU Branch Unconditionally

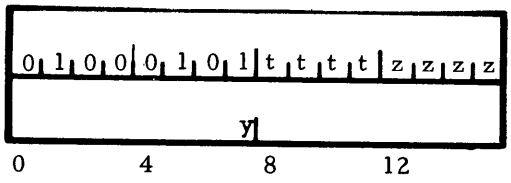
Timing: 2 cycles



The memory address y' is placed in the P-Register, allowing complete memory addressing capability for the fetch of the next instruction.

BAC Branch A Condition

Timing: 1/2 cycles

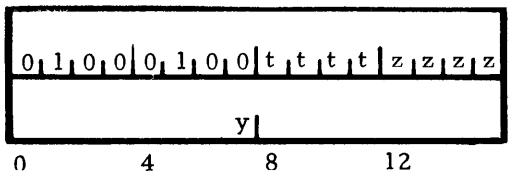


Any single test or combination of tests may be specified in the instruction. The contents of the A-Register are compared algebraically with zero and the result is compared with the corresponding indicator bit in the instruction. If a correspondence exists, the next instruction is fetched from memory location y'. If no correspondence exists, the next sequential instruction is executed. The execution of the BAC instruction does not change either the A-Register or the S-Register. The indicator bit assignments for the instruction are the following:

Branch Condition	Indicator	Bit
(A) < 0	L, Less than	13
(A) > 0	G, Greater than	14
(A) = 0	E, Equal	15

BBC Branch B Condition

Timing: 1/2 cycles

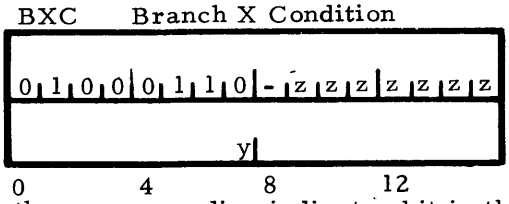


Any single test or combination of tests may be specified in the instruction. The contents of the B-Register are compared algebraically with zero and the result is compared with the corresponding indicator bit in the instruction. If a correspondence exists, the next instruction is fetched from memory location y'. If no correspondence exists, the next sequential instruction is executed. The execution of the BBC instruction does not change either the B-Register or the S-Register. The indicator bit assignments for the instruction are the following:

Branch Condition	Indicator	Bit
(B) < 0	L, Less than	13
(B) > 0	G, Greater than	14
(B) = 0	E, Equal	15

BXC Branch X Condition

Timing: 1/2 cycles

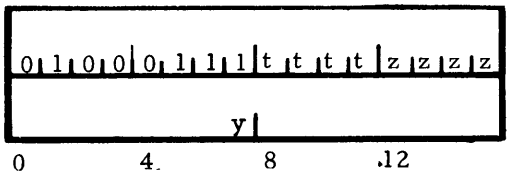


Any single test or combination of tests may be specified in the instruction. The contents of the X-Register specified by bits 12-15 of the instruction are compared algebraically with zero and the result is compared with the corresponding indicator bit in the instruction. If a correspondence exists, the next instruction is fetched from memory location y'. If no correspondence exists, the next sequential instruction is executed. The execution of the BXC instruction does not change either the X-Register or the S-Register. The indicator bit assignments for the instruction are the following:

Branch Condition	Indicator	Bit
(X) < 0	L, Less than	9
(X) > 0	G, Greater than	10
(X) = 0	E, Equal	11

BIC Branch Indicator Condition

Timing: 1/2 cycles

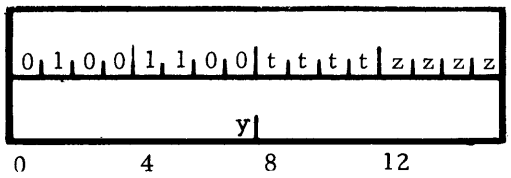


Any single test or combination of tests may be specified in the instruction after the indicator bits in the S-Register have been set/reset by the execution of a CAM, CBM, CXM, CRR, INC or LDS instruction. If a correspondence exists between any of the indicator bits of the instruction and the associated indicators set in the S-Register, then the next instruction is fetched from memory location y'. If no correspondence exists, the next sequential instruction is executed. When a branch on the overflow condition is executed, the overflow indicator in the S-Register is reset. The other indicators in the S-Register are not affected by the execution of the BIC instruction. The indicator bit assignments for the instruction are the following:

Indicator	Bit
O, Overflow	12
L, Less	13
G, Greater	14
E, Equal	15

BST Branch Sense Switch True

Timing: 1/2 cycles

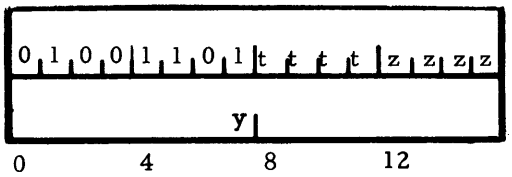


Any single test or combination of tests may be specified in the instruction. If any of the sense switch bits in the instruction are set (equal 1) and the associated sense switches on the console are set, then the next instruction is fetched from memory location y'. Otherwise, the next sequential instruction is executed. The sense switch bit assignments in the instruction are the following:

Sense Switches	Bit
8, 4, 2, 1	12, 13, 14, 15

BSF Branch Sense Switch False

Timing: 1/2 cycles

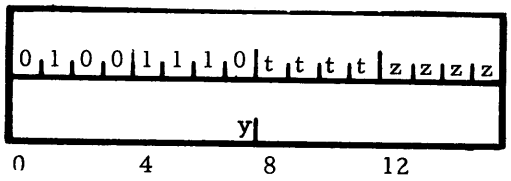


Any single test or combination of tests may be specified in the instruction. If any of the sense switch bits in the instruction are set (equal 1) and the associated sense switches on the console are not set, then the next instruction is fetched from memory location y'. Otherwise, the next sequential instruction is executed. The sense switch bit assignments in the instruction are the following:

Sense Switches	Bits
8, 4, 2, 1	12, 13, 14, 15

BFT Branch Program Flag True

Timing: 1/2 cycles

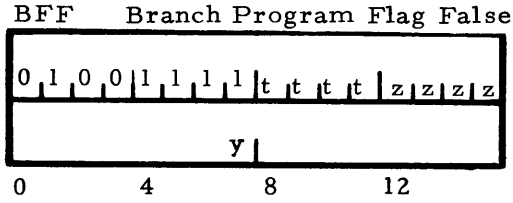


Any single test or combination of tests may be specified in the instruction. If any of the program flag bits in the instruction are set (equal 1) and the associated program flags in the S-Register are set, then the next instruction is fetched from memory location y'. Otherwise, the next sequential instruction is executed. The program flag bit assignments in the instruction and the S-Register are the following:

Program Flags	Bits
8, 4, 2, 1	12, 13, 14, 15

BFF Branch Program Flag False

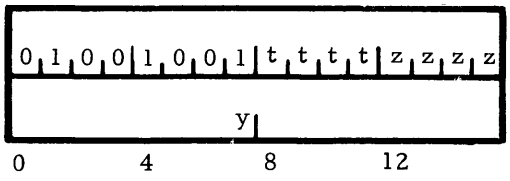
Timing: 1/2 cycles



Any single test or combination of tests may be specified in the instruction. If any of the program flag bits in the instruction are set (equal 1) and the associated program flags in the S-Register are not set, then the next instruction is fetched from memory location y'. Otherwise, the next sequential instruction is executed. The program flag bit assignments in the instruction and the S-Register are the following:

Program Flags	Bits
8, 4, 2, 1	12, 13, 14, 15

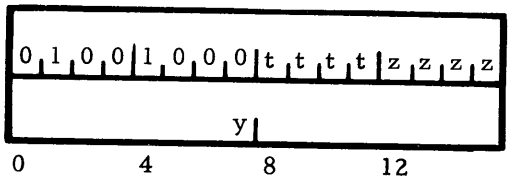
BRE Branch and Reset External Interrupt Timing: 2 cycles



The interrupt latch specified by bits 12-15 of the instruction is reset. Then the memory address y' is placed in the P-Register, allowing complete memory addressing capability for the fetch of the next instruction.

If the Read Only Memory option is implemented, this instruction will operate as described only when an interrupt is being serviced. At any other time the ROM mode flag will be reset after which the memory address y' is placed in the P-Register, allowing complete memory addressing capability for the fetch of the next instruction.

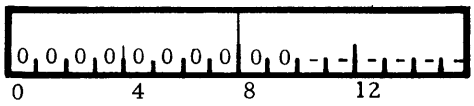
BRI Branch and Reset Internal Interrupt Timing: 2 cycles



The internal interrupt latch specified by bits 12-15 of the instruction is reset. Then the memory address y' is placed in the P-Register, allowing complete memory addressing capability for the fetch of the next instruction.

6.11 CONTROL INSTRUCTIONS

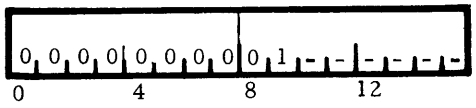
HLT Halt



Timing: 1 cycle

The computer stops with the P-Register specifying the next sequential address. To resume execution of the program, press the RUN button or the HALT and STEP combination. In a multiprocessing or real-time system this instruction may be wired to cause an interrupt rather than stopping processing.

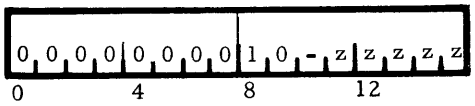
WAT Wait



Timing: 1 cycle

The computer stops with the P-Register specifying the next sequential address. Program execution is resumed if an internal or external interrupt occurs and if that interrupt is enabled. If an interrupt does not occur, program execution can be resumed by pressing the RUN button or the HALT and STEP combination.

RPF Reset Program Flag

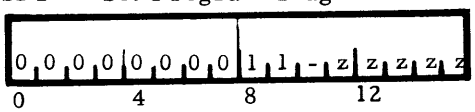


Timing: 1 cycle

The program flags in the S-Register specified by one's in the correspondoning bit positions of the instruction are reset to zero. The program flag bit assignments for both the instruction and the S-Register are the following:

Program Flags	Bits
Overflow, 8, 4, 2, 1	11, 12, 13, 14, 15

SPF Set Program Flag

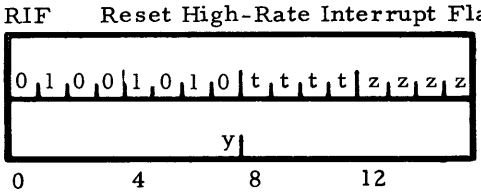


Timing: 1 cycle

The program flags in the S-Register specified by one's in the corresponding bit positions of the instruction are set to one. The program flag bit assignments for both the instruction and the S-Register are the following:

Program Flags	Bits
Overflow, 8, 4, 2, 1	11, 12, 13, 14, 15

RIF Reset High-Rate Interrupt Flag

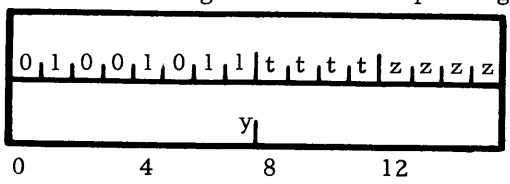


Timing: 3 cycles

The high-rate interrupt flags in the S-Register specified by one's in the corresponding bit positions of the contents of memory location y', which remain unchanged, are reset to zero. The high-rate interrupt flag bit assignments for both the instruction and the S-Register are the following:

High-Rate Interrupt Flags	Bits
0 to 6	0 to 6

SIF Set High-Rate Interrupt Flag



Timing: 3 cycles

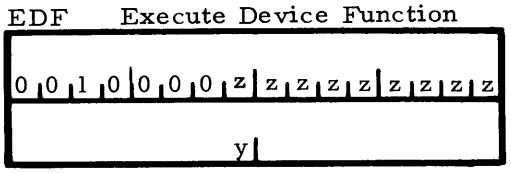
The high-rate interrupt flags in the S-Register specified by one's in the corresponding bit positions of the contents of memory location y', which remain unchanged, are set to one. The high-rate interrupt flag bit assignments for both the instruction and the S-Register are the following:

High-Rate Interrupt Flags	Bits
0 to 6	0 to 6

6.12 INPUT/OUTPUT INSTRUCTIONS

EDF Execute Device Function

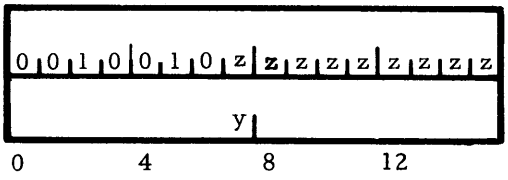
Timing: 1.5 μs



The action specified by bits 7-9 of the instruction is performed by the device specified by bits 10-15. In addition to bits 7-9 of the instruction, the A-Register is available to the controller for further instruction coding, providing a total of 19-bits for specifying an action by an external device. If the EDF command is rejected, a similated BSP to memory location y is affected.

