ALL APPLICATIONS
DIGITAL COMPUTER

DATA PROCESSING
ELEMENT
(DPE)

PROGRAMMER'S
REFERENCE
MANUAL

BR-8184

30 SEPTEMBER 1974

RAYTHEON

UNCLASSIFIED

ALL APPLICATIONS DIGITAL COMPUTER (AADC)
DATA PROCESSING ELEMENT (DPE)
PROGRAMMERS REFERENCE MANUAL

BR-8184

30 September 1974

Prepared for

Naval Air Development Center
Warminster, Pennsylvania 18974

PREPARED BY

RAYTHEON COMPANY
MISSILE SYSTEMS DIVISION
BEDFORD, MASSACHUSETTS

BLANK

TABLE OF CONTENTS

PAGE

PAGE

PAGE

PAGE

PAGE

PAGE

PAGE

PAGE

LIST OF ILLUSTRATIONS

PREFACE

This reference manual is intended to satisfy the contractual requirements for Contract N62269-73-C-0660 for the Naval Air Development Center, Warminster, Pennsylvania. It is expected that the material contained in this document will become a substantial part of a future document to be entitled "Principles of Operation."

This manual is the reference manual for the Data Processing Element (DPE) of the All Application Digital Computer System (AADC). It provides a comprehensive description of the system and of the DPE instruction set. Because the DPE is comprised of two computational elements, the PMU and AP, that have separate instruction sets, a prefix, A-, will be used with the op code number of an AP instruction for purposes of differentiation within this manual. This manual is intended to be self-teaching for a reader that has a basic knowledge of data processing systems.

1.0 INTRODUCTION

The AADC System employs an advanced, powerful general purpose digital system concept. Its concept embraces the computer applications spectrum from the mini-computer at one end to the powerful multiprocessor system at the other end. To accomplish this, the AADC defines modular elements which may be combined to whatever complexity is required by a given application. The modularity is not confined to the assembly of Processing Elements, Memories and I/O Controllers. The Data Processing Element (DPE) itself is separable into two components. One component is a stand alone mini-computer (Program Management Unit) while the other is a powerful Arithmetic Unit.

Salient features of the DPE are:

- Data Insensitive Arithmetic Structure
- Stack oriented accumulator structure to directly execute expressions in infix notation (parenthetical control)
- Implementation of APL primitives
- Virtual Addressing Support
- Pipeline Architecture
- Debugging and Performance Monitoring
- Executive and Security Support
- Efficient Interprocessor Communications

BLANK

2.0 SYSTEM ARCHITECTURE

2.1 System Components

The block diagram (Figure 1) of an AADC configuration shows the system concept and the manner in which the elements interrelate. It should be noted that this is not the only configuration of an AADC.

There are three types of memories available in an AADC, the Block Oriented Random Access Memory (BORAM) which is used for storage of procedure, the Random Access Main Memory (RAMM) which is used for the storage of variables, and the Task Memory (TM) which is used as the local DPE memory.

There are two types of interface units available, the channel which is an AADC internal interface unit, and the Data Communicator Module (DCM) which is an external interface unit.

There are two additional AADC components. One component is a stand alone mini-computer (Program Management Unit) and the other is a powerful Arithmetic Unit (AP).

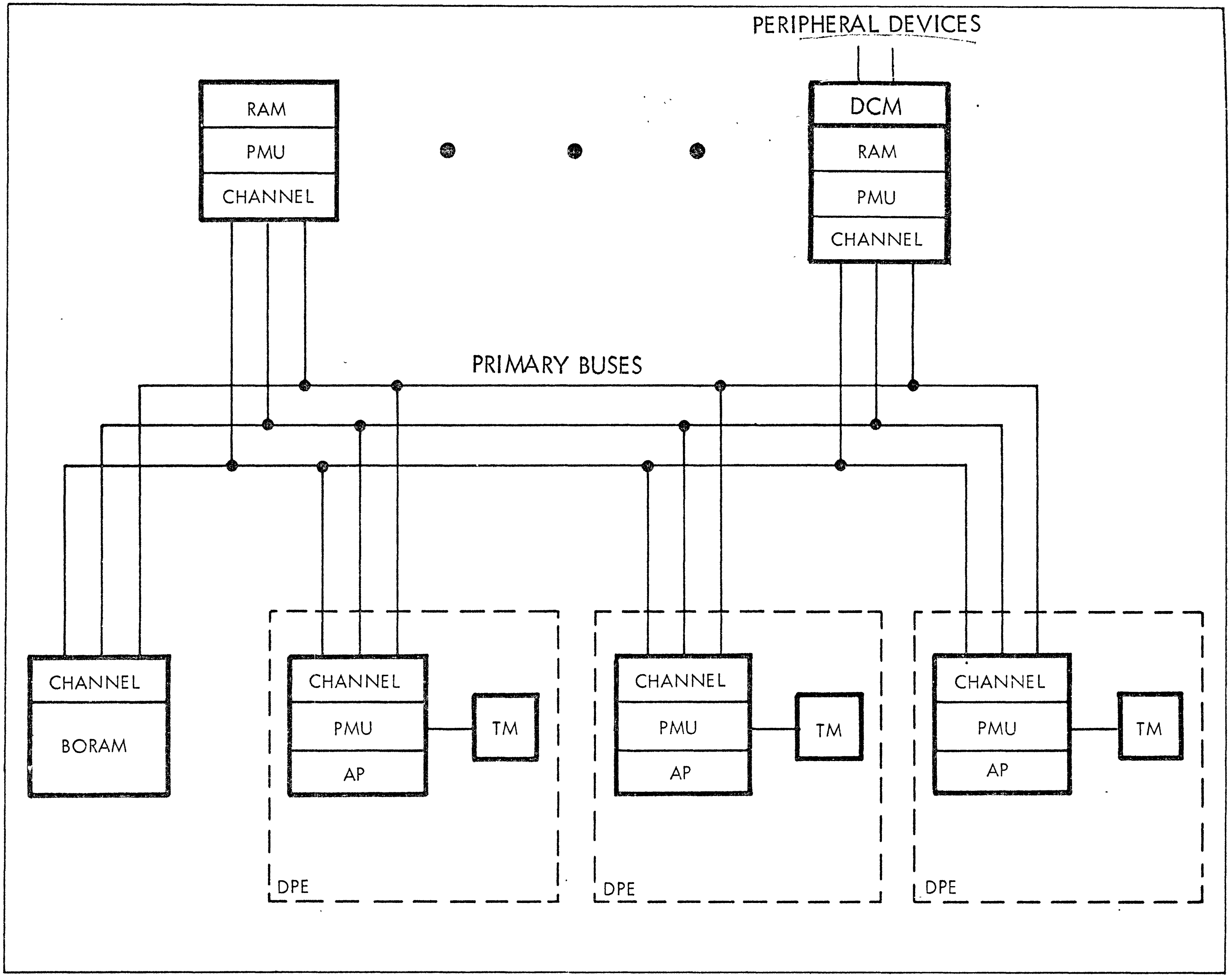


FIGURE 1. AADC SYSTEM CONFIGURATION

1) BORAM (Block Oriented Random Access Memory) - Stores procedure and constants for all programs in pages of 256 words each. Each program segment (or task) may require a number of pages of storage which may or may not be consecutively located in memory. During execution of a particular task, pages will be transferred to the TM of the executing DPE as needed on a demand basis. Transfers are via the primary bus at an uninterrupted rate of 150 nanoseconds per word. Access to the first word of a page is a maximum of two microseconds. In tactical operational use, this memory will function in the read only mode. The write mode will be provided for non-tactical or non-critical applications.

2) RAMM (Random Access Main Memory) - Constitutes the main data storage for the system. The RAMM typically consists of a series of modules, each 8 or 16K words (K=1024), 36 bits per word. Each RAMM has an associated PMU and channel unit which interfaces it with the remainder of the AADC subsystems. Data contained in RAMM can be addressed by any DPE on a single word or multi-word basis. The DPE can use this data directly, or hold it for later use in the TM. Data access time is 250 nsec though transfers are made over the primary bus at 150 nsec per word rate, and access time per word for blocks of data is 150 nsec.

3) TM (Task Memory) - Is a local direct access memory to the mini-computer (PMU) and is considered part of the DPE. The TM typically has 4K of 36 bit words, but could be as large as 64K words. Data transfers are made at 150 nsec per word rate.

4) Channel Unit - Is the common interface between elements of the AADC and the primary bus. It performs the queueing and transfer of information, and coordinates all internal bus transfers.

5) DCM (Data Communicator Module) - Is the interface unit to external devices. It contains serial to parallel converters, and packing and unpacking circuitry. It acts through the channel in the DPE configuration for communications with the computer.

6) DPE (Data Processing Element) - Is a general purpose, programmable processor capable of performing logic and arithmetic operations necessary for handling sequentially organized tasks. The DPE contains a PMU (Program Management Unit), an AP (Arithmetic Processor), a channel and a TM (Task Memory). Procedure pages of program segments or tasks are stored in the TM for execution. Part of the TM is also used for temporary data storage. Control functions of the DPE, including normal instruction and operand fetching, executions of program management type instructions, and interfacing with the other elements via the channel and the primary bus, are handled by the PMU. Arithmetic and logical computations are performed in the AP. This separation of computational functions and control functions permits a highly parallel operation of the two subsystems. Additionally, the PMU is used as a front end memory controller for the RAMM.

