A FUNCTIONAL DESCRIPTION OF THE LITTON L·304H MICROELECTRONIC COMPUTER

MS 37269B

FUNCTIONAL DESCRIPTION OF THE LITTON L-304H MICROELECTRONIC COMPUTER

July 1974

Prepared by:

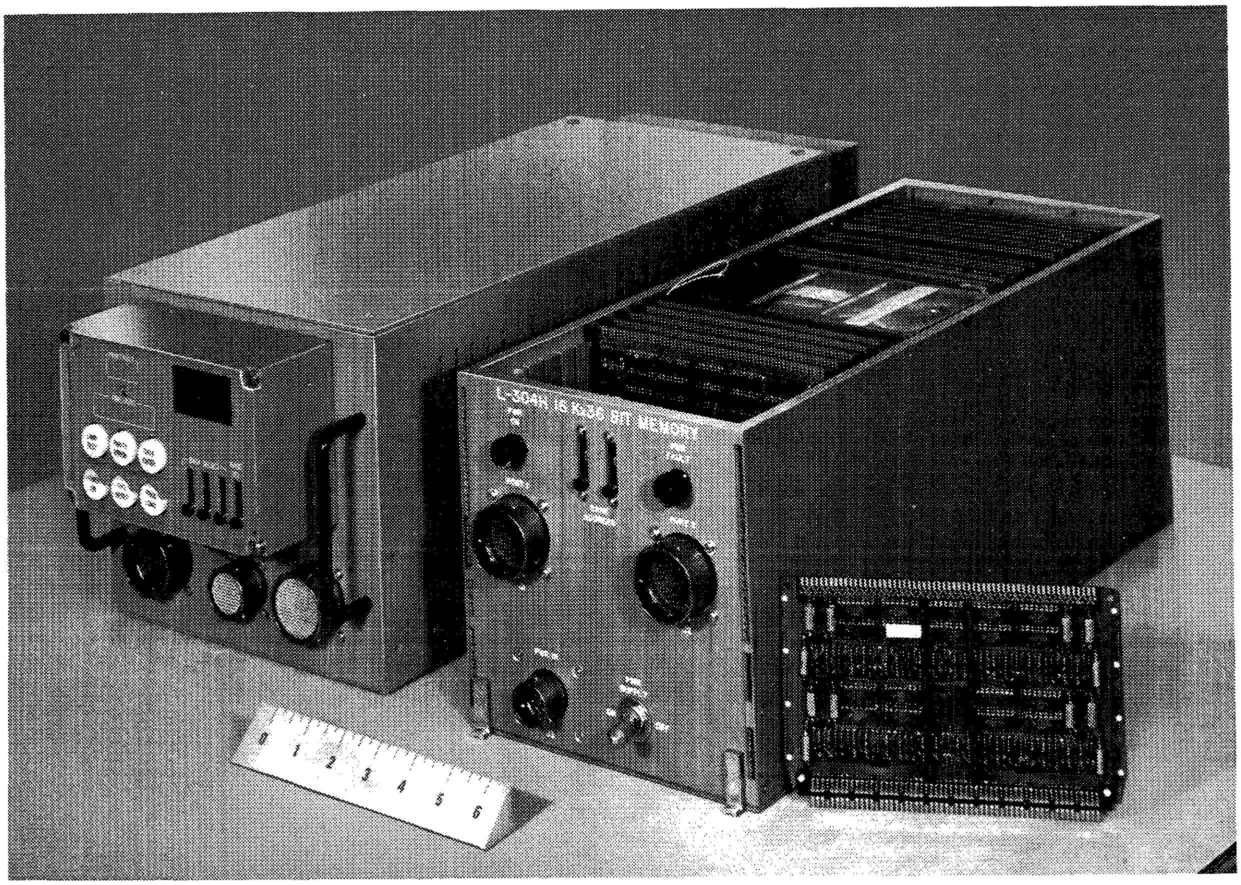
Data Systems Division
Litton Systems, Inc.
8000 Woodley Avenue
Van Nuys, California 91409

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES



Litton L-304H Computer System

SECTION 1 INTRODUCTION

L-304H OFFERS HIGH-SPEED PROCESSING FOR TACTICAL APPLICATIONS

The Litton L-304H Computer provides significantly increased speed and performance capabilities over previous L-304 models.

The L-304H is a successor to the basic Litton L-304 series of computers that have found widespread application among all branches of the military for a variety of real-time data processing applications. The L-304H retains the general characteristics of this basic computer family while offering a performance improvement of approximately four times that of the currently deployed L-304F.

The computer architecture of the L-304 computer systems is specifically designed for real-time application as evidenced by the inclusion of features such as a hardware interrupt handler, hardware multiprogramming control, special instructions, a hardware real-time clock, and provisions for special processing units (SPUs). These and additional features of the L-304H are described in more detail in Table I-1.

Experience with real-time systems throughout the industry has shown that attempts to handle a large number of interrupts on a real-time basis with an executive-type software interrupt handler results in large overhead which decreases the efficiency of the machine directly proportional to the number of interrupts that must be serviced. Separate hardware is provided in the L-304H to determine the source of the interrupt and whether its priority is higher or lower than the program currently under execution. This function is performed concurrently and without interference to the currently executing program. Sixty-four levels of priority are provided, with complete control afforded the programmer for masking and enabling each of the interrupts independently.

Hardware implementation of the multiprogramming function is also provided to minimize the time required for data exchanging in the multiprogramming environment. Since the functions performed in program exchanging are repetitive, hardware implementation of this feature is quite straightforward. As with the handling of interrupts, full software override control is provided to the program.

In design of the L-304 computer family, special instructions were included for real-time processing. As an example, the Gated Compare instruction provides the ability to determine whether a quantity is within or without specified limits. Such a function, of course, is particularly useful in the correlation of incoming sensor data with an existing data file. Hardware mechanization of real-time clocks provides the programmer with the capability to set timing intervals for four different real-time clocks, each with resolution of one millisecond.

The L-304H is fully I/O compatible and upward software compatible with the L-304F. This feature was provided by having the instruction set of the L-304H contain, as a subset, the entire repertoire of the L-304F. Since software compatibility has been maintained with previous versions of the computer, fully developed support software is available. Enhancements to the already powerful L-304 repertoire are provided by new instructions and optional macro capability. The new instructions provide double precision (32-bit) arithmetic; direct "flag" testing of a single bit in memory; and enqueue, dequeue instructions for file control. Special processing units will "custom" functions as required by particular applications.

Implementation of the computer utilizes MSI Schottky circuitry which increases the packaging density and permits an increase in the CPU clock rate.

SYSTEM ORGANIZATION OF THE L-304H COMPUTER

The L-304H computer system achieves maximum flexibility and growth potential in storage, computation, and input/output capability by use of modular design techniques.

The modular design of the L-304H computer permits a variety of system configurations, ranging from those with limited processing, memory, and I/O requirements to systems requiring multiple processors, independent input/output operations, and masses of random access storage. This growth is achieved by adding central processing units, memory units, independent input/output units, and special processing units (SPUs) without changing the basic system implementation.

The L-304H memories are designed with two or four ports; i.e., they contain data lines, control lines, and priority logic which will permit either two or four processors (or processor-like devices) to access the memories independently.

A basic processing system requires the use of a single memory port with the IOUs and SPUs cycle stealing on a common bus with the CPU. (See Figure 1-1, A.) The other port is available for I/O direct memory access.

If additional processing capability is required, a second L-304H processing unit (CPU, IOU, SPU) may be added. (See Figure 1-1, B.)

Maximum memory expansion is achieved with the use of word addressable mass memories and an optional extended memory card. (See Figure 1-1, C.) While the memory bus capacity limits the number of memory units to eight, the number of words directly addressable by L-304H instructions with an expanded EMA is two million. Therefore, mass core memory units each containing 131K words or more may be used. The asynchronous timing relationship between the CPU and the memories permits the access time of the mass memory to be considerably slower from that of the main memory (without creating synchronization problems).

An optional input/output processor (IOP) requiring an independent bus memory would use the four port memory(s), as shown in Figure 1-1, D.

Table I-1. Key Features of the L-304H Computer (Sheet 1 of 2)

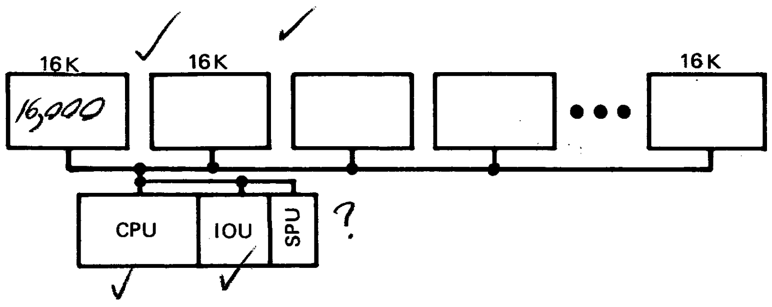
Category	Features
Computer Type	General-purpose; modular, with single or multiple central processing units.
Logic	Synchronous; silicon integrated circuits, TTL SSI, TTL MSI, and Schottky SSI, MSI.
Arithmetic Mode	Parallel; binary; fixed point.
Floating Point Option	Five basic instructions: Compare, Add, Subtract, Multiply, Divide. 32-bit format: sign bit 8-bit exponent, 23-bit function.
Word Length	Memory word – 32 bits plus 4 parity bits; (parity optional) instruction word – 32 bits; data word – 32 bits, 16 bits, 8 bits, 1 bit.
Core Memory	16,384-word assemblies, modular expansion to two million words; overlapping memory cycles; one to four independent access ports; changeable bank address; memory protect by 2048-word block (optional); 450 nanosecond access; less than 750 nanoseconds full cycle; wide temperature cores; power transient protection.
Instructions	89 Basic Instructions: 18 Arithmetic, 36 Data Manipulation and Handling, 26 Jump and Transfer, 4 Input/Output and 5 Miscellaneous. 39 Macro Instructions may be specified.
Addressing	Single and double word addresses; modifiable; full randon addressing capability up to two million words; memory bus limited by line drive capability to maximum of 8 memory banks.
Addressing Range	Without EMA Options: 32,768 words With Standard EMA Option: 131,072 words With Expanded EMA Option: 2,097,152 words
Address Modification	Eight modes: literal, direct, indirect, relative, with selectable indexing; plus register-to-register mode.
Arithmetic	Eight accumulators for each level; eight index registers for each level, overlapped with accumulators; fixed-point binary arithmetic; numbers are signed integers, negative numbers are two's complement.

Table I-1. Key Features of the L-304H Computer (Sheet 2 of 2)

Category	Feature
Programming	Sixty-four independent program levels with externally and internally activated priority-level switching and software override capability; dynamic priority hierarchy; eight process registers for each level to be used as accumulators, index registers, mask registers; 16 memory page control and address registers for each program level (optional); program activity register containing one status and one enable bit for each program level; multilevel priority interrupts; dynamic program relocatability using page/base address registers; special memory test instructions; macro instruction capability.
Data Handling	Logical operations; field, insert, shift, and comparison instruction.
Input/Output	Single word transfers; block transfers by bytes or words (asynchronous, independent of program execution); burst block transfers (synchronous with device); interleaved transfers from several devices; simultaneous I/O and instruction execution, independent I/O controller; seaparate normal and error termination for each device; alarm clock mode (for real-time clock); bootstrap program load.
Maintenance	Central system self-test using hang-up detectors for memory, central processing unit, I/O, real-time clock; automatic detection to gross function or module; modular construction; vast compatibility. Optional self-test logic.
Growth Capability	Special processing units (SPUs) implement classes of individual instructions using any operation code not required for the basic L-304H instruction set (macro). Specialized design for macro instructions allows greater throughput. Six SPUs and one IOU may be associated with each CPU on each Direct Memory Access channel.
EMA (Standard Option)	Extended memory address up to 131K words Memory Protection Parity generation and error detection for both the CPU memory and IOU memory interfaces.

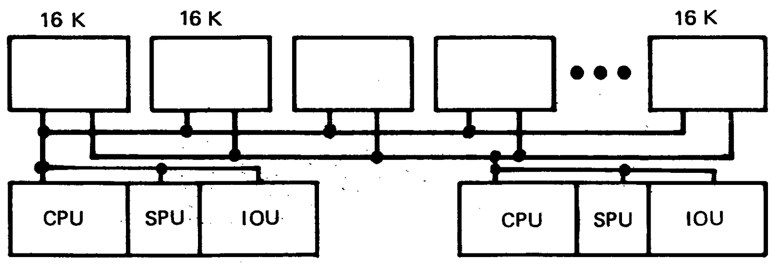
A

- L304H CENTRAL PROCESSING UNIT
- UP TO 131,072 WORDS OF MEMORY
- INPUT/OUTPUT UNIT
- SPECIAL PROCESSING UNIT



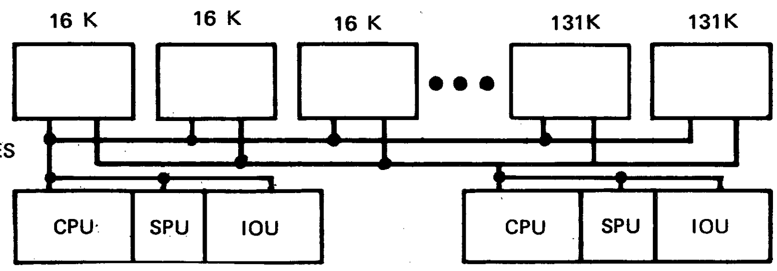
B

- DUAL L304H CENTRAL PROCESSING UNITS
- UP TO 131,072 WORDS OF MEMORY
- INPUT/OUTPUT UNITS
- SPECIAL PROCESSING UNITS



C

- DUAL L304H CENTRAL PROCESSING UNITS
- MIX OF 16K AND 131K WORD MEMORIES
- INPUT/OUTPUT UNITS
- SPECIAL PROCESSING UNITS



D

- DUAL L304H CENTRAL PROCESSING UNITS
- UP TO 131,072 WORDS OF MEMORY
- INPUT/OUTPUT PROCESSORS
- SPECIAL PROCESSING UNITS

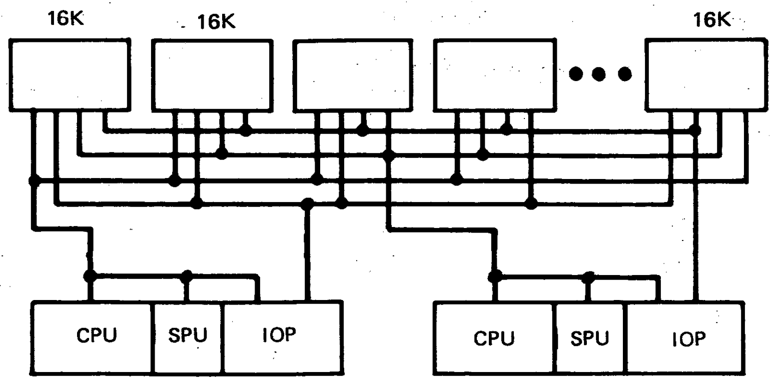


Figure 1-1. Comparison of Sample System Configurations

SECTION 2. MULTIPROGRAMMING CAPABILITY

PRIORITY DEMAND OPERATION ENHANCED BY MULTIPROGRAMMING

Featuring a multiprogramming capability that enables it to execute 64 different programs on a priority demand basis, the computer is uniquely tailored to tactical command and control applications.

One of the most important and unique features of the computer is a hardware implemented system for control of program switching in a multiprogramming environment. Its function and its tactical data system advantages have been proven in operational systems.

The computer's built-in multiprogramming feature allows the processor to execute up to 64 different programs (00 through 77_8 in octal representation), one at a time, on a priority demand basis. Real-time clocks and interval timers under software control are also provided to implement time-sliced multiprogramming if required. The principal feature of the design is that the control for switching from one program to another is built into the hardware with a software override provision.

Each of up to 64 programs is assigned a program level number that corresponds to a bit position within a 64-bit program status register. Bits are set or reset in this register by program or by the I/O unit (Table II-1). A second 64-bit register (the program enable register) provides logical masking on the program status register. Bits in this register are set or reset by programmed instructions. Together, these two 64-bit registers are called the program activity register (PAR). The most significant (highest numbered) one bit of the program status register which has a corresponding one bit in

Table II-1. Program Level Condition Due to Program Activity Register

Situation	PS Bit	PE Bit	Description
Inactive	0	0	Program level is disabled and idle
Waiting	0	1	Program is enabled and is waiting for a response from an external equipment or another program level
Stimulated	1	0	Equipment responded or program was stimulated but the program has not been enabled
Suspended	1	1	Program suspended because program of higher priority is currently being executed or program level change lock has been set
Operating	1	1	Program level is operating

PS: Program Status Bit
PE: Program Enable Bit

3307-6

the program enable register determines the currently active program, as shown in Table II-1. The PAR is held in eight half-word locations in the base memory bank, and is accessible by any program. (The base memory bank contains the PAR along with dedicated addresses for inactive process registers and I/O control words.)

A six-bit number that represents the active program level is logically generated and held in a register called the active program level register. The contents of this register are available to programs by execution of the Load Register Special instruction.

Nine program levels are used for special program functions. These are program levels 70-77 and 00. Program level 77 is reserved for power shutdown. When this condition is detected, program level 77 is automatically entered. Approximately 100 microseconds remain for program clean-up before power is lost.

Program level 76 is reserved for start-up of the computer when power is applied. Under this condition, computer control causes the CPU to enter and execute program level 76 if the system is in the automatic mode.

Program levels 70-73 have been designated as externally forced interrupt levels and may be entered directly via a hard-wired interrupt signal from an external device. The utilization of these levels for forced interrupts does not preclude their use within the normal priority auction configuration.

Program level 74 is entered automatically for a detected memory access violation, a memory parity error, a memory timeout, a program timeout, a device timeout, or an illegal operation code. Status bits are set preserving the source of the error such that the program can sample status and determine both the cause of the problem and the necessary corrective action. Information available to the program includes the error type, the device address, and the memory bank address.

Program level 74 can also be entered directly via a hard-wired interrupt signal from an external device. This permits level 74 to be used as either an additional externally forced interrupt level or as an error level entered either by an internally or externally detected error.

Program level 75 is used for a program trace routine when under control of the programmers' console.

Program level 00 is used for initial program execution following bootstrap program load by the IOU.

RESPONSE TO PRIORITY INTERRUPTS

The computer will respond rapidly to program interrupts in full accord with a preset ranking of program priorities.

The computer has a number of design features that facilitate the capability to respond to program interrupts on a priority basis without the use of complex executive decision operations. Sixty-four program interrupt levels are available, each having its own process registers and page control and address (PCA) registers. This enables 64 independent programs to be simultaneously available for computer operation. Each level has an assigned priority but only operates if that level is the highest priority demanding activation.

An interrupt level can be stimulated either by instruction, by interrupts from the I/O, or by an internally generated alarm. The I/O capability of the computer enables many external interrupts to be quickly serviced. This means that external device interrupts occur for data transfer completion and for interrupts initialed by the device itself. These operations typically require a minimum of program time. The program processing of I/O interrupts is executed on a variable priority basis.

Low-priority functions are interrupted in favor of higher-priority functions. After completion of the higher-priority function, the computer returns to the interrupted lower-priority function without delay and without program duplications. The computer provides the additional feature of a programmable priority level lock-out. This feature guarantees, if so desired, the execution of a sequence of instructions of a low-priority function without being interrupted by a higher-priority function. Normally, the lock-out is set only for the sequence of a few instructions. During lockout, all external interrupts are accepted but not executed. Low-priority functions will invariably receive quick attention since the computer's capability to process data exceeds the requirements of typical workloads by a comfortable margin.

Control over the programs which are operating or suspended is maintained in the status and enable registers. The status register contains 64 bits, where each position represents a program level. The enable register acts as a mask. It is also 64 bits in length and can be used to inhibit a program from operating. Thus, the program operating at any given time is the highest priority program having a coincident one bit in both the status and enable registers unless a program level lock is set. When a program is operating, the set of process registers accessed is that set which corresponds to the operating program level.

Each time that the contents of the status or enable register are accessed during the execution of specific instructions, the logic of the computer will test the program priority levels to determine whether or not a change of level is to occur. If a switch in programs is required, the hardware will preserve the state of the interrupted program and initiate the new program.

Completion of an I/O transfer, as well as device initiated action, may also cause an interruption of a current program. Under these circumstances, the termination word associated with each I/O channel will be accessed, the specified program level will set the corresponding bit in the status register, and a one will be placed in the proper position to indicate the reason for termination of the I/O transfer. The computer logic determines whether or not an interruption is to take place. If an interrupt is to occur, the procedure followed is the same as that described.

In multiprocessor configurations, the capability of interprocessor interrupts exists. If, during the execution of specific instructions, one processor accesses the base memory of another processor, the latter is forced into a program level change – normally to the level enabled by the calling processor. This interrupt is automatically initiated by the calling processor but may be locked out by the called processor.

Externally forced interrupt levels (70-73 and 74) are entered directly upon receipt of a hard-wired interrupt signal from an external device. The priority auction (PAR word search) that normally occurs in response to an external interrupt from the IOU or an internally generated interrupt is bypassed for the forced level interrupt and the program level, as specified by the received signal, is entered directly.

Forced interrupts can be inhibited by the program level lockout. However, an interrupt received during lockout will be saved and will initiate a level change when the lockout is reset.

A level entered in response to a forced interrupt can be interrupted by a higher priority interrupt without the loss of the lower level interrupt.

When several forced interrupts are received together, a priority selection will be made by the CPU such that the higher priority interrupts are processed first.

The priority interrupts may be caused by sources either external or internal to the computer. These interrupts are:

a. External interrupts (from each peripheral device via the IOU)

   (1) True interrupts (externally initiated)
   (2) Normal I/O termination
   (3) Error I/O termination

b. Internal interrupts

(1) Program termination
(2) Memory access violation
(3) Memory parity error
(4) Power interrupt
(5) Timeouts

c. Externally forced interrupt

d. Interprocessor interrupt

PROGRAM LEVEL CHANGE SHOWN AS A FUNCTION OF INTERRUPT

The computer's multiprogramming feature enables the central processing unit to execute 64 different programs on a priority demand basis.

Program level switching with software control is built into the hardware. Its functional advantages in tactical data systems have been proven in operational systems.

The sequence of a program level change is shown step by step in Figure 2-1. An operating program level is indicated by the most significant bit pair in the program activity register which contains a one in both the status and enable registers.

Assume that: (1) an interrupt changes the status bit for a program of higher priority from a zero to a one, (2) this program level is not masked, which is shown by a one in the enable register, and (3) the program level change lock is reset. At the end of the currently executed instruction, a level change is initiated. The active general registers are preserved in the general register area in the base memory bank. The assigned fixed area is determined by the program level which corresponds to the bit position in the program activity register as shown in steps 2 and 3 (Figure 2-1).

The new program level to be activated addresses the base memory bank, as shown in step 4. The set of general registers is transferred by step 5 into the active general registers. In addition, the page control and address information is moved by step 6 from the core memory to the active registers. Both active registers provide for faster access during instruction execution.

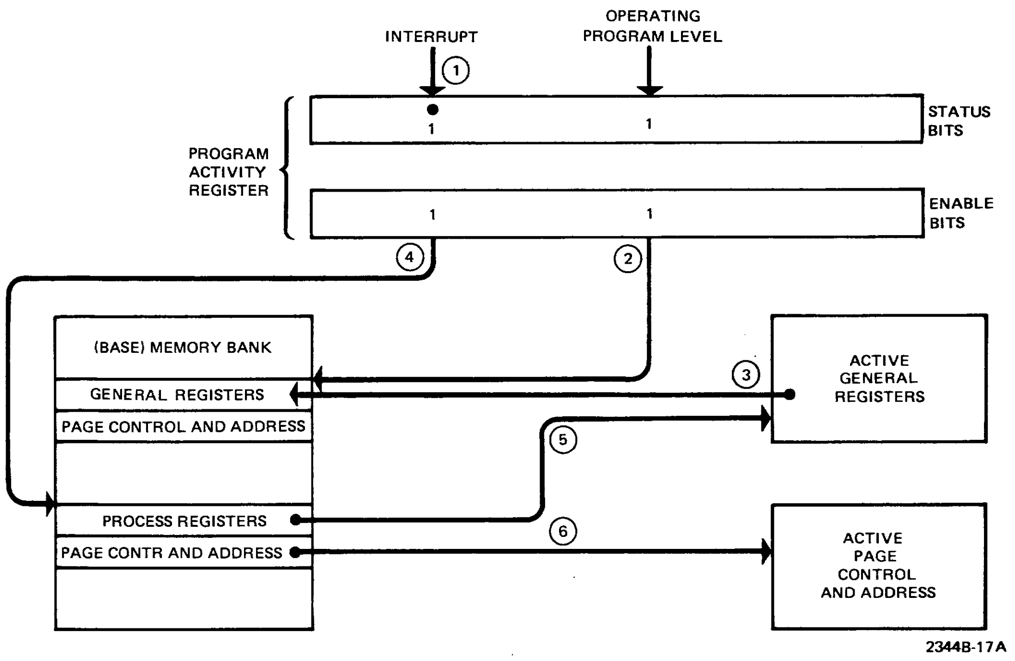


Figure 2-1. Program Level Change

PROGRAM LEVELS LINKED BY INSTRUCTIONS

The hardware/software interface provides a six-fold exit from a program level. Program levels can be linked by using the "call" instructions.

A program level change is initiated by: (1) external interrupt, (2) input/output termination, (3) program termination, (4) memory access violation or parity error, (5) functional time-outs, and (6) power interrupts. A level change occurs normally if the initiated program level is of higher priority, the program level has been enabled, and the program level change lock has been reset.

The program level lock can be set only by specific instructions, and can be set (or reset) by any program. If so desired, the execution of a sequence of instructions of a low-priority function can be completed without being interrupted by a higher priority function. Normally, the lock is set only for the sequence of a few instructions.