WTI Word Transfer In

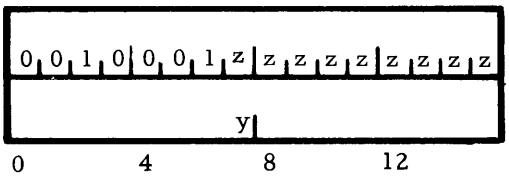
Timing: 1.5 μs



One word (up to 16-bits of information) is transferred into the A-Register from the device specified by bits 10-15 of the instruction. Bits 7-9 represent the order code and are available for special coding at the system level. The interrupt line in the controller specified by bits 10-15 is reset. If the WTI command is rejected, a simulated BSP operation occurs to memory location y.

WTO Word Transfer Out

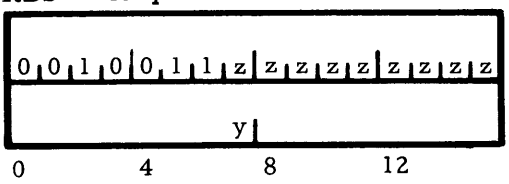
Timing: 1.5 μs



One word (up to 16-bits of information) is transferred from the A-Register to the device specified by bits 10-15 of the instruction. Bits 7-9 represent the order code and are available for special coding at the system level. The interrupt line in the controller specified by bits 10-15 is reset. If the WTO command is rejected, a simulated BSP operation occurs to memory location y.

RDS Request Device Status

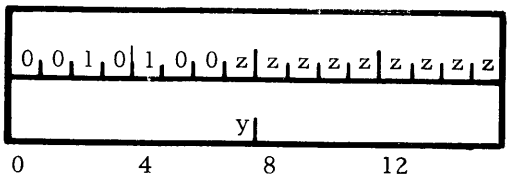
Timing: 1.5 μs



The status of the device specified by bits 10-15 is transferred into the A-Register. If the RDS command is rejected, a simulated BSP operation occurs to memory location y. Bits 7-9 represent the order code and are available for special coding at the system level; if not used, these bits should be zero.

ICI Interrogate Common Interrupts

Timing: 1.5 μs

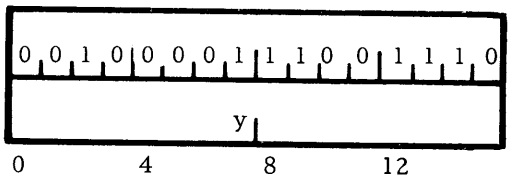


The service-required status of the devices (up to 16) that share a common interrupt line is transferred into the A-Register. Device $XX0000_2$ (bits 1.0-15) corresponds to bit position zero, and each succeeding device up to $XX1111_2$ corresponds to each succeeding bit position up to 15. A one in any of the 16-bit positions indicates that the corresponding device requires service. Bits 7-9 represent the order code and are available for special coding at the system level; if not used, these bits should be zero. If the ICI command is rejected, a simulated BSP operation occurs to memory location y.

6.13 SPECIAL INSTRUCTIONS

ICF Internal Control Function

Timing: 1.5 μs



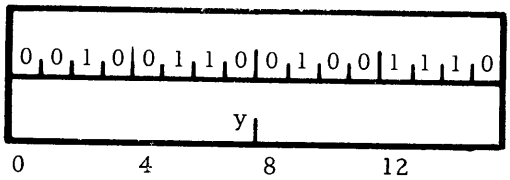
The following CPU internal states are set or reset dependent on whether the corresponding bit position of the A-Register contents is a one or zero, respectively. The second word of the instruction is not used. Setting any of the interrupts will produce the same CPU states normally produced by that interrupt, but will not in itself cause an interrupt to occur.

A-Register Bit	CPU Internal State
0	Power Fail State*
1	Parity Interrupt*
2	Parity Disable
3	Instruction Trap Interrupt*
4	Memory Protect Interrupt*
5	Privileged Instruction Interrupt*
6	System Interrupt*
7	System Interrupt Disable
8	Master Mode Indicator
9-15	Spare

*This internal state cannot be set/reset; it provides status only.

RIS Request Internal Status

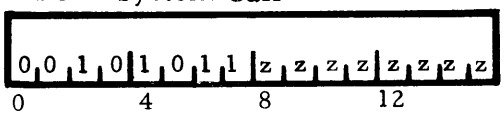
Timing: 1.5 μs



The status of the following CPU internal states are transferred into the A-Register. The second word of the instruction is not used.

A-Register Bit	CPU Internal State
0	Power Fail State
1	Parity Interrupt
2	Parity Disable
3	Instruction Trap Interrupt
4	Memory Protect Interrupt
5	Privileged Instruction Interrupt
6	System Interrupt
7	System Interrupt Disable
8	Master Mode Indicator
9-15	Spare

SYC System Call



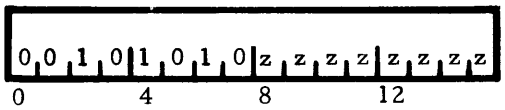
The system call number specified by bits 10-15 of the instruction is added to the fixed address 0080 to specify the address of the location (system call pointer) which contains the address of the first location of the system call subroutine. A simulated BSP to the first location of the system call subrout ne is effected, resulting in the following register stores:

System Call Subroutine

Location	Register
1st	P
2nd	A
3rd	B
4th	S
5th	X12
6th	X13
7th	X14
8th	X15

The master mode flag in the S register is set and the first instruction to be executed is taken from the ninth location of the system call subroutine. Bits 8 and 9 are available for special coding at the system level; if not used, these bits should be zero.

SYR System Return



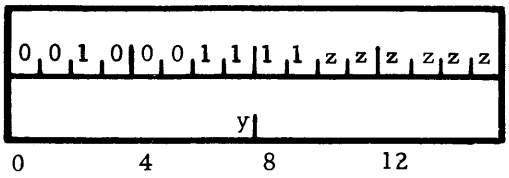
The system call number specified by bits 10-15 of the instruction is added to the fixed address 0080 to specify the address of the location (system call pointer) which contains the address of the first location of the system call subroutine. The contents of the first eight locations of the system call subroutine are restored in the registers as specified below:

System Call Subroutine

Location	Register
1st	P
2nd	A
3rd	B
4th	S
5th	X12
6th	X13
7th	X14
8th	X15

The master mode flag in the S register is reset. The next instruction to be executed is located in the user program and is specified by the P register value stored in the first location of the system call subroutine (normally the instruction immediately following the SYC instruction). Bits 8 and 9 are available for special coding at the system level; if not used, these bits should be zero.

SPM Set Protect Mask



The memory protect mask is transferred from the A register to the MCU (memory control unit) specified by bits 10-15 of the instruction. For a single processor system bits 10-15 equal 14_{10}. The format of the memory protect mask is the following:

Bit	Function
0-1	Bank (65k) Address
2-8	Upper Limit Address 512≤ULA≤65k
9-15	Lower Limit Address 512≤LLA≤65k

If the SPM command is rejected, a simulated BSP operation occurs to memory location Y.

CHAPTER SEVEN
PROGRAMMING AND OPERATION

TEMPO I is a general-purpose, stored program computer. Its operation is governed essentially by a series of program instructions (see preceding section) stored sequentially in the primary core memory.

Once the starting address for the desired program is entered into the P-Register, a simple depression of the RUN button will cause the computer to execute the first program instruction, fetch and execute the next instruction in sequence, and continue through to the final instruction.

Interrupts occasioned by a peripheral desiring to enter data will cause the computer to branch to the original program at the point where it was interrupted.

The "operation" of the TEMPO I computer is therefore largely the task of writing an effective program that makes maximum use of the instructions available. The mechanics of writing such a program are simplified by special programs, already developed and available from Tempo, which can be entered into the computer and used to translate Tempo-mnemonics or Fortran terminology into the TEMPO I machine language. With these "assembly" and "compiler" programs in the computer, instructions and operands can be entered as "data input" and a binary-coded instruction list received as a "data output". This list, normally recorded on punched tape, is then fed back into the computer to be executed.

Most of the front panel controls relate, therefore, to loading the program, checking its status, and when necessary testing or troubleshooting the program and/or the computer hardware.

The Console allows the operator to control and examine virtually every element in the computer through a combination of switches and indicator lights.

The following subsection details the operation of the console.

7.1 OPERATION OF CONSOLE

The various switches and indicator lights shown in Figure 5.2 are used to perform the following functions when operating the Console:

Data Entry ... requires that the CPU be in a HALT state. The appropriate register is selected by raising a REGISTER SELECT switch. The CLEAR switch is depressed clearing the selected register to all "zeros." The "ones" to be inserted are entered by depressing the corresponding DATA INSERT switch. The lights of the DATA DISPLAY continuously display the selected register, and as the Clear and Data Insert switches are depressed the results can be observed. If a mistake in inserted ones is made the Clear must be depressed and all the ones in the data word must be reinserted. Note that the switches to the left of a switch in the Register Select bank have "priority" and, if raised, will override all switches to the right.

Program Load ... requires three sets of information: a) the starting address of the memory block to be loaded (X_{even}); b) the count of words or bytes to be loaded and whether it is byte or word transfer (X_{odd}); and c) the device address of the device to be used to load the program.

Items (a) and (b) are loaded into the selected X-Register pair as described in Data Entry above. Item (c) is loaded into the low order 6-bits (bits 10 through 15) of the I-Register using the Data Entry procedure. The state of the high order 10-bits of the I-Register are not used and may be any state.

The device selected must be prepared to receive an EDF command (this may require a system reset before program load). If the Teletype is the selected device the paper tape is loaded into the reader, the Teletype is put in On-Line mode, and the Reader switch is put in the Start position. The operator should make sure that the N-Register is set to allow interrupts from the Teletype or any other selected device. If any device other than the Teletype is selected, the operator must also load the associated interrupt trap location.

The depression of the PROGRAM LOAD switch selects the device specified by the low order 6-bits of the I-Register, initiates the device via the Program Load line of the I/O bus and transfers the program into the block of memory specified by the X-Register pair. The LOAD light will come on with the depression of the Program Load switch but will be turned off when the X_{odd} count overflows indicating a successful load.

Starting a Program ... requires that the CPU be in the HALT state. The REGISTER SELECT switch is positioned to P and the desired program counter start condition is entered (see Data Entry above). The RUN switch is then depressed.

Single Instruction ... is accomplished by the procedure for starting a program, but instead of depressing the RUN switch, select the Step Mode by pushing the HALT switch. With each push of the HALT switch a single instruction is executed. During the single instruction operation the HALT light remains on.

Register Display ... requires that the computer be in the HALT condition. If it is not, depress the HALT switch. Select the register to be displayed by depressing the desired REGISTER SELECT switch and observe the DATA DISPLAY lights. Note that if one or more Register Select switches are selected the left-most one will be displayed.