2.2 Data Processing Element Components

As indicated earlier, the DPE is a powerful general purpose digital computer which has the property that its sophisticated arithmetic logic may be separated from the computer proper leaving a simple mini-computer. Some factors which highlight the advantages of the DPE over other powerful data processors are:

1) The DPE handles fixed point, floating point and complex arithmetic automatically.

2) The DPE operates on arrays and matrices automatically.

3) The DPE solves algebraic expressions automatically through its ability to interpret parenthetical notation.

The block diagram in Figure 2 shows the DPE internal structure. The DPE contains four major components. They are the arithmetic processor (AP), the Program Management Unit (PMU), the Task Memory (TM) and the Channel.

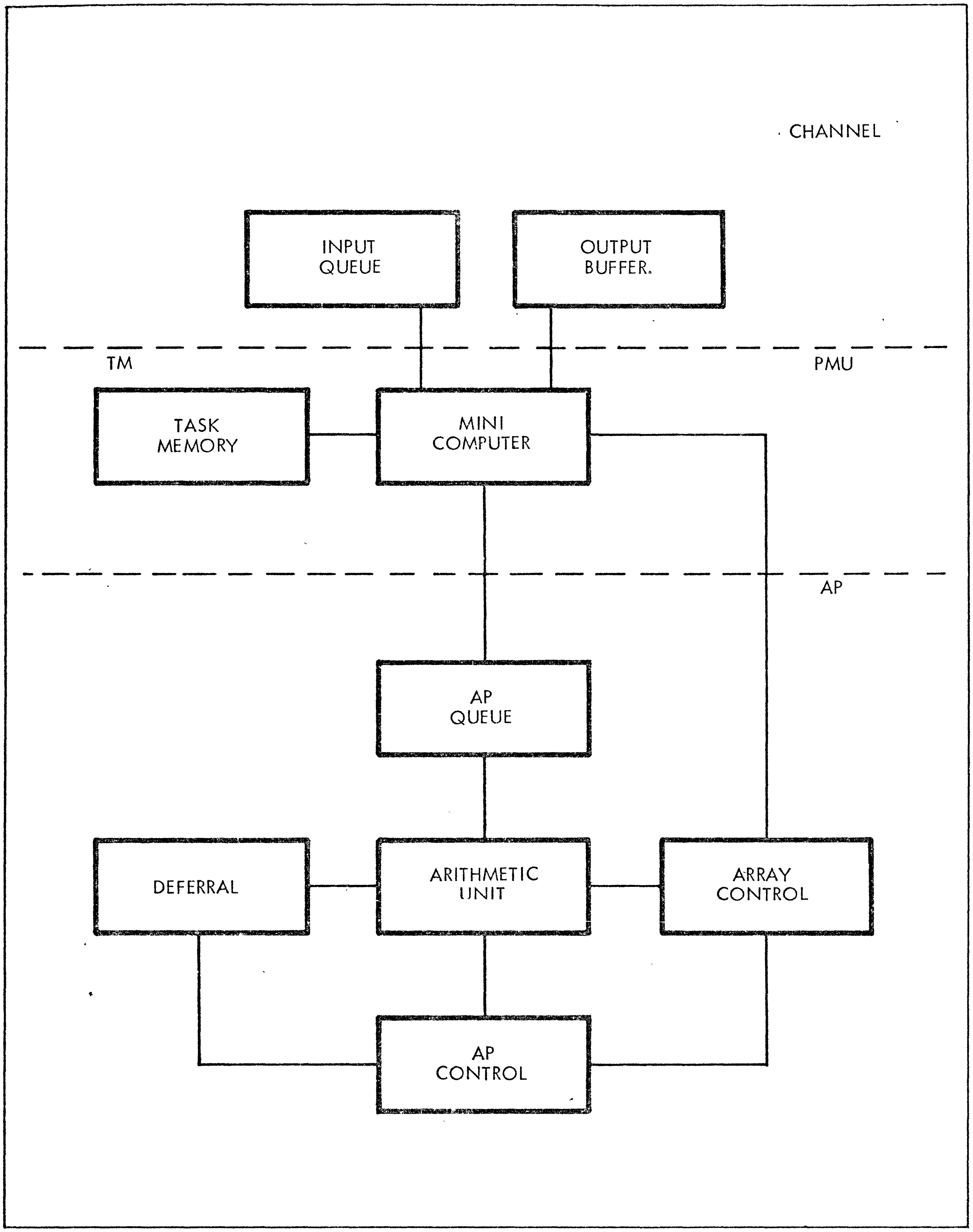


FIGURE 2. DATA PROCESSING ELEMENT

The AP

The AP or Arithmetic Processor, is a data insensitive execution unit which performs arithmetic operations requiring full word precision at high speed. These instructions include Add, Subtract, Multiply, Divide, and Square Root. All operations are performed in floating point. The AP responds to inputs of two kinds. The first type involves receipt of an operation code and operand on which the AP acts according to its instruction set. A second is used in array operations and places the PMU under control of the AP Array Controller, addressing operands and operating as instructed by the AP. Additionally, the AP may interrupt the PMU during scalar-real instruction sequencing for servicing of store conditional transfer instructions, and exception conditions (e.g., exponent overflow).

To implement parenthetical notation, the AP contains a last-in/first-out (LIFO) deferral unit in which information is held for later operation. The operation of this deferral stack will be discussed in section 6.3.9 "Parenthetical Control."

To permit the PMU and AP to run asynchronously in normal operations, a queue (AP Queue) is placed between them. This Queue is a simultaneous read/write scratchpad element that operates on a first-in/first-out basis. It is wide enough to contain an operation code, an operand, and certain necessary control and sequencing information. The queue length visible to the programmer is 13 words in length.

In normal operation, the PMU fetches an instruction (from TM) and an associated operand (from RAMM), and places the necessary data into the Queue. At the other side of the

Queue, the AP sequentially removes these instructions, and executes them. The AP can perform some instructions, such as ADD, faster than the PMU can complete a full fetch; certain other instructions, such as MULTIPLY, take longer than the fetch takes. Use of a Queue tends to average out these differences by permitting the PMU to stockpile instructions during periods when the AP is executing long instructions against the time when a series of short instructions will begin to deplete the queue.

The PMU

The PMU is a mini-computer in that it has its own arithmetic logic and instructions, and may in some applications be a stand alone machine. In the DPE, the PMU acts as the controlling subsystem. It fetches all instructions, performs address translation when necessary and obtains an appropriate operand. It executes its own instruction set and prepares others for transmission to the AP. It formats data requests to outside devices if required and controls the TM. During array processing, the PMU is controlled by the AP.

The TM

Task Memory is a small high speed memory which forms part of the DPE. This memory contains 4096 words of 36 bit length and provides the program which is executed by the DPE. The TM is subdivided into 16 pages of 256 words each. All programs are executed in modules of up to 256 words each.

As program pages are executed and new pages are required, they are brought in as needed and written over current pages. The decision as to which of the current pages are to be written over is decided in accordance with a number of algorithms which are mechanized by the PMU hardware.

The Channel

The Channel is the common interface between elements of the AADC and the primary bus. It performs the queueing and transfer of information, and coordinates the primary bus transfers. The primary buses are bi-directional buses which handle all data and control information transfers between the various computer system elements on a 50 bit parallel basis. Each word is transferred during a 150 nanosecond time slot. The primary bus control scheme assigns non-dedicated time slots on a rotational priority basis. Each channel in the system contains a circuit which, when tied in a closed chain, provides the rotational priority control. A channel desiring the bus, raises an internal demand line and waits notification of bus assignment. Channels having no demand are skipped so all time slots can be used. No element, except when specifically programmed to do so (i.e., BORAM), is permitted to hold on to the bus for more than two time slots. The BORAM locks the bus for one whole page transfer (256 words). Three buses are used to allow for efficient communication between channels as well as to provide redundancy. Two buses are programmatically identical while the third bus is not normally attached to a

DPE channel. This third bus is dedicated to other system functions. Any element desiring to communicate with another can use either bus when free. To make most efficient use of the primary buses, each channel contains an input queue on its input interface with the bus. This makes the destination channel effectively always available to receive data for it on the bus.

3.0 DATA FLOW

The internal flow of data and procedures within an AADC is between the DPE, primary buses and other components. Each of these facets of the system can be explained somewhat independently, but will not be clear outside of the context of a complete system description. The following sections describe the operation and data flow between the primary bus, DPE, and other AADC system components.

3.1 Primary Bus

The system employs a time slotted bus. The time slots are non-dedicated. A distributed equal priority bus controller is used. The system is provided with automatic error retry. In order to allow this system to operate efficiently, the channels in the system are designed with a wide address and are almost always available to receive information.

Bus usage is assigned by the bus controller to a channel. This channel puts its information on the bus for a fixed period of time and then releases it unless the bus has been re-assigned to the same channel. A synchronizing clock is provided which is common to all channels. The time slots are said to be non-dedicated because they are assigned to channels which have asked to send a word, as the requests occur, rather than in a fixed sequence.