Any program is allowed to access the program activity register, which governs the program level change. Thus, any program can call any other program by setting the appropriate bits in the PAR and terminating itself. Specific instructions are available for this purpose. If the called program is the highest priority, it is called immediately; if not, it must wait.

PROTECTION PROVIDED AGAINST POWER LOSS

The on-off sequencer provides for orderly shutdown and startup procedures during power transients as well as application or removal of primary power.

When primary power is initially applied or when power is recovering from a transient, the on-off sequencer provides a System Reset which initializes the CPU and inhibits memory operation while the logic voltage is coming up. When the logic voltage reaches its nominal level, the on-off sequencer removes the System Reset and Memory Inhibit and provides a signal to the CPU that:

a. Sets up a program level change to program level 76

b. Sets the location register to 1610

c. Sets the CPU to the run condition

When a power failure has been detected, the on-off sequencer notifies the CPU, which, in turn

a. Completes the instruction being executed

b. Stores the registers associated with the current program level

c. Initiates a program level change to level 77

Approximately 100 microseconds are then left for the necessary clean-up or bookkeeping operations as specified by the programmer. After this 100-microsecond period, the Memory Inhibit is activated since voltage tolerances can no longer be guaranteed.

Once the shutdown sequence has begun, "start-up" will not be initiated until System Reset has been activated, regardless of the duration of the transient.

MULTIPROGRAMMING CAPABILITY ENHANCED BY FOUR REAL-TIME CLOCKS

Real-time clocks provide prescribed time-base for calling programs involved in peripheral service, system monitors, or fault detection.

The L-304H provides four independent programmable real-time clocks (RTC). Each RTC is accessed via an I/O channel which is serviced by the IOU in the alarm mode. The associated key and termination words provide the agency for interval counting and program interrupts after the specified time has elapsed. These program interrupts can be selectively disabled and temporarily locked out.

Each RTC has a period of 1 millisecond, and each channel has a capacity of 4.1 seconds. Each RTC channel is staggered by 250 microseconds in requesting IOU service.

In addition, the RTC logic provides a monitor for IOU operation: if one RTC channel has not been serviced by the time the next clock is due (a period of 250 microseconds), the IOU timeout bit is posted in the status register, a System Reset is effected, and the CPU is forced to the error program level, 74.

MULTIPROGRAM CAPABILITY FACILITATES PROGRAM LOAD

Program level 00 is reserved for program loading after the IOU completes load of bootstrap program.

After the initial bootstrap program has been loaded by the IOU, the CPU, deactivated by the program load switch and inhibited during the load operation, is started by the IOU. The CPU will obtain the program's starting location from the PLR associated with level 00 where the IOU has stored it during the bootstrap operation. The CPU will then automatically enter level 00 and execute the bootstrap program stored at that level.

PROTECTION PROVIDED AGAINST INTERPROGRAM INTERFERENCE

A technique utilizing memory protection minimizes the possibility of interprogram interference.

The multiprogram use of a computer provides opportunity for interprogram interference by accident or without authorization. To minimize this possibility, memory-protect capabilities are provided in the L-304H Computer.

The memory-protect feature implemented by the PCA registers controls the memory access for instructions and operands as well as a write protection. The mechanization of the computer provides the detailed features in hardware with appropriate software control by program.

A violation of the privilege feature will automatically cause a program level interruption to a specific program level. Control can then be returned to the executive program. The memory protect code is:

00	–	Instruction Fetch, Operand Read and Write permitted
01	–	Only Instruction Fetch and Operand Read permitted
10	–	Only Operand Read permitted
11	–	No Access permitted

SECTION 3. EXTENDED MEMORY ADDRESSING

EXTENDED MEMORY ADDRESSING OPTION PROVIDES POWERFUL TOOL IN MULTIPROGRAM ADDRESSING

The EMA option adds memory protection and parity as well as extended addressing for each program level of the L-304H.

The standard memory addressing capability of the computer is 32,768 words. The EMA (extended memory addressing) option expands this capability to 131,072 words while segmenting all memory into pages of 2048 words. In addition, each page is subject to programmable access control.

This extended addressing and access control is provided by a set of 16 8-bit bytes associated with each of the 64 program levels. These sets, called page control and address (PCA) registers, are stored in reserved locations in the computer base memory while the associated program level is not active. PCA registers for the active program level are held in a fast access "scratch pad" memory within the CPU. During each program level change, the required PCA registers are transferred from the base memory to the scratch pad by the processor hardware.

There is no program access to the scratch pad, but all PCA registers in the base memory are accessible by any program level. Each PCA byte consists of a 6-bit page field and a 2-bit access control field (Figure 3-1). Whenever the computer generates a 15-bit address, the four most significant bits of the address select one of the 16 PCA bytes from the scratch pad memory. The page field, which includes the memory bank and page addresses, is appended to the most significant end of the remaining 11 bits, forming the 17-bit word address. Since the IOU is independent of program levels, the EMA is not active during IOU memory cycles.

At the same time, the access control field is compared with the memory mode lines and if the pending operation is not allowed, the memory request is inhibited, the CPU is notified, and a level change (to error level 74) is initiated. The memory is not accessed. Table III-1 shows the interpretation of the excess control bits.

An expanded EMA option provides the computer with the addressing capability of 2,097,152 words.

MEMORY PARITY GENERATION AND CHECK PROVIDED

Parity on a byte basis is checked or generated for every memory cycle.

If faulty parity is detected, the CPU inhibits execution of the instruction and initiates a transfer to the error program level, 74. If the check fails during a Read-Modify-Write memory cycle, the data, along with the incorrect parity bit is restored to the memory before the transfer is initiated.

Table III-1. Interpretation of Access Control Bits

PAGE ACCESS CONTROL BITS	NAME	ACTION PERMITTED			ACTION INHIBITED		
00	READ-WRITE ACCESS	FETCH INSTRUCTION	READ OPERAND	WRITE OPERAND	–	–	–
01	READ ACCESS	FETCH INSTRUCTION	READ OPERAND	–	–	–	WRITE
10	READ DATA ACCESS	–	READ OPERAND	–	FETCH	–	WRITE
11	NO ACCESS	–	–	–	FETCH	READ	WRITE

2344B-1

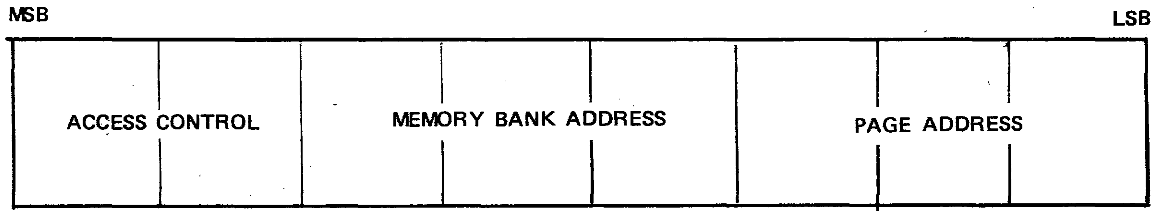


Figure 3-1. PCA Byte Format

SECTION 4. REGISTER ORGANIZATION

GENERAL PURPOSE PROCESS REGISTERS PROVIDE MULTIPURPOSE USE

A set of eight process registers is provided for each of 64 program levels. These registers may be used as index registers, accumulators, and other programming functions.

The computer provides a multiprogram capability that is program-controlled with a priority demand technique. Each of 64 program levels is assigned its own set of process registers. These registers are held in the computer's base memory bank for all temporarily inactive program levels; and in high-speed, scratch-pad memories for currently active program levels. Whenever a program level change occurs, the currently active registers are written, automatically, back into their reserved and dedicated locations within the base memory bank and the new set of registers is loaded into the high-speed memories.

The high-speed memories are constructed from MSI random access memory chips. An array of these elements provides a small memory of 16 half words. Access to this memory is less than 40 nanoseconds.

Use of the process registers as both index registers and accumulators eliminates the requirements for a number of load and store instructions from a program. Often, index quantities are derived from arithmetic operations. Thus, the result is directly available as an index in subsequent instructions.

The use of the process registers as multiple accumulators provides the opportunity to leave partial results in the accumulator without having to preserve them in memory. Partial results can be combined with instructions using the special address feature, which allows register-to-register operations without time consuming memory access.

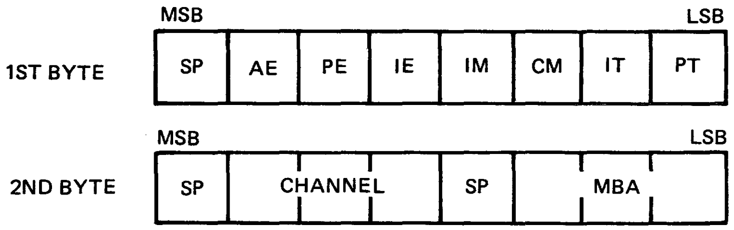
The process registers of any program level may be addressed by any other program by addressing the base memory bank. The process registers of the active program may be addressed by the instruction's S field for indexing operations, by the instruction's H field for use as an accumulator, and by the instruction's operand address. Special addresses ($00\text{-}07_8$) are recognized as process register addresses instead of memory locations.

STATUS REGISTER

The status register contains information pertaining to error/fault conditions existing within the system.

The register is organized as two 8-bit bytes, and may be interrogated by a program via the Load Register Special instruction. This operation will load the status register into the designated process register.

The format of the two bytes is shown below



PT	–	Program Time Out
IT	–	IOU Time-Out
CM	–	CPU/Memory Time-Out
IM	–	IOU/Memory Time-Out
IE	–	IOU Memory Parity Error
PE	–	CPU Memory Parity Error
AE	–	Access Violation
SP	–	Spare
MBA	–	Current Memory Bank Address or Error Memory Bank Address
Channel	–	Active IOU Channel or Error IOU Channel

PREASSIGNED LOCATIONS IN BASE MEMORY

Each base memory has several pre-assigned locations used as temporary storage for registers control words, etc., while that program level, channel, or function is not active.

Figure 4-1 shows the organization of the locations in the base memory. Each location is defined as follows:

General Purpose Process Registers. Eight half words to be used by each program level for accumulators, index registers, etc.

I/O Control Words. Eight key and termination words for each I/O channel. (See section on IOU).

Program Location Registers. One half-word for each program level indicating the current entry point into that program.

Program Activity Registers. Four words indicating the enable and status conditions for each program level.

Page Control and Address Words. Sixteen 8-bit bytes for each program level controlling the memory access and paging for that level.

Note that although locations 01610-01777 are unassigned, the processor, on automatic start-up, is forced to run starting at location 01610.

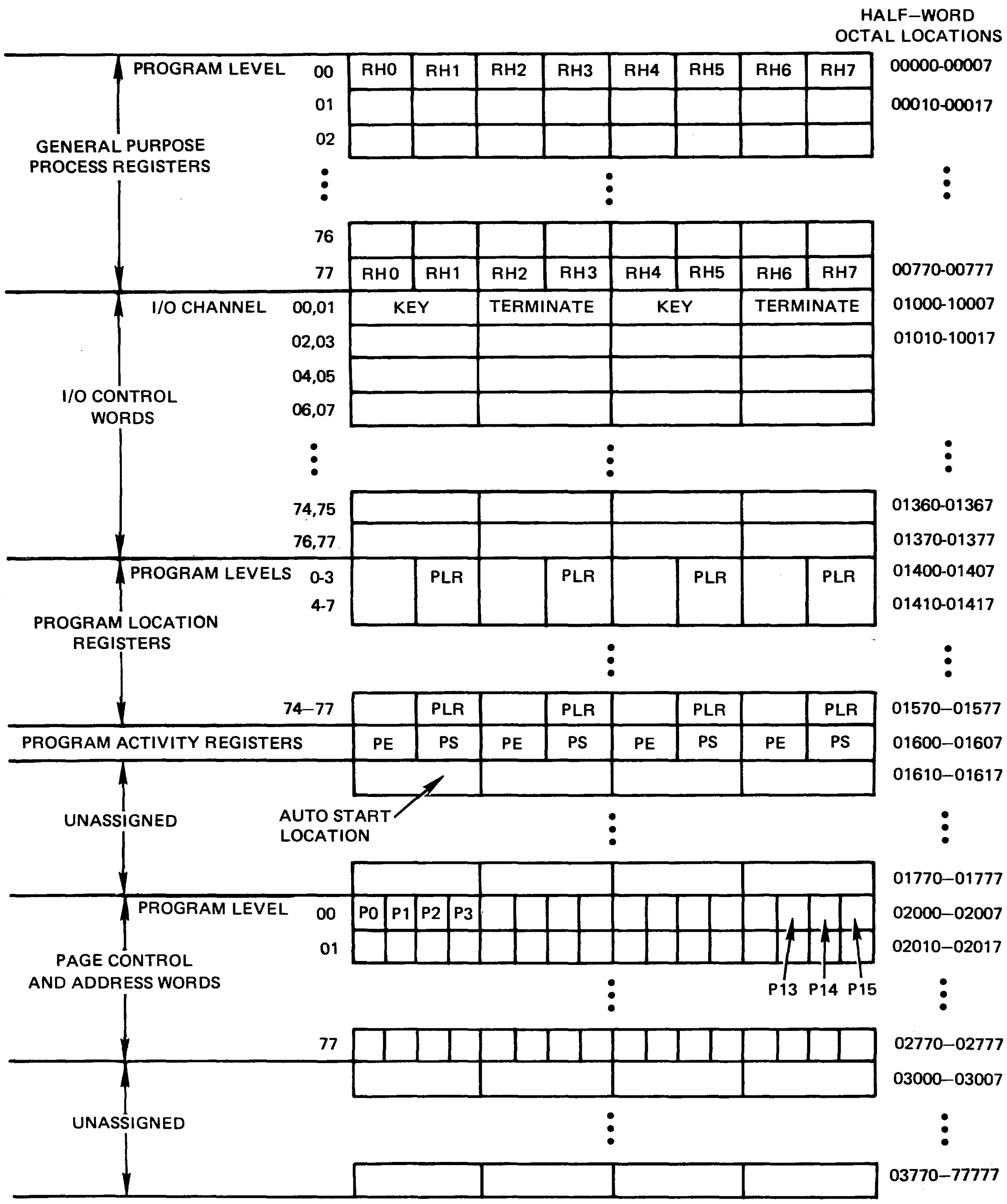


Figure 4-1. Base Memory Map

SECTION 5. INSTRUCTION REPERTOIRE

INSTRUCTION REPERTOIRE TAILORED TO TACTICAL REAL-TIME APPLICATIONS

The computer instruction repertoire provides a number of special instructions to specifically reduce the reaction time of tactical command and control systems.

The computer has been designed to satisfy the requirements of tactical real-time command and control systems. Toward this end, an instruction repertoire has been prepared which provides a number of special instructions to decrease reaction time. The combination of this type of instruction list and the flexibility of programming offers a powerful tool for performing real-time missions. The selection of each instruction was made after intensive trade-off analysis and discussions which reflected Litton's extensive experience in tactical data processing requirements. The instructions may be grouped in seven classes: arithmetic, data manipulation, data handling, jump, transfer, input/output, and miscellaneous instructions. The arithmetic instructions encompass a comprehensive computational capability and include Add, Add Double Precision, Subtract, Subtract Double Precision, Multiply, and Divide with a number of variations to simplify the programming operation. The jump group includes compare instructions which allow algebraic, logical, and gated (within limits) comparisons.

Data manipulation instructions include logical operations which enable data to be matched and merged, and act as an adjunct to the data handling capability. The Move and Insert and the Move and Zero instructions give an excellent capability to move groups of bits of arbitrary length about the process registers. A full complement of shift instructions, including a reflect operation is also provided. Four instructions are incorporated which facilitate set and reset of individual bits within a data word.

The Store all Zero instruction is used to clear memory or register cells.

The Transfer and Jump instructions include a wide range of control instructions to assist the programmer in organizing and controlling his program sequence and provide a capability to respond to external stimuli. General purpose (GP) register test instructions are used to test and modify registers and provide program looping. Instructions are provided to test a single bit for its status. A Load Special Instruction provides internal/external status to the programmer to facilitate fault diagnosis. The Enque/Deque instruction provides a simplified method for handling common data files.

Input/output instructions initiate and enable communication to be set up with peripheral devices for automatic independent data transfers. Two instructions allow single transfers between a device and a GP register.

The L-304H was designed to accommodate the addition of a special processing unit tailored to a customer's specific application. These SPUs offer the capability to perform macro instructions and allow execution of complex algorithms efficiently under software control. The SPU has direct access to both the memories and the hardware process registers. Thirty-nine operation codes have been reserved for the macro operations.

An example of an SPU is the extended performance arithmetic unit which provides double precision multiply and divide operations and floating point multiply, divide, add, subtract, and compare instructions.

A summary of the 128 instructions, by type, includes:

Arithmetic: 18 fixed point

Data Manipulation: 6 logic, 6 shift, 8 set/reset bit

Data Handling: 7 load (registers), 2 store (registers), 4 move, 2 exchange, 1 store zero

Transfer (Branch): 4 GP register test, 3 control transfer, 4 GP register modify

Jump: 7 compare, 8 test bit and skip on match

Miscellaneous: 5 miscellaneous

Macro: 39 codes available

FLEXIBILITY OF DATA WORD FORMATS

The data word size in the Litton computer is determined at the programmer's option.

The programmer may select data words of a single bit, a "byte" of eight bits, a half-word of 16 bits, and a full word of 32 bits. The data word size is a function of the programmer's option on the instruction to be executed. On arithmetic and logical instructions, either half-word or word operations are allowed. Special instructions are provided for test and modification of single bits within a half-word. Special byte handling instructions are also provided. Full-word products are generated on multiply instructions; full-word dividends are used in divide instructions; full-word register pairs are allowed in shift instructions, and addition and subtraction can be performed on full-word operands.

Memory addresses are considered as half-word addresses because the instruction's operand address field contains a single bit (in bit position 0, called the W field). This bit will select either the left-most 16 bits of a selected word if it is zero (W = 0), or it will select the right-most 16 bits of a selected word if it is one (W = 1). On word operations, the W bit of the instruction is ignored, and the address is treated as an even number. Therefore, all word addresses are even numbers.

Bytes of eight bits are selected on half-word instructions as either the "upper byte" or "lower byte" by the instruction's operation code. Single bit positions of a byte are selected by use of the instruction word's H field.

Bytes of a variable length and position are selected and controlled during the move instructions by a mask contained in the instruction's CA field.

The storage of partial words and full words within the process registers and memory is shown in Figures 5-1 through 5-4.

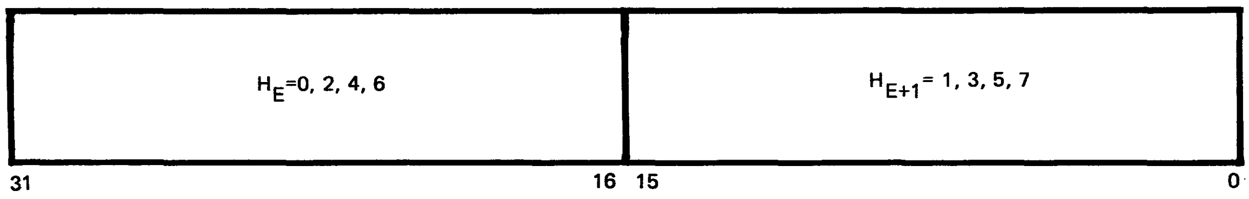


Figure 5-1. Process Register Pair

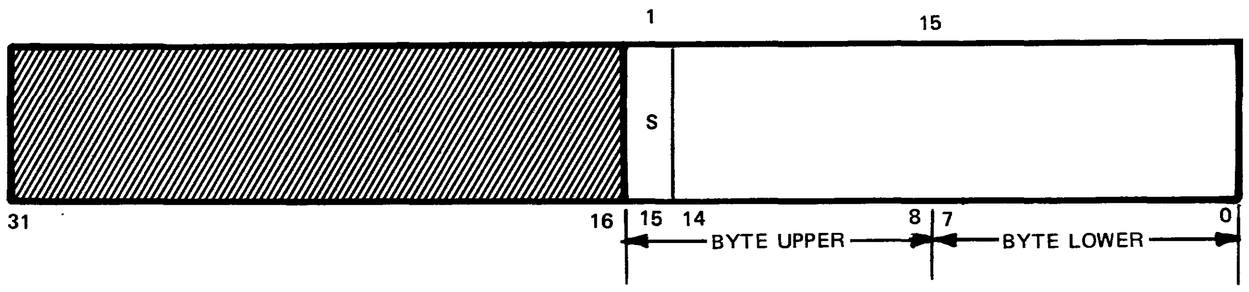


Figure 5-2. Operand Selected from Memory Output when W = 1

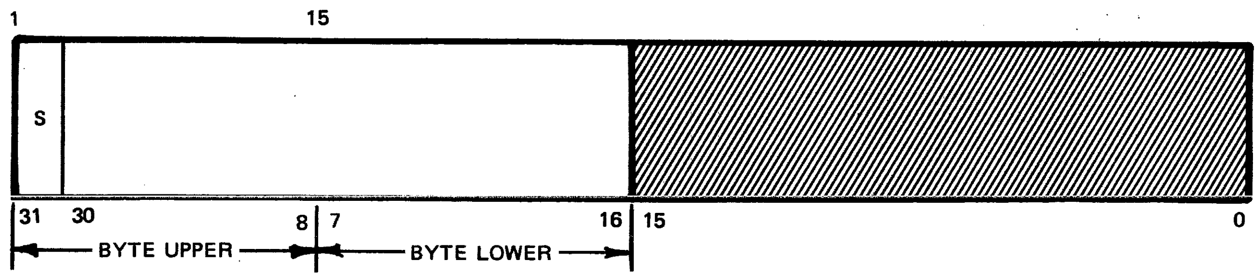


Figure 5-3. Operand Selected from Memory Output when W = 0

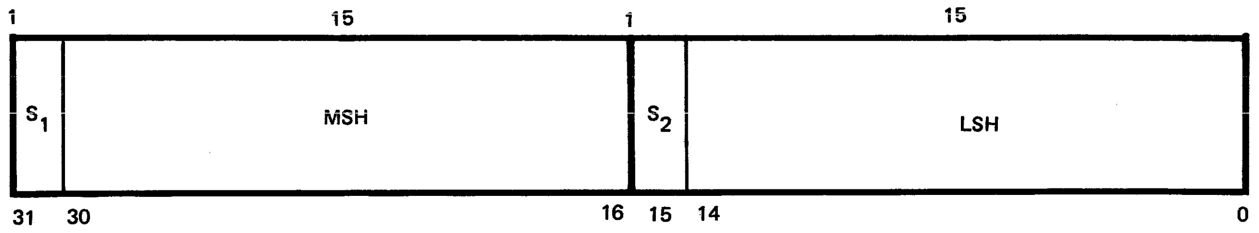


Figure 5-4. Full Word Format

A data word may contain a numerical or logical quantity. Numerical quantities are treated as signed integers. Logical quantities include unsigned numbers (e.g., addresses) or collections of individual bits and fields.

In numerical words, the most significant bit is treated as the algebraic sign. If this bit is a zero, the sign is positive. If this bit is a one, the sign is negative. The sign bit of a numerical half-word is automatically sign extended (value repeated) to the left 16-bit positions when a numerical full-word operand is required.

Negative numbers are represented in two's complement form. Two's complement representation has advantages in arithmetic compatibility with unsigned numbers and in elimination of a negative zero value.

A 32-bit algebraic operand is provided with two sign bits (S1 and S2) which are always equal (Figure 5-4).

INSTRUCTION WORD FORMAT

The Litton computer operates with a 32-bit instruction word containing five fields.

The normal use and definition of each instruction word field is as follows (Figure 5-5):

a. F Field – This 7-bit field is the instruction operation code. The operation code is represented by a 3-digit octal number.

b. H Field – The H field is a 3-bit binary number that selects one of eight process registers to be used as the accumulator by the instruction. Process registers are addressed by H = 0, 1, 2, · · ·, 7 on all program levels.

c. M Field – The M field is a 3-bit code that provides up to eight instruction address options as follows:

M = 0, Direct Address
M = 1, Direct Address with Indexing
M = 2, Literal
M = 3, Literal with Indexing
M = 4, Indirect
M = 5, Indexed, Indirect
M = 6, Indirect, Indexed
M = 7, Relative with Indexing Option

These options are described in more detail in a subsequent discussion of addressing modes.

d. S Field – The S field is a 3-bit field that selects one of the eight process registers to be used as an index register on modes M = 1, 3, 5, 6, and 7. The S field addresses the same set of process registers as the H field on a program level. A different set of eight process registers is provided for each of 64 program levels.

e. CA Field – The operand address field, CA, is a 16-bit field that may either be an address of a word in memory or may be a 16-bit operand itself. The CA field is used as an operand in the literal address modes, M = 2 or 3.

When the CA field is used as an address, it is considered to consist of three subfields: D, A, and W. These three subfields provide memory address extension of up to 18 bits for addressing 16-bit words or 17 bits for addressing 32-bit words.

f. D Subfield – The D subfield is a 4-bit binary number that selects one of 16 page control and address registers. The selected register contains six bits which are appended to the most significant end of the A subfield to yield a 17-bit operand address. Although an overflow out of the D subfield may occur during indexing, a carry is not allowed to propagate from the D subfield to the S field.

g. A Subfield – The A subfield is an 11-bit binary address. The 11-bit address will select one of 2048 32-bit words within a memory module. The particular memory module is selected by the three MSBs of the 6-bit page control and address register (described previously).

h. W Subfield – During an operand fetch, the 1-bit W subfield specifies left or right half of the 32-bit memory output to be used as a 16-bit operand. If W is a zero, the left half of the word is used. If W is a one, the right half is used. The W subfield is ignored during an instruction fetch.