Read or Write Memory ... allows any number of words to be read from or written into memory. Set the desired starting address into P as described in Data Entry above. Next select the M-Register and place the READ/WRITE switch in the READ position. Each time the INITIATE switch is depressed the P counter will address the memory and bring one word to the M-Register. The value read from memory can be observed in the DATA DISPLAY indicators. Each time the Initiate switch is pushed the program counter is also incremented, thus a block of memory may be read one word at a time.

In order to write into memory, select the WRITE mode via the READ/WRITE switch. Enter the desired address into the program counter as described above, then re-select the M-Register. Enter data into M through the DATA INSERT and CLEAR switches and depress the INITIATE switch. The data in M will be written into the memory address provided by the program counter which is incremented with each depression of the Initiate switch. A repeated sequence of loading the input data into M and pressing the Initiate switch will load a block of data into memory.

Read or Write Register Memory ... takes advantage of the fact that the contents of any X-Register can be displayed or altered by means of controls on the console. Outputs from the Register Memory communicate with the remainder of the system via the D-Register. So the selection of RM is a selection of the D-Register. Read and write into a selected X-Register therefore requires a transfer to and from the D-Register. The front panel controls for the Register Memory are labeled "RM."

To display the contents of a particular X-Register, it must be selected by inserting the appropriate binary code into the RM ADDRESS switches and selecting the RM and READ switches. When the INITIATE switch is then depressed the contents of the selected X-Register will be loaded into the D-Register. If the D-Register has been selected by the Register Select switch (RM position), then the contents of it (and therefore the contents of the selected X-Register) will be displayed in the lights of the DATA DISPLAY.

To alter the contents of the Register Memory a particular X-Register must be selected by inserting the appropriate binary code into the RM ADDRESS switches and selecting the RM-WRITE mode. The REGISTER SELECT switch is then placed in the RM position and data is entered into the D-Register through the use of the CLEAR and DATA INSERT switches. The contents of the D-Register will then be transferred to the selected X-Register when the INITIATE switch is depressed.

Basic Bootstrap ... is a technique for loading a program into a computer without the Program Load option. The Teletype, or other selected input device, is used to enter the program after a manual bootstrap entry is made via the Memory Write operation described above. The bootstrap program is a series of instructions for entering data from the selected input device.

APPENDIX A
TABLE OF POWERS OF TWO

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

OCTAL-DECIMAL INTEGER CONVERSION TABLE

0000 to 0777 (Octal)	0000 to 0511 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

1000 to 1777 (Octal)	0512 to 1023 (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

APPENDIX B
HEXADECIMAL/DECIMAL CONVERSION

B-1. HEXADECIMAL/DECIMAL CONVERSION CHART—INTEGERS

Table D-1 on the following pages is used to convert hexadecimal numbers to decimal and decimal numbers to hexadecimal. In the descriptions that follow, the explanation of each step is followed by an example in parentheses.

a. Hexadecimal to Decimal Conversion. - Locate the first two digits (38) of the hexadecimal number (387) in the left column. Follow the line of figures across the page to the column headed by the low order digit (7). The decimal number (0951) located at the junction of the horizontal line and the vertical column is the equivalent of the hexadecimal number.

b. Decimal to Hexadecimal Conversion. - Locate the decimal number (0951) in the body of the table. The two high order digits (38) of the hexadecimal number are in the left column on the same line, and the low order digit (7) is at the top of the column. Thus, the hexadecimal number 387 is equal to the decimal number 0951.

c. Extended Hexadecimal. - The table to the left gives the decimal, binary and hexadecimal coding for the full range of four binary bits, from zero through F_{16} and 15_{10}. To convert a four-digit hexadecimal number to decimal, determine the decimal value of the three low order hexadecimal digits in the main table, and add the value for the high order digit, as shown in the extended chart to the right.

For conversion of decimal values beyond the main table, deduct the largest number in the table at the right that will yield a positive result. The related digit is the high order hexadecimal digit. Determine the three remaining hexadecimal digits by converting the product of the above subtraction of the main table.

Dec	Bin	Hex	Dec	Bin	Hex
0	0000	0	8	1000	8
1	0001	1	9	1001	7
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

Hex	Dec	Hex	Dec
1000	4096	9000	36864
2000	8192	A000	40960
3000	12288	B000	45056
4000	16384	C000	49152
5000	20480	D000	53248
6000	24576	E000	57344
7000	28672	F000	61440
8000	32768		

APPENDIX B-1 HEXADECIMAL/DECIMAL CONVERSION—INTEGERS

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00 _	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
01 _	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
02 _	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
03 _	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
04 _	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
05 _	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
06 _	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
07 _	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
08 _	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
09 _	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A _	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B _	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C _	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D _	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E _	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F _	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255
10 _	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
11 _	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
12 _	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
13 _	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
14 _	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
15 _	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
16 _	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
17 _	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
18 _	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
19 _	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A _	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B _	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C _	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D _	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E _	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F _	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511
20 _	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
21 _	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
22 _	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
23 _	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
24 _	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
25 _	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
26 _	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
27 _	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
28 _	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
29 _	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A _	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B _	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C _	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D _	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E _	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F _	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
30 _	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
31 _	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
32 _	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
33 _	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
34 _	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
35 _	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
36 _	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
37 _	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
38 _	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
39 _	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A _	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B _	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C _	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D _	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E _	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F _	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

Table B-1. Hexadecimal/Decimal Convertion – Integers (Continued)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
40 _	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
41 _	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
42 _	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
43 _	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
44 _	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
45 _	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
46 _	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
47 _	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
48 _	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
49 _	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A _	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B _	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C _	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D _	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E _	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F _	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
50 _	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
51 _	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
52 _	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
53 _	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
54 _	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
55 _	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
56 _	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
57 _	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
58 _	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
59 _	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A _	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B _	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C _	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D _	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E _	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F _	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535
60 _	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
61 _	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
62 _	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
63 _	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
64 _	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
65 _	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
66 _	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
67 _	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
68 _	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
69 _	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A _	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B _	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C _	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D _	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E _	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F _	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791
70 _	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
71 _	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
72 _	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
73 _	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
74 _	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
75 _	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
76 _	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
77 _	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
78 _	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
79 _	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A _	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B _	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C _	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D _	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E _	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F _	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047

Table B-1. Hexadecimal/Decimal Conversion – Integers (Continued)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
80 _	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
81 _	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
82 _	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
83 _	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
84 _	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
85 _	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
86 _	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
87 _	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
88 _	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
89 _	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A _	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B _	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C _	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D _	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E _	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F _	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
90 _	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
91 _	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
92 _	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
93 _	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
94 _	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
95 _	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
96 _	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
97 _	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
98 _	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
99 _	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A _	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B _	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C _	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D _	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E _	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F _	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559
A0 _	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A1 _	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A2 _	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A3 _	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A4 _	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A5 _	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A6 _	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A7 _	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A8 _	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A9 _	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA _	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB _	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC _	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD _	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE _	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF _	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B0 _	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B1 _	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B2 _	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B3 _	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B4 _	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B5 _	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B6 _	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B7 _	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B8 _	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B9 _	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA _	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB _	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC _	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD _	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE _	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF _	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071

Table B-1. Hexadecimal/Decimal Conversion – Integers (Continued)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C0_	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C1_	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C2_	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C3_	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C4_	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C5_	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C6_	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C7_	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C8_	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C9_	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA_	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB_	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC_	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD_	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE_	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF_	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
D0_	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D1_	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D2_	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D3_	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D4_	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D5_	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D6_	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D7_	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D8_	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D9_	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA_	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB_	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC_	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD_	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE_	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF_	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583
E0_	3584	3585	3586	3587	3583	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E1_	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E2_	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E3_	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E4_	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E5_	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E6_	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E7_	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E8_	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E9_	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA_	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB_	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC_	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED_	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE_	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF_	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F0_	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F1_	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F2_	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F3_	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F4_	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F5_	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F6_	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F7_	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F8_	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F9_	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA_	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB_	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC_	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD_	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE_	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF_	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

B-2. HEXADECIMAL/DECIMAL CONVERSION CHART—FRACTIONS

Table D-2 on the following five pages may be used for converting fractions.

a. Decimal-to-Hexadecimal. - Locate the decimal fraction (0. 1534) in the table. If the exact figure is not shown, locate the next higher and lower fractions (0. 15332031), (0. 15356445). The first two digits of the hexadecimal fraction are at the top of the column (0. 27). To locate the third digit, determine by observation or subtraction the smaller difference between the known fraction and each of the found fractions. The smaller difference identifies the correct line (0. 004). The hexadecimal equivalent is 0. 274. If the hexadecimal fraction is required to more places, multiply the decimal fraction by 16 and develop integers as successive terms of the hexadecimal fraction. Using the previous sample decimal fraction:

$$
\begin{array}{r}
0.1534 \\
\underline{\times 16} \\
2.4544 \\
\underline{\times 16} \\
7.2704 \\
\underline{\times 16} \\
4.3264 \\
\underline{\times 16} \\
5.2224
\end{array}
\qquad 0.1534_{10} = 0.2745_{16}
$$

b. Hexadecimal-to-Decimal Conversion. - Locate the first two digits (0. 27) of the hexadecimal fraction (0. 274) in the horizontal row of column headings. Locate the third digit (0. 004) in the leftmost column of the table. Follow the 0. 004 line horizontally to the right to the 0. 27 column. The decimal equivalent is 0. 15332031. The decimal fractions in the table were carried to eight places and rounded. If 12 places are required, or if the hexadecimal fraction exceeds the capacity of the table, express the hexadecimal fraction as powers of 16 (expansion). For example:

$$
\begin{aligned}
0.2745_{16} &= 2(16^{-1}) + 7(16^{-2}) + 4(16^{-3}) + 5(16^{-4}) \\
&= 2(0.0625) + 7(0.00390625) + 4(0.000244140625) + 5(0.0000152587890625) \\
&= 0.1533966064453125_{10}
\end{aligned}
$$

TABLE B-2. HEXADECIMAL/DECIMAL CONVERSION — FRACTIONS

	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09	.0A	.0B	.0C	.0D	.0E	.0F
.000	.00000000	.00390625	.00781250	.01171875	.01562500	.01953125	.02343750	.02734375	.03125000	.03515625	.03906250	.04296875	.04687500	.05078125	.05468750	.05859375
.001	.00024414	.00415039	.00805664	.01196289	.01586914	.01977539	.02368164	.02758789	.03149414	.03540039	.03930664	.04321289	.04711914	.05102539	.05493164	.05883789
.002	.00048828	.00439453	.00830078	.01220703	.01611328	.02001953	.02392578	.02783203	.03173828	.03564453	.03955078	.04345703	.04736328	.05126953	.05517578	.05908203
.003	.00073242	.00463867	.00854492	.01245117	.01635742	.02026367	.02416992	.02807617	.03198242	.03588867	.03979492	.04370117	.04760742	.05151367	.05541992	.05932617
.004	.00097656	.00488281	.00878906	.01269531	.01660156	.02050781	.02441406	.02832031	.03222656	.03613281	.04003906	.04394531	.04785156	.05175781	.05566406	.05957031
.005	.00122070	.00512695	.00903320	.01293945	.01684570	.02075195	.02465820	.02856445	.03247070	.03637695	.04028320	.04418945	.04809570	.05200195	.05590820	.05981445
.006	.00146484	.00537109	.00927734	.01318359	.01708984	.02099609	.02490234	.02880859	.03271484	.03662109	.04052734	.04443359	.04833984	.05224609	.05615234	.06005859
.007	.00170898	.00561523	.00952148	.01342773	.01733398	.02124023	.02514648	.02905273	.03295898	.03686523	.04077148	.04467773	.04858398	.05249023	.05639648	.06030273
.008	.00195313	.00585938	.00976563	.01367188	.01757813	.02148438	.02539063	.02929688	.03320313	.03710938	.04101563	.04492188	.04882813	.05273438	.05664063	.06054688
.009	.00219727	.00610352	.01000977	.01391602	.01782227	.02172852	.02563477	.02954102	.03344727	.03735352	.04125977	.04516602	.04907227	.05297852	.05688477	.06079102
.00A	.00244141	.00634766	.01025391	.01416016	.01806641	.02197266	.02587891	.02978516	.03369141	.03759766	.04150391	.04541016	.04931641	.05322266	.05712891	.06103516
.00B	.00268555	.00659180	.01049805	.01440430	.01831055	.02221680	.02612305	.03002930	.03393555	.03784180	.04174805	.04565430	.04956055	.05346680	.05737305	.06127930
.00C	.00292969	.00683594	.01074219	.01464844	.01855469	.02246094	.02636719	.03027344	.03417969	.03808594	.04199219	.04589844	.04980469	.05371094	.05761719	.06152344
.00D	.00317383	.00708008	.01098633	.01489258	.01879883	.02270508	.02661133	.03051758	.03442383	.03833008	.04223633	.04614258	.05004883	.05395508	.05786133	.06176758
.00E	.00341797	.00732422	.01123047	.01513672	.01904297	.02294922	.02685547	.03076172	.03466797	.03857422	.04248047	.04638672	.05029297	.05419922	.05810547	.06201172
.00F	.00366211	.00756836	.01147461	.01538086	.01928711	.02319336	.02709961	.03100586	.03491211	.03881836	.04272461	.04663086	.05053711	.05444336	.05834961	.06225586