The bus controller decides which channel will use a certain time slot during the preceeding one, so that no time is lost for this determination. The assignment is said to be equal priority because no channel has easier access to the bus than any other. Some channels may send more words in consecutive slots than others according to fixed rules, but do not have any greater ability to obtain the bus.

The controller is distributed among the channels in a daisy chain fashion. That is, each channel receives a signal representing status of channels physically before it in the chain. According to preset logic, this signal is passed on to remaining channels. Other information on the bus is in a party line form. That is, the same wires are used for input to and output from every channel in the system. This type of bus is bi-directional.

Each channel which is capable of receiving information is supplied with an input queue buffer which is eight instruction words in length.

This means that all information sent to a channel will almost always be accepted by the receiver. In fact, the receiver can be executing an instruction while receiving and accepting a group of others which will wait their turn to be processed. The receiving channel is said to be essentially "always available". Sending channels will ordinarily await an indication of its instruction's being processed before attempting to continue sending to the same channel.

In order to ensure correctness of receipt of the information, parity checking is performed by the receiver on received words, but in order to avoid overly long time slots, this checking is done during the next time slot after receipt. The error, if one occurred, is signaled in the second time slot after receipt on a separate cable. Correct receipt is acknowledged, to account for the absence of the indicated receiving unit. That is, correct receipt is positively signaled. Furthermore, irregular, but possible conditions such as receiver input queue full can be signaled.

When an error or receiver busy is signalled, the sender, which is constrained to maintain sent information in its Output Queue, retries the transfer. In order to maintain proper sequencing of messages, a receiving channel once it has rejected a word due to parity error or busy condition, it will reject with the 'busy' code all the words that may be received by it in the two time slots following the arrival of the first rejected word.

When the first rejected word is retried, if a second parity error is detected by the receiving channel, an emergency Transfer and Stack Kernel 0 (Op Code 54) instruction addressed to the Executive will be generated by the receiving channel and sent through its Output Queue to the Internal Bus.

A transmitting channel receiving a parity error Status Return for the second time for the same word will:

- send an emergency Transfer and Stack Kernel 0 (Op Code 54) instruction addressed to the Executive, passing through its Output Queue to the Primary Bus.

- Keep retrying the transmission of the rejected word until either the word is accepted or the channel receives an emergency reset command from the Executive.

In the system described, a channel wishing to send information raises an internal control line indicating this and signalling its portion of the bus controller. Some time later, this channel is granted the bus. It sends its information during the indicated time slot, then releases the bus if required.

A channel maintains three address counters associated with its output queue.

1) the next queue location to be filled
2) the next queue location to be transmitted
3) the queue location after the last validated transmission

When an error or busy signal is received, no further transmission is permitted. If the channel continues to possess the bus, it transmits the No Transmission Code.

The channel may now obtain the bus for purposes of error or busy retry. Retry is made for only one word at a time. A second parity error Status Return received is signalled to the executive. Busy signals repeat the retry until accepted. When the queue validation counter equals the next queue location to be transmitted counter, normal continuous operation may be resumed. The Error Retry bit is set if the nature of the error was parity error.

If several words were sent with one having an error, only the incorrect or busy words need to be retried. If one word of a two word instruction which was sent is signalled as being an error, both words must be retried. Thus, the first word of a two word instruction is only considered as sent and validated when the second word of the instruction has also been received and validated.

Channels are provided as three bus units or as two bus units with extension capability to three buses.

The bus system described can run continuously with a transfer taking place in each time slot and will sometimes only degrade temporarily, when errors occur.

A further enhancement to the bus system is the AADC wide address. In this system, instructions placed on the bus are steered by their address field to the proper receiving element. The source element identifies itself using the source field provided as part of each transmission. Thus, any command request to an element may be placed on the bus with a destination address, and be certain that the element will pick up the request and process it. The element addressed can be anything from a main memory to a simple peripheral. The word addressed can be a register, a bank of switches or a word in a memory.

As an additional possibility, some elements may be designed so that they can be instructed to request channel transmission on only some of the available buses, so that an element can be given a bus continuously or so that a bad bus can be removed from the system, allowing further error free operation with reduced total throughput.

The word formats for the two word types, command and data, which are transmitted on the bus are illustrated in Figure 3 and are discussed in the following paragraphs.

3.1.1 Transmission Types (Commands and Data) (Bits 37-39)

The channels use the bus to transmit both commands and data to other subsystems. Presently, the bus system can specify the following types of transmission through 3 coded lines.

Bit Pattern

0 No Transmission 000

There is no valid transmission on the bus. Receivers are to ignore the remainder of the bus content. The transmitting device possesses the bus, but did not want it.

1 Continue Interrupted Process 001

This is a special transmission code which will be discussed elsewhere in detail. The command normally causes immediate activation of the receiving device to initiate an operation previously suspended.

2 Data 010

The bus contains data. The receiving subsystem must be expecting this data. It is placed in the data queue.

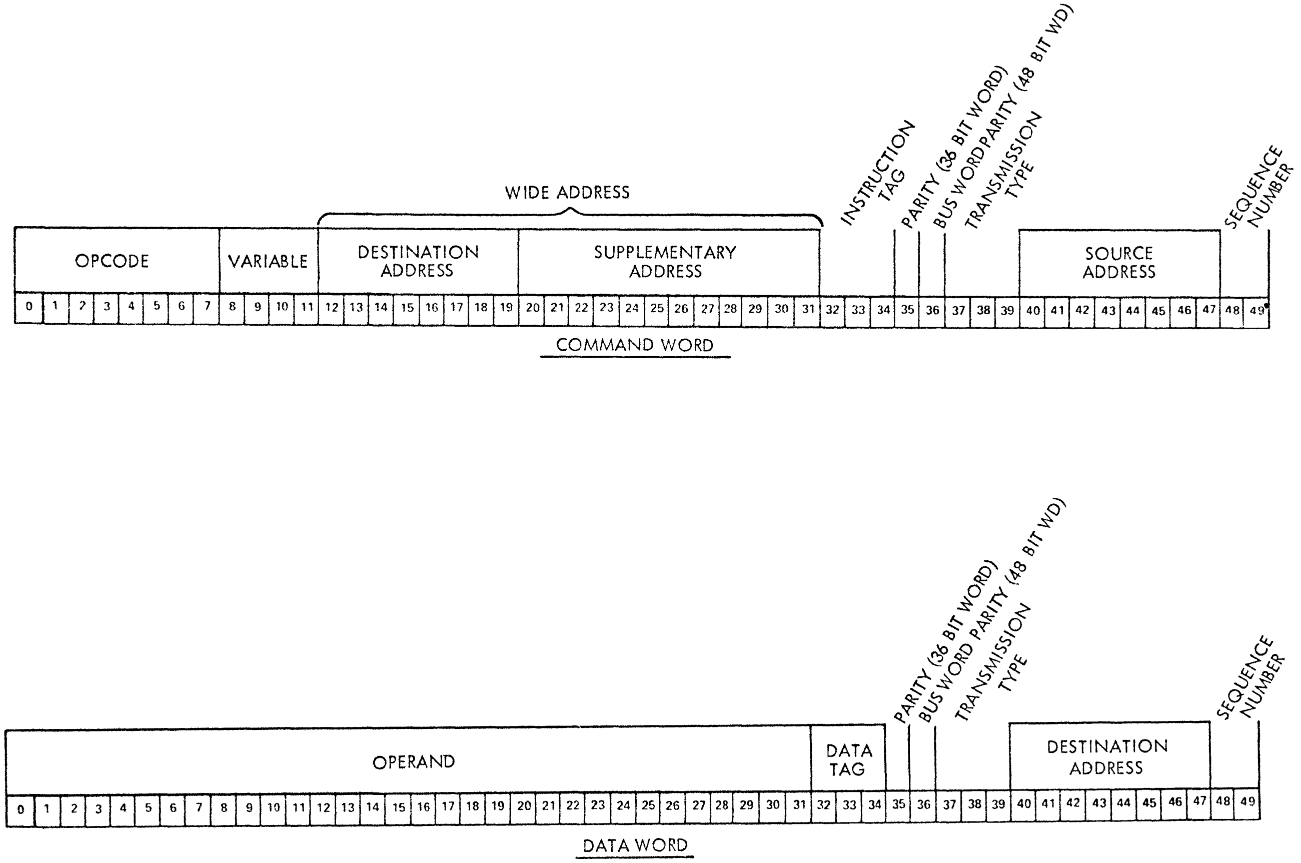


Figure 3. Bus Word Format

Bit Pattern

3 Data and End of Block 011

Same as 2, with the additional information that is the last word of a variable block transfer. This code is generated by a transmitting subsystem under defined conditions.

4 Single Word Command 100

This indicates an instruction interrupt is on the bus.

5 Two Word Command 101

This code identifies the next word on the bus as being destined for the same device as this one. All other devices must unconditionally refuse the next word. Similarly, this device must unconditionally accept the next word and place it in the input instruction queue.

6 Single Word Emergency Command 110

This code identifies that this command is to be the next to be executed. Normal processing is interrupted and will not necessarily continue correctly.