LEGEND:

F:	OPERATION CODE
M:	ADDRESSING MODE SELECTOR
H:	ACCUMULATOR DESIGNATOR
S:	INDEX DESIGNATOR
CA:	OPERAND ADDRESS FIELD
D:	PAGE DESIGNATOR
A:	WORD ADDRESS DESIGNATOR
W:	HALFWORD ADDRESS DESIGNATOR

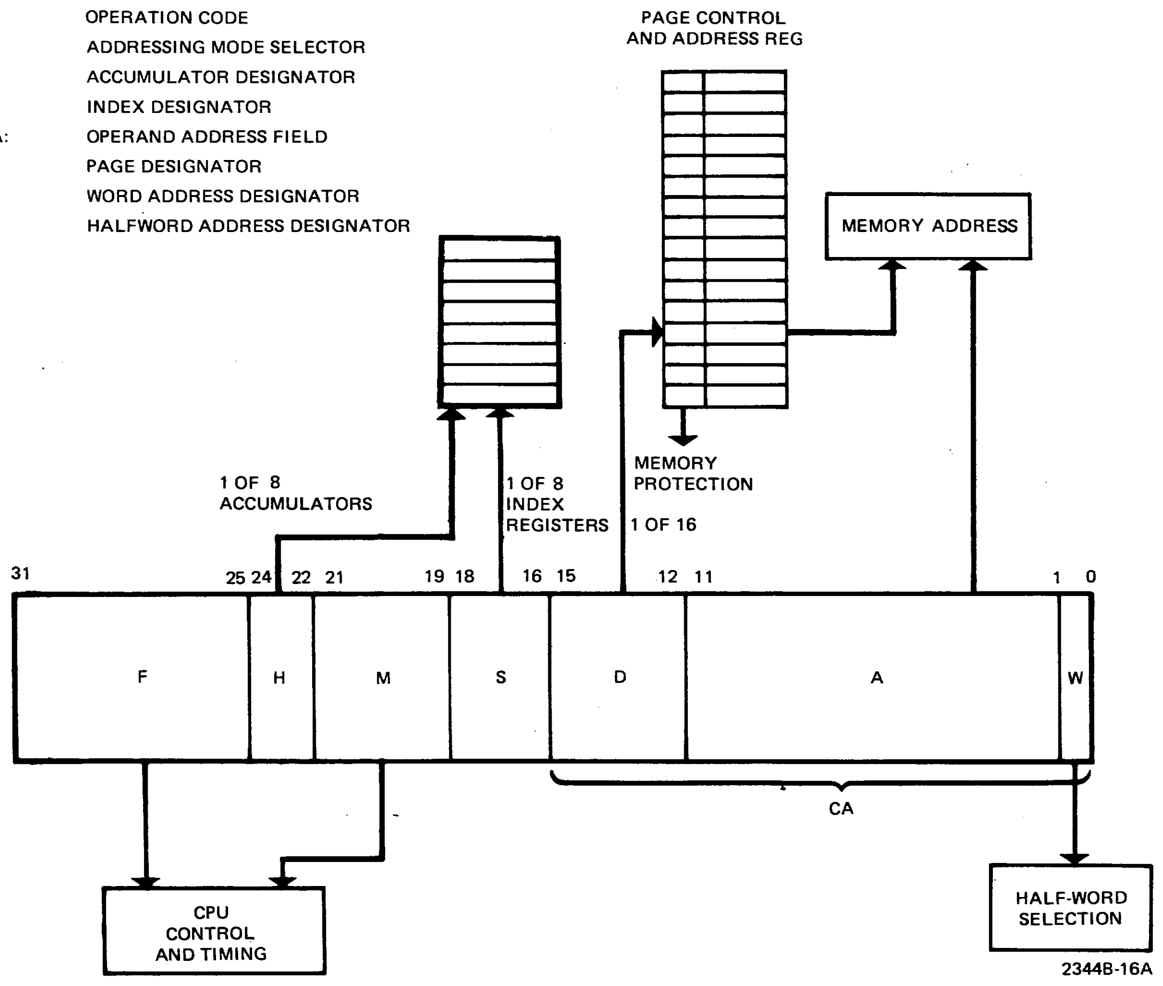


Figure 5-5. Instruction Word Format

ADDRESSING MODES ENHANCE PROGRAMMING

The availability of eight addressing modes in the Litton computer provides the programmer with a powerful tool.

The eight addressing modes may be grouped into four major classes: literal, direct, relative, and indirect. Indexing is applied appropriately to each of the modes.

The literal mode uses information contained within the instruction format as an operand. Therefore, a memory access is eliminated which results in a saving in storage space as well as in execution time.

The direct mode is the most commonly used addressing mode. In this mode, the operand address is contained within the instruction.

The relative mode provides an addressing capability which allows referencing to the location of the current instruction. A relocation of a program will not disturb the relationship between the instruction and the referenced operand. It is recommended that the operand be located close to the instruction since an insertion or deletion of instructions in a program requires modification of the relative address.

The indirect addressing mode with indexing is applicable to reentrant subroutines because the index is added after accessing of the indirect address.

Double indexing is provided through use of the indirect options when CA = 0-7. Under those conditions, CA will specify a second index register.

INSTRUCTION OPTIONS PROVIDED BY MULTIMODE ADDRESSING

Eight simply structured addressing modes provided by the instruction's M field allows flexibility in operand selection.

The eight address modes combined with the 128 instruction codes give the computer several hundred useful instruction options. The M field is used to select the address option. These options are provided in almost all instructions. Table V-1 summarizes the possible operand addresses that can be generated with the eight modes.

The addressing modes are defined as follows:

a. M = 0, Direct Address – The CA field of the instruction word is not modified. The D subfield directly or indirectly selects the specific memory module. The A and W subfields specify a half-word operand address in the specified memory module. On transfer instructions or full 32-bit operands, W is not used.

b. M = 1, Direct Address with Indexing – The CA field of the instruction word is added to the contents of the index register selected by the S field. Overflow on this addition is not detected. The sum replaces the CA field within the instruction word register, and is used as the new operand address field as in M = 0.

c. M = 2, Literal – The CA field represents a 16-bit half-word. This word may be an operand, mask, instruction address, or shift number.

d. M = 3, Literal with Indexing – The CA field represents a 16-bit half-word as in mode 2. The CA field is added to the contents of the process register selected by the S field before the instruction operation takes place. This provides a useful double operation on many instructions. Overflow is detected on this addition for those instructions which could cause arithmetic overflow (F = X10 through X17, X30, X31). If it does not occur for instructions F110-F117, F130, and F131, the next instruction in sequence is skipped. The sum replaces the CA field within the instruction word register, and is used as the new instruction operand, as in M = 2.

e. M = 4, Indirect – The contents of the memory word that are addressed by the 16-bit address. This "indirect" address replaces the CA field within the instruction word register and is used as in M = 0.

f. M = 5, Indexed, Indirect – The 16-bit CA field of the instruction word is added to the contents of the process register selected by the S field. Overflow on this addition is not detected. The sum is used to address a 16-bit word in memory that replaces the CA field of the instruction in the instruction word register, and is used as the new operand address field as in M = 0.

g. M = 6, Indirect, Indexed – The 16-bit CA field of the instruction selects a 16-bit word in memory which is added to the process register selected by the S field. No overflow detection occurs. The sum replaces the CA field of the instruction in the instruction word register, and is used as the new operand address field, as in M = 0.

h. M = 7, Relative – This address option mode operates in two ways, depending upon the S field of the instruction word.

If the S field is all zeros, the contents of the instruction location register, LL, are added to the D and A subfields of the instruction. The W bit is not modified. No overflow is detected on this addition. The sum replaces the D and A subfields of the instruction within the instruction word register. This operation yields an address that is relative to the address on the next instruction in sequence.

If the S field of the instruction is not zero, the 16-bit CA field of the instruction word is added to the contents of the LL register. The sum is then added to the contents of the process register selected by the S field. This operation yields an indexed address that is relative to the address of the next instruction in sequence.

The modified CA field is then used to specify the operand address as in M = 0, unless it is a transfer type instruction (F = 34 through 36, 40 through 43), in which case the CA field is used as the operand as in M = 2.

LITTON COMPUTER CHARACTERIZED BY POWERFUL INSTRUCTION REPERTOIRE

The computer instructions include general-purpose as well as special instructions useful in real-time tactical applications.

The instructions, grouped into seven classes, are listed in Tables V-1 through V-8. Instruction execution times are based on address modes 0 or 1: direct with or without indexing.

The following symbols and notations are used frequently in the definition of the instructions.

Letter	Definition
CA	The instruction word's CA field. It is a 16-bit number that is used to address a location in memory or, in modes 2 and 3 (Literal), CA is used as a 16-bit operand.
d	Represents a full-word (32 bits) as in (H)d or (Y)d.
F	The F field of the instruction word defining the operation code.
H	The H field of the instruction word specifying the accumulator.
Hd	Hd or (H)d refers to a processor register pair. The LSB of the H field is ignored for instructions requiring the use of a processor register pair.
(H)	The contents of the process register that is selected by the H field of the instruction word.
H_6	Process register 6.
H_E	The even numbered register of a process register pair (0, 2, 4, 6) which contains the most significant half of a full-word.
H_{E+1}	The odd numbered register of a process register pair (1, 3, 5, 7) which contains the least significant half of a full word.
LL	Represents the contents of the instruction location register (15 bits). This number is the address of the next instruction in normal sequence.
LP	Program level register.
M	The M field of the instruction word specifying the address mode.
m-n	Denotes the bit positions of a word. $(Y)_{4-0}$ denotes the five LSBs of the operand.
n	The number of bit positions shifted on shift type instructions.

Letter	Definition
S	The S field of the instruction word specifying the index register.
(S)	The contents of the process register that is selected by the S field of the instruction word. A 16-bit word that is generally used for address indexing.
T	Transfer address.
SWJ	The set of three conditional jump switches.
SWH	The set of three conditional halt switches.
Y	Operand address.
(Y)	Operand - (Y) = Y for the literal mode.
$(Y)_B$	A single bit of the operand.
$(Y)_{BL}$	A single bit of the lower byte of the operand.
$(Y)_{BU}$	A single bit of the upper byte of the operand.
$\overline{(\ )}$	One's complement unless otherwise noted.
( )	Parentheses represent the contents of the memory location that is addressed by the word within the parentheses.
$\neq$	Not equal to.
$\lvert\ \rvert$	Absolute value.
$A \cdot B$	Logical AND function.
A/B	Logical inclusive OR function.
$A \oplus B$	Logical exclusive OR function.
A + B	Addition.
A - B	Subtraction.
A x B	Multiplication.
$A \div B$	Division.

Table V-1. Data Operand and Transfer Instruction Address

Mode			Action		
M-Field	S-Field	Name	Operand Address	Operand	Transfer and Execute Instruction Address
0	0-7	Direct	Y = CA	Z = (Y)	T = (Y)
1	0-7	Direct with Indexing	Y = CA + (S)	Z = (Y)	T = (Y)
2	0-7	Literal		Z = CA	T = CA
3	0-7	Literal with Indexing		Z = CA + (S)	T = CA + (S)
4	0-7	Indirect	Y = (CA)	Z = (Y)	T = (Y)
5	0-7	Indirect with Indexing	Y = (CA + (S))	Z = (Y)	T = (Y)
6	0-7	Indirect with Second Address Indexing	Y = (CA) + (S)	Z = (Y)	T = (Y)
7	0	Relative	Y = CA + LL	Z = (Y)	T = CA + LL
7	1-7	Relative with Indexing	Y = CA + LL + (S)	Z = (Y)	T = CA + LL + (S)

NOTES:

CA:	Contents of the Instruction Address Field	Y:	Value of the Operand Address
LL:	Contents of the Location Register	Z:	Value for the Operand
M:	Mode Designator	T:	Value for the Transfer Address
S:	Index Designator	( ):	Contents of

Table V-2. Class: Arithmetic Instructions (Sheet 1 of 2)

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
Arith	ADD	010	$(H) + (Y) \rightarrow H$	Add	1.60
	SUB	011	$(H) - (Y) \rightarrow H$	Subtract	1.60
	RAD*	012	$(Y) + (H) \rightarrow Y$	Replace Add	1.95
	RUB*	013	$(Y) - (H) \rightarrow Y$	Replace Subtract	1.95
	ADA	014	$(H) + \|(Y)\| \rightarrow H$	Add Absolute	1.60
	SBA	015	$(H) - \|(Y)\| \rightarrow H$	Subtract Absolute	1.60
	MPY	030	$(H_{E+1}) \times (Y) \rightarrow Hd$	Multiply	3.36
	DIV	031	$(H)d \div (Y) \rightarrow H_{E+1}$ Remainder $\rightarrow H_E$	Divide	7.20
Arith and Skip	ADD	110	$(H) + (Y) \rightarrow H$ Skip next location if <u>no</u> overflow	Add	1.60
	SUB	111	$(H) - (Y) \rightarrow H$ Skip next location if <u>no</u> overflow	Subtract	1.60
	RAD*	112	$(Y) + H \rightarrow Y$ Skip next location if <u>no</u> overflow	Replace Add	1.95
	RUB*	113	$(Y) - (H) \rightarrow Y$ Skip next location if <u>no</u> overflow	Replace Subtract	1.95
	ADA	114	$(H) + \|(Y)\| \rightarrow H$ Skip next location if <u>no</u> overflow	Add Absolute	1.60

*Literal modes 2 and 3 will cause the instruction to act as a NOP.

Table V-2. Class: Arithmetic Instructions (Sheet 2 of 2)

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
Arith and Skip	SBA	115	(H) − |(Y)| → H Skip next location if no overflow	Subtract Absolute	1.60
	ADP	120	(H)d + (Y)d → Hd Skip next location if no overflow	Add Double Precision	1.92
	SDP	121	(H)d − (Y)d → Hd Skip next location if no overflow	Subtract Double Precision	1.92
	MPY	130	(H_{E+1}) X (Y)→Hd Skip next location if no overflow	Multiply	3.36
	DIV	131	(H)d ÷ Y → H_{E+1} Remainder → H_E Skip next location if no overflow	Divide	7.20

Table V-3. Class: Data Manipulation Instructions (Sheet 1 of 3)

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
Logic	EOR	020	$(H) \oplus (Y) \rightarrow H$	Exclusive Or	1.60
	IOR	021	$(H) / (Y) \rightarrow H$	Inclusive Or	1.60
	AND	022	$(H) \cdot (Y) \rightarrow H$	Logical And	1.60
	RER*	024	$(Y) \oplus (H) \rightarrow Y$	Replace Exclusive Or	1.95
	RIR*	025	$(Y) / (H) \rightarrow Y$	Replace Inclusive Or	1.95
	RAN*	026	$(Y) \cdot (H) \rightarrow Y$	Replace Logical And	1.95
Shift	SLL	044	(H)d shifted logically, left, circularly n places as specified by $(Y)_{4\text{-}0}$. $H_{31} \rightarrow H_0$ $H_{15} \rightarrow H_{16}$	Shift Long Left	2.72 8 Shifts
	NLL	045	(H)d shifted algebraically, left, open n places as specified by $(Y)_{4\text{-}0}$ or until $H_{31} \neq H_{30}$. Count residue $\rightarrow$ S. $H_{31} \rightarrow H_{31}$ $H_{15} \rightarrow H_{15}$ $H_{14} \rightarrow H_{16}$ $0 \rightarrow H_0$	Normalize Long Left	2.88 8 Shifts
	SLR	056	(H)d shifted logically, right, circularly n places as specifed by $(Y)_{4\text{-}0}$. $H_0 \rightarrow H_{31}$ $H_{16} \rightarrow H_{15}$	Shift Long Right	2.72 8 Shifts
	SAR	057	(H)d shifted algebraically, right, open n places as specified by $(Y)_{4\text{-}0}$. $H_{31} \rightarrow H_{31}$ $H_{15} \rightarrow H_{15}$ $H_{16} \rightarrow H_{14}$ $H_{31} \rightarrow H_{30}$	Shift Algebraically Right	2.72 8 Shifts

*Literal modes 2 or 3 will cause the instruction to act as a NOP.

Table V-3. Class: Data Manipulation Instructions (Sheet 2 of 3)

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
Shift	SNC	046	(H_{E+1})shifted logically, left, circularly n places as specified by $(Y)_{4-0}$. $H_{15} \rightarrow H_0$ (H_E) + number of ones shifted $\rightarrow H_E$	Shift And Count	2.72 8 Shifts
	RFT	047	(H_E) shifted logically left and (H_{E+1}) shifted logically right n places as specified by $(Y)_{4-0}$. $H_0 \rightarrow H_{16}$ $H_{31} \rightarrow H_{15}$	Reflect	2.72 8 Shifts
Set/Reset Bit	SBL*	060	a) 1 → (Y)BL as specified by H. b) An interprocessor interrupt is set if other PAR address is accessed.	Set Lower Bit	1.95
	SBU*	061	a) 1 → (Y)BU as specified by H. b) An interprocessor interrupt is set if other PAR address is accessed.	Set Upper Bit	1.95
	RBL*	062	0 → (Y)BL as specified by H	Reset Lower Bit	1.95
	RBU*	063	0 → (Y)BU as specified by H	Reset Upper Bit	1.95
	SBL*	160	a) 1 → (Y)BL as specified by H. b) External interrupt lockout is reset c) An internal interrupt is set if PAR address is accessed. d) An interprocessor interrupt is set if other PAR address is accessed	Set Lower Bit	1.95
	SBU*	161	a) 1 → (Y)BU as specified by H b) External interrupt lockout is reset c) An internal interrupt is set if PAR address is accessed d) An interprocessor interrupt is set if other PAR address is accessed	Set Upper Bit	1.95

*Literal modes 2 or 3 will cause the instruction to act as a NOP.

Table V-3. Class: Data Manipulation Instructions (Sheet 3 of 3)

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
Set/Reset Bit	RBL*	162	a) $0 \rightarrow$ (Y)BL as specified by H b) External interrupt lockout is reset c) An internal interrupt is set if PAR address is accessed	Reset Lower Bit	**1.95**
	RBU*	163	a) $0 \rightarrow$ (Y)BU as specified by H b) External interrupt lockout is reset c) An internal interrupt is set if PAR address is accessed.	Reset Upper Bit	**1.95**

*Literal modes 2 or 3 will cause the instruction to act as a NOP.

Table V-4. Class: Data Handling Instructions (Sheet 1 of 3)

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
Load Register	LDR	004	$(Y) \rightarrow H$	Load Register	1.60
	LDD	006	$(Y)d \rightarrow Hd$	Load Double	1.76
	LDA	016	$\lvert(Y)\rvert \rightarrow H$	Load Absolute	1.60
	LDA	116	$\lvert(Y)\rvert \rightarrow H$ Skip next location if <u>no</u> overflow: $(Y) \neq -1$.	Load Absolute	1.60
	LDC	017	$\overline{(Y)} \rightarrow H$ NOTE: 2's complement	Load Complement	1.60
	LDC	117	$\overline{(Y)} \rightarrow H$ NOTE: 2's complement Skip next location if <u>no</u> overflow: $(Y) \neq -1$.	Load Complement	1.60
	LRS	104	LL/LP/base memory address/status $\rightarrow$H as specified by the S field. The M and CA fields are not used by this instruction.	Load Register Special	1.00
Store Zeros	STZ	072	$0 \rightarrow (Y)$	Store all Zeros	1.95
Store Register	STR*	005	$(H) \rightarrow Y$	Store Register	1.95
	STD*	007	$(H)d \rightarrow Yd$	Store Double	1.95
Exchange	EXC*	002	$(Y) \rightarrow H$ $(H) \rightarrow Y$	Exchange	1.95
	EXD*	003	$(Y)d \rightarrow Hd$ $(H)d \rightarrow Yd$	Exchange Double	2.27

*Literal modes 2 or 3 will cause the instruction to act as a NOP.

Table V-4. Class: Data Handling Instructions (Sheet 2 of 3)

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
Move	MVI	071	a) (H) shifted logically right, circularly n places as specified by M field. $H_0 \rightarrow H_{15}$ b) $[(H)\text{ shifted} \cdot C_A] \,/\, [(S) \cdot \overline{CA}] \rightarrow S$ c) $(H) \rightarrow H$ unless $H = S$ NOTES: 1. Address options do not exist for this instruction. 2. CA = mask	Move and Insert	1.92 4 Shifts
	MVI	171	a) M = 0 (H) shifted logically left, circularly 8 places. $H_{15} \rightarrow H_0$ $M \neq 0$ (H) shifted logically left, circularly n places as specified by the M field. $H_{15} \rightarrow H_0$ b) $[(H)_{\text{shifted}} \cdot C_A] \,/\, [(S) \cdot \overline{CA}] \rightarrow S$ c) $(H) \rightarrow H$ unless $H = S$ NOTES: 1. Address options do not exist for this instruction. 2. CA = mask	Move and Insert	2.24 8 Shifts
	MVZ	070	a) (H) shifted logically right, circularly n places as specified by the M field. $H_0 \rightarrow H_{15}$ b) $(H)\text{ shifted} \cdot CA \rightarrow S$ c) $(H) \rightarrow H$ unless $H = S$ NOTES: 1. Address options do not exist for this instruction. 2. CA = mask	Move and Zero	1.60 4 Shifts

Table V-4. Class: Data Handling Instructions (Sheet 3 of 3)

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
Move	MVZ	170	a) M = 0 (H) shifted logically left, circularly 8 places. $H_{15} \rightarrow H_0$ M ≠ 0 (H) shifted logically left, circularly n places as specified by the M field $H_{15} \rightarrow H_0$ b) (H) shifted · CA → S c) (H) → H unless H = S NOTES: 1. Address options do not exist for this instruction 2. CA = mask	Move and Zero	1.92 8 Shifts

Table V-5. Class: Jump Instructions (Sheet 1 of 3)

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
Algebraic Compare	JTW	037	$(Y) < (H) : LL + 2 \rightarrow LL$ (next loc) $(Y) = (H) : LL + 4 \rightarrow LL$ $(Y) > (H) : LL + 6 \rightarrow LL$	Jump Three Ways	1.60 $Y \leqslant H$ 1.76 $Y > H$
	CJL	050	$(Y) \geqslant (H) : LL + 2 \rightarrow LL$ (next loc) $(Y) < (H) : LL + 4 \rightarrow LL$	Compare, Jump if Less	1.60
	CJE	051	$(Y) \neq (H) : LL + 2 \rightarrow LL$ (next loc) $(Y) = (H) : LL + 4 \rightarrow LL$	Compare, Jump if Equal	1.60
	CJU	052	$(Y) = (H) : LL + 2 \rightarrow LL$ (next loc) $(Y) \neq (H) : LL + 4 \rightarrow LL$	Compare, Jump if not Equal	1.60
	CJG	053	$(Y) \leqslant (H) : LL + 2 \rightarrow LL$ (next loc) $(Y) > (H) : LL + 4 \rightarrow LL$	Compare, Jump if Greater	1.60
Gated Compare	GCI	054	$\|(Y) - (H)\| > (H_6) : LL + 2 \rightarrow LL$ (next loc) $\|(Y) - (H)\| \leqslant (H_6) : LL + 4 \rightarrow LL$	Gated Comparison, Jump if Inside	1.76
	GCO	055	$\|(Y) - (H)\| \leqslant (H_6) : LL + 2 \rightarrow LL$ (next loc) $\|(Y) - (H)\| > (H_6) : LL + 4 \rightarrow LL$	Gated Comparison, Jump if Outside	1.76
Test Bit	TLZ	064*	$(Y)_{BL} \neq 0 : LL + 2 \rightarrow LL$ (next loc) $(Y)_{BL} = 0 : LL + 4 \rightarrow LL$ BL = Lower bit as specified by H	Test Lower Bit, Jump if Zero	1.60

*Literal modes 2 or 3 will cause the instruction to act like a NOP

Table V-5. Class: Jump Instructions (Sheet 2 of 3)

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
Test Bit	TUZ	065*	$(Y)_{BU} \neq 0 : LL + 2 \rightarrow LL$ (next loc) $(Y)_{BU} = 0 : LL + 4 \rightarrow LL$ BU = Upper bit as specified by H	Test Upper Bit, Jump if Zero	1.60
	TLF	066*	$(Y)BL \neq 1 : LL + 2 \rightarrow LL$ (next loc) $(Y)BL = 1 : LL + 4 \rightarrow LL$ BL = Lower bit as specified by H	Test Lower Bit, Jump if One	1.60
	TUF	067*	$(Y)BU \neq 1 : LL + 2 \rightarrow LL$ (next loc) $(Y)BU = 1 : LL + 4 \rightarrow LL$ BU = Upper bit as specified by H	Test Upper Bit, Jump if One	1.60
	TLZ	164*	a. $(Y)_{BL} \neq 0 : LL + 2 \rightarrow LL$ (next loc) $(Y)_{BL} = 0 : LL + 4 \rightarrow LL$ BL = Lower bit as specified by H b. Set external interrupt lockout	Test Lower Bit, Jump if Zero	1.60
	TUZ	165*	a. $(Y)BU \neq 0 : LL + 2 \rightarrow LL$ (next loc) $(Y)BU = 0 : LL + 4 \rightarrow LL$ BU = Upper bit as specified by H b. Set external interrupt lockout	Test Upper Bit, Jump if Zero	1.60

*Literal modes 2 or 3 will cause the instruction to act like a NOP.

Table V-5. Class: Jump Instructions (Sheet 3 of 3)

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
Test Bit	TLF	166*	a. $(Y)BL \neq 1 : LL + 2 \rightarrow LL$ (next loc) $(Y)BL = 1 : LL + 4 \rightarrow LL$ BL = Lower bit as specified by H b. Set external interrupt lockout	Test Lower Bit, Jumper if One	1.60
	TUF	167*	a. $(Y)BU \neq 1 : LL + 2 \rightarrow LL$ (next loc) $(Y)BU = 1 : LL + 4 \rightarrow LL$ BU = Upper bit as specified by H. b. Set external interrupt lockout	Test Upper Bit, Jump if One	1.60

*Literal modes 2 or 3 will cause the instruction to act like a NOP.