	.10	.11	.12	.13	.14	.15	.16	.17	.18	.19	.1A	.1B	.1C	.1D	.1E	.1F
.000	.06250000	.06640625	.07031250	.07421875	.07812500	.08203125	.08593750	.08984375	.09375000	.09765625	.10156250	.10546875	.10937500	.11328125	.11718750	.12109375
.001	.06274414	.06665039	.07055664	.07446289	.07836914	.08227539	.08618164	.09008789	.09399414	.09790039	.10180664	.10571289	.10961914	.11352539	.11743164	.12133789
.002	.06298828	.06689453	.07080078	.07470703	.07861328	.08251953	.08642578	.09033203	.09423828	.09814453	.10205078	.10595703	.10986328	.11376953	.11767578	.12158203
.003	.06323242	.06713867	.07104492	.07495117	.07885742	.08276367	.08666992	.09057617	.09448242	.09838867	.10229492	.10620117	.11010742	.11401367	.11791992	.12182617
.004	.06347656	.06738281	.07128906	.07519531	.07910156	.08300781	.08691406	.09082031	.09472656	.09863281	.10253906	.10644531	.11035156	.11425781	.11816406	.12207031
.005	.06372070	.06762695	.07153320	.07543945	.07934570	.08325195	.08715820	.09106445	.09497070	.09887695	.10278320	.10668945	.11059570	.11450195	.11840820	.12231445
.006	.06396484	.06787109	.07177734	.07568359	.07958984	.08349609	.08740234	.09130859	.09521484	.09912109	.10302734	.10693359	.11083984	.11474609	.11865234	.12255859
.007	.06420898	.06811523	.07202148	.07592773	.07983398	.08374023	.08764648	.09155273	.09545898	.09936523	.10327148	.10717773	.11108398	.11499023	.11889648	.12280273
.008	.06445313	.06835938	.07226563	.07617188	.08007813	.08398438	.08789063	.09179688	.09570313	.09960938	.10351563	.10742188	.11132813	.11523438	.11914063	.12304688
.009	.06469727	.06860352	.07250977	.07641602	.08032227	.08422852	.08813477	.09204102	.09594727	.09985352	.10375977	.10766602	.11157227	.11547852	.11938477	.12329102
.00A	.06494141	.06884766	.07275391	.07666016	.08056641	.08447266	.08837891	.09228516	.09619141	.10009766	.10400391	.10791016	.11181641	.11572266	.11962891	.12353516
.00B	.06518555	.06909180	.07299805	.07690430	.08081055	.08471680	.08862305	.09252930	.09643555	.10034180	.10424805	.10815430	.11206055	.11596680	.11987305	.12377930
.00C	.06542969	.06933594	.07324219	.07714844	.08105469	.08496094	.08886719	.09277344	.09667969	.10058594	.10449219	.10839844	.11230469	.11621094	.12011719	.12402344
.00D	.06567383	.06958008	.07348633	.07739258	.08129883	.08520508	.08911133	.09301758	.09692383	.10083008	.10473633	.10864258	.11254883	.11645508	.12036133	.12426758
.00E	.06591797	.06982422	.07373047	.07763672	.08154297	.08544922	.08935547	.09326172	.09716797	.10107422	.10498047	.10888672	.11279297	.11669922	.12060547	.12451172
.00F	.06616211	.07006836	.07397461	.07788086	.08178711	.08569336	.08959961	.09350586	.09741211	.10131836	.10522461	.10913086	.11303711	.11694336	.12084961	.12475586

	.20	.21	.22	.23	.24	.25	.26	.27	.28	.29	.2A	.2B	.2C	.2D	.2E	.2F
.000	.12500000	.12890625	.13281250	.13671875	.14062500	.14453125	.14843750	.15234375	.15625000	.16015625	.16406250	.16796875	.17187500	.17578125	.17968750	.18359375
.001	.12524414	.12915039	.13305664	.13696289	.14086914	.14477539	.14868164	.15258789	.15649414	.16040039	.16430664	.16821289	.17211914	.17602539	.17993164	.18383789
.002	.12548828	.12939453	.13330078	.13720703	.14111328	.14501953	.14892578	.15283203	.15673828	.16064453	.16455078	.16845703	.17236328	.17626953	.18017578	.18408203
.003	.12573242	.12963867	.13354492	.13745117	.14135742	.14526367	.14916992	.15307617	.15698242	.16088867	.16479492	.16870117	.17260742	.17651367	.18041992	.18432617
.004	.12597656	.12988281	.13378906	.13769531	.14160156	.14550781	.14941406	.15332031	.15722656	.16113281	.16503906	.16894531	.17285156	.17675781	.18066406	.18457031
.005	.12622070	.13012695	.13403320	.13793945	.14184570	.14575195	.14965820	.15356445	.15747070	.16137695	.16528320	.16918945	.17309570	.17700195	.18090820	.18481445
.006	.12646484	.13037109	.13427734	.13818359	.14208984	.14599609	.14990234	.15380859	.15771484	.16162109	.16552734	.16943359	.17333984	.17724609	.18115234	.18505859
.007	.12670898	.13061523	.13452148	.13842773	.14233398	.14624023	.15014648	.15405273	.15795898	.16186523	.16577148	.16967773	.17358398	.17749023	.18139648	.18530273
.008	.12695313	.13085938	.13476563	.13867188	.14257813	.14648438	.15039063	.15429688	.15820313	.16210938	.16601563	.16992188	.17382813	.17773438	.18164063	.18554688
.009	.12719727	.13110352	.13500977	.13891602	.14282227	.14672852	.15063477	.15454102	.15844727	.16235352	.16625977	.17016602	.17407227	.17797852	.18188477	.18579102
.00A	.12744141	.13134766	.13525391	.13916016	.14306641	.14697266	.15087891	.15478516	.15869141	.16259766	.16650391	.17041016	.17431641	.17822266	.18212891	.18603516
.00B	.12768555	.13159180	.13549805	.13940430	.14331055	.14721680	.15112305	.15502930	.15893555	.16284180	.16674805	.17065430	.17456055	.17846680	.18237305	.18627930
.00C	.12792969	.13183594	.13574219	.13964844	.14355469	.14746094	.15136719	.15527344	.15917969	.16308594	.16699219	.17089844	.17480469	.17871094	.18261719	.18652344
.00D	.12817383	.13208008	.13598633	.13989258	.14379883	.14770508	.15161133	.15551758	.15942383	.16333008	.16723633	.17114258	.17504883	.17895508	.18286133	.18676758
.00E	.12841797	.13232422	.13623047	.14013672	.14404297	.14794922	.15185547	.15576172	.15966797	.16357422	.16748047	.17138672	.17529297	.17919922	.18310547	.18701172
.00F	.12866211	.13256836	.13647461	.14038086	.14428711	.14819336	.15209961	.15600586	.15991211	.16381836	.16772461	.17163086	.17553711	.17944336	.18334961	.18725586

Table B-2. Decimal-to-Hexadecimal Conversion – Fractions (Continued)

	.30	.31	.32	.33	.34	.35	.36	.37	.38	.39	.3A	.3B	.3C	.3D	.3E	.3F
.000	.18750000	.19140625	.19531250	.19921875	.20312500	.20703125	.21093750	.21484375	.21875000	.22265625	.22656250	.23046875	.23437500	.23828125	.24218750	.24609375
.001	.18774414	.19165039	.19555664	.19946289	.20336914	.20727539	.21118164	.21508789	.21899414	.22290039	.22680664	.23071289	.23461914	.23852539	.24243164	.24633789
.002	.18798828	.19189453	.19580078	.19970703	.20361328	.20751953.	.21142578	.21533203	.21923828	.22314453	.22705078	.23095703	.23486328	.23876953	.24267578	.24658203
.003	.18823242	.19213867	.19604492	.19995117	.20385742	.20776367	.21166992	.21557617	.21948242	.22338867	.22729492	.23120117	.23510742	.23901367	.24291992	.24682617
.004	.18847656	.19238281	.19628906	.20019531	.20410156	.20800781	.21191406	.21582031	.21972656	.22363281	.22753906	.23144531	.23535156	.23925781	.24316406	.24707031
.005	.18872070	.19262695	.19653320	.20043945	.20434570	.20825195	.21215820	.21606445	.21997070	.22387695	.22778320	.23168945	.23559570	.23950195	.24340820	.24731445
.006	.18896484	.19287109	.19677734	.20068359	.20458984	.20849609	.21240234	.21630859	.22021484	.22412109	.22802734	.23193359	.23583984	.23974609	.24365234	.24755859
.007	.18920898	.19311523	.19702148	.20092773	.20483398	.20874023	.21264648	.21655273	.22045898	.22436523	.22827148	.23217773	.23608398	.23999023	.24389648	.24780273
.008	.18945313	.19335938	.19726563	.20117188	.20507813	.20898438	.21289063	.21679688	.22070313	.22460938	.22851563	.23242188	.23632813	.24023438	.24414063	.24804688
.009	.18969727	.19360352	.19750977	.20141602	.20532227	.20922852	.21313477	.21704102	.22094727	.22485352	.22875977	.23266602	.23657227	.24047852	.24438477	.24829102
.00A	.18994141	.19384766	.19775391	.20166016	.20556641	.20947266	.21337891	.21728516	.22119141	.22509766	.22900391	.23291016	.23681641	.24072266	.24462891	.24853516
.00B	.19018555	.19409180	.19799805	.20190430	.20581055	.20971680	.21362305	.21752930	.22143555	.22534180	.22924805	.23315430	.23706055	.24096680	.24487305	.24877930
.00C	.19042969	.19433594	.19824219	.20214844	.20605469	.20996094	.21386719	.21777344	.22167969	.22558594	.22949219	.23339844	.23730469	.24121094	.24511719	.24902344
.00D	.19067383	.19458008	.19848633	.20239258	.20629883	.21020508	.21411133	.21801758	.22192383	.22583008	.22973633	.23364258	.23754883	.24145508	.24536133	.24926758
.00E	.19091797	.19482422	.19873047	.20263672	.20654297	.21044922	.21435547	.21826172	.22216797	.22607422	.22998047	.23388672	.23779297	.24169922	.24560547	.24951172
.00F	.19116211	.19506836	.19897461	.20288086	.20678711	.21069336	.21459961	.21850586	.22241211	.22631836	.23022461	.23413086	.23803711	.24194336	.24584961	.24975586