7 Two Word Emergency 111

This code identifies that this command is to be the next to be executed. As with 5, the next word should be accepted as well. As with 6, normal processing is interrupted.

3.1.2 Sequence Number Field (Bits 48 and 49)

This 2-bit field can be set at 00, 01 or 10 and must be returned with the data requested by the source element. The source channel transmits this number any time its controlling element expects a data reply. When the source element does not expect a reply, as in the case of a single word write command, the channel transmits the currently valid transmission number. This means that a channel can issue up to 3 commands with data returns pending, and properly sequence the returns. If the returned data has the correct transmission number, the data is accepted. If the data has the correct number and is also EOB, then the expected sequence number register contained in the channel may be moved up to the next expected sequence number. Any data received in the incorrect sequence will be rejected by the channel with a busy signal. The fourth combination of the sequence number bits (11) is used only by elements placing commands in the channel output queue to indicate to the channel that the element is expecting a reply. The current sequence number incremented by one is to be used as the sequence number when this command is sent out on the AADC Internal Bus.

3.1.3 Data/Instruction Field (Bits 0-35)

The format of the 36 bits correspond to the Data or Instruction formats of the DPE.

3.1.4 Source/Destination Field (Bits 40-47)

Elements are specified by an 8 bit number. The 8 bit number is identical to the high order 8 bits of a standard AADC 20 bit wide address. When commands are sent to an element, the address information contained within the 36 bit instruction word contains the Destination information to specify the Receiving element. The element address is specified by bits 12-19 in the command word.

Since commands require an eventual return of information to the originating element, the 8 bit field associated with a transmission command specifies the Source Subsystem in bits 40-47.

Data is transmitted only to an element which requested the data. Elements which request data from other elements must be ready to receive the returned data when it is provided. When a bus contains data, the associated 8 bit field contains the destination information (bits 40-47).

Each channel has two names, physical and logical. The physical name is wired in by the channels physical position in the AADC housing. Upon system reset, the logical name is made to be the physical name. The AADC Executive can, via the Set System Parameter (Op Code 25) instruction, determine the logical name of the channel. Independent of the logical name, all Emergency Commands are sent to the physical name of the channel.

One channel code has been reserved for specifying the executive system wherever it resides in the system. This is code FF (hexadecimal notation).

3.2 Channel

The channel control must perform sequencing of queue addressing output buffer selection, as well as decisions about the function to be performed as a result of information sent on the bus.

The channel unit contains four major parts which are the bus itself including the drivers, a bus controller circuit, the input queue, and an output queue. The bus and bus controller were discussed in Section 3.1. The Input and Output Queues will be discussed in the following sections.

3.2.1 Input Queue

The input queue is divided into two parts conceptually. These are Command and Data. The physical component is 16 words in length, with commands using 8 words and data using 8 words.

Information is stored into these queues in a first-in/first-out fashion. Of the information on the bus only the retried bit is stripped off. All other information is placed into the queue.

The queue will be considered empty when the next word to be read equals the next word to be written and the last operation was a read from the queue. The queue will be considered full when the next word to be read equals the next word to be written and the last operation was a write into the queue.

The queue full signal is used to refuse further information from the internal buses.

The routing of data and command is a function of the transmission field. In the case of the second word of a two word command, the information is always placed in the command queue.

Two deviations from this are the no transfer and continue bus tags. No transfer indicates that the device which has the bus has nothing to send. This word is ignored by the receiving channel unit.

Continue,. when received, should be used by the element to format as a data word and is placed in the data queue.

An Emergency command is allowed to interfere with any operation in progress and will be passed through after validation.

Emergency commands bypass all commands presently waiting in the input queue, appearing immediately at the front of the queue.

As soon as the words received by the channel are validated and placed in the Input Queue attempts are made to send them out to the DPE one after another.

3.2.2 Output Queue

Before sending the information out on the AADC Internal Bus, the channel unit first places all command words and data words received from the DPE into the Output Queue. Up to 16 words mixed, command and/or data, can be entered into the queue in the sequence of arrival.

The main purpose of the Output Queue is to provide a variable depth word buffer between the DPE, capable of sending a continuous stream of data words during burst mode, and the AADC Primary Bus with its automatic error retry requirement.

4.0 PROGRAM MANAGEMENT UNIT (PMU)

The PMU is a digital data processor with a 36 bit word length. It is capable of performing operations using half-word (16 bit), and full word (32 bit) operands. It performs arithmetic operations using sign magnitude integer arithmetic. The most significant bit of an operand is the sign bit: 0 means positive, 1 means negative.

4.1 PMU Registers

4.1.1 Addressable Registers

The registers in the following subparagraphs are addressable for the specified PMU operations.

4 1.1.1 General Scratchpad Registers

The PMU has one general scratchpad register set, consisting of 32 16-bit registers designated SP [0] through SP [31] . Each SP register is capable of holding a half-word operand for arithmetic, shift and logical functions, and as temporary storage. SP [0] - SP [15] are directly addressable by designators in the instruction words. SP [1] through SP [7] may also be used as index registers, addressable by another set of designators in the instruction words.

For half-word operations SP [0] - SP [15] are each selectable as a 16-bit accumulator. For those operations in which a full word accumulator is required, a 32 bit register is made available by concatenating one register selected from SP [0] - SP [15] with the corresponding register selected from SP [16] - SP [31] , respectively. The low bank of 16 registers will contain the most significant portion of the operand while the high bank, SP [16] - SP [31] , will contain the least significant portion of the operand.

Within the general scratchpad set, all are available as pointers and operand registers. However, the following registers are dedicated to automatic functions in the DPE configuration and any programmatic contents may be destroyed.

SP [15]	Interrupt Stack Pointer
SP [16]	Used for Array Control in DPE Configurations
SP [17]	
SP [18]	
SP [19]	
SP [20]	
SP [21]	
SP [22]	
SP [23]	
SP [24]	
SP [25]	
SP [26]	Array Stack Deferral Pointer
SP [27]	Deferral Overflow/Underflow Stack Pointer
SP [28]	Address of Indirect Dimension Word
SP [29]	P Counter Temporary Storage
SP [30]	Internal Registers Temporary Storage
SP [31]	Parameter Stack Linkage

4.1.1.2 Program Address Register

The PMU has a 16 bit program address register which is designated P. The contents of P specify the address of the next instruction. The computer increments by one the contents of P for each instruction. Instructions which cause program jumps enter P with the address of the instruction to which program control is transferred. When the operations specified by the current instruction are completed, the contents of P are then used to obtain the next instruction.

4.1.1.3 Interval Timer Register

The PMU has a 16 bit Interval Timer Register. When enabled, the contents of the Interval Timer Register decrement at an interval of 307.2 microseconds. The Interval Timer Trap Signal is generated when the contents of the Interval Timer Register equals zero. The Interval Timer Register is loaded and the decrementing sequence is enabled under program control. The total elapsed time capacity for the Interval Timer Register is approximately 20.1314 seconds.

4.1.1.4 Source Registers

The PMU contains two source registers. The Program Source register (P-source) contains the 8 bit code of the external system element that requested the presently running program. The Interrupt Source register (I-source) contains the 8 bit code of the external system element that supplied the most recent external instruction. If the PMU is processing an external instruction, the I-source register is defined as containing the active source for that instruction. If the PMU is processing an internal instruction, the P-source register holds the active source.

4.1.1.5 Halt (HALT) Indicator

When the Halt Indicator is set (HALT = 1), the PMU will not perform program operations, but will only respond to externally generated instructions or internal traps. The Halt Indicator is set when bit 11 in either an Interval Timer Control/Halt instruction (Op Code 32) or a Transfer to Executive instruction (Op Code 37) is set and the instruction is executed, or the indicator is set by the Reset line. The Halt Indicator is reset (HALT = 0) by the execution of a Proceed instruction (Op Code 01), or a Return Stack to P and Proceed instruction (Op Code 53), or a Transfer and Stack instruction (Op Code 51).

4.1.1.6 Trap Level Register

The PMU contains a five bit register which maintains the trap priority level of the presently running program (see section 6.0).

4.1.1.7 Parity Error Inhibit (PEI) Indicator

When this indicator is set (PEI = 1), the parity error trap signal will not be generated and no response will be made to a parity error. This indicator is set or reset by bit 9 of the Set System Parameter instruction (Op Code 25).

4.1.1.8 Procedure Kernel Register

This two-bit register indicates which of the four lowest page areas of memory that the current program is referenced in.

4.1.1.9 Data Kernel Register

This two-bit register indicates which of the four lowest page areas of memory that is referenced for the data being used for current program.

4.1.1.10 Upper Bound Register

This four-bit register is used to designate the highest page area, within the 16 least significant page areas of local memory which are subject to automatic page replacement (see Section 4.3.3.2). This register is set with bits 8-11 of the Set Task Parameters instruction (Op Code 29).

4.1.1.11 Lower Bound Register

This two-bit register is used to designate the highest page area, within the four page areas normally reserved for kernel information, which are not subject to automatic page replacement (see Section 4.3.3.2). This register is set with the least significant bits (bits 14-15) of the Lower Bound field (bits 12-15) of the Set Task Parameters instruction (Op Code 29).