Table V-6. Class: Transfer Instructions (Sheet 1 of 2)

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
Register Modify	DTX	032	a) (H) − 2 → H : 0 → H on wrap around b) T → LL if H ≠ 0 LL + 2 → LL if H = 0 (next loc)	Decrement by Two and Transfer	1.60
	DOX	033	a) (H) − 1 → H : 0 → H on wrap around b) T → LL if H ≠ 0 LL + 2 → LL if H = 0 (next loc)	Decrement by One and Transfer	1.60
	ITX	132	a) (H) + 2 → H : 0 → H on wrap around b) T → LL if H ≠ 0 LL + 2 → LL if H = 0 (next loc)	Increment by Two and Transfer	1.60
	IOX	133	a) (H) + 1 → H : 0 → H on wrap around b) T → LL if H ≠ 0 LL + 2 → LL if H = 0 (next loc)	Increment by One and Transfer	1.60
Register Test	XEZ	040	(H) = 0 : T → LL (H) ≠ 0 : LL + 2 → LL (next loc)	Transfer if H = 0	1.60
	XNZ	041	(H) ≠ 0 : T → LL (H) = 0 : LL + 2 → LL (next loc)	Transfer if H ≠ 0	1.60
	XNG	042	$(H)_{15} = 1$: T → LL $(H)_{15} \neq 1$: LL + 2 → LL (next loc)	Transfer if H is Negative	1.60

Table V-6. Class: Transfer Instructions (Sheet 2 of 2)

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
Register Test	XPS	043	$(H)_{15} = 0 : T \rightarrow LL$ $(H)_{15} \neq 0 : LL + 2 \rightarrow LL$ (next loc)	Transfer if H is positive	1.60
Control Transfer	XFR	034	$T \rightarrow LL$	Transfer Unconditional	1.60
	XLK	035	$T \rightarrow LL$ $LL + 2 \rightarrow H$ (next loc)	Transfer Unconditional and Store Link	1.60
	XSW	036	$H_b = SWJ_b : T \rightarrow LL$ $H_b \neq SWJ_b : LL + 2 \rightarrow LL$ (next loc) Comparison is by bit match	Transfer on Console Transfer Switch	1.60

Table V-7. Class: Input/Output Instructions

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
I/O	DEC	074	$(Y)_{15-8} \rightarrow$ I/O data lines$_{15-8}$ $(Y)_{5-0} \rightarrow$ I/O data lines$_{5-0}$	External Device Command	4.16
	DES	174	a. $(Y)_{15-8} \rightarrow$ I/O data lines$_{15-8}$ $(Y)_{5-0} \rightarrow$ I/O address lines$_{5-0}$ b. Reset external interrupt lockout. Set internal interrupt. Reset PAR status bit associated with existing level.	External Device Command and Suicide	5.31
	OFR	076	$(H)_d \rightarrow$ I/O data lines$_{31-0}$ $(Y)_{5-0} \rightarrow$ I/O address lines$_{5-0}$	Output from Register	4.32
	ITR	075	I/O data lines$_{31-0} \rightarrow H_d$ $(Y)_{5-0} \rightarrow$ I/O address lines$_{5-0}$	Input to Register	4.48

Table V-8. Class: Miscellaneous Instructions

Subclass	Mnemonic	Function Code	Description	Name	Execution Time (μsec) m = 0/1
Misc	EXE	001	a. LL $\rightarrow$ LL b. The next instruction is accessed from address T.	Execute	1.60
	HLT	000	Halt if $H_b = SWH_b$. Comparison is by bit match.	Halt	1.60
	MBA	027	a. (Y) $\rightarrow$ memory module addressed by $(Y)_{14-15}$ b. Memory status $\rightarrow H_d$ if $H \neq 0$	Memory Bank Assignment	2.16 H = 0 2.56 H $\neq$ 0
	QED	124*	a. $(Y) \oplus (H) \rightarrow Y$ b. Set Y_0 if $[(Y) \oplus (H)] \neq 0$ c. LL + 4 $\rightarrow$ LL if the following equation is true $[(Y) \oplus (H)] = 0 \cdot Y_0 = 1$ or $[(Y) \oplus (H)] \neq 0 \cdot Y_0 = 0$ LL + 2 $\rightarrow$ LL (next loc) if the above equation is false d. Set external interrupt lockout	Queue Table Enque and Deque	1.95
	NOP	077	No operation	No operation	0.80
Macro	TBD	Any Unused Code	a. Macro is activated upon access of instruction from memory. b. CPU halts and waits for Function End signal from SPU. c. CPU accesses next instruction from memory upon receipt of the Function End signal. NOTE: The SPU can generate the Function End signal immediately upon recognition of its instruction when it is advantageous to execute the macro instruction concurrently with subsequent CPU instructions.	TBD	TBD

*Modes 2 or 3 will cause the instruction to act as a NOP.

A number of special instructions are provided to simplify programming and reduce program execution time.

Special instructions that will simplify programming include:

Queue Table Enque and Deque

This special instruction provides a more convenient method for handling common data files between multiple processor and program level, as well as providing a suitable method of handling multiple I/O termination to a common program level (Figure 5-6).

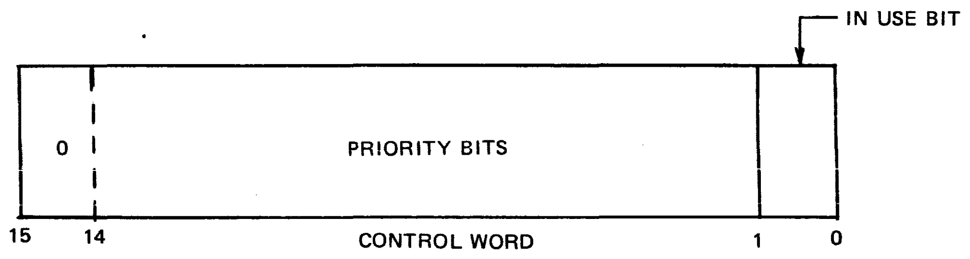
This is accomplished by using a designated control word for each data file. The control word is composed of an "in use" bit (bit 0) and 14 priority bits (14-1). Bit 15 must always be zero.

The same instruction is used for both enqueing and dequeing. A processor may use the file without being called only when the control word contains all zeros. If the file is in use when initially accessed by a processor, the appropriate queue bit will be set in the control word but the next location will be skipped in memory. The processor must then wait to be called before using the table.

When a processor has finished using the table, it must again execute the Queue Table instruction. The processor queue bit and the "in use" bit will be reset if no other queue bits are set in the instruction. The next location will be accessed by the processor to return it to its next routine.

If other queue bits are still set when the dequeing instruction is executed, the "in use" bit will not be reset and the next location in memory will be skipped. The processor will then determine the next highest priority (relative bit position) and issue a call to that processor by setting a bit in its PAR word and generating an interprocessor interrupt (SBL and SBU instructions).

The interrupt lock-out is set during this instruction to prevent a program level change from occurring before the processor completes the routines necessary for assuring an orderly transition in the usage of the table.



		(Y) INITIAL	(Y) FINAL	SKIP	(H)
ENQUE	FILE NOT IN USE	000000_8	002001_8	NO	002001_8
	FILE IN USE	040001_8	042001_8	YES	002001_8
DEQUE	LAST USER	002001_8	000000_8	NO	002001_8
	NOT LAST USER	042001_8	040001_8	YES	002001_8

Figure 5-6. Queue Table Instruction

Move Instruction

The two Move instructions allow any number of bits in one register to be addressed by means of a mask and moved to any position in another register.

The Move and Insert instruction (Figure 5-7) allows the bits to be stored into the designated area of the register without disturbing the remaining contents of that register.

The Move and Zero instruction (Figure 5-8) permits the storage of the bits into the designated area of the register but sets the remaining bits to zero.

This capability greatly facilitates the handling of "packed" data which in turn minimizes memory requirements by allowing several short words to be stored into a single memory word.

EQUATION: $\left[\text{(H) SHIFTED} \cdot \text{CA}\right] \,/\, \left[\text{(S)} \cdot \overline{\text{CA}}\right] \rightarrow \text{S}$

	H	CA MASK	S
INITIAL	0 [101 101] 000 101 000	0 000 000 [111 111] 000	0 101 100 [101 000] 111
(H) SHIFTED	1 010 000 [101 101] 000		
(H)S·CA	0 000 000 [101 101] 000		
(S)·$\overline{\text{CA}}$			0 101 100 [000 000] 111
FINAL RESULT	0 101 101 000 101 000		0 101 100 [101 101] 000

Figure 5-7. Move and Insert

EQUATION: (H) SHIFTED·CA $\rightarrow$ S

	H	CA MASK	S
	0 [101 101] 000 101 000	0 000 000 [111 111] 000	0101 100 101 000 111
(H) SHIFTED	1 010 000 [101 101] 000		
(H)S · CA	0 000 000 [101 101] 000		
FINAL	0 101 101 000 101 000		0 000 000 [101 101] 000

Figure 5-8. Move and Zero

Gated Comparison

The Gated Comparison instructions allow comparisons to be made between a value in memory and the contents of one of the process registers plus or minus a designated gate value. This gate value must be stored in a specific process register prior to the execution of the gated comparison.

The program can be made to branch if the value in memory falls inside or outside the gate range. These instructions are extremely useful where values are known only approximately or where some tolerance is allowed on either side of an expected value.

Arithmetic and Skip Instructions

All arithmetic instructions are provided with alternate operation codes to program exit within the same program level upon detection of an arithmetic overflow or error condition. Instances in which this skip capability provides an error exit include the following:

a. When the adder capacity is exceeded during the execution of any Add or Subtract, Load Absolute, or Load Complement instructions.

b. In the execution of a Multiply or Divide instruction where both operands are negative full scale.

c. In the execution of a Divide instruction where both operands are equal in magnitude and either have like signs or the sign of the divisor is positive.

d. In the execution of a Divide instruction where the absolute magnitude of the dividend is greater than the absolute magnitude of the divisor.

e. In the execution of a Divide instruction when the divisor = 0.

f. In the execution of a double precision Add or Subtract instruction where the sign associated with the least significant half of the operand or processor register pair is unlike the sign associated with the most significant half.

g. In Mode 3, Literal with Indexing, where arithmetic overflow occurs during the modification of the literal operand.

The skip capability is implemented by permitting the program to skip the next sequential location in memory during an arithmetic operation when a legal condition exists. A software link to a subroutine is established by storing an appropriate branch instruction in the next sequential location following the arithmetic instruction. The branch to the subroutine is only executed when an illegal condition occurs. The subroutine is used to analyze the result for appropriate action and then return control to the main program.

PROGRAM SEQUENCING

The instruction access is shown step by step beginning with the instruction location and ending with the instruction stored in the instruction register.

The instruction access or staticizing is shown in the accompanying illustration (Figure 5-9). The encircled numbers show the individual sequential steps.

The instruction location which is the address of the instruction to be addressed is contained in the 16-bit location counter. The instruction location register has three fields. All computer address formats are by half words, and all instructions are of 32-bit length and are located in even-numbered word addresses. Thus, the least significant address bit is ignored. The D field serves a dual purpose: step 1 verifying (through the page control and address register) that a "fetch" of the instruction is permitted, and step 2 providing the page address bits to be appended to the A field.

If memory access is not authorized, then instruction is not accessed and error program level is initiated. The A field and the appended address bits will access the memory (step 3). This memory cell contains the instruction which is transferred into the instruction register by step 4.

The instruction location address is incremented by 2 which corresponds to two half-word addresses. The result replaces the instruction location in the location counter. Both of these steps are shown as 5 and 6.

A program level change will preserve the process registers in core memory and thus the instruction location address is also automatically retained.

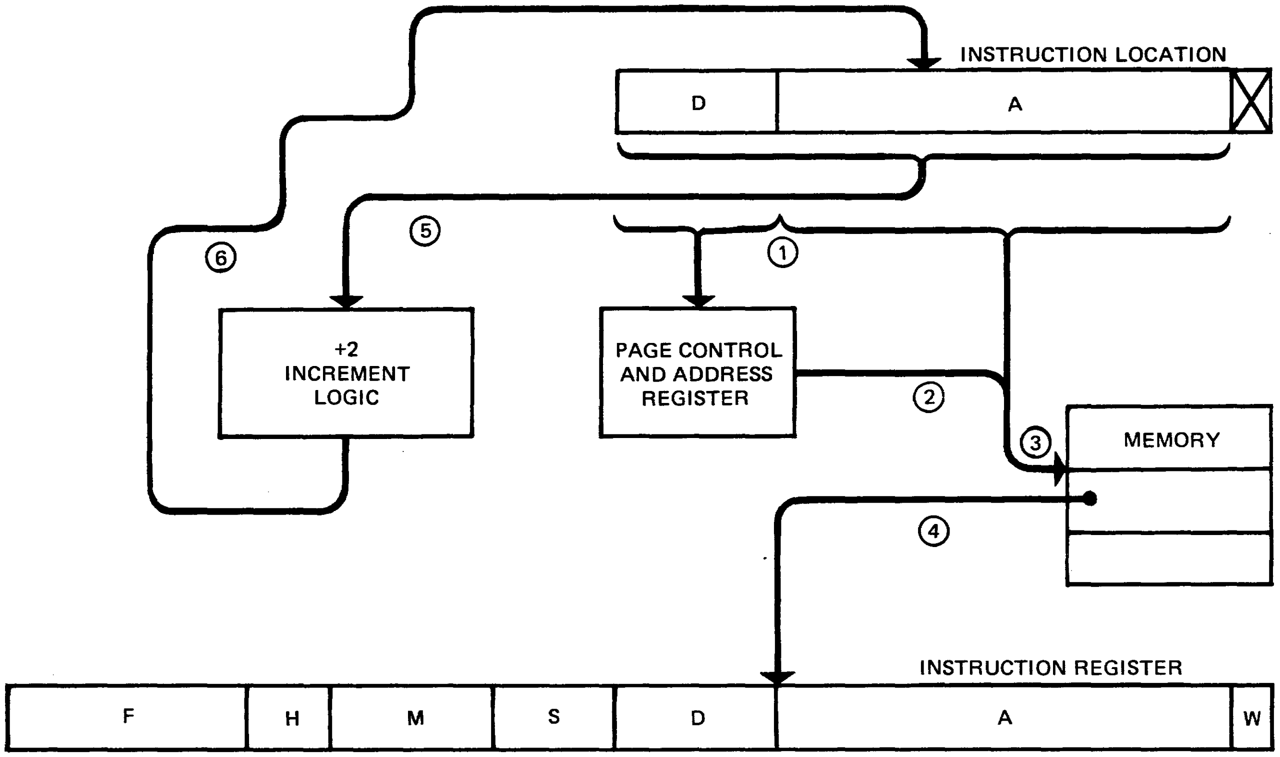


Figure 5-9. Instruction Access Process

MEMORY CONTROL OF THE INSTRUCTION'S ADDRESS FIELD

The memory control includes the page control and the page address. The page control provides full memory protection.

The memory control consists of the page control (PC) and the page address (PA). Each program level has its own set of controls, each of which contains 16 page control and address (PCA) registers (Figure 5-10).

Page Address

The four most significant bits of an address field are labeled as D and are used to select one of the 16 PCA registers. The PA field within this register consists of six bits which are appended to the A field of the address. The 11-bit A field addresses one of 2048 words within a page.

The translation of the D field through the PA register by pages can be considered as the base register concept with a module of 2048 words Thus, a dynamic program relocation can be achieved by changing the content of the PA register. The size of the PA field provides memory addresses of up to 131K words. The D field is also used for memory access control as described in the following paragraphs.

Page Control

The memory access control is provided selectively for each program level and selectively for each 2048 words of memory for the full address range of 131K words. As described in the preceding paragraphs, the D field of the address accesses the PCA register which contains two page control bits in addition to the page address.

The page control bits provide control for: (1) fetching of instructions, (2) read of operands, and (3) write of operands. The two-bit page control field allows for four combinations of control as shown in the accompanying diagram.

The fetch instruction control provides control for the access of instructions from memory. It specifically prevents the execution of instruction from a data base by accidental or unauthorized transfers.

The read operand control prevents reading of operands from memory. Thus, unauthorized access of data base including instructions of programs can be denied. The write operand control prevents destruction of information in memory.

A violation of any of the controls will inhibit the execution of the instruction, set a bit in the computer's status register, and transfer control to the error program level.

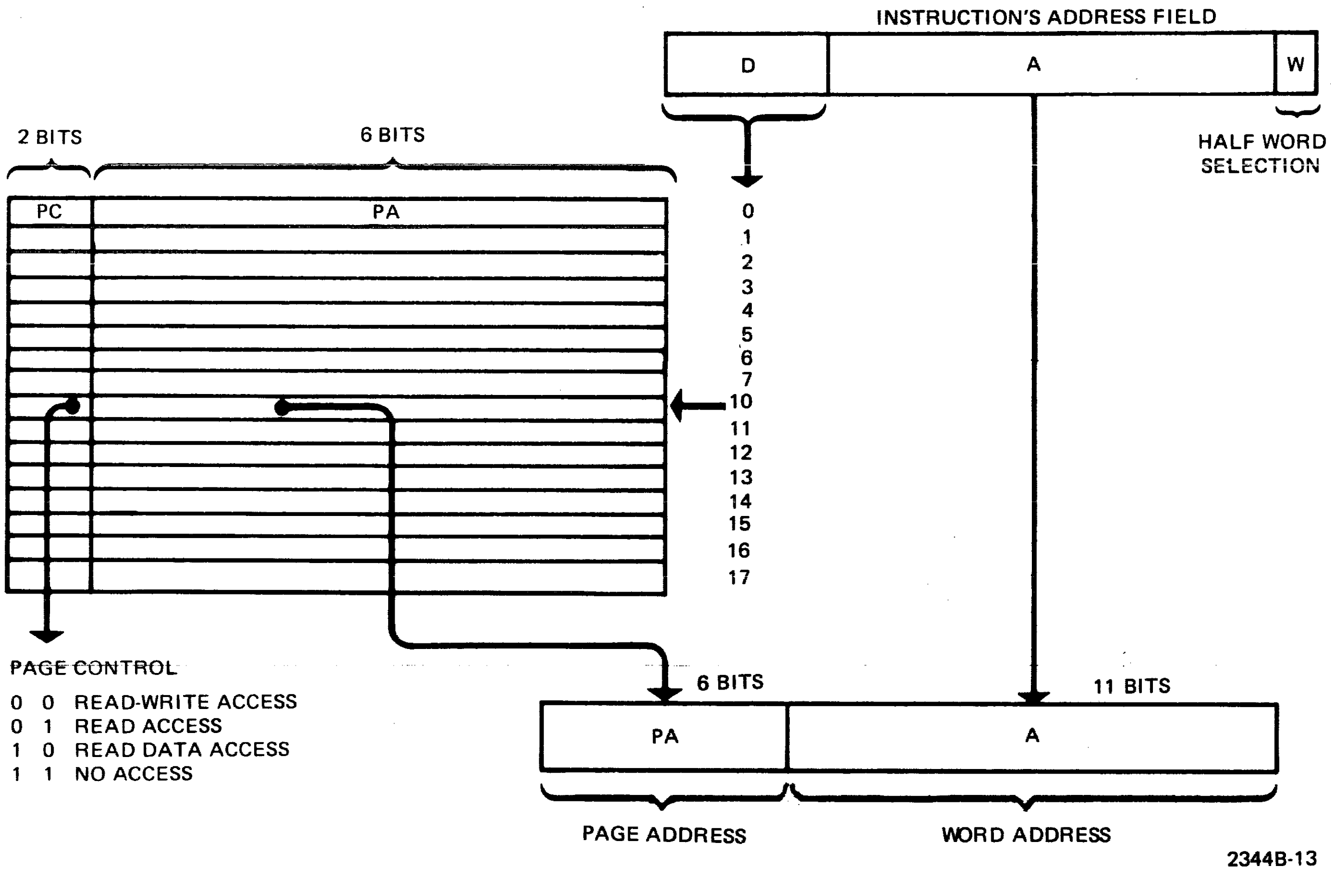


Figure 5-10. Page Control and Address

SERIES OF OPERATIONS IMPLEMENTED VIA EACH INSTRUCTION EXECUTION

Each instruction execution accomplishes a series of steps often requiring several instructions in less advanced computers.

Figure 5-11 shows the instruction format as it is stored in the instruction register.

The F and M fields contain the command function of the instruction; they are decoded in the operation controller as shown in step 1. Assuming an arithmetic type of instruction is under consideration, the following steps continue to be performed.

The S field, as shown in steps 2 and 3, refers to the index register which is retrieved from the process registers. The index is added to the instruction's address field consisting of D, A, W. The sum is a new D, A, W quantity as shown in step 5.

As previously described, step 6 will use the PCA to verify the authorization of memory access and to translate the D field into the page address. Step 7 shows the appending of these bits to the A field and the operand access from memory. It is assumed that the M-mode field of the instruction specified a direct memory access. Otherwise appropriate address modes would be applied.

The operand as retrieved from memory is shown in step 8. A half-word command operation would use the W-address bit to select the appropriate half-word to which the operation has to be applied. The instruction's H field selects, in step 9, an accumulator from the process registers. Steps 8, 10, and 11 show the combination of the selected accumulator with the operand from memory. The result of the operation replaces the accumulator.

Other commands may replace content of memory or apply only to the process registers as described in the instruction repertoire.

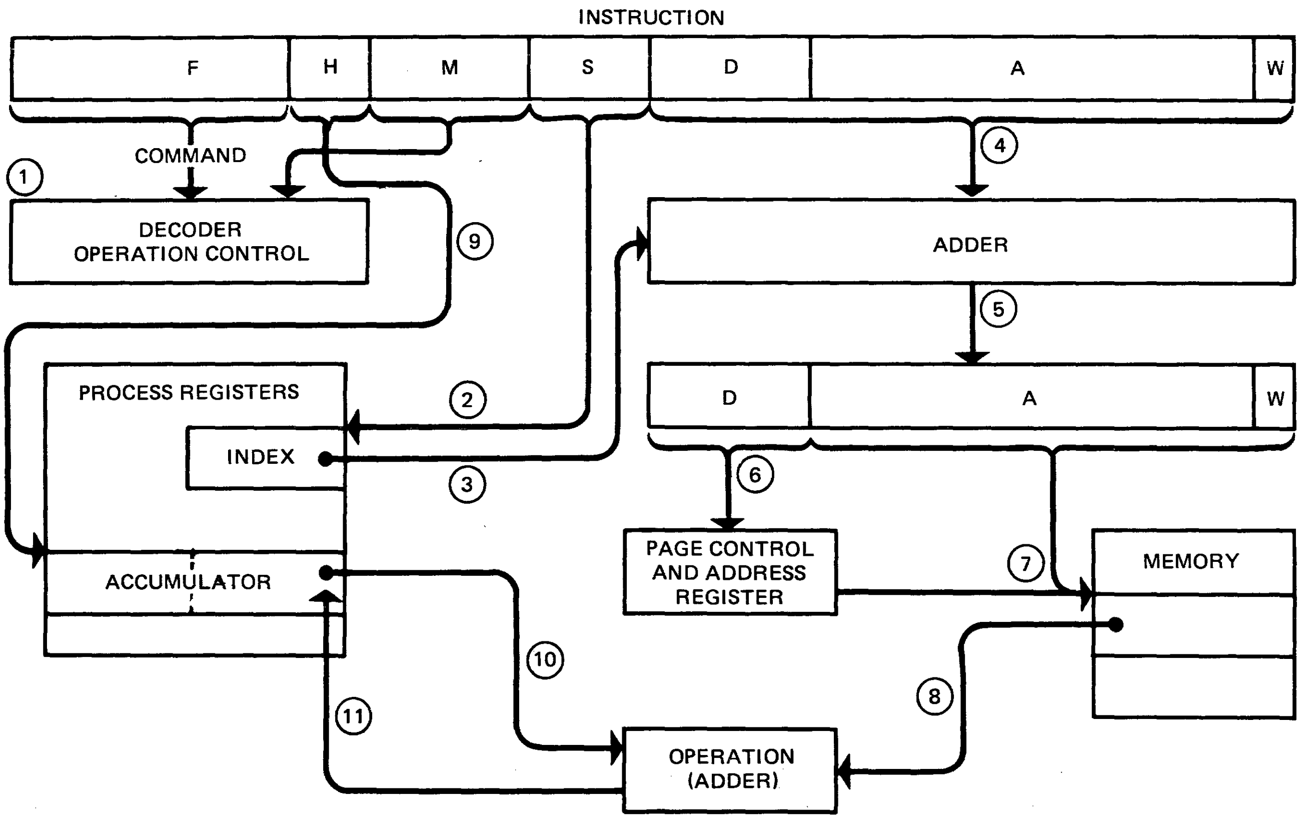


Figure 5-11. Instruction Execution Sequence

FACTORS AFFECTING INSTRUCTION EXECUTION TIME

The instruction execution time depends upon the instruction type, selected options, and the location of the operands.

The instruction execution time is variable and depends upon several parameters. The programmer can drastically reduce the execution time by selecting the appropriate combinations of influencing factors.

The instruction execution time in the computer depends upon four major functions to be performed and upon four hardware parameters. Each instruction sequences through the following four steps:

1. Instruction access
2. Generation of operand address
3. Operand access
4. Execution of command

The execution time also depends upon:

a. Availability of the memory
b. Memory cycle time
c. Use of process register

The instruction access time depends upon the location of the instruction. If the instruction and operand are located in the same memory bank, the operation will take the maximum time. The time decreases as instructions and operands are located in the different banks which can be accessed simultaneously.