	.40	.41	.42	.43	.44	.45	.46	.47	.48	.49	.4A	.4B	.4C	.4D	.4E	.4F
.000	.25000000	.25390625	.25781250	.26171875	.26562500	.26953125	.27343750	.27734375	.28125000	.28515625	.28906250	.29296875	.29687500	.30078125	.30468750	.30859375
.001	.25024414	.25415039	.25805664	.26196289	.26586914	.26977539	.27368164	.27758789	.28149414	.28540039	.28930664	.29321289	.29711914	.30102539	.30493164	.30883789
.002	.25048828	.25439453	.25830078	.26220703	.26611328	.27001953	.27392578	.27783203	.28173828	.28564453	.28955078	.29345703	.29736328	.30126953	.30517578	.30908203
.003	.25073242	.25463867	.25854492	.26245117	.26635742	.27026367	.27416992	.27807617	.28198242	.28588867	.28979492	.29370117	.29760742	.30151367	.30541992	.30932617
.004	.25097656	.25488281	.25878906	.26269531	.26660156	.27050781	.27441406	.27832031	.28222656	.28613281	.29003906	.29394531	.29785156	.30175781	.30566406	.30957031
.005	.25122070	.25512695	.25903320	.26293945	.26684570	.27075195	.27465820	.27856445	.28247070	.28637695	.29028320	.29418945	.29809570	.30200195	.30590820	.30981445
.006	.25146484	.25537109	.25927734	.26318359	.26708984	.27099609	.27490234	.27880859	.28271484	.28662109	.29052734	.29443359	.29833984	.30224609	.30615234	.31005859
.007	.25170898	.25561523	.25952148	.26342773	.26733398	.27124023	.27514648	.27905273	.28295898	.28686523	.29077148	.29467773	.29858398	.30249023	.30639648	.31030273
.008	.25195313	.25585938	.25976563	.26367188	.26757813	.27148438	.27539063	.27929688	.28320313	.28710938	.29101563	.29492188	.29882813	.30273438	.30664063	.31054688
.009	.25219727	.25610352	.26000977	.26391602	.26782227	.27172852	.27563477	.27954102	.28344727	.28735352	.29125977	.29516602	.29907227	.30297852	.30688477	.31079102
.00A	.25244141	.25634766	.26025391	.26416016	.26806641	.27197266	.27587891	.27978516	.28369141	.28759766	.29150390	.29541016	.29931641	.30322266	.30712891	.31103516
.00B	.25268555	.25659180	.26049805	.26440430	.26831055	.27221680	.27612305	.28002930	.28393555	.28784180	.29174805	.29565430	.29956055	.30346680	.30737305	.31127930
.00C	.25292969	.25683594	.26074219	.26464844	.26855469	.27246094	.27636719	.28027344	.28417969	.28808594	.29199219	.29589844	.29980469	.30371094	.30761719	.31152344
.00D	.25317383	.25708008	.26098633	.26489258	.26879883	.27270508	.27661133	.28051758	.28442383	.28833008	.29223633	.29614258	.30004883	.30395508	.30786133	.31176758
.00E	.25341797	.25732422	.26123047	.26513672	.26904297	.27294922	.27685547	.28076172	.28466797	.28857422	.29248047	.29638672	.30029297	.30419922	.30810547	.31201172
.00F	.25366211	.25756836	.26147461	.26538086	.26928711	.27319336	.27709961	.28100586	.28491211	.28881836	.29272461	.29663086	.30053711	.30444336	.30834961	.31225586

	.50	.51	.52	.53	.54	.55	.56	.57	.58	.59	.5A	.5B	.5C	.5D	.5E	.5F
.000	.31250000	.31640625	.32031250	.32421875	.32812500	.33203125	.33593750	.33984375	.34375000	.34765625	.35156250	.35546875	.35937500	.36328125	.36718750	.37109375
.001	.31274414	.31665039	.32055664	.32446289	.32836914	.33227539	.33618164	.34008789	.34399414	.34790039	.35180664	.35571289	.35961914	.36352539	.36743164	.37133789
.002	.31298828	.31689453	.32080078	.32470703	.32861328	.33251953	.33642578	.34033203	.34423828	.34814453	.35205078	.35595703	.35986328	.36376953	.36767578	.37158203
.003	.31323242	.31713867	.32104492	.32495117	.32885742	.33276367	.33666992	.34057617	.34448242	.34838867	.35229492	.35620117	.36010742	.36401367	.36791992	.37182617
.004	.31347656	.31738281	.32128906	.32519531	.32910156	.33300781	.33691406	.34082031	.34472656	.34863281	.35253906	.35644531	.36035156	.36425781	.36816406	.37207031
.005	.31372070	.31762695	.32153320	.32543945	.32934570	.33325195	.33715820	.34106445	.34497070	.34887695	.35278320	.35668945	.36059570	.36450195	.36840820	.37231445
.006	.31396484	.31787109	.32177734	.32568359	.32958984	.33349609	.33740234	.34130859	.34521484	.34912109	.35302734	.35693359	.36083984	.36474609	.36865234	.37255859
.007	.31420898	.31811523	.32202148	.32592773	.32983398	.33374023	.33764648	.34155273	.34545898	.34936523	.35327148	.35717773	.36108398	.36499023	.36889648	.37280273
.008	.31445313	.31835938	.32226563	.32617188	.33007813	.33398438	.33789063	.34179688	.34570313	.34960938	.35351563	.35742188	.36132813	.36523438	.36914063	.37304688
.009	.31469727	.31860352	.32250977	.32641602	.33032227	.33422852	.33813477	.34204102	.34594727	.34985352	.35375977	.35766602	.36157227	.36547852	.36938477	.37329102
.00A	.31494141	.31884766	.32275391	.32666016	.33056641	.33447266	.33837891	.34228516	.34619141	.35009766	.35400391	.35791016	.36181641	.36572266	.36962891	.37353516
.00B	.31518555	.31909180	.32299805	.32690430	.33081055	.33471680	.33862305	.34252930	.34643555	.35034180	.35424805	.35815430	.36206055	.36596680	.36987305	.37377930
.00C	.31542969	.31933594	.32324219	.32714844	.33105469	.33496094	.33886719	.34277344	.34667969	.35058594	.35449219	.35839844	.36230469	.36621094	.37011719	.37402344
.00D	.31567383	.31958008	.32348633	.32739258	.33129883	.33520508	.33911133	.34301758	.34692383	.35083008	.35473633	.35864258	.36254883	.36645508	.37036133	.37426758
.00E	.31591797	.31982422	.32373047	.32763672	.33154297	.33544922	.33935547	.34326172	.34716797	.35107422	.35498047	.35888672	.36279297	.36669922	.37060547	.37451172
.00F	.31616211	.32006836	.32397461	.32788086	.33178711	.33569336	.33959961	.34350586	.34741211	.35131836	.35522461	.35913086	.36303711	.36694336	.37084961	.37475586

Table B-2. Decimal-to-Hexadecimal Conversion – Fractions (Continued)

	.60	.61	.62	.63	.64	.65	.66	.67	.68	.69	.6A	.6B	.6C	.6D	.6E	.6F
.000	.37500000	.37890625	.38281250	.38671875	.39062500	.39453125	.39843750	.40234375	.40625000	.41015625	.41406250	.41796875	.42187500	.42578125	.42968750	.43359375
.001	.37524414	.37915039	.38305664	.38696289	.39086914	.39477539	.39868164	.40258789	.40649414	.41040039	.41430664	.41821289	.42211914	.42602539	.42993164	.43383789
.002	.37548828	.37939453	.38330078	.38720703	.39111328	.39501953	.39892578	.40283203	.40673828	.41064453	.41455078	.41845703	.42236328	.42626953	.43017578	.43408203
.003	.37573242	.37963867	.38354492	.38745117	.39135742	.39526367	.39916992	.40307617	.40698242	.41088867	.41479492	.41870117	.42260742	.42651367	.43041992	.43432617
.004	.37597656	.37988281	.38378906	.38769531	.39160156	.39550781	.39941406	.40332031	.40722656	.41113281	.41503906	.41894531	.42285156	.42675781	.43066406	.43457031
.005	.37622070	.38012695	.38403320	.38793945	.39184570	.39575195	.39965820	.40356445	.40747070	.41137695	.41528320	.41918945	.42309570	.42700195	.43090820	.43481445
.006	.37646484	.38037109	.38427734	.38818359	.39208984	.39599609	.39990234	.40380859	.40771484	.41162109	.41552734	.41943359	.42333984	.42724609	.43115234	.43505859
.007	.37670898	.38061523	.38452148	.38842773	.39233398	.39624023	.40014648	.40405273	.40795898	.41186523	.41577148	.41967773	.42358398	.42749023	.43139648	.43530273
.008	.37695313	.38085938	.38476563	.38867188	.39257813	.39648438	.40039063	.40429688	.40820313	.41210938	.41601563	.41992188	.42382813	.42773438	.43164063	.43554688
.009	.37719727	.38110352	.38500977	.38891602	.39282227	.39672852	.40063477	.40454102	.40844727	.41235352	.41625977	.42016602	.42407227	.42797852	.43188477	.43579102
.00A	.37744141	.38134766	.38525391	.38916016	.39306641	.39697266	.40087891	.40478516	.40869141	.41259766	.41650391	.42041016	.42431641	.42822266	.43212891	.43603516
.00B	.37768555	.38159180	.38549805	.38940430	.39331055	.39721680	.40112305	.40502930	.40893555	.41284180	.41674805	.42065430	.42456055	.42846680	.43237305	.43627930
.00C	.37792969	.38183594	.38574219	.38964844	.39355469	.39746094	.40136719	.40527344	.40917969	.41308594	.41699219	.42089844	.42480469	.42871094	.43261719	.43652344
.00D	.37817383	.38208008	.38598633	.38989258	.39379883	.39770508	.40161133	.40551758	.40942383	.41333008	.41723633	.42114258	.42504883	.42895508	.43286133	.43676758
.00E	.37841797	.38232422	.38623047	.39013672	.39404297	.39794922	.40185547	.40576172	.40966797	.41357422	.41748047	.42138672	.42529297	.42919922	.43310547	.43701172
.00F	.37866211	.38256836	.38647461	.39038086	.39428711	.39819336	.40209961	.40600586	.40991211	.41381836	.41772461	.42163086	.42553711	.42944336	.43334961	.43725586

	.70	.71	.72	.73	.74	.75	.76	.77	.78	.79	.7A	.7B	.7C	.7D	.7E	.7F
.000	.43750000	.44140625	.44531250	.44921875	.45312500	.45703125	.46093750	.46484375	.46875000	.47265625	.47656250	.48046875	.48437500	.48828125	.49218750	.49609375
.001	.43774414	.44165039	.44555664	.44946289	.45336914	.45727539	.46118164	.46508789	.46899414	.47290039	.47680664	.48071289	.48461914	.48852539	.49243164	.49633789
.002	.43798828	.44189453	.44580078	.44970703	.45361328	.45751953	.46142578	.46533203	.46923828	.47314453	.47705078	.48095703	.48486328	.48876953	.49267578	.49658203
.003	.43823242	.44213867	.44604492	.44995117	.45385742	.45776367	.46166992	.46557617	.46948242	.47338867	.47729492	.48120117	.48510742	.48901367	.49291992	.49682617
.004	.43847656	.44238281	.44628906	.45019531	.45410156	.45800781	.46191406	.46582031	.46972656	.47363281	.47753906	.48144531	.48535156	.48925781	.49316406	.49707031
.005	.43872070	.44262695	.44653320	.45043945	.45434570	.45825195	.46215820	.46606445	.46997070	.47387695	.47778320	.48168945	.48559570	.48950195	.49340820	.49731445
.006	.43896484	.44287109	.44677734	.45068359	.45458984	.45849609	.46240234	.46630859	.47021484	.47412109	.47802734	.48193359	.48583984	.48974609	.49365234	.49755859
.007	.43920898	.44311523	.44702148	.45092773	.45483398	.45874023	.46264648	.46655273	.47045898	.47436523	.47827148	.48217773	.48608398	.48999023	.49389648	.49780273
.008	.43945313	.44335938	.44726563	.45117188	.45507813	.45898438	.46289063	.46679688	.47070313	.47460938	.47851563	.48242188	.48632813	.49023438	.49414063	.49804688
.009	.43969727	.44360352	.44750977	.45141602	.45532227	.45922852	.46313477	.46704102	.47094727	.47485352	.47875977	.48266602	.48657227	.49047852	.49438477	.49829102
.00A	.43994141	.44384766	.44775391	.45166016	.45556641	.45947266	.46337891	.46728516	.47119141	.47509766	.47900391	.48291016	.48681641	.49072266	.49462891	.49853516
.00B	.44018555	.44409180	.44799805	.45190430	.45581055	.45971680	.46362305	.46752930	.47143555	.47534180	.47924805	.48315430	.48706055	.49096680	.49487305	.49877930
.00C	.44042969	.44433594	.44824219	.45214844	.45605469	.45996094	.46386719	.46777344	.47167969	.47558594	.47949219	.48339844	.48730469	.49121094	.49511719	.49902344
.00D	.44067383	.44458008	.44848633	.45239258	.45629883	.46020508	.46411133	.46801758	.47192383	.47583008	.47973633	.48364258	.48754883	.49145508	.49536133	.49926758
.00E	.44091797	.44482422	.44873047	.45263672	.45654297	.46044922	.46435547	.46826172	.47216797	.47607422	.47998047	.48388672	.48779297	.49169922	.49560547	.49951172
.00F	.44116211	.44506836	.44897461	.45288086	.45678711	.46069336	.46459961	.46850586	.47241211	.47631836	.48022461	.48413086	.48803711	.49194336	.49584961	.49975586