4.1.1.12 Executive Mode Indicator

This indicator, when set (= 1), designates that the PMU is in the executive mode and capable of performing all privileged functions. When this indicator is clear (= 0), the PMU is not in the executive mode. (See Section 4.2.2.)

4.1.1.13 Data Addressing Mode Indicator

This indicator, when set (= 1), designates that the data is to be accessed virtually, through references located in the Data Kernel. When clear (= 0), this indicator designates that the data is to be accessed directly from local memory. This indicator is set or reset by bit 23 of the Set Task Parameters instruction (Op Code 29).

4.1.1.14 Replacement Algorithm Register

This two-bit register contains the code for the presently operational replacement algorithm (see Section 4.3.3.2). This register is set with the contents of bits 18-19 of the Set Task Parameters instruction (Op Code 29).

4.1.2 Non-Addressable Registers

The register in the following subparagraph is not directly addressable by the programmer, but is included here for definition. It is referenced later in the manual in the discussion of virtual addressing.

4.1.2.1 Procedure Page Register

This non-addressable eight-bit register contains the direct local memory address of the page of procedure currently being executed. The contents of this register are appended to the eight least significant bit of the program counter P, the displacement field, to obtain the direct local memory address of the next instruction to be executed. This register is set with the contents of bits 0-7 of the procedure kernel word during a procedure page fetch. (See Section 4.3.3.2.)

4.2 Modes of Operation

A DPE can operate in either one of two modes, Problem or Executive.

4.2.1 Problem Mode

In this mode of operation, the Executive Mode Indicator is cleared to zero. This is the normal mode for interpreting procedure. All security protect mechanisms involved with virtual addressing are invoked. (See Section 4.2.2.2.2).

4.2.2 Executive Mode

In this mode of operation, Executive Mode Indicator set to one, a DPE possesses capabilities that would otherwise be illegal when there is an attempt to use them. The following description is in three parts.

- Entrance to Executive Mode
- Executive Mode Capabilities
- Exit from Executive Mode

4.2.2.1 Entrance into Executive Mode

There is only one way to enter the Executive Mode, the receipt of an external Transfer and Stack Kernel 0 instruction (Op Code 54) addressed to the channel which recognizes the resource name hex "FF." The Transfer and Stack Kernel 0 instruction in this structure is a two word bus command. The first word, the instruction itself, invokes the Executive at a fixed entry point, a transfer to the instruction located at the virtual address hex "FF00." The Executive, once invoked, must obtain the interrupt mask data word from its channel by executing a Write Word From Input instruction (Op Code 03). This mask word contains the necessary information for the Executive to determine what actions to perform. (Note: Further interrupts are inhibited until the interrupt mask word is fetched.)

The two word Transfer and Stack Kernel 0 instruction is transmitted to the Executive in the normal course of computation by one of two possible mechanisms: programmed or channel interrupt.

4.2.2.1.1 <u>Programmed</u>

When a DPE wishes to interrupt the Executive, it executes a Transfer to Executive instruction (Op Code 37). This instruction causes the creation of the two word Transfer and Stack Kernal 0 command. The first word of the Transfer and Stack Kernel 0 instruction has been described in Section 4.2.2.1. The second word has the following format:

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

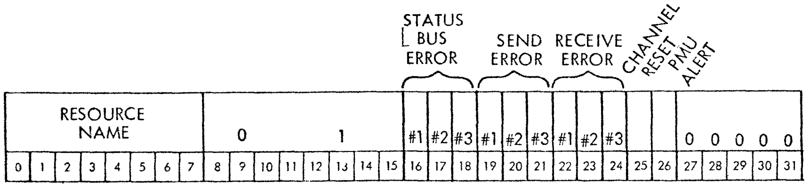
Bits 0-7 contain the resource name of the resource performing the Transfer to Executive instruction. The contents of this field are derived from the Resource Name field (bits 16-23) of the Set System Parameter instruction (Op Code 25).

Bits 8-15 are cleared to zeroes, which indicates that this interrupt was due to the execution of a Transfer to Executive instruction.

Bits 16-31, the interrupt mask, are the effective address field of the Transfer to Executive instruction (normally instruction bits 16-31 as an immediate). The meaning of the interrupt mask will be established by system software conventions.

4.2.2.1.2 Channel Interrupt

The channel, in response to certain conditions that are described below, sends an interrupt to the executive. The channel creates a two word Transfer and Stack Kernel 0 command. The first word has the same format as an interrupt generated as a result of the execution of a Transfer to Executive instruction. The second word has the following format.



Bits 16-26 signify the occurrence of various interrupt conditions that the channel has detected. These conditions are independent and more than one may occur simultaneously. It should be noted that once a condition is detected by the channel and transmitted to the executive, subsequent occurrences of the same condition will not cause another executive interrupt unless the channel received a Set System Parameter instruction (Op Code 25) in the interim. (See Appendix I).

Bits 8-15 signify the interrupt originates from the channel. Bits 0-7 are the resource name for the originating channel.

It should be noted, nothing prevents an executive from sending an interrupt to itself.

4.2.2.2 Executive Mode Capabilities

When, a DPE is in the executive mode of operation, it possesses capabilities not present in the problem mode. These capabilities involve two areas: the security system and emergency commands.

4.2.2.2.1 Emergency Commands

When a DPE is in the executive mode, two word emergency commands can be transmitted to an AADC resource via the Command Subsystem instructions (Op Codes 47 and 4F). If bit 8 of the Command Subsystem instruction is set to one, a two word command is transmitted that has a transmission tag designating emergency commands. See Section 4.3.2 for a description of emergency commands.

4.2.2.2.2 Security System

When the DPE is in the executive mode, all security violations involving the protection mechanisms for the kernel area cannot occur, i.e., read, write and command protect. Security violations that result due to an improper data structure still occur in the executive mode. For example, transferring to word data or attempting to execute the load page instruction with word data.

The kernel protect violations do not occur in the executive mode. It should be noted that this is true whether the executive is attempting to access the kernel in its own task memory, or is attempting to access a kernel area in a remote resource.

4.2.2.3 Exit From Executive Mode

The executive exits from the executive mode by executing a Return Stack to P instruction (Op Code 52) or a Return Stack to P and Proceed instruction (Op Code 53). Upon completion of this instruction, the DPE is either in the problem mode or a previous level of the executive. It is not possible to return to the executive mode when a return stack instruction is executed in the problem mode. There is a hardware override which leaves the DPE in the problem mode.

4.3 Detailed Instruction Performance

Instructions for the DPE are of three types: external, emergency, and internal. External instructions and emergency commands are received by the DPE over the primary bus system. Internal instructions are obtained from local memory, either as the result of a trap or a normal program counter instruction fetch or they are obtained from the primary bus specifically as the word following a Command Subsystem/Address Modifications (Op Code 47).

4.3.1 External Instructions

Any pending external instruction is processed immediately at the conclusion of the instruction currently being processed. External instructions are received over the primary bus system in a 50-bit format (see Section 3.1). The external instruction is the 36-bit high order (bit positions 0-35) portion of the incoming 50-bit transmission word, and has the following format.

OP CODE								SPA				XXXX				ADDRESS																M	AP	X	X
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Bits 0-7 specify the operation to be performed.

Bits 8-11 specify a particular scratchpad register or specify a special purpose function depending upon the op code.

Bits 12-5 are unused.

Bits 16-31 specify the operand or the address of the operand.

Bit 32 define whether a memory reference is to be made or not.

Bit 33 define whether this is a PMU or an AP instruction.

Bit 34-35 are unused.

The above defined fields have the same meaning as for internal instructions,(see Section 4.3.4.1 for further definition of these fields).

Address modification (bits 12-15) and parity (bit 35) checking are not performed by the PMU for external instructions. The instruction trace (bit 34) trap also is not generated for any external instruction.

4.3.2 Emergency Commands

An "emergency command" is a mechanism to enable the executive processor (or any other permitted device, such as the control panel) to interrupt a channel and cause the "emergency command" to be the next thing transmitted via its secondary bus.

Identified by the Transmission Code "111" the emergency command is a two word instruction. Although any two-word instruction can theoretically be sent, the present AADC design contains a mechanism for producing the emergency transmission code only in conjunction with either of the COMMAND SUBSYSTEM instructions.

This, the first word received by the channel will be a routing word, and the second will specify the action to be performed.

As previously mentioned, the channel, when receiving an emergency command addressed to one of its secondary bus devices, transmits the word, with it emergency code, as its next transmission. This is the only type of command information that takes precedence over data, and the integrity of the program of the addressed processor cannot be maintained -- the emergency command will interrupt an array being read or, in the case of the DPE, a pipeline of data requests. Recovery from these interruptions will probably be impossible. Sending a HALT command prior to the emergency command will not necessarily be effective in alleviating this problem unless sufficient time is allowed to ensure that the HALT was accepted.

Emergency commands may also effect other subsystems connected to the same channel. When the channel places the first word on the secondary bus, it begins to count the clock pulses. If 4096 clock pulses pass and the receipt of the command has not been acknowledged, the channel raises its SYSTEM MASTER RESET line and resets all the subsystems on the secondary bus.