The time also depends upon the memory cycle time and availability. Each memory is accessed by the central processing unit and the input/output unit. If one of these two units has control of a memory, the second unit must wait for completion of service for the first unit. If both units attempt to simultaneously access the same memory, the input/output unit is given priority.

The generation of the operand address depends upon the selected address mode as specified by the instruction's mode field. The literal mode gives the fastest execution since no memory access for operand access is involved. The indirect addressing requires one memory access.

The operand access time depends upon the same parameters as described for the instruction access. The fastest execution time is achieved for register-to-register operations.

The execution of the command depends upon the selected instruction. It normally involves the access of an accumulator from the process registers, the combined operation between operand and accumulator, and the return to the result into the process register. As an example, variations in "add" instruction times are shown in Table V-9.

Table V-9. Add Instruction Times in Microseconds

	Access Type	Memory/Memory	Memory/Register	Register/Memory	Register/Register
	Data Location Instruction → Indirect Address → Operand →	 Memory X Memory X Memory X	 Memory X Memory X Process Register	 Memory X Process Register Memory X	 Memory X Process Register Process Register
ADDRESS MODES	M = 0 Direct	1.60	1.28	1.60	1.28
	M = 1 Direct Indexed	1.60	1.28	1.60	1.28
	M = 2 Literal	1.12	1.12	1.12	1.12
	M = 3 Literal Indexed	1.12	1.12	1.12	1.12
	M = 4 Indirect	2.40	2.08	2.08	1.60
	M = 5 Indexed Indirect	2.40	2.08	2.08	1.60
	M = 6 Indirect Indexed	2.40	2.08	2.08	1.60
	M = 7, S = 0 Relative	1.92	1.60	1.92	1.60
	M = 7, S ≠ 0 Relative Indexed	1.92	1.60	1.92	1.60

SECTION 6. INPUT/OUTPUT SYSTEM

MODULAR FEATURES OF INPUT/OUTPUT UNIT

The input/output unit is modular, with a direct memory access channel.

The input/output unit (IOU) is independent of the CPU with the exception of common power supplies and a common channel to memory. Data throughput is enhanced by the simultaneous operation of the CPU and IOU.

Parallel, simultaneous operation of processing units allows peak processing loads to be accommodated. By the appropriate choice of memory configuration, the CPU-IOU system can be matched to increasing processing load requirements. For example, availability of two memory units allows an increase in the capability of simultaneous CPU-IOU operations over that possible with a single memory unit. The direct memory access channel is shared in a manner which minimizes the data transfer time.

Modularity of the IOU provides the following advantages:

a. Additional IOU capability can be obtained simply.

b. The IOU can be interfaced with a separate memory port for high data rate applications.

c. The IOU can be operated at twice normal clock frequency for high data rate applications.

d. Special purpose IOUs can be added to the system without affecting the CPU or memory.

IOU ADDRESSES UP TO 56 PERIPHERAL DEVICES

The IOU can address up to 56 external peripheral devices and multiplex the transfer of data between the devices and the memory.

The IOU has the capability of addressing 64 devices. However, eight addresses are used for internal processor functions (real-time clock, maintenance functions, etc.).

The 56 device addresses are divided into seven groups of eight addresses, with an input/output exchange unit required to multiplex data transfer within each group of eight addresses.

The data exchange can consist of a 32-bit word or an 8-bit byte. The IOU controls packing and unpacking of the bytes into the 32-bit memory word. Parity is written with each memory word and checked as each word is read.

In addition to the data interface, 31 control and address lines are provided to the peripheral devices. All lines are twisted pairs with a common ground system. Peripheral device to IOU distances of 30 meters can be accommodated.

INPUT/OUTPUT MULTIPLEXING OCCURS ON A PRIORITY BASIS

Communication with peripheral devices can be initiated as an input/output instruction is executed in the CPU (programmed input/output), or upon the request of the peripheral device and under control of the IOU command words (automatic input/output.

Programmed input/output instructions have highest priority for access to the peripheral devices. An automatic data transfer will be interrupted while the CPU instruction is executed. These instructions are used to exchange data with the device at the convenience of the CPU program. Examples are: reading device status, commanding a specific device function, and transferring program status to the device.

These input/output instructions are:

a. Output From Register (OFR) instruction (Figure 6-1). This instruction causes the contents of the process register pair addressed by the instruction H field to be transmitted to the device addressed by the six least significant bits of the instruction operand.

   The S field selects one of eight process registers for address modification on modes M = 1, 3, 5, and 7 if S ≠ 0. Bits 15 to 6 of the operand are not used.

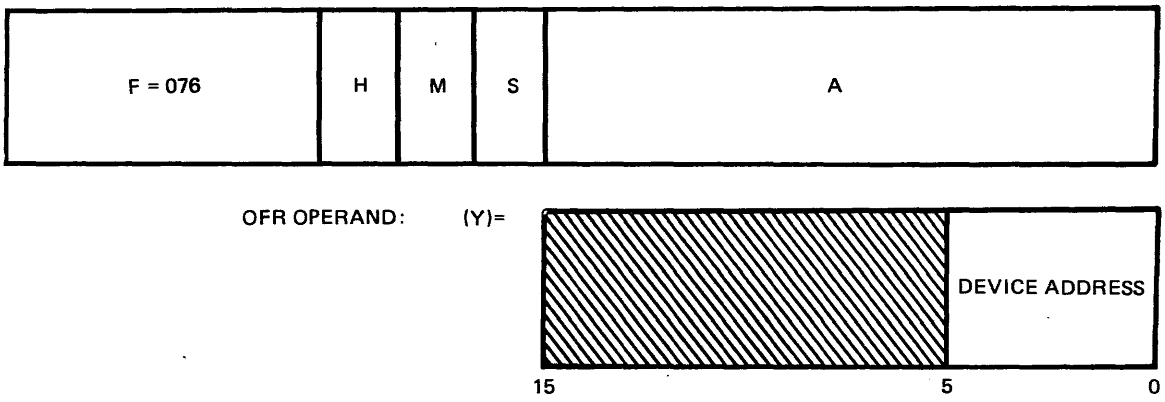


Figure 6-1. OFR Instruction Word

b. Input to Register (ITR) instruction (Figure 6-2). This instruction causes the device addressed by the six least significant bits of the instruction operand to transmit up to 32 bits of data. This data is stored in the process register pair specified by the H field of the instruction.

The S field selects one of eight process registers for address modification on modes M = 1, 3, 5, and 7 if S ≠ 0. Bits 15 to 6 of the operand are not used.

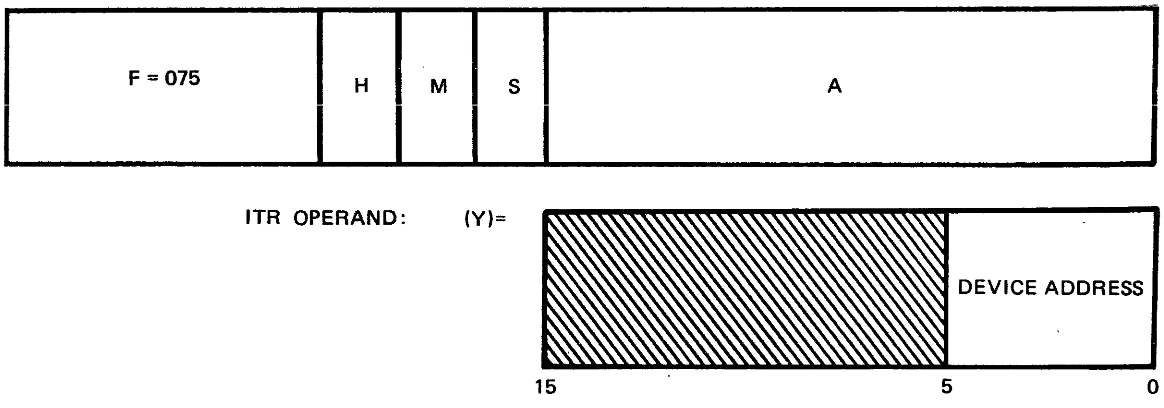


Figure 6-2. ITR Instruction Word

c. Device Command (DEC) instruction (Figure 6-3). This instruction causes the most significant eight bits (bit positions 15 to 8) of the instruction operand to be transmitted to the I/O interface unit that was addressed by the least significant six bits (bits 5 to 0 of the operand). The H field is not used on this instruction.

All modes, M, of address modification are allowed; however, Mode 2, the literal mode, or Mode 0, the direct mode, will be the most common.

The S field selects one of eight process registers for address modification on modes M = 1, 3, 5, and 7 if S ≠ 0.

If F = 174, an automatic program level change will occur. At the end of the DEC instruction the Program Activity Register is obtained from memory. The active bit for the current program level is reset and the next highest program level is entered. This operation is called program level suicide. In addition, external interrupt lockout is removed if it was in effect.

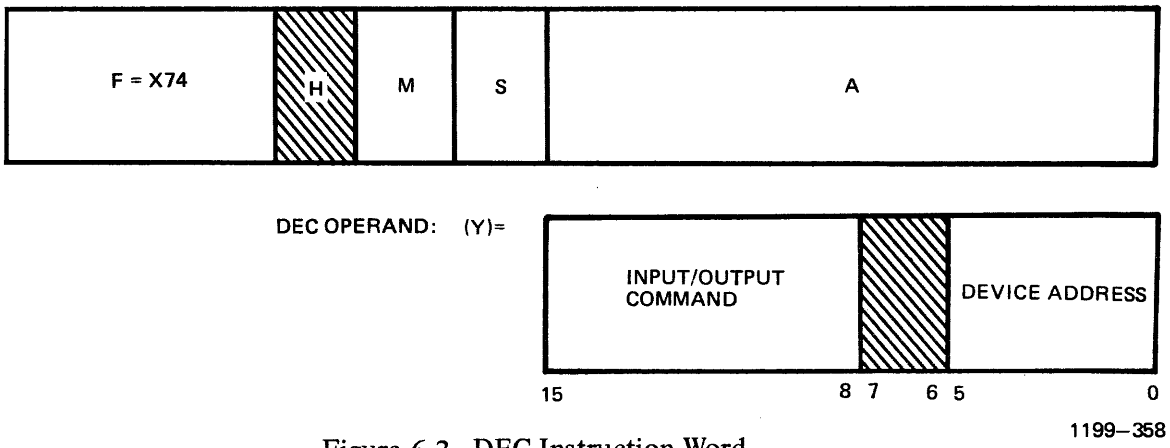


Figure 6-3. DEC Instruction Word

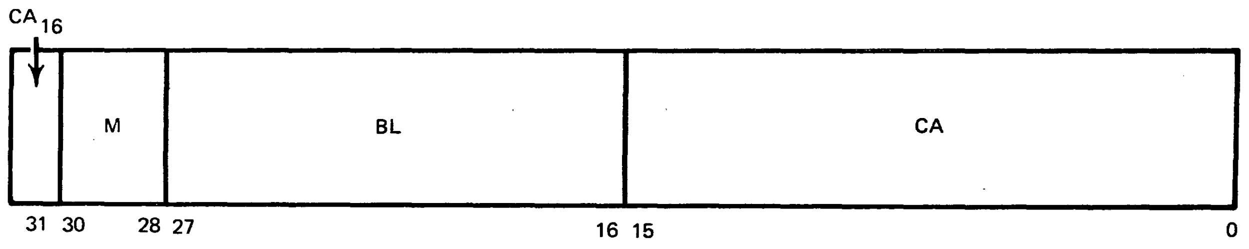
Automatic input/output operations are handled in the order of their address; a request from the device at address 77_8 has priority over an address 76_8 request; an address 00_8 request has lowest priority. Two IOU command words for each of 77_8 addresses are stored in a memory designated as the "base" memory. The key word defines the mode of I/O operation, the block length and the current address of data in memory; it controls the transfer of data initiated by a request from the device. The termination word defines two CPU program levels to be activated for a normal or error termination and tag bits defining the type of termination; it controls interruption of the CPU at the end of an I/O operation.

When an I/O interface unit is ready to receive data from the computer or is ready to transmit data to the computer, it will set the request line at its I/O station in a true state. The IOU will cause the interface unit to transmit its substation address by sending an enable signal to the interface unit. The IOU will read from memory the key word associated with the station and substation address.

Figure 6-4 shows the I/O key word format. The fields of the key word are defined as:

M = Mode, bit positions 30 to 28. This 3-bit specifies one of the following modes of operation designated in octal digits:

0 Inactive
1 Alarm
2 Block word input
3 Block word output
4 Block character input, least significant character (LSC) first
5 Block character output, LSC first
6 Block character input, most significant character (MSC) first
7 Block character output, MSC first



1424–18

Figure 6-4. I/O Key Word

BL = Block length, bit positions 27 to 16. This 12-bit field specifies either the number of words or characters in a block data transmission of the number of time increments in the alarm mode. In 32-bit word transfers, BL can specify up to 4095 words. In the character mode, BL can specify up to 4095 characters packed into 1024 words; thus, the two least significant bits of BL specify the character position in the word. After each word or character transmission or after each time increment, BL is decremented until a block length of Zero is reached.

CA = Current address, bit positions 31, 15 to 0. This 17-bit field specifies the running address of the memory location from which the current 32-bit word is (or four 8-bit characters are) accessed in an output operation, or in which it is stored in an input operation. After each word transmission in a block data transfer, CA is incremented.

When the block length, BL, field of the I/O key word has been decremented to Zero, the IOU will obtain the termination word to signal the programmer that data communication is complete for that particular I/O substation. Figure 6-5 shows the I/O termination word format. The fields of the termination word are defined as:

EPL = Error Program level, bits 21 to 16. This 6-bit field is the binary number of the program level to be entered if an error condition during the I/O data communication occurred (E bit or R bit is set) or an external interrupt signal from the I/O substation (I bit is set) occurred. This program level number is preset by the program.

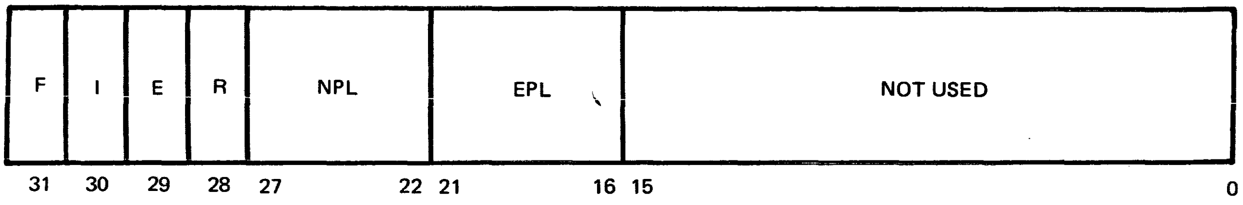


Figure 6-5. I/O Termination Word

1424–19

NPL = Normal program level, bits 27 to 22. This 6-bit field is the binary number of the program level to be entered when the key word block length field, BL, reaches Zero and no error condition was detected (F bit is set). This program level number is preset by the program.

R = Channel malfunction, bit 28. The R bit is set whenever an I/O operation is terminated by the recognition of an error condition by the processor. The processor recognizes a channel malfunction by an inactive mode (mode zero) or an initial block length of zero.

E = Device error, bit 29. The E bit is set whenever an I/O operation is terminated due to the recognition of an error condition by the interface unit. The interface unit notifies the processor by the transmission of a device error signal during an I/O data transfer.

I = Indicator, bit 30. The I bit is set whenever the processor receives the indicator signal from an interface unit following the acknowledgement of its request. The indicator signal is used primarily for the accomplishment of an interface unit initiated program interruption.

F = Block complete, bit 31. The F bit is set following the completion of the transfer of a block of data. This is considered a normal termination of an I/O operation. The F bit is not set if an abnormal terminating condition (the detection of an error condition) occurs prior to the specified number of transmissions or the receipt of an indicator signal.

SEVEN MODES OF AUTOMATIC INPUT/OUTPUT OPERATIONS PROVIDED

Data transfer by word or byte provides adaptability of the IOU to many different applications.

M = 0, inactive mode

If the IOU detects that the key word mode field is all zeros, it will indicate an I/O error condition. It will obtain the corresponding termination word, set its R bit, and will transmit and end-of-transmission signal to the interface unit. The EPL field of the termination word is used to specify the status bit in the program activity register that is set to a one. The key word is not modified.

M = 1, alarm mode

In the alarm mode, the block length field, BL, of the key word is used as an elapsed time counter. Each time a request from the substation address with this key word is serviced, the BL field is decremented; the CA field is not modified and no data transmission occurs. The request signal is usually provided from an interface unit with a real-time clock source.

When the BL field reaches zero, the corresponding termination word is obtained from memory and its F bit set to a one. In this mode, the NPL field of the word is used to set the desired status bit in the program activity register. An end-of-transmission signal is transmitted to the interface.

Since the BL field is composed of 12 bits, up to 4095 time intervals may be counted without program intervention. The programmer has normal access to the key words, since they are addressable in memory.

M = 2 or 3, block word input or output

These modes are the normal modes for transmission of full words. Words may be any size from 1 to 32 bits, but each request for an input or output addresses one word in memory and the address field of the key word, CA, is incremented to the next word address. The block length field, BL, is decremented for each word received or transmitted.

M = 4 or 5, block character input or output, least significant character first

Data is received or transmitted as 8-bit characters. This data is unpacked from a 32-bit word on input from right to left. Each 32-bit word is separated into four 8-bit character fields. The least significant two bits of the key word block length field, BL, counts the character position within the data word, as shown in Figure 6-6.

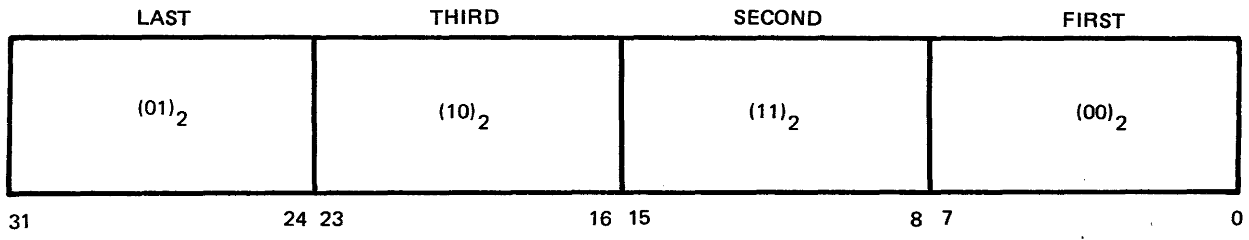


Figure 6-6. Character Positions, Least Significant First 1424–20

As characters are received or transmitted they are not shifted in position within the 32-bit word. They are properly placed or transmitted according to the character number. Block length may be any number from 1 to 4095_{10}.

The current address field, CA, of the key word in a character mode is incremented each time the least significant two bits of the block length field decrements from $(01)_2$ to $(00)_2$.

M = 6 or 7, block character input or output, most significant character first

These modes are similar to modes 4 and 5 except that characters are counted left to right. The least significant two bits of the key word block length field counts the character positions within the data word, as shown in Figure 6-7.

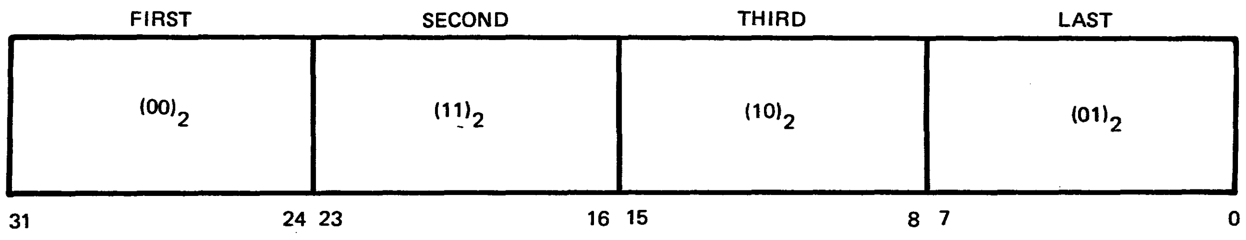


Figure 6-7. Character Positions, Most Significant First 1424–21

IOU SYSTEM PROVIDES BOTH A BURST AND BLOCK MODE OF DATA TRANSFER

The IOU can accommodate peripheral devices with different data transfer rates.

Devices requiring a high data rate utilize the burst mode; the block mode can be used for lower data rates. In either case, the limitation on IOU data rates is determined by the memory unit; the highest rates are possible when the IOU and CPU are addressing different memory units.

The block mode of data transfer involves fetching the key word and transfer of one data word for each request from the device. After each sequence the request next in priority is processed by the IOU.

Block Transmission Input. The IOC examines the eight I/O request lines when it is not already executing an I/O operation. When an I/O request is detected, the IOU enters the phases which are provided for I/O control. If two or more I/O requests are received simultaneously, the highest numbered station will be serviced first.

A block of data will be received (input) by the IOU upon completion of the following sequence of events:

a. The request from the interface unit is acknowledged by activating the enable line for that station.

b. The interface unit specifies the substation on the three address lines, and then transmits a ready signal. The processor waits until this ready signal is received.

c. A sync signal is sent to the interface unit. The sync signal causes the interface unit to deactivate its request if it is in the block mode. The interface unit places data on the lines and then transmits another ready signal. The processor waits until the ready signal is received.

d. The data is taken from the data lines by an internal control signal.

e. Steps a, b, and c are repeated for each request until the block length of the key word reaches zero. An end-of-transmission signal is sent to the interface unit if the entire block of data has been received.

Block Transmission Output. A block of data will have been transmitted (output) by the IOU upon completion of the following sequence:

a. The request signal from the interface unit is acknowledged by activating the enable line of that station.

b. The interface unit specifies the substation on the three address lines, and then transmits a ready signal. The processor waits until the ready signal is received.

c. A sync signal is sent to the interface unit to deactivate its request if it is in the block mode.

d. The processor waits until another ready signal is received from the interface unit.

e. The data is put on the data lines by the data bus enable control.

f. The data is clocked into the interface unit with a strobe control signal.

g. Steps a, b, c, and d are repeated until the block length of the key word reaches zero. An end-of-transmission signal is sent to the interface unit if the entire block of data has been transmitted.

The burst mode of data transfer involves fetching the key word once in response to a device request followed by multiple data transfers. Data transfer continues until the block length reaches zero as detected by the IOU, or the device deactivates its request. A higher priority request will interrupt the burst mode data transfer which will then be re-established after the higher priority request is serviced.

Since memory cycles have to be time-shared with the CPU, data rates are effected by the memory interleave between the CPU and the IOU. Table VI-1 provides the typical and maximum data rates for all modes of operation.

Table VI-1. Data Rates for I/O Operations

	Maximum (kHz)	Typical (kHz)
Burst Input	458	450
Burst Output	625	590
Block Input	230	220
Block Output	218	217

REAL TIME SYSTEM PROGRAMS CALLED BY I/O INTERRUPT HAVE PROGRAMMABLE PRIORITY

Flexibility in the programming of real-time systems is provided by variable program priority. Reconfiguration of externally driven programs during operation is effected by an efficient machine architecture.

Real time systems require that programs be activated in response to external signals. The relative priority of such programs must be set to allow completion of all programs between interrupts. It is advantageous if the relative priority of programs is not final but can be adjusted to differing operational environments.

The activation of CPU program levels is controlled by various control words stored in the "base" memory unit. Some of these words are modified by hardware in addition to being available to the programmer.

The steps involved in calling a program are outlined in the following paragraphs to illustrate the ease with which program priority can be changed.

The IOU termination word designates two of 64 possible program levels to be called upon termination of the I/O operation. This can be due to an interrupt from the peripheral device or to completion of a data transfer operation.

Address in Base of Termination Word

$(001000_8 + 4\ DA_8 + 2_8)$ = [TAGS, NPL, EPL, SPARE]

The CPU is interrupted by the IOU after a status bit is written in the program activity register (PAR) corresponding to the program level designated in the termination word. The CPU then determines the highest program level which has been enabled and initiates a program level change if required. The program has control of the program levels which can be activated by means of the PAR enable list.

Address in Base of Program Activity Register

PL 17_8 to 00_8 (1600_8) = [PL ENABLE, PL STATUS]
PL 37_8 to 20_8 (1602_8) = [PL ENABLE, PL STATUS]
PL 57_8 to 40_8 (1604_8) = [PL ENABLE, PL STATUS]
PL 77_8 to 60_8 (1606_8) = [PL ENABLE, PL STATUS]

During the program level change, the program location register and the eight process registers of the new program are fetched from memory and placed in hardware registers. Also, if the extended memory address option is part of the CPU, the address of relevant pages must be fetched.

Address in Base of Program Location Register

$(1400_8 + 2\ PL_8)$ = [SPARE, PROGRAM LOCATION]

Changing the priority of the program called by an I/O interrupt requires changing the control words described above. If spare program levels are available, a simple priority change can be effected by

disabling the old and enabling the new program level in the program activity register, copying the old program location in the new program location register (also the process registers), and changing the program level fields of the termination word.

A single interrupt from a peripheral device will interrupt the CPU after a delay of 3.6 microseconds. The program level change in the CPU will be completed in 14 microseconds if the search for priority is a maximum (assuming no extended memory addressing).

PROGRAM LOADING PROVIDED BY HARDWARE BOOTSTRAP CONTROL

The IOU can provide the bootstrap function at any device address. An initial program of 1023 instructions can be loaded by byte or 4095 instructions can be loaded by word.

The program load feature is activated by a program load signal on the IOU interface. Both the CPU and IOU must be in the halted condition and all peripheral devices reset before this signal is activated. Activation of the program load signal causes the IOU to examine the first device request as a source of four bytes of data defining the key word controlling the program load. These four bytes are stored in the IOU key word register. The next device request is controlled by the key word and program data is stored at the designated current address until the block length reaches zero. At this time, an interrupt to program level 00_8 is sent to the CPU, and the IOU leaves the bootstrap mode.

The CPU will begin program execution at the location specified by the initial value of the designated current address.