	.80	.81	.82	.83	.84	.85	.86	.87	.88	.89	.8A	.8B	.8C	.8D	.8E	.8F
.000	.50000000	.50390625	.50781250	.51171875	.51562500	.51953125	.52343750	.52734375	.53125000	.53515625	.53906250	.54296875	.54687500	.55078125	.55468750	.55859375
.001	.50024414	.50415039	.50805664	.51196289	.51586914	.51977539	.52368164	.52758789	.53149414	.53540039	.53930664	.54321289	.54711914	.55102539	.55493164	.55883789
.002	.50048828	.50439453	.50830078	.51220703	.51611328	.52001953	.52392578	.52783203	.53173828	.53564453	.53955078	.54345703	.54736328	.55126953	.55517578	.55908203
.003	.50073242	.50463867	.50854492	.51245117	.51635742	.52026367	.52416992	.52807617	.53198242	.53588867	.53979492	.54370117	.54760742	.55151367	.55541992	.55932617
.004	.50097656	.50488281	.50878906	.51269531	.51660156	.52050781	.52441406	.52832031	.53222656	.53613281	.54003906	.54394531	.54785156	.55175781	.55566406	.55957031
.005	.50122070	.50512695	.50903320	.51293945	.51684570	.52075195	.52465820	.52856445	.53247070	.53637695	.54028320	.54418945	.54809570	.55200195	.55590820	.55981445
.006	.50146484	.50537109	.50927734	.51318359	.51708984	.52099609	.52490234	.52880859	.53271484	.53662109	.54052734	.54443359	.54833984	.55224609	.55615234	.56005859
.007	.50170898	.50561523	.50952148	.51342773	.51733398	.52124023	.52514648	.52905273	.53295898	.53686523	.54077148	.54467773	.54858398	.55249023	.55639648	.56030273
.008	.50195313	.50585938	.50976563	.51367188	.51757813	.52148438	.52539063	.52929688	.53320313	.53710938	.54101563	.54492188	.54882813	.55273438	.55664063	.56054688
.009	.50219727	.50610352	.51000977	.51391602	.51782227	.52172852	.52563477	.52954102	.53344727	.53735352	.54125977	.54516602	.54907227	.55297852	.55688477	.56079102
.00A	.50244141	.50634766	.51025391	.51416016	.51806641	.52197266	.52587891	.52978516	.53369141	.53759766	.54150391	.54541016	.54931641	.55322266	.55712891	.56103516
.00B	.50268555	.50659180	.51049805	.51440430	.51831055	.52221680	.52612305	.53002930	.53393555	.53784180	.54174805	.54565430	.54956055	.55346680	.55737305	.56127930
.00C	.50292969	.50683594	.51074219	.51464844	.51855469	.52246094	.52636719	.53027344	.53417969	.53808594	.54199219	.54589844	.54980469	.55371094	.55761719	.56152344
.00D	.50317383	.50708008	.51098633	.51489258	.51879883	.52270508	.52661133	.53051758	.53442383	.53833008	.54223633	.54614258	.55004883	.55395508	.55786133	.56176758
.00E	.50341797	.50732422	.51123047	.51513672	.51904297	.52294922	.52685547	.53076172	.53466797	.53857422	.54248047	.54638672	.55029297	.55419922	.55810547	.56201172
.00F	.50366211	.50756836	.51147461	.51538086	.51928711	.52319336	.52709961	.53100586	.53491211	.53881836	.54272461	.54663086	.55053711	.55444336	.55834961	.56225586

Table B-2. Decimal-to-Hexadecimal Conversion – Fractions (Continued)

	.90	.91	.92	.93	.94	.95	.96	.97	.98	.99	.9A	.9B	.9C	.9D	.9E	.9F
.000	.56250000	.56640625	.57031250	.57421875	.57812500	.58203125	.58593750	.58984375	.59375000	.59765625	.60156250	.60546875	.60937500	.61328125	.61718750	.62109375
.001	.56274414	.56665039	.57055664	.57446289	.57836914	.58227539	.58618164	.59008789	.59399414	.59790039	.60180664	.60571289	.60961914	.61352539	.61743164	.62133789
.002	.56298828	.56689453	.57080078	.57470703	.57861328	.58251953	.58642578	.59033203	.59423828	.59814453	.60205078	.60595703	.60986328	.61376953	.61767578	.62158203
.003	.56323242	.56713867	.57104492	.57495117	.57885742	.58276367	.58666992	.59057617	.59448242	.59838867	.60229492	.60620117	.61010742	.61401367	.61791992	.62182617
.004	.56347656	.56738281	.57128906	.57519531	.57910156	.58300781	.58691406	.59082031	.59472656	.59863281	.60253906	.60644531	.61035156	.61425781	.61816406	.62207031
.005	.56372070	.56762695	.57153320	.57543945	.57934570	.58325195	.58715820	.59106445	.59497070	.59887695	.60278320	.60668945	.61059570	.61450195	.61840820	.62231445
.006	.56396484	.56787109	.57177734	.57568359	.57958984	.58349609	.58740234	.59130859	.59521484	.59912109	.60302734	.60693359	.61083984	.61474609	.61865234	.62255859
.007	.56420898	.56811523	.57202148	.57592773	.57983398	.58374023	.58764648	.59155273	.59545898	.59936523	.60327148	.60717773	.61108398	.61499023	.61889648	.62280273
.008	.56445313	.56835938	.57226563	.57617188	.58007813	.58398438	.58789063	.59179688	.59570313	.59960938	.60351563	.60742188	.61132813	.61523438	.61914063	.62304688
.009	.56469727	.56860352	.57250977	.57641602	.58032227	.58422852	.58813477	.59204102	.59594727	.59985352	.60375977	.60766602	.61157227	.61547852	.61938477	.62329102
.00A	.56494141	.56884766	.57275391	.57666016	.58056641	.58447266	.58837891	.59228516	.59619141	.60009766	.60400391	.60791016	.61181641	.61572266	.61962891	.62353516
.00B	.56518555	.56909180	.57299805	.57690430	.58081055	.58471680	.58862305	.59252930	.59643555	.60034180	.60424805	.60815430	.61206055	.61596680	.61987305	.62377930
.00C	.56542969	.56933594	.57324219	.57714844	.58105469	.58496094	.58886719	.59277344	.59667969	.60058594	.60449219	.60839844	.61230469	.61621094	.62011719	.62402344
.00D	.56567383	.56958008	.57348633	.57739258	.58129883	.58520508	.58911133	.59301758	.59692383	.60083008	.60473633	.60864258	.61254883	.61645508	.62036133	.62426758
.00E	.56591797	.56982422	.57373047	.57763672	.58154297	.58544922	.58935547	.59326172	.59716797	.60107422	.60498047	.60888672	.61279297	.61669922	.62060547	.62451172
.00F	.56616211	.57006836	.57397461	.57788086	.58178711	.58569336	.58959961	.59350586	.59741211	.60131836	.60522461	.60913086	.61303711	.61694336	.62084961	.62475586

	.A0	.A1	.A2	.A3	.A4	.A5	.A6	.A7	.A8	.A9	.AA	.AB	.AC	.AD	.AE	.AF
.000	.62500000	.62890625	.63281250	.63671875	.64062500	.64453125	.64843750	.65234375	.65625000	.66015625	.66406250	.66796875	.67187500	.67578125	.67968750	.68359375
.001	.62524414	.62915039	.63305664	.63696289	.64086914	.64477539	.64868164	.65258789	.65649414	.66040039	.66430664	.66821289	.67211914	.67602539	.67993164	.68383789
.002	.62548828	.62939453	.63330078	.63720703	.64111328	.64501953	.64892578	.65283203	.65673828	.66064453	.66455078	.66845703	.67236328	.67626953	.68017578	.68408203
.003	.62573242	.62963867	.63354492	.63745117	.64135742	.64526367	.64916992	.65307617	.65698242	.66088867	.66479492	.66870117	.67260742	.67651367	.68041992	.68432617
.004	.62597656	.62988281	.63378906	.63769531	.64160156	.64550781	.64941406	.65332031	.65722656	.66113281	.66503906	.66894531	.67285156	.67675781	.68066406	.68457031
.005	.62622070	.63012695	.63403320	.63793945	.64184570	.64575195	.64965820	.65356445	.65747070	.66137695	.66528320	.66918945	.67309570	.67700195	.68090820	.68481445
.006	.62646484	.63037109	.63427734	.63818359	.64208984	.64599609	.64990234	.65380859	.65771484	.66162109	.66552734	.66943359	.67333984	.67724609	.68115234	.68505859
.007	.62670898	.63061523	.63452148	.63842773	.64233398	.64624023	.65014648	.65405273	.65795898	.66186523	.66577148	.66967773	.67358398	.67749023	.68139648	.68530273
.008	.62695313	.63085938	.63476563	.63867188	.64257813	.64648438	.65039063	.65429688	.65820313	.66210938	.66601563	.66992188	.67382813	.67773438	.68164063	.68554688
.009	.62719727	.63110352	.63500977	.63891602	.64282227	.64672852	.65063477	.65454102	.65844727	.66235352	.66625977	.67016602	.67407227	.67797852	.68188477	.68579102
.00A	.62744141	.63134766	.63525391	.63916016	.64306641	.64697266	.65087891	.65478516	.65869141	.66259766	.66650391	.67041016	.67431641	.67822266	.68212891	.68603516
.00B	.62768555	.63159180	.63549805	.63940430	.64331055	.64721680	.65112305	.65502930	.65893555	.66284180	.66674805	.67065430	.67456055	.67846680	.68237305	.68627930
.00C	.62792969	.63183594	.63574219	.63964844	.64355469	.64746094	.65136719	.65527344	.65917969	.66308594	.66699219	.67089844	.67480469	.67871094	.68261719	.68652344
.00D	.62817383	.63208008	.63598633	.63989258	.64379883	.64770508	.65161133	.65551758	.65942383	.66333008	.66723633	.67114258	.67504883	.67895508	.68286133	.68676758
.00E	.62841797	.63232422	.63623047	.64013672	.64404297	.64794922	.65185547	.65576172	.65966797	.66357422	.66748047	.67138672	.67529297	.67919922	.68310547	.68701172
.00F	.62866211	.63256836	.63647461	.64038086	.64428711	.64819336	.65209961	.65600586	.65991211	.66381836	.66772461	.67163086	.67553711	.67944336	.68334961	.68725586

Table B-2. Decimal-to-Hexadecimal Conversion — Fractions (Continued)