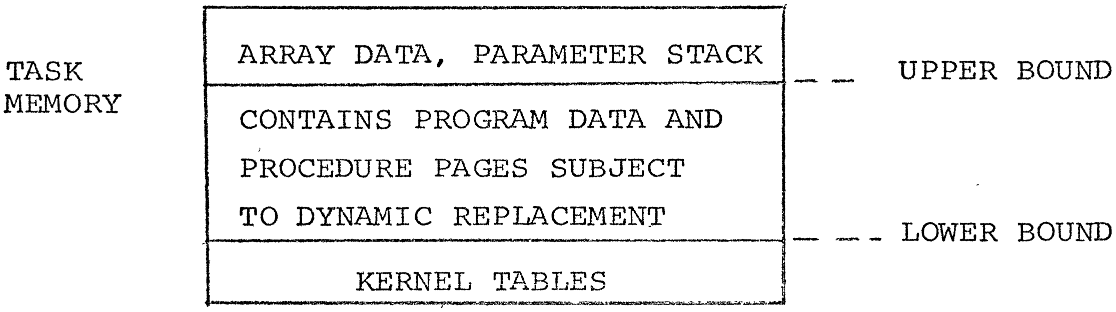
When the Channel is known to be reset, the first subsequent instruction must not be an emergency instruction. After receipt of this first instruction, emergency instructions can be properly received.

4.3.3. Internal Instructions

Internal instructions are obtained either in response to a trap, a Command Subsystem/Address Modification instruction (Op Code 47) or by a program counter reference of memory. Such a reference to memory will occur when no external instruction is pending, the HALT indicator is zero and no trap of higher priority than the present trap register value is pending. In such circumstances, the contents of P are used to obtain an instruction.

4.3.3.1 Virtual Addressing Mechanism

In normal operation, a processing element operates on a single segment of a program until completion. It then performs another. Each of these segments.is termed a program module. Associated with each program module is a "kernel area" which may contain a maximum of 4 kernel tables and which normally remain resident in TM throughout the running of the program module. This kernel typically contains, among other information, the direct BORAM addresses of the remaining pages of the program module. It may also contain RAMM data addresses, and control information necessary to run the program. Kernels associated with other program modules (as well as the Executive kernel, if required) may also be resident in task memory. The remainder of TM is used to hold currently running program pages, subject to dynamic replacement; and also is used to hold the parameter stack and array data required by the arithmetic processor.


TASK MEMORY
ARRAY DATA, PARAMETER STACK
UPPER BOUND
CONTAINS PROGRAM DATA AND PROCEDURE PAGES SUBJECT TO DYNAMIC REPLACEMENT
LOWER BOUND
KERNEL TABLES

The virtual addressing mode is the normal mode in the AADC system. In this mode, a fetch cycle is implemented to access procedure or word data from BORAM or RAMM respectively. via the Channel. When data is absolutely addressed, procedure is virtually addressed and all data is resident in Task Memory (kernel data entry not accessed).

The Virtual Addressing Mechanism's ultimate objective is to calculate the absolute address of the operand referenced by the effective virtual address of the instruction executed. The effective virtual address is used in this description to mean the 16 bit address obtained after indexing and all modes of indirection are completed.

4.3.3.2 Normal Instruction Fetch

If the present instruction does not specify a transfer or escape, and if a Page Carry condition (see Section 4.3.3.2.2) does not exist, and if a pending trap is not about to be honored, the PMU will obtain its next instruction in the following manner.

4.3.3.2.1 Program Counter Operation

The Program Counter P has the following format:

PAGE								DISPLACEMENT							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

To fetch an instruction, an address is formed by appending the Procedure Page Register (see Section 4.1.2.1) to the displacement field of P.

PROCEDURE PAGE REGISTER								DISPLACEMENT							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

This address is used as an absolute local memory address of the instruction to be fetched and executed. After this instruction is fetched, P is incremented by one to point to the next sequential instruction. If the instruction thus obtained does not specify a transfer or an escape and if the contents of the displacement field (bits 8-15) of P were hex FF, a Page Carry condition exists.

4.3.3.2.2 Page Carry

Upon completion of the fetch of an instruction, P is incremented by one. If a carry from bit 8 to bit 7 occurs during this process, a page carry exists. A page carry indicates that the instruction just fetched was located at word 255 of the present page. Since the process of incrementing P also incremented the page field of P, the procedure kernel must be accessed to determine if the next sequential procedure page is resident in local memory. Thus, a page carry condition results in a control sequence being entered which unconditionally transfers to the instruction indicated by the incremented P.

For example, if the present P indicates page 10, location 255, incrementing P results in P indicating page 11, location 0. The Procedure Kernel entry associated with page 11 is accessed to determine if page 11 is resident in local memory.

4.3.3.3 Procedure Page Fetch

When a Page Carry condition exists, or when the instruction just executed was a Transfer causing a change in P, or an escape instruction, a procedure page fetch is performed.

The contents of the page field (bits 0-7) of the program counter are appended to the contents of the Procedure Kernel Register (see Section 4.1.1.8) to form a local memory address of a kernel entry word.

PROCEDURE KERNEL REGISTER								P COUNTER PAGE FIELD							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

It should be remembered that the Procedure Kernel Register is a 2-bit register. Therefore, in the above format these two bits occupy bit positions 6 and 7 with zeroes in bits 0-5.

4.3.3.3.1 Kernel Entry Word

The Kernel Entry Word accessed has the format:

TM PAGE								KERNEL LOAD	KERNEL TRACE	RESIDENT	PAGE/WORD	WIDE ADDRESS																				READ PROTECT	WRITE PROTECT	COMMAND PROTECT	PARITY
																																R	W	C	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

A kernel entry word consists of 36 bits, 32 of which are directly used in processing. The remaining 4 bits, the least significant byte, contains certain security-oriented control information. These bits are interpreted as follows:

a) Read-Protect (Bit 32) - When this bit is set, the program is not permitted to read the desired information unless the PMU is in the executive mode. If this bit is discovered set in the kernel in the course of a normal virtual Read fetch cycle, the PMU halts and signals the Exécutive if the trap routine so indicates.

b) Write-Protect (Bit 33) - When this bit is set, the program is not permitted to write into the indicated virtual segment unless the PMU is in the executive mode. If this bit is discovered set in the kernel in the course of a normal Store type command, the PMU halts and signals the Executive if the trap routine so indicates.

c) Command Protect (Bit 34) - This bit, when set, indicates that the associated kernel word contains a command to be interpreted by another subsystem. Certain PMU instructions permit a program to issue commands to other subsystems. These instructions may only issue those commands located in the kernel and marked by this bit.

d) Parity Check (Bit 35) - This bit assures an odd parity within the 36 bit word if no single error exists.

If the kernel entry indicates word oriented data (bit 11 is zero), a Command Violation trap occurs. All procedure must be page oriented. (Note: An Execute instruction interprets data as an instruction and therefore uses the Operand Kernel Page.)

4.3.3.3.2 Resident Procedure

If the Command Protect (Bit 34) bit of the kernel entry word is one or if the Page/Word (bit 11) bit is zero (indicating word data), the Command Protect Trap is generated and the remainder of the procedure fetch is aborted. If bit 34 is zero and bit 11 is one (indicating paged procedure), the contents of the Resident (bit 10) bit is examined. If the page is shown as being resident (bit 10 is one), the contents of the TM Page Field (bits 0-7) are placed in the Procedure Page Register (see Section 4.1.2.1) and the next instruction is fetched according to a normal instruction fetch (Section 4.3.3.2).

4.3.3.3.3 Non-Resident Procedure

If the kernel entry word indicates paged procedure (bit 11 is 1), not resident (bit 10 is zero), and the Command Protect bit (bit 34) is a zero, a standard Instruction Transmission Word (see Section 4.3.5.2.7) is created with an Op Code 06 (Read, Page to Output instruction) and transmitted on the primary bus system. The 20-bit System Address of the transmission word is equal to bits 12-31 of the kernel entry word. At this point, the PMU prepares to accept a page transmission by determining the appropriate local memory location in which to place the first word of the received page. The page replacement sequence that is performed is described in Section 4.3.3.3.4. The page location in which the incoming page will be stored is placed into bits 4-7 of the kernel entry word (bits 0-3 are always zero because of the maximum 4K size of the task memory), the residency bit (bit 10) is set to one and the kernel entry word is rewritten in the local memory location from which it was read. The PMU also places the contents of bits 0-7 of the updated kernel entry word in the Procedure Page Register and writes the incoming page into sequential local memory locations beginning with the address formed by augmenting the contents of the Procedure Page Register with hex 00. In addition, the PMU locates the kernel entry word, if any, that referred to the contents of the page being overlayed and clears the residency bit (bit 10) to zero. At this point the page replacement sequence is complete and the next instruction is fetched according to a normal instruction fetch (Section 4.3.3.2).

4.3.3.3.4 Replacement Algorithm

Whenever non-resident paged information is brought in, the selection of a local memory page location must be made. The DPE allows a selection from four different types of algorithms. A pair of registers, Lower Bound Register and Upper Bound Register, are provided which maintain the bound of the local memory area which is subject to replacement. It is within this area that the automatic algorithms work. The algorithm to be used is contained in the Replacement Algorithm Register and is set, along with the two boundary registers, by the execution of the Set Task Parameter instruction (Op Code 29). The four replacement algorithms are:

- Programmer Specified
- First In/First Out
- Random
- Sequential Fill/Random

4.3.3.3.4.1 Programmer Specified

If the contents of the Replacement Algorithm Register are 00, the starting location for the page store is the contents of bits 0-7 of the kernel entry word augmented by hex 00. In this case, the residency bit (bit 10) of the kernel entry word is set to one and the kernel entry word is rewritten into the local memory location from which it was read.