THE IOU SYSTEM IS POWER FAIL SAFE

In the event of power interruption, operational shut-down and start-up occur without loss of memory data.

The IOU uses the same power sequencer as the CPU. Power interrupt during IOU operation results in an orderly termination of the data transfer currently taking place. The termination word R bit is set for the device transferring data at the time power failure is detected, and a system reset signal is sent to all devices on the IOU interface.

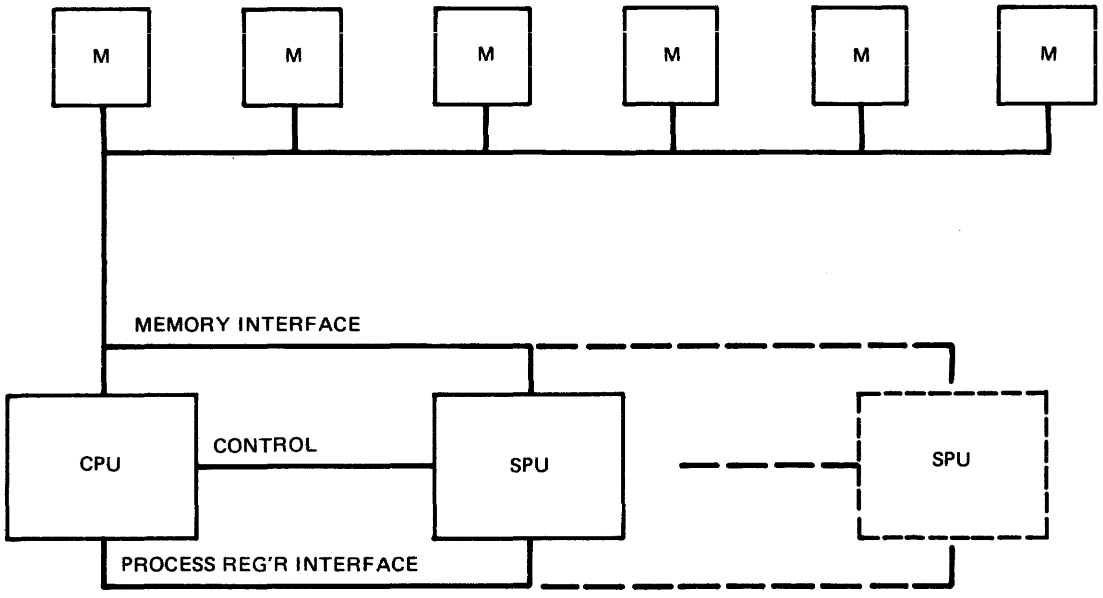
During power start-up, the system reset signal is sent to all devices on the IOU interface. A command from the CPU program is necessary before an I/O device becomes active, thereby insuring that the IOU command words have been initialized before automatic I/O operation starts.

SECTION 7. SPECIAL PROCESSING UNITS

Special processing units (SPUs) may be added to adapt the L-304H to unique processing or to expand on the basic instruction set. Examples of expansion of the basic instruction set are: floating point instructions, cordic conversion instruction, bit string instructions, transcendental functions, square root, etc.

The SPUs implement classes of or individual instructions using operation codes not required for the basic L-304H instruction set. These instructions are called macro-instructions. SPUs tailored to the macro-instruction(s) greatly increase processor throughput.

SPUs interface directly with memory via the DMA channel. Control signals between the CPU and SPU determine the interaction required between the two units. All 16 (16 bit) active registers (8 process plus 8 scratchpad) may be used by a SPU. Figure 7-1 shows a typical interconnect configuration.



L-304H-2

Figure 7-1. SPU Interconnection Diagram

The SPU control lines are: Instruction Fetch, Active Registers Available, Operand Fetch, Address Phase Enable, Location Register Increment, and Function End.

The Instruction Fetch line is activated by the CPU each time an instruction is accessed by the CPU. The line alerts the SPU(s) to monitor the instruction on the memory interface. If the instruction is a macro, a SPU must interpret the proper action. The CPU halts on detection of a macro, issues a Active Register Available signal and removes the Instruction Fetch signal. The SPU(s) now has access to the eight basic process registers plus eight scratchpad registers not used by the CPU. The eight scratchpad registers are not automatically stored in memory on a power loss or program level change. The Function End signal is generated by the operating SPU when the SPU has completed its task or no longer requires the CPU to remain halted. The CPU's response to the Function End signal is to deactivate the Active Register Available signal and resume processing.

The SPU instructs the CPU to fetch the operand by activating the Address Phase Enable line. The CPU then executes the designated address mode of operation and activates the Operand Fetch line when the operand is available to the SPU. The CPU then halts, issues the Active Registers Available signal, and waits to be released by the Function End signal from the SPU.

An SPU communicates with memory in the same manner as the IOU. The memory addresses provided by the SPU may require interpretation by the extended memory addressing (EMA) option if supplied. Activation of the Address Interpret line accomplishes this function.

An SPU designed to continue a task after releasing the CPU from the halted condition may still access memory via the DMA interface, but will no longer have access to the "active" registers of the CPU. On completion of a task this SPU must inform the CPU by using an interrupt line or control word. If the EMA option is provided, the SPU must inform the CPU by using an interrupt line or control word. If the EMA option is provided, the SPU must address memory using absolute addresses or the CPU must be placed in the non-interrupt mode prior to accessing the macro instruction. The SPU must activate the Address Interrupt line to the EMA if the memory address requires interpretation.

PROGRAM SKIP CAPABILITY PROVIDED FOR ALL SPUs

All SPUs are provided with the capability of generating a program skip operation.

The provision of a skip capability for all SPUs permits a program exit within the same program level for SPU arithmetic error conditions or any other SPU condition where a jump from the main routine is desirable. The SPU initiates this skip capability by activating the Location Register Increment line prior to issuing the Function End signal. The CPU increments the location register before accessing the next instruction, thus skipping the next instruction in memory.

The intent of the SPU concept is to accommodate specific and unique customer requirements without modifying the basic CPU structure.

To increase the capability of an already powerful instruction set, an extended performance arithmetic unit has been designed as an optional addition to the basic L-304H. This unit provides a full set of floating point instructions and completes the set of full-word, fixed-point arithmetic instructions.

This SPU uses the same instruction format as the L-304H and is capable of directly accessing the process register in the CPU. Therefore, all memory register operations place the result in the process register of the CPU.

Floating Point Arithmetic

The purpose of the floating-point instruction set is to perform calculations using operands with a wide range of magnitude and yielding results scaled to preserve precision.

A floating-point number consists of a signed exponent and a signed fraction. The quantity expressed by this number is the fraction multiplied by the power of 2 as defined by the exponent. The exponent is expressed in excess 128 binary notation; the fraction is expressed as a binary number having a radix point to the left of the high-order digit. Operations may be either register-to-register or storage-to-register.

To preserve maximum precision, addition, subtraction, multiplication, and division are performed with normalized results.

Floating Point Number Representation

A floating point number consists of a signed exponent and a signed fraction. The quantity of this number is a 23-bit signed fraction multiplied by the power of 2 as defined by the exponent.

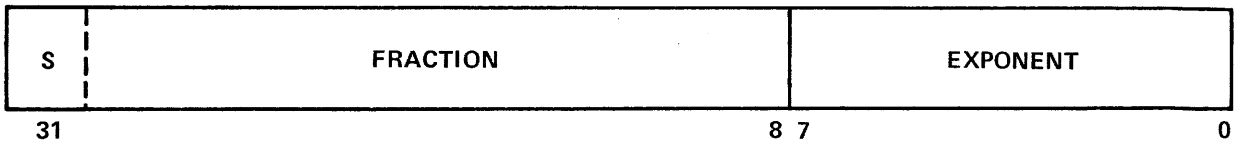
The exponent is an 8-bit binary number with a range from -128 to +127. A zero exponent has a binary value of all zeroes. A negative exponent is expressed in 2's complement notation.

The fraction is a 24-bit binary number consisting of a sign bit and 23 bits of magnitude. The range of the fraction may be expressed as follows:

$$-1 \leqslant F < +1$$

Negative fractions are represented in 2's complement form. A true zero is defined as an all zero word.

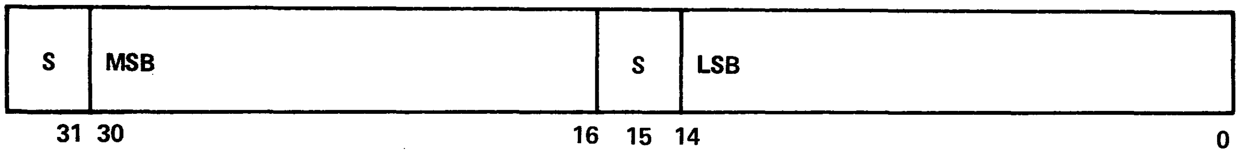
Floating Point Format



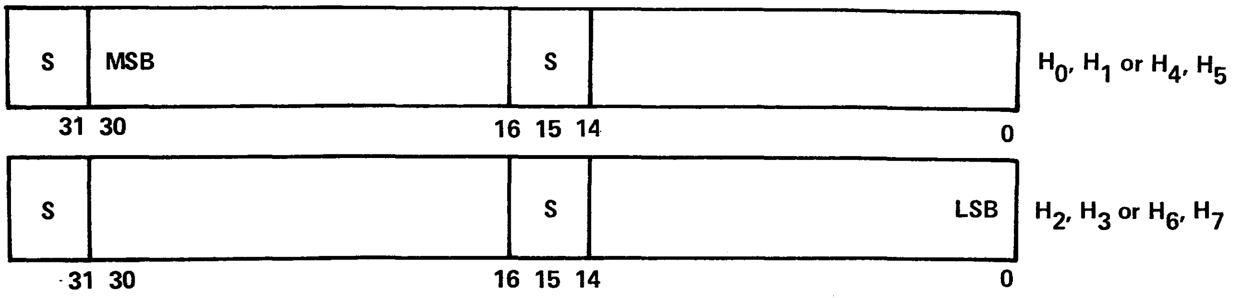
The first bit in the format is the sign bit (S). The subsequent 23 bits provide 23 bits of magnitude for the fraction. The last 8 bits are occupied by the exponent.

Double Precision Number Representation

A double precision number is represented by a 31-bit signed fraction. The sign of the most significant half of the word is repeated in the least significant half of the word. This is consistent with the existing double precision format in the L-304H.



Sixty-four-bit data cards are used by the Long Multiply and Long Divide instructions. The format is consistent with the 32-bit format.



Floating-Point Normalization

A quantity can be represented with the greatest precision by a floating-point number of given fraction length when that number is normalized. A normalized floating-point number has the sign bit and high-order fraction bit in opposite sense. If this condition is not met, the number is said to be unnormalized. The process of normalization consists of shifting the fraction left until the high-order binary digit is of opposite sense from the sign bit and reducing the characteristic by the number of digits shifted. A zero fraction cannot be normalized; therefore, its associated characteristic remains unchanged when normalization is called.

Normalization usually takes place when the intermediate arithmetic result is changed to the final result. This function is called "post-normalization." All instructions assume that floating-point numbers are normalized. Operands not pre-normalized are detected as an error.

Floating Point Add (FAD) Instruction

The FAD instruction performs the addition of two 32-bit floating point numbers. The operand (Y)d is added algebraically to the contents of the process register pair (H)d and the normalized sum is placed in (H)d.

A number of error conditions can occur prior to or as a result of the steps required to add the two numbers. If an error is detected, a status bit is set defining the error, the process registers are left unaltered, and the next instruction in memory is executed. When the trap capability is utilized, the next instruction is skipped when no error occurs.

The types of errors that can occur within this instruction are as follows:

1. One or both of the numbers were not prenormalized.
2. Exponent underflow occurred during post-normalization.
3. Exponent overflow occurred as a result of a fraction overflow.

Floating Point Subtract (FSB) Instruction

The FSB instruction performs the subtraction of two 32 bit floating point numbers. The operand (Y)d is subtracted algebraically from the contents of the process register pair (H)d and the normalized difference is placed in (H)d.

The same error conditions exist for the FSB instruction as for the FAD instruction. When the trap instruction is used, the instruction will execute the next instruction in sequence when an error occurs or skip over the next instruction when an error does not occur.

Floating Point Multiply (FMP) Instruction

The FMP instruction performs a multiplication of the operand (Y)d and the process register pair (H)d. The post-normalized product is truncated to 24 bits and stored in (H)d.

Error conditions that occur during the execution of the multiply instruction will cause a status bit to be set defining the error and cause the next instruction in memory to be executed. The process register pair will be left unaltered for any error condition. The trap instruction will cause the next instruction to be skipped when an error condition does not occur.

Errors that can occur are as follows:

1. One or both of the numbers were not prenormalized.
2. Exponent overflow occurred during the initial addition of the exponents (positive exponents).
3. Exponent overflow occurred during the initial addition of the exponents (negative exponents).
4. Exponent overflow occurred due to a product overflow (+1). A product overflow which results from a (-1) multiplicand and a (-1) multiplier causes the product to be shifted right one place and the exponent increased by one.
5. Exponent underflow occurred during the post-normalization cycle.

When either of the numbers to be multiplied contains a zero fraction, a true zero (all bits are zero) is stored into the process register pair.

Floating Point Divide (FDV) Instruction

The FDV instruction performs a division of the process register pair (H)d by the operand (Y)d. The quotient is placed in Hd. No remainder is preserved.

Errors that can occur for the FDV instruction are as follows:

1. One or both of the numbers are not prenormalized.
2. Divisor = zero
3. Exponent overflow or underflow occurs during the initial subtraction of exponents.
4. Exponent overflow occurs when it is necessary to correct the condition where the divisor fraction is not greater in magnitude than the dividend fraction

When the dividend contains a zero fraction, a true zero (all bits are zero) is stored into the process register pair.

Floating Point Compare Instructions

The compare instructions perform an algebraic comparison between the double length operand (Y)d and the process register pair (H)d. The operand and process registers are left unaltered by these instructions.

The three instructions used in performing the comparisons are as follows:

1. FCL: The next instruction in memory is skipped if $Yd < Hd$.
2. FCE: The next instruction in memory is skipped if $Yd = Hd$.
3. FCG: The next instruction in memory is skipped if $Yd > Hd$.

It is assumed that all numbers are pre-normalized prior to executing the compare instructions. If the numbers are not pre-normalized, an error bit is set and the next instruction in memory is executed.

Double Precision Multiply (DMY) Instruction

The DMY instruction performs a multiplication of the double length operand (Y)d and a process register pair (H)d. The 32-bit truncated product is stored in the process register pair (H)d.

Multiplication of a (-1) multiplier by a (-1) multiplicand will result in an error condition. The process register pair is left unaltered and the next instruction in memory is executed. Use of the trap instruction will cause the next location in memory to be skipped if an error does not exist.

Double Precision Divide (DDV) Instruction

The dividend (H)d is divided by the divisor (Y)d. The quotient is placed in Hd and the remainder is discarded. An illegal divide will occur if the dividend is greater than the divisor or the dividend is equal to the divisor and has the same sign. An illegal divide will cause an error bit to be set and the process register pair will be left unaltered.

A trap instruction will cause the next instruction in memory to be skipped if no error condition exists.

Double Precision Compare Instructions

The compare instructions perform an algebraic comparison between the double length operand (Y)d and the process register pair (H)d. The operand and process registers are left unaltered by these instructions.

The instructions used in performing the comparisons are as follows:

1. DCL: The next instruction in memory is skipped if $Yd < Hd$.
2. DCE: The next instruction in memory is skipped if $Yd = Hd$.
3. DCG: The next instruction in memory is skipped if $Yd > Hd$.

Long Multiply (LMY) Instruction

The process register pair (H)d is multiplied by the operand (Y)d and the 64-bit product is stored in Hd and Hd + 1. The multiplicand is restricted to $H_{(0,1)}$ or $H_{(4,5)}$. The product will be stored in $H_{(0,1,2,3)}$ or $H_{(4,5,6,7)}$.

A (+1) product will result in an error condition. The process registers are left unaltered and the next instruction in memory is executed. Use of the trap instruction will cause the next location in memory to be shipped if no error condition exists.

Long Divide (LDV) Instruction

The 64-bit dividend (H)d, d + 1 is divided by the divisor (Y)d. The remainder is stored in Hd and the 32-bit quotient is stored in (H)d + 1.

The 64-bit dividend will be fetched from either $H_{(0,1,2,3)}$ or $H_{(4,5,6,7)}$.

An illegal divide will occur if the dividend is greater than the divisor or the dividend is equal to the divisor and has like sign. An illegal divide will cause an error bit to be set and the process registers will be left unaltered.

A trap instruction will cause the next instruction in memory to be skipped if no error condition exists.

Packaging and Power

The extended performance arithmetic unit is packaged on four logic cards which are inserted directly into the processor unit ATR case.

No additional power is required other than that already provided by the processor power supply.

Table VII-1. Summary of Instructions For Extended Performance Arithmetic Option

Mnemonic	Code	Operation	Name	Execution Times (μsec)
FAD	140*	Hd + Yd $\rightarrow$ Hd	Floating Point Add	5.24
FSB	141*	Hd - Yd $\rightarrow$ Hd	Floating Point Subtract	5.24
FMP	142*	Hd x Yd $\rightarrow$ Hd	Floating Point Multiply	6.24
FDV	143*	$\frac{Hd}{Yd} \rightarrow$ Hd	Floating Point Divide	12.64
FAD	150*	Hd + Yd $\rightarrow$ Hd†	Floating Point Add-Trap	5.24
FSB	151*	Hd - Yd $\rightarrow$ Hd†	Floating Point Subtract-Trap	5.24
FMP	152*	Hd x Yd $\rightarrow$ Hd†	Floating Point Multiply-Trap	6.24
FDV	153*	$\frac{Hd}{Yd} \rightarrow$ Hd	Floating Point Divide-Trap	12.64
FCL	101*	Skip Next Location If Y < H	Floating Point Compare, Jump is Less	2.44
FCE	102*	Skip Next Location If Y = H	Floating Point Compare, Jump if Greater	2.44
FCG	103*	Skip Next Location If Y > H	Floating Point Compare, Jump if Greater	2.44
DMY	144	Hd x Yd $\rightarrow$ Hd	Double Precision Multiply	7.24
DDV	145	$\frac{Hd}{Yd} \rightarrow$ Hd	Double Precision Divide	15.64
DMY	154	Hd x Yd $\rightarrow$ Hd†	Double Precision Multiply-Trap	7.24
DDV	155	$\frac{Hd}{Yd} \rightarrow$ Hd†	Double Precision Divide-Trap	15.64
DCL	105	Skip Next Location If Y < H	Double Precision Compare, Jump if Less	2.44
DCE	106	Skip Next Location If Y = H	Double Precision Compare, Jump if Less	2.44
DCG	107	Skip Next Location If Y > H	Double Precision Compare, Jump if Less	2.44
LMY	146	(H)d x (Y)d $\rightarrow$ Hd, d + 1	Long Multiply	7.64
LDV	147	$\frac{(Hd, d + 1)}{(Y)d} \rightarrow$ Hd + 1 Remainder $\rightarrow$ Hd	Long Divide	16.04
LMY	156	(H)d x (Y)d $\rightarrow$ Hd, d + 1†	Long Multiply-Trap	7.64
LDV	157	$\frac{(H)d, d + 1}{(Y)d} \rightarrow$ Hd + 1 Remainder $\rightarrow$ Hd†	Long Divide–Trap	16.04

*Literal Modes 2 or 3 will cause the instruction to act as an NOP.
†Skip Next Location If No Error

SECTION 8. L-304H SOFTWARE

The L-304H Computer incorporates a software package which includes a comprehensive set of operator-controlled utility/facility programs and a computer test program for use in detecting and isolating computer failures.

Operating System

The L-304H utilizes the standard operating system provided for all the Litton family of L-304 computers. This system has been developed to provide effective communication between the user and the computer system, and to aid in efficient use of the system.

The operating system is a group of programs that controls the loading and execution of object programs with provisions for relocatability and for card, tape (magnetic or paper), and printer handling. Troubleshooting aids are also provided in the form of memory dumps and program trace performed at operator-selected points. In addition the L-304 operating system provides the required flexibility to allow use of the computer in varying memory configurations with no less than 16K words as a minimum.

The L-304 operating system is categorized in three major groups: (1) the Resident Supervisor, (2) the Loading Control Programs, and (3) the Support Programs. Of these, only those programs in the Resident Supervisor category must be in core storage at all times.

The programs in the Resident Supervisor category can be further considered as three groups designated as Control Programs, Service Programs, and I/O Programs. The Control Programs are those which determine the course of action to be taken by the L-304 operating system. The Service Programs are those routines which perform specific non-I/O functions, such as the Octal Conversion Routine. Most of these routines are open; i.e., they may be called directly by the User Program, perform their function, and return to the User Program. The I/O Programs handle the L-304 peripherals in the manner best suited for the L-304 operating system. These routines are also open for use by the User Program, providing that the method of peripheral handling is suitable to the user's problem.

A block diagram of the L-304 operating system and related software is shown in Figure 8-1 and 8-2.

The Resident Supervisor structure is illustrated in Figure 8-3. The programs in this group also fall into three categories: (1) the System Loading Control Programs, (2) the Functional Input Programs, and (3) the Modification Programs. This structure is illustrated in Figure 8-4. The L-304 operating system Loading Control Programs consist of all programs other than the Resident Supervisor which may be used by L-304 during a load operation.