	.B0	.B1	.B2	.B3	.B4	.B5	.B6	.B7	.B8	.B9	.BA	.BB	.BC	.BD	.BE	.BF
.000	.68750000	.69140625	.69531250	.69921875	.70312500	.70703125	.71093750	.71484375	.71875000	.72265625	.72656250	.73046875	.73437500	.73828125	.74218750	.74609375
.001	.68774414	.69165039	.69555664	.69946289	.70336914	.70727539	.71118164	.71508789	.71899414	.72290039	.72680664	.73071289	.73461914	.73852539	.74243164	.74633789
.002	.68798828	.69189453	.69580078	.69970703	.70361328	.70751953	.71142578	.71533203	.71923828	.72314453	.72705078	.73095703	.73486328	.73876953	.74267578	.74658203
.003	.68823242	.69213867	.69604492	.69995117	.70385742	.70776367	.71166992	.71557617	.71948242	.72338867	.72729492	.73120117	.73510742	.73901367	.74291992	.74682617
.004	.68847656	.69238281	.69628906	.70019531	.70410156	.70800781	.71191406	.71582031	.71972656	.72363281	.72753906	.73144531	.73535156	.73925781	.74316406	.74707031
.005	.68872070	.69262695	.69653320	.70043945	.70434570	.70825195	.71215820	.71606445	.71997070	.72387695	.72778320	.73168945	.73559570	.73950195	.74340820	.74731445
.006	.68896484	.69287109	.69677734	.70068359	.70458984	.70849609	.71240234	.71630859	.72021484	.72412109	.72802734	.73193359	.73583984	.73974609	.74365234	.74755859
.007	.68920898	.69311523	.69702148	.70092773	.70483398	.70874023	.71264648	.71655273	.72045898	.72436523	.72827148	.73217773	.73608398	.73999023	.74389648	.74780273
.008	.68945313	.69335938	.69726563	.70117188	.70507813	.70898438	.71289063	.71679688	.72070313	.72460938	.72851563	.73242188	.73632813	.74023438	.74414063	.74804688
.009	.68969727	.69360352	.69750977	.70141602	.70532227	.70922852	.71313477	.71704102	.72094727	.72485352	.72875977	.73266602	.73657227	.74047852	.74438477	.74829102
.00A	.68994141	.69384766	.69775391	.70166016	.70556641	.70947266	.71337891	.71728516	.72119141	.72509766	.72900391	.73291016	.73681641	.74072266	.74462891	.74853516
.00B	.69018555	.69409180	.69799805	.70190430	.70581055	.70971680	.71362305	.71752930	.72143555	.72534180	.72924805	.73315430	.73706055	.74096680	.74487305	.74877930
.00C	.69042969	.69433594	.69824219	.70214844	.70605469	.70996094	.71386719	.71777344	.72167969	.72558594	.72949219	.73339844	.73730469	.74121094	.74511719	.74902344
.00D	.69067383	.69458008	.69848633	.70239258	.70629883	.71020508	.71411133	.71801758	.72192383	.72583008	.72973633	.73364258	.73754883	.74145508	.74536133	.74926758
.00E	.69091797	.69482422	.69873047	.70263672	.70654297	.71044922	.71435547	.71826172	.72216797	.72607422	.72998047	.73388672	.73779297	.74169922	.74560547	.74951172
.00F	.69116211	.69506836	.69897461	.70288086	.70678711	.71069336	.71459961	.71850586	.72241211	.72631836	.73022461	.73413086	.73803711	.74194336	.74584961	.74975586

	.C0	.C1	.C2	.C3	.C4	.C5	.C6	.C7	.C8	.C9	.CA	.CB	.CC	.CD	.CE	.CF
.000	.75000000	.75390625	.75781250	.76171875	.76562500	.76953125	.77343750	.77734375	.78125000	.78515625	.78906250	.79296875	.79687500	.80078125	.80468750	.80859375
.001	.75024414	.75415039	.75805664	.76196289	.76586914	.76977539	.77368164	.77758789	.78149414	.78540039	.78930664	.79321289	.79711914	.80102539	.80493164	.80883789
.002	.75048828	.75439453	.75830078	.76220703	.76611328	.77001953	.77392578	.77783203	.78173828	.78564453	.78955078	.79345703	.79736328	.80126953	.80517578	.80908203
.003	.75073242	.75463867	.75854492	.76245117	.76635742	.77026367	.77416992	.77807617	.78198242	.78588867	.78979492	.79370117	.79760742	.80151367	.80541992	.80932617
.004	.75097656	.75488281	.75878906	.76269531	.76660156	.77050781	.77441406	.77832031	.78222656	.78613281	.79003906	.79394531	.79785156	.80175781	.80566406	.80957031
.005	.75122070	.75512695	.75903320	.76293945	.76684570	.77075195	.77465820	.77856445	.78247070	.78637695	.79028320	.79418945	.79809570	.80200195	.80590820	.80981445
.006	.75146484	.75537109	.75927734	.76318359	.76708984	.77099609	.77490234	.77880859	.78271484	.78662109	.79052734	.79443359	.79833984	.80224609	.80615234	.81005859
.007	.75170898	.75561523	.75952148	.76342773	.76733398	.77124023	.77514648	.77905273	.78295898	.78686523	.79077148	.79467773	.79858398	.80249023	.80639648	.81030273
.008	.75195313	.75585938	.75976563	.76367188	.76757813	.77148438	.77539063	.77929688	.78320313	.78710938	.79101563	.79492188	.79882813	.80273438	.80664063	.81054688
.009	.75219727	.75610352	.76000977	.76391602	.76782227	.77172852	.77563477	.77954102	.78344727	.78735352	.79125977	.79516602	.79907227	.80297852	.80688477	.81079102
.00A	.75244141	.75634766	.76025391	.76416016	.76806641	.77197266	.77587891	.77978516	.78369141	.78759766	.79150391	.79541016	.79931641	.80322266	.80712891	.81103516
.00B	.75268555	.75659180	.76049805	.76440430	.76831055	.77221680	.77612305	.78002930	.78393555	.78784180	.79174805	.79565430	.79956055	.80346680	.80737305	.81127930
.00C	.75292969	.75683594	.76074219	.76464844	.76855469	.77246094	.77636719	.78027344	.78417969	.78808594	.79199219	.79589844	.79980469	.80371094	.80761719	.81152344
.00D	.75317383	.75708008	.76098633	.76489258	.76879883	.77270508	.77661133	.78051758	.78442383	.78833008	.79223633	.79614258	.80004883	.80395508	.80786133	.81176758
.00E	.75341797	.75732422	.76123047	.76513672	.76904297	.77294922	.77685547	.78076172	.78466797	.78857422	.79248047	.79638672	.80029297	.80419922	.80810547	.81201172
.00F	.75366211	.75756836	.76147461	.76538086	.76928711	.77319336	.77709961	.78100586	.78491211	.78881836	.79272461	.79663086	.80053711	.80444336	.80834961	.81225586

	.D0	.D1	.D2	.D3	.D4	.D5	.D6	.D7	.D8	.D9	.DA	.DB	.DC	.DD	.DE	.DF
.000	.81250000	.81640625	.82031250	.82421875	.82812500	.83203125	.83593750	.83984375	.84375000	.84765625	.85156250	.85546875	.85937500	.86328125	.86718750	.87109375
.001	.81274414	.81665039	.82055664	.82446289	.82836914	.83227539	.83618164	.84008789	.84399414	.84790039	.85180664	.85571289	.85961914	.86352539	.86743164	.87133789
.002	.81298828	.81689453	.82080078	.82470703	.82861328	.83251953	.83642578	.84033203	.84423828	.84814453	.85205078	.85595703	.85986328	.86376953	.86767578	.87158203
.003	.81323242	.81713867	.82104492	.82495117	.82885742	.83276367	.83666992	.84057617	.84448242	.84838867	.85229492	.85620117	.86010742	.86401367	.86791992	.87182617
.004	.81347656	.81738281	.82128906	.82519531	.82910156	.83300781	.83691406	.84082031	.84472656	.84863281	.85253906	.85644531	.86035156	.86425781	.86816406	.87207031
.005	.81372070	.81762695	.82153320	.82543945	.82934570	.83325195	.83715820	.84106445	.84497070	.84887695	.85278320	.85668945	.86059570	.86450195	.86840820	.87231445
.006	.81396484	.81787109	.82177734	.82568359	.82958984	.83349609	.83740234	.84130859	.84521484	.84912109	.85302734	.85693359	.86083984	.86474609	.86865234	.87255859
.007	.81420898	.81811523	.82202148	.82592773	.82983398	.83374023	.83764648	.84155273	.84545898	.84936523	.85327148	.85717773	.86108398	.86499023	.86889648	.87280273
.008	.81445313	.81835938	.82226563	.82617188	.83007813	.83398438	.83789063	.84179688	.84570313	.84960938	.85351563	.85742188	.86132813	.86523438	.86914063	.87304688
.009	.81469727	.81860352	.82250977	.82641602	.83032227	.83422852	.83813477	.84204102	.84594727	.84985352	.85375977	.85766602	.86157227	.86547852	.86938477	.87329102
.00A	.81494141	.81884766	.82275391	.82666016	.83056641	.83447266	.83837891	.84228516	.84619141	.85009766	.85400391	.85791016	.86181641	.86572266	.86962891	.87353516
.00B	.81518555	.81909180	.82299805	.82690430	.83081055	.83471680	.83862305	.84252930	.84643555	.85034180	.85424805	.85815430	.86206055	.86596680	.86987305	.87377930
.00C	.81542969	.81933594	.82324219	.82714844	.83105469	.83496094	.83886719	.84277344	.84667969	.85058594	.85449219	.85839844	.86230469	.86621094	.87011719	.87402344
.00D	.81567383	.81958008	.82348633	.82739258	.83129883	.83520508	.83911133	.84301758	.84692383	.85083008	.85473633	.85864258	.86254883	.86645508	.87036133	.87426758
.00E	.81591797	.81982422	.82373047	.82763672	.83154297	.83544922	.83935547	.84326172	.84716797	.85107422	.85498047	.85888672	.86279297	.86669922	.87060547	.87451172
.00F	.81616211	.82006836	.82397461	.82788086	.83178711	.83569336	.83959961	.84350586	.84741211	.85131836	.85522461	.85913086	.86303711	.86694336	.87084961	.87475586

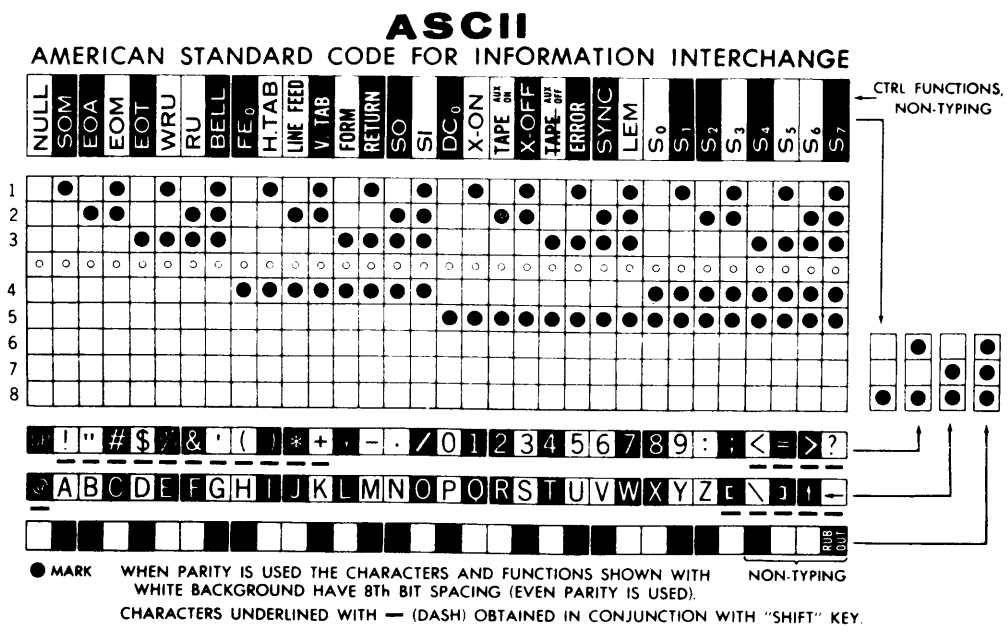
Table B-2. Decimal-to-Hexadecimal Conversion — Fractions (Continued)