Also, if bits 0-3 of the referenced kernel word are not 0000, the page to be replaced is determined by bits 0-7 of the just read kernel word (regardless of the specified replacement algorithm). In the last case it should be noted that even though the replaced page was not generated by the automatic replacement logic, the automatic replacement logic assumes that it has. Consequently, the call of successive page replacements via this means may, for example, make the replaceable area look filled (Sequential Fill/ Random Algorithm) even though a page was never replaced in the dynamically replaceable area.

4.3.3.3.4.2 First In/First Out

If the contents of the Replacement Algorithm Register are 01, the page location in which the new page is to be stored is determined in the following manner. Immediately after an Initiate New Task instruction (Op Code 28), the new page is placed in the page location immediately above that specified by the Lower Bound Register. Thereafter, successive pages are placed in the next sequential higher page locations, until a page has been placed in the page location specified by the Upper Bound Register. Subsequent pages are placed in the location immediately above that specified by the Lower Bound Register, and successive pages are stored in successive locations as before.

4.3.3.3.4.3 Random

If the contents of the Replacement Algorithm Register are 10, the random algorithm is used and the page location is determined in the following manner. The internal counter that is used to determine the page location according to the first in/first out algorithm is allowed to continuously count in the range from one plus the contents of the Lower Bound Register to the value of the Upper Bound Register. Whenever the counter reaches the value of the Upper Bound Register, it is reset to its lower value. For each page replacement according to the random algorithm, the current value of the counter is used as the page location for storing the incoming page.

4.3.3.3.4.4 Sequential Fill/Random

If the Replacement Algorithm Register contents are 11, the Sequential Fill/Random algorithm is used. Immediately after an Initiate New Task instruction (Op Code 28), the new page is placed in the page location immediately above that specified by the Lower Bound Register. Thereafter, successive pages are placed in the next sequential higher page locations, until a page has been placed in the location specified by the Upper Bound Register. Subsequent pages are then replaced according to the random algorithm described in section 4.3.3.3.4.3.

4.3.3.4 Non Standard Procedure Page Fetch

In honoring a trap, the PMU performs a Procedure Page Fetch according to Section 4.3.3.3 except that Word 255 of the Procedure Kernel is used as the kernel word for all cases.

4.3.3.5 Examplex of Instruction Fetch

Figures 4 and 5 are illustrative diagrams of the steps which occur during the instruction fetch cycle for resident paged procedure and non resident paged procedure, respectively. Hexadecimal notation is used in these examples, where applicable.

4.3.3.5.1 Resident Paged Procedure (Figure 4)

In this example the current contents of the Program Counter "P" indicate that the last instruction (number 255 decimal) of virtual page 0A is to be fetched. The local memory address of this instruction (11FF) is obtained by catenating the contents of the Procedure Page Register (11) to the displacement field of P (FF). After this instruction is fetched, P is incremented by one which results in a page carry from bit 8 to bit 7. As a result, the Procedure Kernel Page Register contents for this program module are appended to the page field of P to obtain the local address of the kernel word. This kernel word indicates resident paged procedure which is located in local memory page 06. This local memory page 06 is loaded into the Procedure Page Register and appended to the displacement field of P (00) to obtain the local memory address (0600) for the virtual address (0B00) contained in P.

4.3.3.5.2 Non-Resident Paged Procedure (Figure 5)

For this example it is assumed that the same conditions and steps were followed as in the previous example up to the examination of the Kernel Word located in local memory address 020B. In this example the kernel word indicates that the paged procedure is non-resident (bit 10 is zero). Therefore, a "Read Page To Output" instruction is created and transmitted over the internal bus to the BORAM. For this example the PMU address is 00 and the BORAM address is 03. The required procedure page is located in address 4D6 in the BORAM. Upon receiving the "Read Page To Output" instruction,