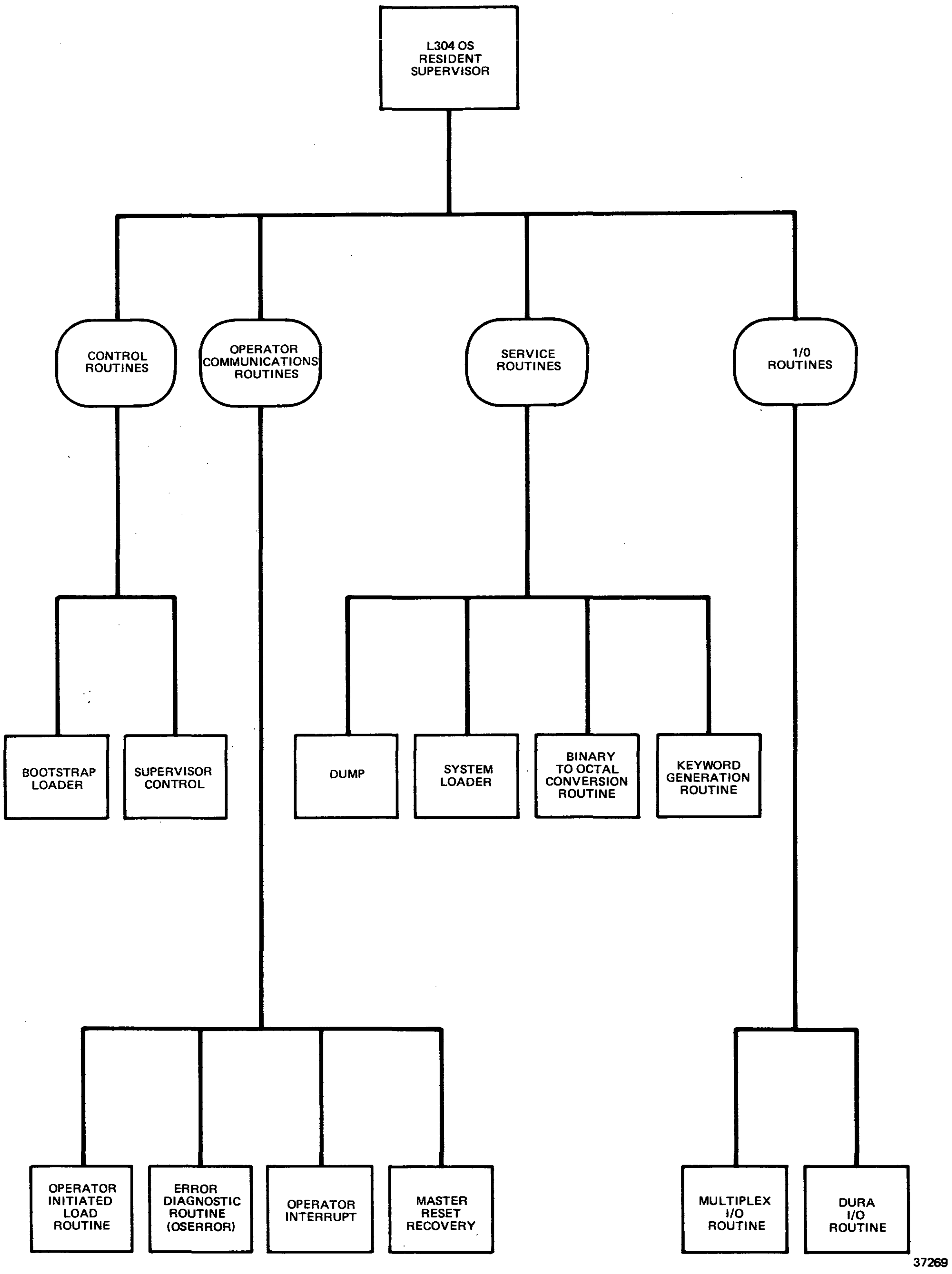


Figure 8-1. L-304 OS Resident Supervisor

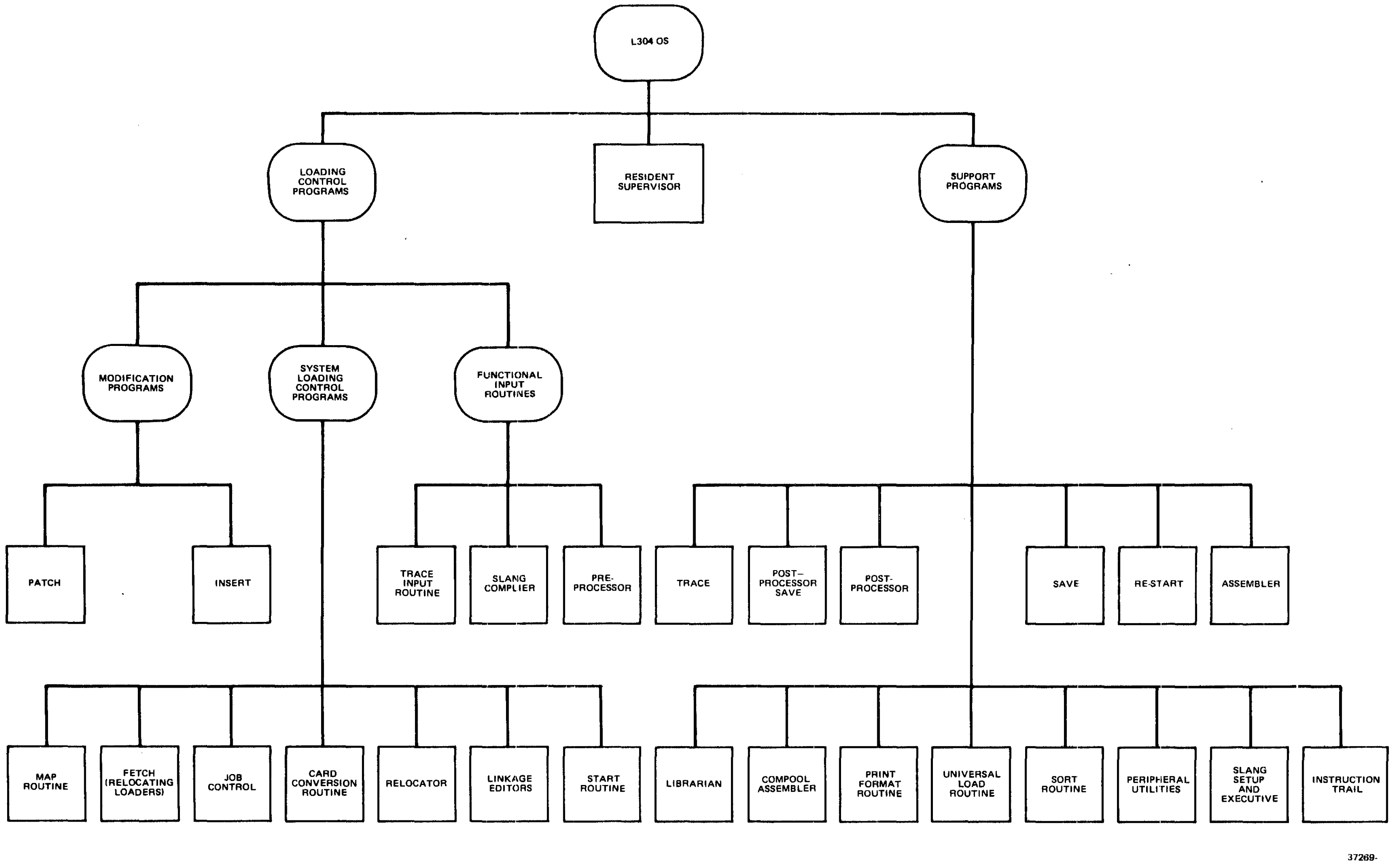


Figure 8-2. L-304 Operating System

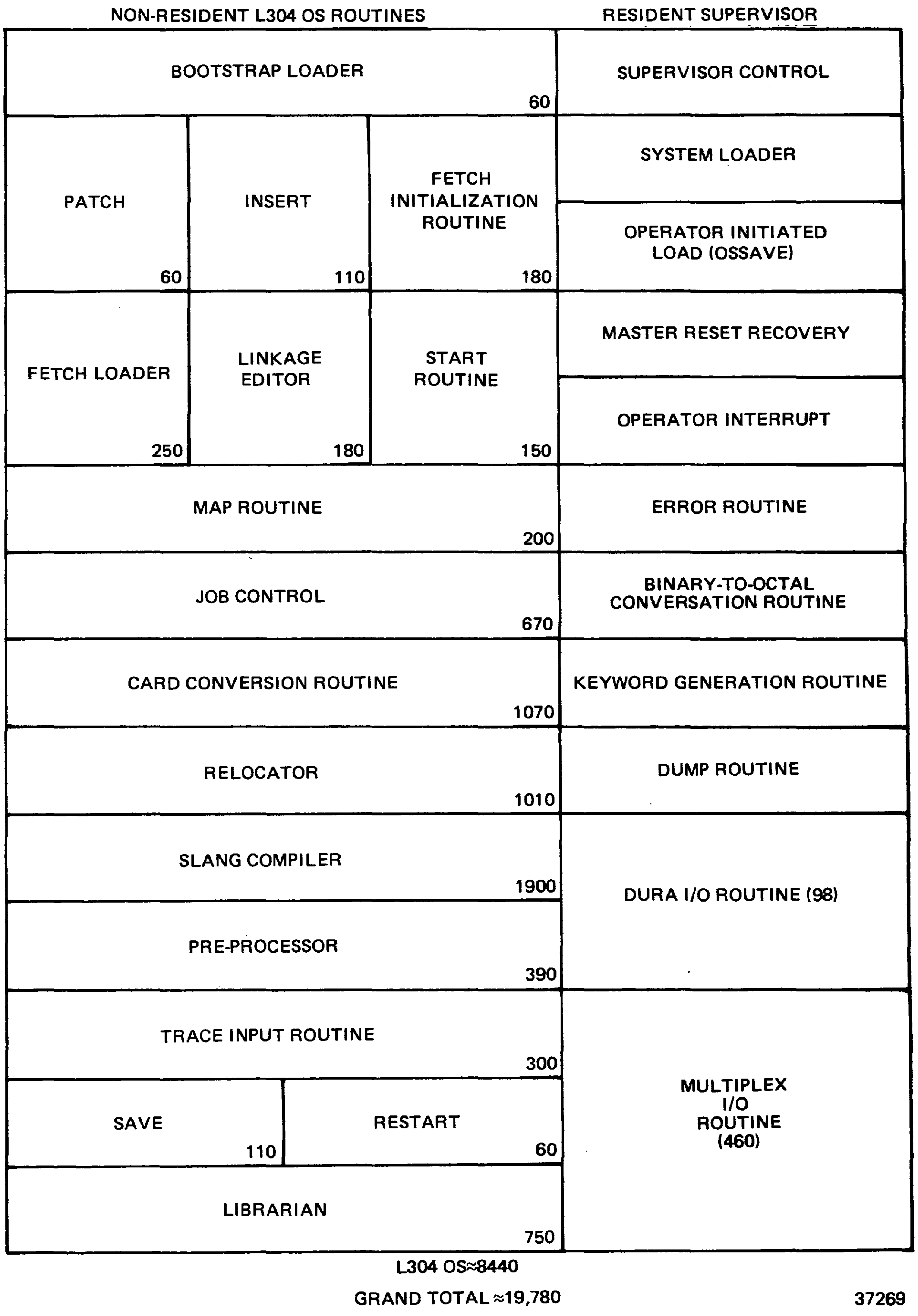


Figure 8-3. Resident Supervisor Structure

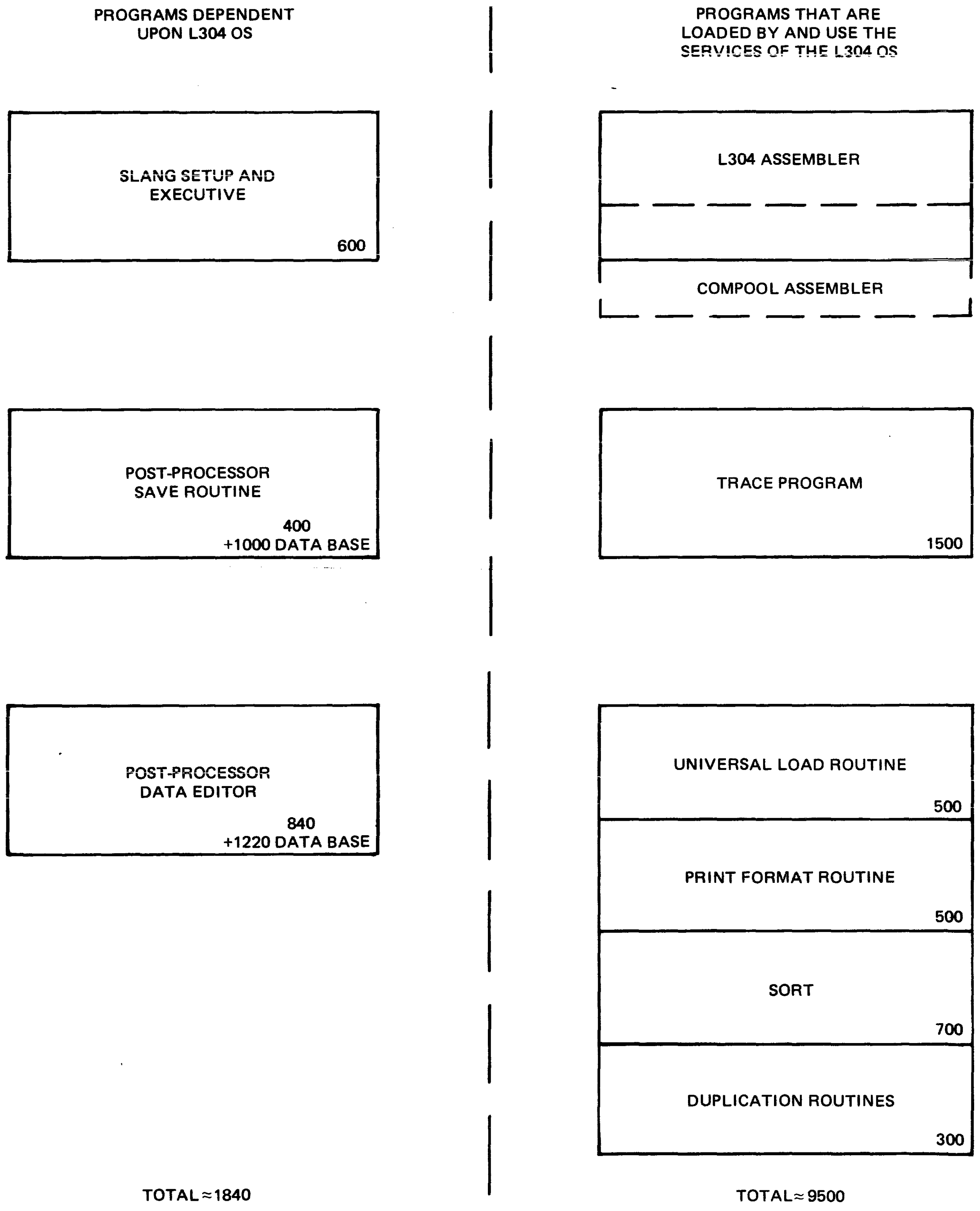


Figure 8-4. L-304 OS Support Programs

Assembly Program

The L-304 Assembly Program is available for use on a Litton L-304, or IBM 360 or IBM 370 computer.

The assembly language is relatively standard, i.e., each instruction (statement), excluding the label and comments field, consists of an operator and variable field. The operator (op code) identifies the operation (add, subtract, etc.); the variable field generally represents such things as storage locations, general registers, immediate data, or constant values. The variable field in assembly language can contain from 1 to 4 subfields, depending on the instruction class.

The assembly program translates or processes assembler-language programs into machine language for execution by the computer. The program written in the assembler language (used as input to the Assembler) is called the source program; the machine-language program produced as output from the Assembler is called the object program. The translation or processing procedure performed by the Assembler to produce the object program is called assembling. The object program produced is also referred to as an assembly. Program statements (source statements) written in assembler language may consist of: a label to identify the statement; a symbolic operation code (mnemonic) to identify the function the statement represents; and a variable field, consisting of one to four subfields to designate the data or storage locations used in the operation, and space for comments.

Symbolic instruction statements are one-for-one representations of L-304 machine instructions

The assembler language provides for the symbolic representation of any addresses, machine components (such as registers), and actual values needed in source statements. Also provided is a variety of forms of data representations: decimal, octal, hexadecimal, or character representation of binary machine values. (The programmer selects the representation best suited to express a given piece of data.)

The following example illustrates the use of the label (name), operation, variable field (operand), and comments entries as they appear on the coding form. An "Add" instruction has been labeled by the symbol LABL; the operation entry (ADD) is the mnemonic operation code for an add operation; and the variable field 5, 6 designates the two general registers whose contents are to be added.

1 8	10 14	16 71		73 80
LABEL	OPERA-TION	VARIABLE FIELD	COMMENTS	IDENTI-FICATION
LABL	ADD	5, 6	ADD REGISTER 5 TO 6. STORE SUM IN 6.	100

The basic structure of the assembler language is a source statement consisting of a label entry, an operation entry, and a variable field entry. The label entry (optional) is a symbol. The operation entry (mandatory) is a mnemonic operation code representing an assembler instruction. A variable field entry, if used, may be numeric or symbolic or a combination of numeric and symbolic. The variable field entry may consist of from zero to four subfields, depending upon the type of operation code specified.

Service Routines

The L-304 Service Routines listed below constitute functions which are used by most program systems. The routines may be loaded by the Operating System Program and entered from external programs or they may be entered directly by operator control.

Mathematical Routines

This group is divided into the following five subsections:

a. Trigonometric Functions

- Sine/Cosine
- Tangent
- Arc Tangent
- Arc Sine/Arc Cosine

b. Square Root

c. Double Precision Arithmetic Functions

- Add/Subtract
- Multiply
- Divide

d. Logarithmic Functions (Base 2, e, or 10)

e. Exponential Function (X^y)

Conversion Routines

The routines within this group will:

a. Format 16-bit data words into EBCDIC characters for binary, octal, hex, or BCD output
b. Pack binary, octal, hex, or BCD EBCDIC character inputs into 16-bit words
c. Convert binary to BCD or BCD to binary. The Edit will convert a signed binary fraction to a signed BCD fraction and format in EBCDIC (with leading zeros suppressed and inserted decimal point).

Interceptor/Simulator Program

The L-304 Interpreter/Simulator is a group of seven modules (separate assemblies) designed to simulate the execution of any L-304 program on the IBM 360. The simulator is written in IBM 360 Basic Assembly Language and can be run on any model of the IBM 360 computer which is operating under the S6360 Operating System. The results of a simulator run will, with the specific exceptions, be identical to the results that would be obtained if the same program were to be executed on an L-304 computer.

The simulator is designed to accept as input the output of the L-304 Assembler Program. This input consists of a machine language representation of the programmed instructions. The user of the simulator indicates, via control cards:

a. The trace or dump features that he wishes to use.
b. The peripheral device(s) he wishes to simulate.
c. The modification he wishes to make to the assembled program(s).
d. The address and program level at which the execution of the L-304 program is to start.
e. The maximum number of lines of output or the maximum number of L-304 instructions he wishes to execute in this run.

The seven modules of the simulator have specific functions.

The Input Module has four major functions:

1. To read in the control cards
2. To read in the binary deck(s)
3. To resolve any external linkages in the L-304 programs
4. To relocate, if necessary, the data in the binary deck(s).

The Universal Load Routine deformats the control cards read by the Input Module and stores the data from the various fields in the 360 core.

The Instruction Interpreter Module performs the execution of all L-304 instructions except the three I/O instructions and maintains the L-304 real time clock.

The Peripheral Simulator simulates the peripheral devices that normally interface with the L-304.

The Debugging Module is responsible for generating program traces, and dump output.

The I/O Control Simulator simulates both the L-304 IOC and the I/O devices defined by the customer that are to be present on the L-304. This module also maintains the L-304 real time clock.

General Machine Test Program (GMT)

The GMT program consists of a set of program modules which exercise and test the functions of the central processor unit, memory unit(s) and I/O control unit. The program is constructed to facilitate computer troubleshooting by providing an error halt option at each test point and an option to loop on a given test.

The program is loaded into memory from card decks using the bootstrap load feature of the L-304.

The program modules of the GMT have specific functions described as follows:

a. Instruction Test – all instructions (except I/O) and addressing modes are tested.

b. Internal Interrupt Test – all logic related to program level call is tested except for I/O stimulators.

c. Real Time Clock Test – the L-304 program controlled clock is tested along with I/O stimulated program level call.

d. Computer Time-Out Logic Test – central processor, memory, and I/O control timeout logic tested.

e. Watchdog Timer Test – the programmed controlled times is tested.

f. Data Entry and Display Test – the operator communication functions are tested

g. Extended Memory Addressing – all logic associated with memory access by program is tested

h. Memory Test – one or more memories are tested and the test is directed to locating failures in the data lines, addressing lines, and memory control logic.

SECTION 9. SYSTEM PACKAGING, POWER, AND CONTROLS

Design and packaging of the L-304H Computer makes it suitable for use in a wide variety of rugged tactical environments.

Each module of the airborne version of the L-304H Computer is packaged in a three-quarter long ATR case, which measures 7.5 inches by 7.5 inches by 19.5 inches and weighs approximately 30 pounds. One ATR case contains the processor unit, including the control, arithmetic, and input/output circuits. Another case, containing the memory unit, includes a 16,384 word core stack and associated memory circuitry (Figures 9-1 and 9-2).

Provision for growth is afforded by the modular construction technique used throughout both units. Memory size can be increased by simply adding 16K memory units in a plug-in arrangement. Multiple processor capability is likewise achieved by the modular addition of processor units.

Standard electronic circuits, consisting of high-speed Schottky SSI and MSI elements are used throughout the processor and memory units. Eight types of SSI and 14 types of MSI circuits are used, with components mounted on four- to six-layer printed circuit cards constructed by standard plated-through hole technique, with ground and power layers provided.

Heat generated on the cards is conducted through integral bar-type heat sinks to the sides of the ATR case where spring pressure is applied to maintain positive contact to cold plates on the side of the structure. With the use of cold plates, components are isolated from air flow contaminants.

The processor and memory units are powered by integral power supplies. Input power is 115 volts, 400 Hertz. Processor power consumption is 220 watts; memory power consumption is 260 watts under nominal operating conditions. Automatic, orderly shutdown is provided by power sequencing circuitry, and power loss protection is provided by power fault interrupt to the CPU. Operation in both steady-state and transient conditions is as specified in MIL-STD-704A for category B equipment.

External interface connectors are mounted either on the front faces of the ATR cases or on the rear depending upon specific customer applications. VAST test connectors are provided on the sides of the cases and on the edges of the printed circuit cards.

The ATR structures are dip-brazed assemblies which give extremely high strength-to-weight ratios and incorporate two side cold plates of finned aluminum. As a result of the dip-brazing process, the fins become an integral part of the primary structure and provide a direct, unimpaired thermal path for all internal components. Cooled air is not required, and the computer uses only forced ambient air for heat dissipation.

The L-304H Computer is designed for installation in a variety of ground, shipboard, and airborne environments, and meets the following military specifications:

MIL-E-5400N	Class 2X
MIL-E-16400	Classes 3 and 4
MIL-I-461A	Notice 3
MIL-E-4158	
MIL-STD-704A	Category B

Processor Unit

The processor unit ATR case includes the central processing unit (CPU), input/output unit (IOU), and associated power supply. Twenty logic card connectors are provided. The CPU uses 11 printed circuit cards, the IOU uses 3 cards, and 6 card slots are provided as spares for expansion. Each card is 6 inches x 7 inches and contains a mix of SSI and MSI components up to a maximum of 80 microcircuits. A single wire-wrap plate is used to interconnect the cards with 180-pin fork and blade connectors used to connect the cards to the plate. Functional partitioning and byte slicing maximize fault isolation techniques.

Access to the functional cards and power supply assemblies is gained by removing the top shear cover of the unit case. Removal of the bottom shear cover provides access to the wiring side of the wire-wrap plate. All covers are fitted with EMI gaskets.

The power supply furnishes a regulated +5 volt, 40 ampere output to the processor unit. Distributed power is routed through the VAST connector to accommodate VAST testing requirements. A terminating cap (mating connector) is provided to return direct power during operation, and each card is supplied power through a separate power cable. The power supply has been packaged with four separate, removable units. In addition to providing thermal advantages, this packaging of elements accommodates testing, fault isolation, and replacement of elements. The four units are: (1) switch assembly, (2) rectifier assembly, (3) auxiliary regulator card assembly, and (4) main regulator control card assembly. These four units plug in directly to the single processor unit wire wrap plate.

The switch and rectifier assemblies are modular subassemblies using a one-piece heat sink which mounts discrete components directly to its surface. The heat sink is then mounted directly to the system cold plates, providing a direct thermal path to system cooling.

The two card assemblies are two-sided printed circuit cards with discrete components mounted on a bar type heat sink similar to the logic cards. Thermal conduction is achieved through spring clips to primary structure which is attached to the cold plates. Electrical connections are made through 30 and 90-pin fork and blade type connectors.

Sense circuitry has been added to protect the power supply and to automatically shut down power under the following conditions:

a. Overvoltage
b. Overcurrent
c. Overtemperature
d. Loss of phase
e. Over/under voltage input

Automatic restart will occur under the following conditions:

a. The temperature decreases to the operating level.

b. The input voltage reaches the level required by specification.

Fuses have been added within the power supply to protect the unit from an internal short-circuit condition.

Memory Unit

The memory unit ATR case contains six types of 17 printed circuit cards, a 16K core stack, and the memory power supply. The circuit cards are 4 inches x 7 inches and are interconnected by a single wire-wrap plate with fork and blade connectors used to connect the cards to the plate. The 16K by 36-bit core stack is a single plug-in unit, and contains integral sense amplifier circuits. Four rails on the corners of the core stack provide physical support and location fixing. In addition, the rails provide thermal conductive cooling paths for conducting heat to the cold wall side plates. The core stack uses 14 mil cores, selected for optimum performance and reliability.

The 16K memory power supply provides regulated direct current voltages to the 16K core memory unit. The power supply accepts 3-phase, 115-vac, 400-Hz power input and furnishes the memory with the following voltages:

+5v $\pm$10 percent @ 11.5 amps

+5v $\pm$5 percent @ 1.0 amp

-5v $\pm$5 percent @ 0.5 amp

+23v $\pm$7 percent @ 0.5 amp

+32v $\pm$10 percent @ 0.7 amp

+12v variable (8.5 to 12 amps) matched to stack characteristics

The power supply consists of six major assemblies:

Main Chassis. Contains the high power components and serves as the housing assembly for the power supply.

Component Chassis. Contains the output rectifier filters.

Four Circuit Cards. Auxiliary converter, $\pm$5v regulator, +5v regulator, and logic card.

Heat is removed from the power supply via the thermal conduction from the main chassis to the cold plates. The power supply provides a single 132-pin output connector which contains both the output voltages and test points internal to the power supply. The test points were selected to provide compatibility with VAST testing requirements. In addition, the power supply provides a fault signal to the memory to indicate a power supply malfunction; it also contains the same startup, shutdown, and protective circuitry features as the processor power supply.

System Control Panel

The system control panel (SCP) for the L-304H, shown in Figure 9-3, is a self-contained subunit which can be mounted on the front of the processor case or located remotely.

The panel indicators and switches are grouped as follows:

Indicators	Data Display
Functional Switches	Data & Base Select Switches

INDICATORS

RUN: Illuminates when CPU is running.

OVER TEMP: Illuminates when processor case temperature reaches the warning level; if temperature increases to the upper limit, system power is shut down.

PWR FAULT PROC: Illuminates when the processor power supply fails.

PWR FAULT MEM: Illuminates when a memory power supply fails.

IOU MEM TIME OUT: Illuminates on a detected failure for the IOU-memory interface.

IOU TIME OUT: Illuminates on a detected failure for the IOU-external device interface.

CPU MEMORY TIME OUT: Illuminates on a detected failure for the CPU-memory interface.

PROGRAM TIME OUT: Illuminates on failures such as an internal CPU failure, an illegal operation code, or a time out of the CPU-SPU interface.