	.E0	.E1	.E2	.E3	.E4	.E5	.E6	.E7	.E8	.E9	.EA	.EB	.EC	.ED	.EE	.EF
.000	.87500000	.87890625	.88281250	.88671875	.89062500	.89453125	.89843750	.90234375	.90625000	.91015625	.91406250	.91796875	.92187500	.92578125	.92968750	.93359375
.001	.87524414	.87915039	.88305664	.88696289	.89086914	.89477539	.89868164	.90258789	.90649414	.91040039	.91430664	.91821289	.92211914	.92602539	.92993164	.93383789
.002	.87548828	.87939453	.88330078	.88720703	.89111328	.89501953	.89892578	.90283203	.90673828	.91064453	.91455078	.91845703	.92236328	.92626953	.93017578	.93408203
.003	.87573242	.87963867	.88354492	.88745117	.89135742	.89526367	.89916992	.90307617	.90698242	.91088867	.91479492	.91870117	.92260742	.92651367	.93041992	.93432617
.004	.87597656	.87988281	.88378906	.88769531	.89160156	.89550781	.89941406	.90332031	.90722656	.91113281	.91503906	.91894531	.92285156	.92675781	.93066406	.93457031
.005	.87622070	.88012695	.88403320	.88793945	.89184570	.89575195	.89965820	.90356445	.90747070	.91137695	.91528320	.91918945	.92309570	.92700195	.93090820	.93481445
.006	.87646484	.88037109	.88427734	.88818359	.89208984	.89599609	.89990234	.90380859	.90771484	.91162109	.91552734	.91943359	.92333984	.92724609	.93115234	.93505859
.007	.87670898	.88061523	.88452148	.88842773	.89233398	.89624023	.90014648	.90405273	.90795898	.91186523	.91577148	.91967773	.92358398	.92749023	.93139648	.93530273
.008	.87695313	.88085938	.88476563	.88867188	.89257813	.89648438	.90039063	.90429688	.90820313	.91210938	.91601563	.91992188	.92382813	.92773438	.93164063	.93554688
.009	.87719727	.88110352	.88500977	.88891602	.89282227	.89672852	.90063477	.90454102	.90844727	.91235352	.91625977	.92016602	.92407227	.92797852	.93188477	.93579102
.00A	.87744141	.88134766	.88525391	.88916016	.89306641	.89697266	.90087891	.90478516	.90869141	.91259766	.91650391	.92041016	.92431641	.92822266	.93212891	.93603516
.00B	.87768555	.88159180	.88549805	.88940430	.89331055	.89721680	.90112305	.90502930	.90893555	.91284180	.91674805	.92065430	.92456055	.92846680	.93237305	.93627930
.00C	.87792969	.88183594	.88574219	.88964844	.89355469	.89746094	.90136719	.90527344	.90917969	.91308594	.91699219	.92089844	.92480469	.92871094	.93261719	.93652344
.00D	.87817383	.88208008	.88598633	.88989258	.89379883	.89770508	.90161133	.90551758	.90942383	.91333008	.91723633	.92114258	.92504883	.92895508	.93286133	.93676758
.00E	.87841797	.88232422	.88623047	.89013672	.89404297	.89794922	.90185547	.90576172	.90966797	.91357422	.91748047	.92138672	.92529297	.92919922	.93310547	.93701172
.00F	.87866211	.88256836	.88647461	.89038086	.89428711	.89819336	.90209961	.90600586	.90991211	.91381836	.91772461	.92163086	.92553711	.92944336	.93334961	.93725586

	.F0	.F1	.F2	.F3	.F4	.F5	.F6	.F7	.F8	.F9	.FA	.FB	.FC	.FD	.FE	.FF
.000	.93750000	.94140625	.94531250	.94921875	.95312500	.95703125	.96093750	.96484375	.96875000	.97265625	.97656250	.98046875	.98437500	.98828125	.99218750	.99609375
.001	.93774414	.94165039	.94555664	.94946289	.95336914	.95727539	.96118164	.96508789	.96899414	.97290039	.97680664	.98071289	.98461914	.98852539	.99243164	.99633789
.002	.93798828	.94189453	.94580078	.94970703	.95361328	.95751953	.96142578	.96533203	.96923828	.97314453	.97705078	.98095703	.98486328	.98876953	.99267578	.99658203
.003	.93823242	.94213867	.94604492	.94995117	.95385742	.95776367	.96166992	.96557617	.96948242	.97338867	.97729492	.98120117	.98510742	.98901367	.99291992	.99682617
.004	.93847656	.94238281	.94628906	.95019531	.95410156	.95800781	.96191406	.96582031	.96972656	.97363281	.97753906	.98144531	.98535156	.98925781	.99316406	.99707031
.005	.93872070	.94262695	.94653320	.95043945	.95434570	.95825195	.96215820	.96606445	.96997070	.97387695	.97778320	.98168945	.98559570	.98950195	.99340820	.99731445
.006	.93896484	.94287109	.94677734	.95068359	.95458984	.95849609	.96240234	.96630859	.97021484	.97412109	.97802734	.98193359	.98583984	.98974609	.99365234	.99755859
.007	.93920898	.94311523	.94702148	.95092773	.95483398	.95874023	.96264648	.96655273	.97045898	.97436523	.97827148	.98217773	.98608398	.98999023	.99389648	.99780273
.008	.93945313	.94335938	.94726563	.95117188	.95507813	.95898438	.96289063	.96679688	.97070313	.97460938	.97851563	.98242188	.98632813	.99023438	.99414063	.99804688
.009	.93969727	.94360352	.94750977	.95141602	.95532227	.95922852	.96313477	.96704102	.97094727	.97485352	.97875977	.98266602	.98657227	.99047852	.99438477	.99829102
.00A	.93994141	.94384766	.94775391	.95166016	.95556641	.95947266	.96337891	.96728516	.97119141	.97509766	.97900391	.98291016	.98681641	.99072266	.99462891	.99853516
.00B	.94018555	.94409180	.94799805	.95190430	.95581055	.95971680	.96362305	.96752930	.97143555	.97534180	.97924805	.98315430	.98706055	.99096680	.99487305	.99877930
.00C	.94042969	.94433594	.94824219	.95214844	.95605469	.95996094	.96386719	.96777344	.97167969	.97558594	.97949219	.98339844	.98730469	.99121094	.99511719	.99902344
.00D	.94067383	.94458008	.94848633	.95239258	.95629883	.96020508	.96411133	.96801758	.97192383	.97583008	.97973633	.98364258	.98754883	.99145508	.99536133	.99926758
.00E	.94091797	.94482422	.94873047	.95263672	.95654297	.96044922	.96435547	.96826172	.97216797	.97607422	.97998047	.98388672	.98779297	.99169922	.99560547	.99951172
.00F	.94116211	.94506836	.94897461	.95288086	.95678711	.96069336	.96459961	.96850586	.97241211	.97631836	.98022461	.98413086	.98803711	.99194336	.99584961	.99975586

APPENDIX C

TELETYPE CODE



APPENDIX D. INSTRUCTION LIST

Load and Store Instructions (10)

	Cycles	
LDA	2	Load A from Memory①
STA	2	Store A in Memory
LDB	2	Load B from Memory
STB	2	Store B in Memory
LDX	3	Load X_i from Memory②
STX	3	Store X_i in Memory
LDS	3	Load S from Memory
STS	3	Store S in Memory
LIE	3	Load N from Memory
SIE	3	Store N in Memory

Arithmetic Instructions (14)

	Cycles	
AMA	2	Add Memory to A
AMB	2	Add Memory to B
AMX	3	Add Memory to X_i
AAM	3	Add A to Memory
AXM	4	Add X_i to Memory
SMA	2	Subtract Memory from A
SMB	2	Subtract Memory from B
SMX	3	Subtract Memory from X_i
MUL	5.8us	Multiply Memory by B
DIV	6.3us	Divide A and B by Memory
INC	3	Increment Memory & Compare with 0
CAM	2	Compare A with Memory
CBM	2	Compare B with Memory
CXM	3	Compare X_i with Memory

Logical Instructions (9)

	Cycles	
ANA	2	AND A and Memory
ANB	2	AND B and Memory
ANX	3	AND X_i and Memory
ORA	2	OR A and Memory
ORB	2	OR B and Memory
ORX	3	OR X_i and Memory
EOA	2	Exclusive OR A and Memory
EOB	2	Exclusive OR B and Memory
EOX	3	Exclusive OR X_i and Memory

① One or two word instruction. Add one cycle if two word.

② X_i = Any X-Register specified by $i \cdot 0 \leq i \leq 15$.

③ R_i = A or B or any X-Register.

Register Instructions (13)

	Cycles	
TRF	1	Transfer R_1 to $R_2$③
IRT	1	Increment R_1 and Transfer to R_2
DRT	1	Decrement R_1 and Transfer to R_2
TRT	1	Two's Complement R_1 and Transfer to R_2
ORT	1	One's Complement R_1 and Transfer to R_2
EXG	1	Exchange R_1 with R_2
CRZ	1	Clear R_1 and R_2 to Zero
ARR	1	Add R_1 and R_2
SRR	1	Subtract R_1 from R_2
CRR	1	Compare R_1 with R_2
ANR	1	AND R_1 and R_2
ORR	1	OR R_1 and R_2
EOR	1	Exclusive OR R_1 and R_2

Shift Instructions (19) 1.15 + .125n us

LLA	Logical Shift Left A
LLB	Logical Shift Left B
LLD	Logical Shift Left Double
LRA	Logical Shift Right A
LRB	Logical Shift Right B
LRD	Logical Shift Right Double
ALA	Arithmetic Shift Left A
ALB	Arithmetic Shift Left B
ALD	Arithmetic Shift Left Double
ARA	Arithmetic Shift Right A
ARB	Arithmetic Shift Right B
ARD	Arithmetic Shift Right Double
CLA	Circular Shift Left A
CLB	Circular Shift Left B
CLD	Circular Shift Left Double
CRA	Circular Shift Right A
CRB	Circular Shift Right B
CRD	Circular Shift Right Double
BRV	Bit Reversal

Control (6)

	Cycles	
HLT	1	Halt
RPF	1	Reset Program Flag
SPF	1	Set Program Flag
RIF	3	Reset High Rate Interrupt Flag
SIF	3	Set High Rate Interrupt Flag
WAT	1	Wait

Branch Instructions (15)

	Cycles	
BUR	1	Branch Unconditionally Relative
BAR	1	Branch A Condition Relative
BIR	1	Branch Indicator Condition Relative
BSP	2	Branch and Store P-Register
BRU	2	Branch Unconditionally
BAC	2	Branch A Condition
BBC	2	Branch B Condition
BXC	2	Branch X_i Condition
BIC	2	Branch Indicator Condition
BST	2	Branch Sense Switch True
BSF	2	Branch Sense Switch False
BFT	2	Branch Program Flag True
BFF	2	Branch Program Flag False
BRE	2	Branch and Reset External Interrupt
BRI	2	Branch and Reset Internal Interrupt

Input/Output (5)

EDF	1.5μs	Executive Device Function
WTI	1.5μs	Word Transfer In
WTO	1.5μs	Word Transfer Out
RDS	1.5μs	Request Device Status
ICI	1.5μs	Interrogate Common Interrupts

Special (5)

ICF	Internal Control Function
RIS	Request Internal Status
SPM	Set Protect Mask
SYC	System Call
SYR	System Return