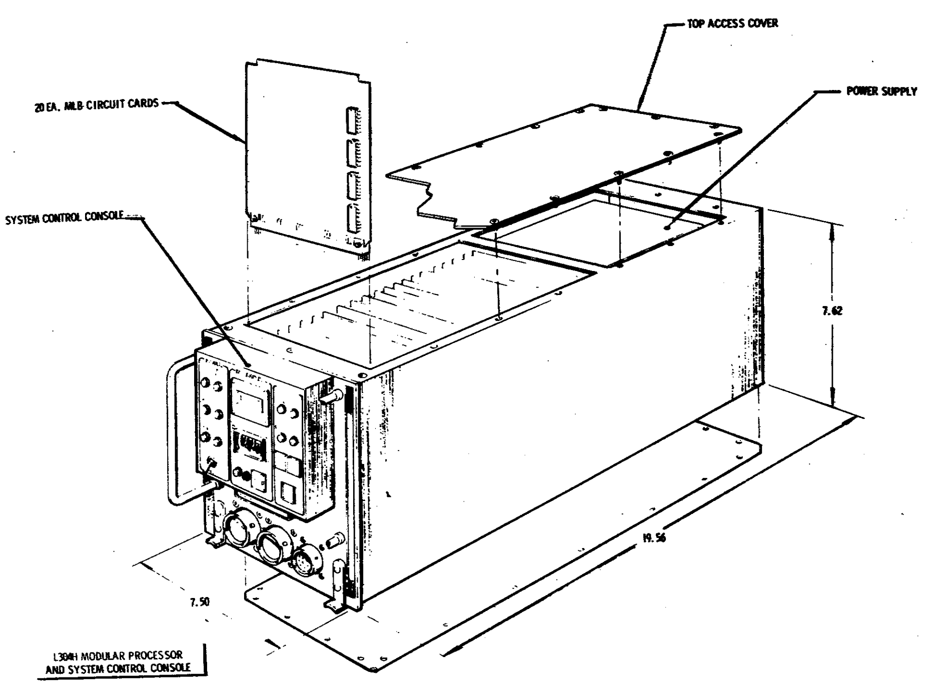


Figure 9-1. L-304H Processor Packaging

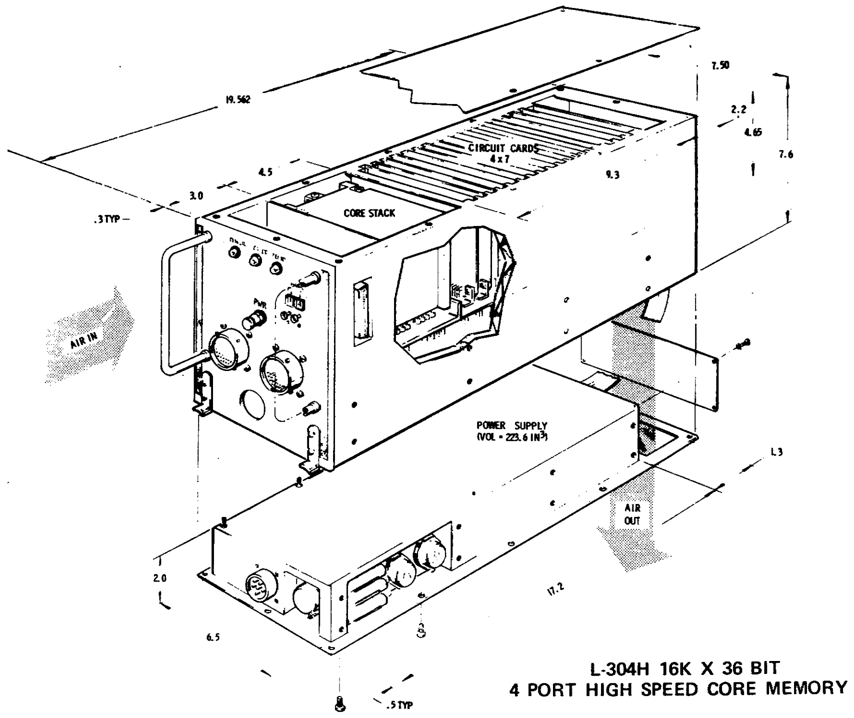


Figure 9-2. L-304H Memory Packaging

FUNCTIONAL SWITCHES

LAMP TEST: Actuation causes all switch caps and indicators to illuminate and forces zeroes into the data display.

PARITY ERROR: Switchcap illuminates when memory parity error is detected; actuation of switch resets display.

DATA ENTER: Actuation interrupts the processor and indicates to the program that the operator has entered a code into the DATA SELECT switches.

POWER: Actuation turns on the processor power supply.

SYS RESET: Actuation causes a simulated power failure and resets the processor and memory to their initial conditions; release of the switch simulates restoration of power, i.e., the processor is forced to program level 76_8 and started.

PROG LOAD: Actuation initially causes a system reset; upon recovery, the CPU remains inactive and the Bootstrap signal is sent to the IOU.

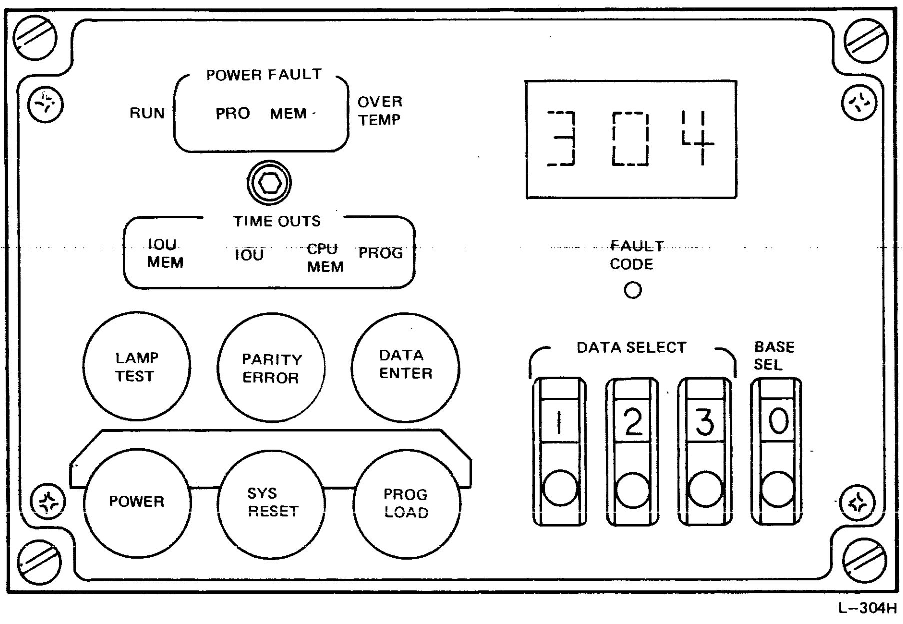


Figure 9-3. L-304H System Control Panel

DATA DISPLAY: Three-digit decimal LED readout under program control which provides program communication with operator; the FAULT CODE indicator calls attention to the fact that a fault code is present.

DATA SELECT: Three decimal switches used by operator to insert data into processor; the switches can be interrogated at any time by the program, but the DATA ENTER switch must be actuated to interrupt the running program.

BASE SEL: Eight position switch by which the operator designates which of the memories is to serve as the base memory.

In addition to the controls and indicators on the system control console, each 16K memory has a prime power on/off switch and indicator, a memory bank address switch, and a power fault indicator.

Litton

DATA SYSTEMS

NTDS UN[illegible] COMPUTER

AN/USQ-20 *Repertoire of Instructions*

01	Right SHift • Q	Shift (Q) Right by Y
02	Right SHift • A	Shift (A) Right by Y
03	Right SHift • AQ	Shift (AQ) Right by Y
04*	COMpare • A, • Q, • AQ	Sense (j); $(A)_i = (A)_f$
05	Left SHift • Q	Shift (Q) Left by Y
06	Left SHift • A	Shift (A) Left by Y
07	Left SHift • AQ	Shift (AQ) Left by Y
10	ENTer • Q	$Y \rightarrow Q$
11	ENTer • A	$Y \rightarrow A$
12	ENTer • B^n	$Y \rightarrow B^j$
13^	EXternal - FunCTion • C^n	$\hat{j} \neq 0$ or 1, $(Y) \rightarrow C^{\hat{j}}$, $\hat{j} = 0$ or 1, See Note.
14	SToRe • Q	$(Q) \rightarrow Y$; $k=0, Q' \rightarrow Q$
15	SToRe • A	$(A) \rightarrow Y$; $k=4, A' \rightarrow A$
16	SToRe • B^n	$(B)^j \rightarrow Y$
17^	SToRe • C^n	$(C)^{\hat{j}} \rightarrow Y$
20	ADD • A	$(A) + Y \rightarrow A$
21	SUBtract • A	$(A) - Y \rightarrow A$
22	MULtiply	$(Q)\,Y \rightarrow AQ$
23*	DIVide	$(AQ)/Y \rightarrow Q.\ R \rightarrow A_f$
24	RePLace • A+Y	$(A)+(Y) \rightarrow Y$ & A
25	RePLace • A−Y	$(A)-(Y) \rightarrow Y$ & A
26*	ADD • Q	$(Q) + Y \rightarrow Q.\ (A)_i = (A)_f$ } j interpretation
27*	SUBtract • Q	$(Q) - Y \rightarrow Q.\ (A)_i = (A)_f$ } reversed for A&Q
30	ENTer • Y+Q	$Y + (Q) \rightarrow A$
31	ENTer • Y−Q	$Y - (Q) \rightarrow A$
32	SToRe • A+Q	$(A)+(Q) \rightarrow Y$ & A
33	SToRe • A−Q	$(A)-(Q) \rightarrow Y$ & A
34	RePLace • Y+Q	$(Y)+(Q) \rightarrow Y$ & A
35	RePLace • Y−Q	$(Y)-(Q) \rightarrow Y$ & A
36	RePLace • Y+1	$(Y)+1 \rightarrow Y$ & A
37	RePLace • Y−1	$(Y)-1 \rightarrow Y$ & A
40*	ENTer • LP**	$L[Y(Q)] \rightarrow A$; j=2, even parity; j=3, odd parity
41	ADD • LP	$L[Y(Q)] + (A) \rightarrow A$
42	SUBtract • LP	$(A) - L[Y(Q)] \rightarrow A$
43	COMpare • MASK	$(A) - L[Y(Q)]$ SENSE (j), $(A) + L[Y(Q)]$; $(A)_j = (A)_f$
44*	RePLace • LP	$L(Y)(Q) \rightarrow Y$ & A; j=2, even parity; j=3, odd parity
45	RePLace • A+LP	$L(Y)(Q) + (A) \rightarrow Y$ & A
46	RePLace • A−LP	$(A) - L(Y)(Q) \rightarrow Y$ & A
47	SToRe • LP	$L(A)(Q) \rightarrow Y$; $(A)_i = (A)_f$
50	SELective • SET	SET $(A)_n$ FOR $Y_n = 1$
51	SELective • CP**	COMPLEMENT $(A)_n$ FOR $Y_n = 1$
52	SELective • CL**	CLEAR $(A)_n$ FOR $Y_n = 1$
53	SELective • SU**	$Y_n \rightarrow (A)_n$ FOR $(Q)_n = 1$

54	Replace SElective • SET	SET $(A)_n$ FOR $(Y)_n = 1$, $\rightarrow Y$ & A
55	Replace SElective • CP	COMPLEMENT $(A)_n$ FOR $(Y)_n = 1$, $\rightarrow Y$ & A
56	Replace SElective • CL	CLEAR $(A)_n$ FOR $(Y)_n = 1$, $\rightarrow Y$ & A
57	Replace SElective • SU	$(Y)_n \rightarrow (A)_n$ FOR $(Q)_n = 1$, $\rightarrow Y$
60	JumP (arithmetic)	Jump to Y if j-condition is satisfied.
61	JumP (manual)	(see JP & RJP j-Designators)
62^	JumP (if • C^n has ACTIVE INput buffer)	Jump to Y if $C^{\hat{j}}$ input buffer active } (see JP & RJP j-Designators)
63^	JumP (if • C^n has ACTIVE OUTput buffer)	Jump to Y if $C^{\hat{j}}$ output buffer active } (see JP & RJP j-Designators)
64	Return JumP (arithmetic)	Jump to Y+1 and $P+1 \rightarrow Y_L$ if j condition is
65	Return JumP (manual)	satisfied (see JP & RJP j-Designators)
66^	TERMinate • C^n • INPUT	Terminate input buffer on channel $\hat{j}$
67^	TERMinate • C^n • OUTPUT	Terminate output buffer on channel $\hat{j}$
70*	RePeaT	Execute NI Y times
71	BSKip • B^n	$(B)^j = Y$, skip NI and clear $(B)^j$, $(B)^j \neq Y$, Advance B^j and read NI
72	BJumP • B^n	$(B)^j = 0$, read NI; $(B)^j \neq 0$, $(B)^j - 1$ and jump to address Y
73^	INput • C^n (without monitor mode)	Buffer IN on $C^{\hat{j}}$; $\hat{k}=3, (Y) \rightarrow (00100+\hat{j})$; $\hat{k}=1, (Y)_L \rightarrow (00100+\hat{j})_L$; $\hat{k}=0, Y \rightarrow (00100+\hat{j})_L$.
74^	OUTput • C^n (without monitor mode)	Buffer OUT on $C^{\hat{j}}$; $\hat{k}=3, (Y) \rightarrow (00120+\hat{j})$; $\hat{k}=1, (Y)_L \rightarrow (00120+\hat{j})_L$; $\hat{k}=0, Y \rightarrow (00120+\hat{j})_L$.
75^	INput • C^n (with • MONITOR mode)	Buffer IN on $C^{\hat{j}}$ with mon. $\hat{k}=3, (Y) \rightarrow (00100+\hat{j})$; $\hat{k}=1, (Y)_L \rightarrow (00100+\hat{j})_L$; $\hat{k}=0, Y \rightarrow (00100+\hat{j})_L$. mon. inter. at $00040+\hat{j}$
76^	OUTput • C^n (with • MONITOR mode)	Buffer OUT on $C^{\hat{j}}$ with mon. $\hat{k}=3, (Y) \rightarrow (00120+\hat{j})$; $\hat{k}=1, (Y)_L \rightarrow (00120+\hat{j})_L$; $\hat{k}=0, Y \rightarrow (00120+\hat{j})_L$. mon. inter. at $00060+\hat{j}$

— NO-OPeration	CS-1 Mono-codes
— ComPlement • A or • Q	CS-1 Mono-codes
— CLear • A, • Q, • B^n, or Y	CS-1 Mono-codes
— Remove Interrupt Lockout	CS-1 Mono-codes
— Remove Interrupt Lockout and JumP • Y	CS-1 Mono-codes
— TEST • CO or • CI	CS-1 Mono-codes

**LP - Logical Product   CP - Complement   SU - Substitute   CL - Clear   *, ^ } Special j & k Designators (see opposite side of card)   Y - The operand; Y or (Y)

NOTE: Skip NI if other Computer (on channel 0 or 1) has input buffer active. ~~Execute twice.~~

NTDS UNIT COMPUTER

AN/USQ-20 *Repertoire of Instructions*

j-DESIG.

j	(Not applicable on * or ^) Skip Code
0	(no skip)
1	SKIP
2	Q POS
3	Q NEG
4	A ZERO
5	A NOT Zero
6	A POS
7	A NEG

NORMAL k-DESIGNATORS

	READ		STORE		REPLACE		
k	Code	Origin	Code	Dest.	Code	Origin	Dest.
0	'blank'	U_L	Q	Q	'not used'	—	—
1	L	M_L	L	M_L	L	M_L	M_L
2	U	M_U	U	M_U	U	M_U	M_U
3	W	M	W	M	W	M	M
4	X	XU_L	A	A	'not used'	—	—
5	LX	XM_L	CPL	Cpl M_L	LX	XM_L	M_L
6	UX	XM_U	CPU	Cpl M_U	UX	XM_U	M_U
7	A	A	CPW	Cpl M	'not used'	—	—

LEGEND

M – Memory word (30 bits)	Cpl – Complement
M_L – Lower half memory word	A – A-register
M_U – Upper half memory word	Q – Q-register
X – Sign bit extended	U – U-register

$\hat{j}$-DESIGNATORS

(4 bits)

$\hat{j}$ Occupies 4 bit positions and represents C^n where n may be $0 — 15_8$
The instruction word assumes the format:

f	$\hat{j}$	$\hat{k}$	b	y
29 – –24	23 – 20	19 18	17 – 15	14 – –0

$\hat{k}$-DESIGNATORS

(2 bits)

$\hat{k}$	EX–FCT 13	STR•C^n 17	JP 62 63	IN•C^n, OUT•C^n 73 75 74 76
0	'not used'	'not used'	'blank'	'blank'
1	'not used'	'not used'	L	L
2	'not used'	'not used'	U	'not used'
3	W	W	W	W

JP & RJP j-DESIGNATORS

j	JP 60	RJP 64	JP 61	RJP 65
0	(No Jump)*		(Uncond. Jump)	
1	(Uncond. Jump)*		KEY 1	
2	Q POS		KEY 2	
3	Q NEG		KEY 3	
4	A ZERO		STOP	
5	A NOT Zero		STOP 5	
6	A POS		STOP 6	
7	A NEG		STOP 7	
$\hat{j}$	62 $\hat{j}$		63 $\hat{j}$	
$0\text{-}15_8$	C^n ACTIVE IN		C^n ACTIVE OUT	

*60 Clears interrupt & bootstrap modes.

* j-DESIGNATORS

j	COM•A, •Q, •AQ 04		DIV 23	ADD•Q 26	SUB•Q 27	ENT•LP 40	RPL•LP 44	RPT 70		
0	(no skip)		(no skip)	(no skip)		(no skip)		(no mod.)	Y of NE = Y	
1	(unconditional skip)		SKIP	SKIP		SKIP		ADV	Y of NE = Y+1	
2	Y LESS	Y ≤ (Q)	NO Over Flow	A POS		EVEN parity		BACK	Y of NE = Y−1	
3	Y MORE	Y > (Q)	Over Flow	A NEG		ODD parity		ADD B	Y of NE = Y + B^b	
4	Y IN	(Q) ≥ Y and Y > (A)	A ZERO	Q ZERO		A ZERO		Rpl. Inc.	Y of NE = Y $\boxed{+B^6}$	✓
5	Y OUT	(Q) < Y or Y ≤ (A)	A NOT Zero	Q NOT Zero		A NOT Zero		ADV R	Y of NE = Y+1 $\boxed{+B^6}$	✓
6	Y LESS	Y ≤ (A)	A POS	Q POS		A POS		BACK R	Y of NE = Y−1 $\boxed{+B^6}$	✓
7	Y MORE	Y > (A)	A NEG	Q NEG		A NEG		ADD B R	Y of NE = Y + B^b $\boxed{+B^6}$	✓

✓ B^6 Increment if NI is RPL class; increments Y address for the store portion of the replace.
NE — Next execution

Table IV-6. Index of Instructions by Function Code

M 7.

FUNCTION CODE	MNEMONIC CODE	PAGE NUMBER IN TEXT	INSTRUCTION
00	MLT	2-54	HALT
01	EXE	2-50	EXECUTE
02	EXC	2-17	EXCHANGE
03	EXD	2-17	EXCHANGE DOUBLE
04	LDR	2-12	LOAD RH
05	STR	2-12	STORE RH
06	LDD	2-13	LOAD DOUBLE
07	STD	2-13	STORE DOUBLE
10	ADD	2-18	ADD
11	SUB	2-19	SUBTRACT
12	RAD	2-19	REPLACE ADD
13	RUB	2-19	REPLACE SUBTRACT
14	ADA	2-20	ADD ABSOLUTE
15	SBA	2-20	SUBTRACT ABSOLUTE
16	LDA	2-14	LOAD ABSOLUTE
17	LDC	2-14	LOAD COMPLEMENT
20	EOR	2-22	EXCLUSIVE OR
21	IOR	2-23	INCLUSIVE OR
22	AND	2-23	LOGICAL AND
23	MBD	2-51	MEMORY BANK DESIGNATOR
24	RER	2-24	REPLACE EXCLUSIVE OR
25	RIR	2-25	REPLACE INCLUSIVE OR
26	RAN	2-26	REPLACE LOGICAL AND
27	MBA	2-51	MEMORY BANK ASSIGNMENT
30	MPY	2-21	MULTIPLY
31	DIV	2-21	DIVIDE
32	DTX	2-31	DECREMENT RH BY 2, TRANSFER IF RH ≠ 0
32 (E = 1)	ITX	2-32	TRANSFER AND INCREMENT RH BY 2
33	DOX	2-32	DECREMENT RH BY 1, TRANSFER IF RH ≠ 0
33 (E = 1)	IOX	2-33	TRANSFER AND INCREMENT RH BY 1
34	XFR	2-33	TRANSFER UNCONDITIONAL
35	XLK	2-33	TRANSFER UNCONDITIONAL AND STORE LINK
36	XSW	2-34	TRANSFER ON CONSOLE TRANSFER SWITCH
37	JTW	2-36	JUMP THREE-WAY

FUNCTION CODE	MNEMONIC CODE	PAGE NUMBER IN TEXT	INSTRUCTION
40	XEZ	2-34	TRANSFER IF RH = 0
41	XNZ	2-35	TRANSFER IF RH ≠ 0
42	XNG	2-35	TRANSFER IF RH IS NEGATIVE
43	XPS	2-35	TRANSFER IF RH IS POSITIVE
44	SLL	2-27	SHIFT LONG LEFT
45	NLL	2-28	NORMALIZE LONG LEFT
46	SNC	2-30	SHIFT AND COUNT
47	RFT	2-30	REFLECT
50	CJL	2-36	COMPARE, JUMP IF LESS
51	CJE	2-37	COMPARE, JUMP IF EQUAL
52	CJU	2-37	COMPARE, JUMP IF UNEQUAL
53	CJG	2-38	COMPARE, JUMP IF GREATER
54	GCI	2-38	GATED COMPARISON, JUMP IF INSIDE
55	GCO	2-38	GATED COMPARISON, JUMP IF OUTSIDE
56	SLR	2-29	SHIFT LONG RIGHT
57	SAR	2-29	SHIFT ALGEBRAICALLY RIGHT
60 (OR 61)	SET	2-52	SET BIT
62 (OR 63)	CLR	2-53	CLEAR BIT
64 (OR 65)*	SKZ	2-39	SKIP IF BIT IS ZERO
66 (OR 67)*	SKN	2-40	SKIP IF BIT IS ONE
70	MVZ	2-16	MOVE AND ZERO
71	MVI	2-14	MOVE AND INSERT
72	STZ	2-54	STORE ALL ZEROS
74	DEV	2-44	SPECIAL DEVICE COMMAND
75	ITR	2-45	INPUT TO REGISTER
76	OFR	2-45	OUTPUT FROM REGISTER
77	NOP	2-49	NO OPERATION

*INDICATES DUAL FUNCTION CODE FOR IDENTICAL INSTRUCTION.

2502-8

Table IV-5. L-304F Instruction Index

ARITHMETIC INSTRUCTIONS

INSTRUCTION	FUNCTION CODE	MNEMONIC CODE	PAGE NUMBER
ADD ABSOLUTE	14	ADA	2-20
ADD	10	ADD	2-18
DIVIDE	31	DIV	2-21
MULTIPLY	30	MPY	2-21
REPLACE ADD	12	RAD	2-19
REPLACE SUBTRACT	13	RUB	2-19
SUBTRACT ABSOLUTE	15	SBA	2-20
SUBTRACT	11	SUB	2-19

DATA TRANSMISSION INSTRUCTIONS

INSTRUCTION	FUNCTION CODE	MNEMONIC CODE	PAGE NUMBER
EXCHANGE	02	EXC	2-17
EXCHANGE DOUBLE	03	EXD	2-17
LOAD ABSOLUTE	16	LDA	2-14
LOAD COMPLEMENT	17	LDC	2-14
LOAD DOUBLE	06	LDD	2-13
LOAD RH	04	LDR	2-12
MOVE AND INSERT	71	MVI	2-14
MOVE AND ZERO	70	MVZ	2-16
STORE DOUBLE	07	STD	2-13
STORE RH	05	STR	2-12

LOGIC INSTRUCTIONS

INSTRUCTION	FUNCTION CODE	MNEMONIC CODE	PAGE NUMBER
LOGICAL AND	22	AND	2-23
EXCLUSIVE OR	20	EOR	2-22
INCLUSIVE OR	21	IOR	2-23
REPLACE LOGICAL AND	26	RAN	2-26
REPLACE EXCLUSIVE OR	24	RER	2-24
REPLACE INCLUSIVE OR	25	RIR	2-25

SHIFT INSTRUCTIONS

INSTRUCTION	FUNCTION CODE	MNEMONIC CODE	PAGE NUMBER
NORMALIZE LONG LEFT	45	NLL	2-28
SHIFT ALGEBRAICALLY RIGHT	57	SAR	2-29
SHIFT LONG LEFT	44	SLL	2-27
SHIFT LONG RIGHT	56	SLR	2-29
SHIFT AND COUNT	46	SNC	2-30
REFLECT	47	RFT	2-30

INPUT/OUTPUT INSTRUCTIONS

INSTRUCTION	FUNCTION CODE	MNEMONIC CODE	PAGE NUMBER
SPECIAL DEVICE COMMAND	74	DEV	2-44
INPUT TO REGISTER	75	ITR	2-45
OUTPUT FROM REGISTER	76	OFR	2-45

TRANSFER INSTRUCTIONS

INSTRUCTION	FUNCTION CODE	MNEMONIC CODE	PAGE NUMBER
DECREMENT RH BY 1, TRANSFER IF RH ≠ 0	33	DOX	2-32
DECREMENT RH BY 2, TRANSFER IF RH ≠ 0	32	DTX	2-31
TRANSFER AND INCREMENT RH BY 1	33 (E = 1)	IOX	2-33
TRANSFER AND INCREMENT RH BY 2	32 (E = 1)	ITX	2-32
TRANSFER IF RH = 0	40	XEZ	2-34
TRANSFER UNCONDITIONAL	34	XFR	2-33
TRANSFER UNCONDITIONAL AND STORE LINK	35	XLK	2-33
TRANSFER IF RH IS NEGATIVE	42	XNG	2-35
TRANSFER IF RH ≠ 0	41	XNZ	2-35
TRANSFER IF RH IS POSITIVE	43	XPS	2-35
TRANSFER ON CONSOLE TRANSFER SWITCH	36	XSW	2-34

JUMP INSTRUCTIONS

INSTRUCTION	FUNCTION CODE	MNEMONIC CODE	PAGE NUMBER
COMPARE, JUMP IF EQUAL	51	CJE	2-37
COMPARE, JUMP IF GREATER	53	CJG	2-38
COMPARE, JUMP IF LESS	50	CJL	2-36
COMPARE, JUMP IF UNEQUAL	52	CJU	2-37
GATED COMPARISON, JUMP IF INSIDE	54	GCI	2-38
GATED COMPARISON, JUMP IF OUTSIDE	55	GCO	2-38
JUMP THREE WAY	37	JTW	2-36
SKIP IF BIT IS ONE	66 (OR 67) *	SKN	2-40
SKIP IF BIT IS ZERO	64 (OR 65) *	SKZ	2-39

MISCELLANEOUS INSTRUCTIONS

INSTRUCTION	FUNCTION CODE	MNEMONIC CODE	PAGE NUMBER
CLEAR BIT	62 (OR 63)	CLR	2-53
EXECUTE	01	EXE	2-50
HALT	00	HLT[4]	2-54
MEMORY BANK ASSIGNMENT	27	MBA	2-51
MEMORY BANK DESIGNATOR	23	MBD	2-51
NO OPERATION	77	NOP	2-49
SET BIT	60 (OR 61)	SET	2-52
STORE ALL ZEROS	72	STZ	2-54