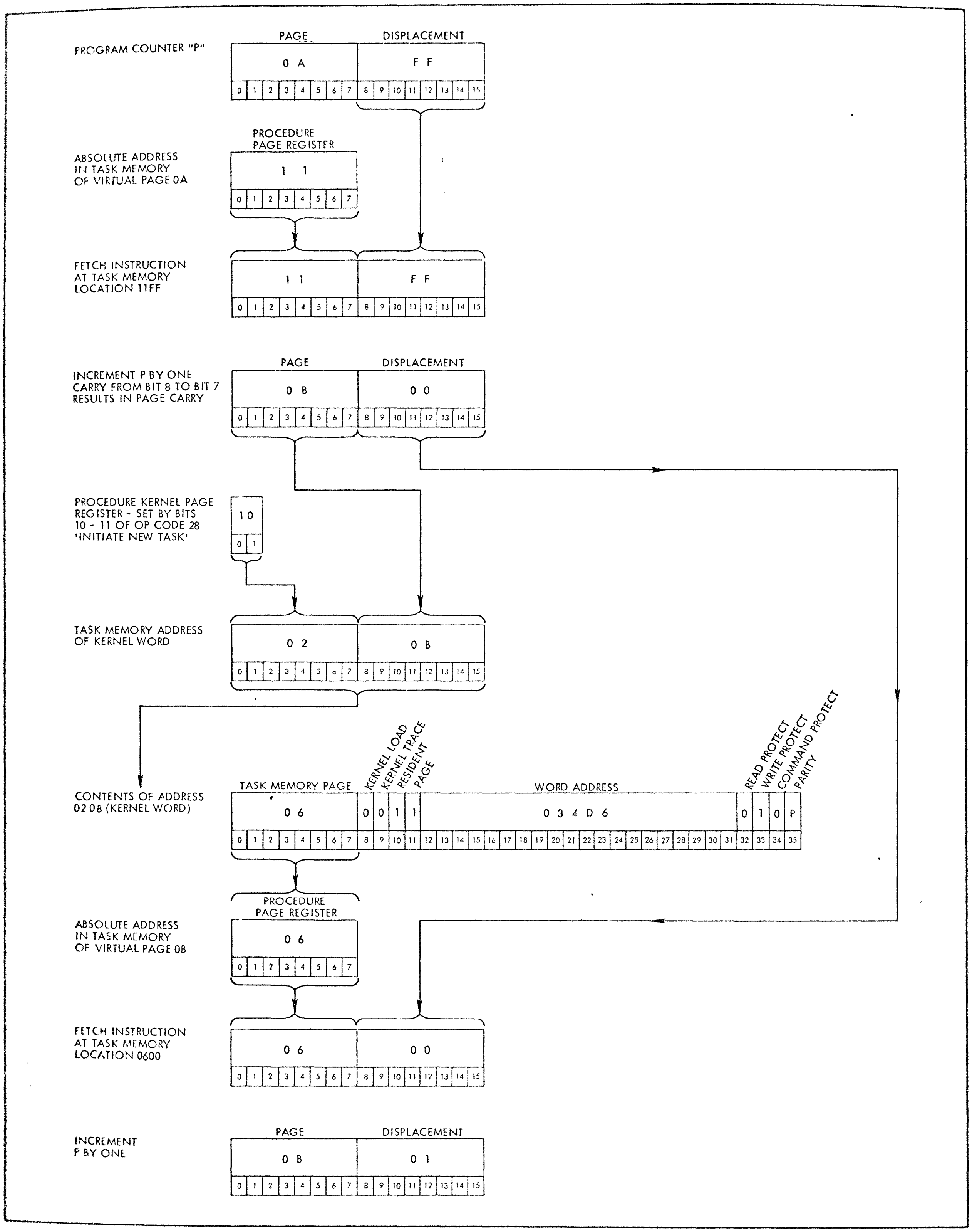


Figure 4. Virtual Addressing - Resident Paged Procedure

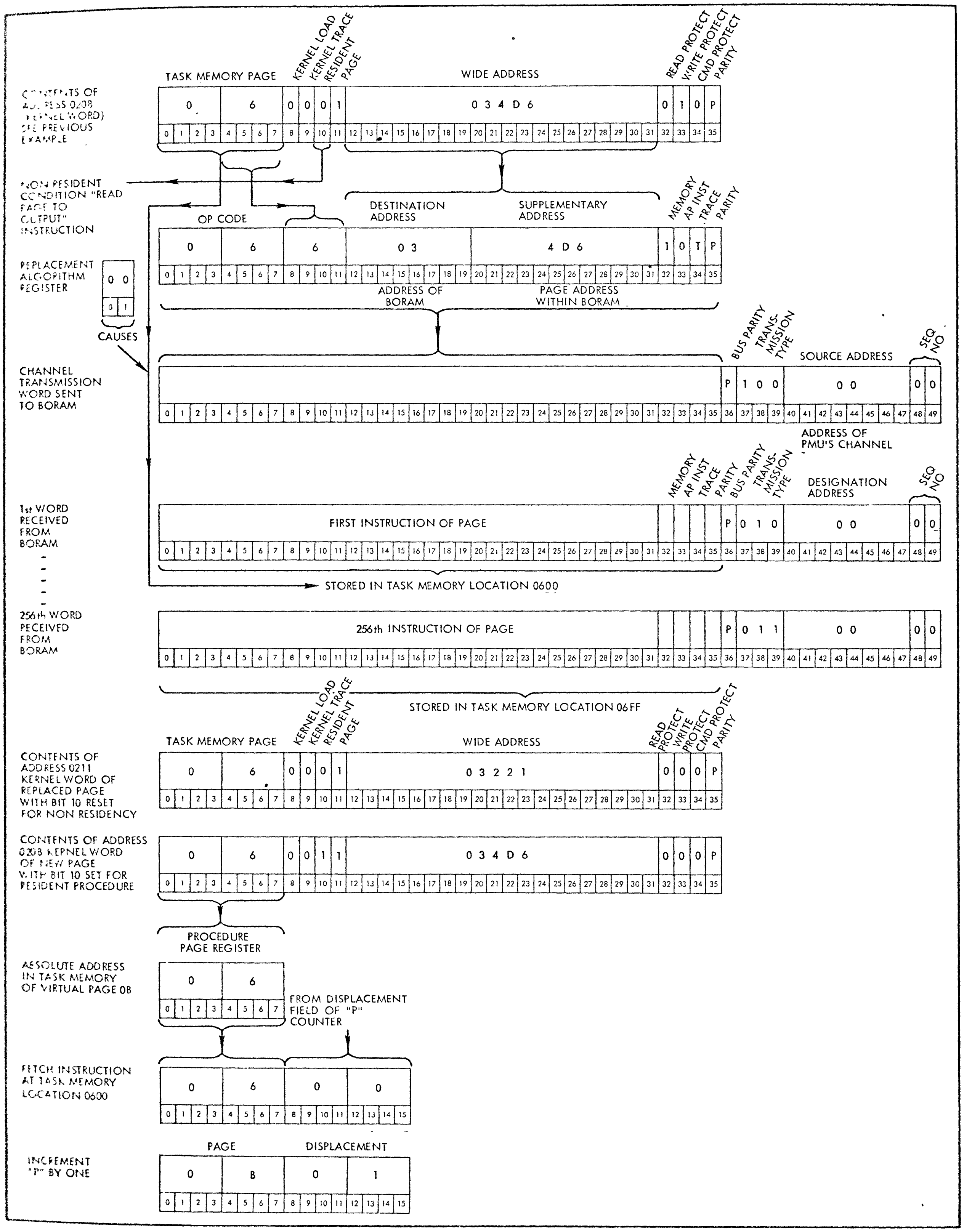

CONTENTS OF ADDRESS 020B (KERNEL WORD) SEE PREVIOUS EXAMPLE
TASK MEMORY PAGE
KERNEL LOAD
KERNEL TRACE
RESIDENT PAGE
WIDE ADDRESS
READ PROTECT
WRITE PROTECT
CMD PROTECT
PARITY
0
6
0 0 0 1
0 3 4 D 6
0 1 0 P
NON RESIDENT CONDITION "READ PAGE TO OUTPUT" INSTRUCTION
OP CODE
DESTINATION ADDRESS
SUPPLEMENTARY ADDRESS
MEMORY
AP INST
TRACE
PARITY
0
6
6
0 3
4 D 6
1 0 T P
ADDRESS OF BORAM
PAGE ADDRESS WITHIN BORAM
REPLACEMENT ALGORITHM REGISTER
0 0
CAUSES
CHANNEL TRANSMISSION WORD SENT TO BORAM
BUS PARITY
TRANSMISSION TYPE
SOURCE ADDRESS
SEQ NO
P 1 0 0
0 0
0 0
ADDRESS OF PMU'S CHANNEL
1st WORD RECEIVED FROM BORAM
MEMORY
AP INST
TRACE
PARITY
BUS PARITY
TRANSMISSION TYPE
DESIGNATION ADDRESS
SEQ NO
FIRST INSTRUCTION OF PAGE
P 0 1 0
0 0
0 0
STORED IN TASK MEMORY LOCATION 0600
256th WORD RECEIVED FROM BORAM
256th INSTRUCTION OF PAGE
P 0 1 1
0 0
0 0
STORED IN TASK MEMORY LOCATION 06FF
CONTENTS OF ADDRESS 0211 KERNEL WORD OF REPLACED PAGE WITH BIT 10 RESET FOR NON RESIDENCY
TASK MEMORY PAGE
KERNEL LOAD
KERNEL TRACE
RESIDENT PAGE
WIDE ADDRESS
READ PROTECT
WRITE PROTECT
CMD PROTECT
PARITY
0
6
0 0 0 1
0 3 2 2 1
0 0 0 P
CONTENTS OF ADDRESS 020B KERNEL WORD OF NEW PAGE WITH BIT 10 SET FOR RESIDENT PROCEDURE
0
6
0 0 1 1
0 3 4 D 6
0 0 0 P
PROCEDURE PAGE REGISTER
ABSOLUTE ADDRESS IN TASK MEMORY OF VIRTUAL PAGE 0B
0
6
FROM DISPLACEMENT FIELD OF "P" COUNTER
FETCH INSTRUCTION AT TASK MEMORY LOCATION 0600
0
6
0
0
INCREMENT "P" BY ONE
PAGE
DISPLACEMENT
0
B
0
1

Figure 5.. Virtual Addressing - Non-Resident Paged Procedure - Programmer Specified Replacement

the BORAM executes the instruction by accessing its location 4D6 and causing the transmission of the full 256 word page to the PMU. Upon receiving the words from the BORAM, the PMU stores the words in consecutive locations in local memory page 06, as determined by the page field of the Kernel Word because the replacement is programmer specified. The kernel word for the paged procedure which formally was stored in page location 06 is tagged non-resident (bit 10 is reset to zero). In this example, this kernel word is located in memory location 0211. The kernel word, memory location 020B, for the page just read into memory is marked resident. From this point on the same procedure as described in the previous example is followed to obtain the final instruction.

4.3.4 Instruction Word Interpretation

4.3.4.1 Instruction Format

The general format for both PMU and AP instructions are essentially the same and is given in Figure 6.

Bits 0-3 are termed the HI-OP bits, bits 4-7 are termed the LO-OP bits. HI-OP and LO-OP values are specified in hexadecimal notation. These two bytes, in combination, are termed the Operation Code (or Op Code), and specify the operation to be performed.

When bit 33 equals zero, indicating a PMU instruction, bits 8-11 indicate the scratchpad register (SPA) (or register pair in the case of full word instructions) which is to be used as the accumulator. When bit 33 is set to one, indicating an AP instruction, bits 8-11 are termed the parenthetical field. Bit 8 is an arithmetic precision bit and bits 9-11 specify parenthetical action.

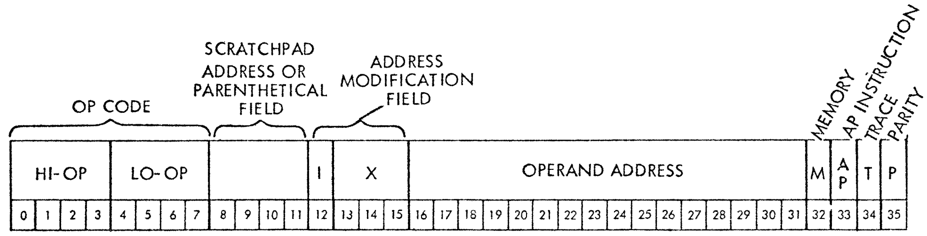


FIGURE 6 GENERAL INSTRUCTION FORMAT

Bits 12-15 specify requisite address modification. Bit 12 specifies whether the address derived by this instruction is the effective address of the operand or the address of another address. When bit 12 is set, an indirect addressing cycle is performed. Bits 13-15 specify one of seven index registers (SP [1] - [7]) to be used to index the instruction. If no indexing is desired, an index code of "000" is specified. Indexing occurs prior to Indirect Addressing.

Bits 16-31 specify the address of the operand. The PMU will typically be in "Virtual Address Mode," and this field thus specifies the virtual address of the operand. In this case, bits 16-23 specify the location of the RAMM segment address in the kernel, and bits 24-31 specify the displacement of the particular operand in that segment. In cases where direct addressing of task memory mode has been specified, bits 20-31 give the direct address. When both bit 12 and bit 32 are zero, the contents of bits 16-31 are taken as an immediate operand.

Bit 32, when set, specifies if a memory access is to be made.

Bit 34, when set, will cause the instruction trace trap to be raised upon completion of the execution sequence.

Bit 35 specifies odd parity for the instruction word.

4.3.4.2 Parity (Bit 35)

Parity of the instruction word (bit 35) is checked after the instruction is read from local memory. If parity does not check, the Parity Error Trap is enabled. Parity is generated on all local memory writes. Thereafter, bit 35 is ignored in instruction processing.

4.3.4.3 Addressing Modification (Bits 12 thru 15)

In all internal instructions, the address portion of the instruction word may be modified by indexing or indirect addressing or both. Indexing precedes indirect address modifications.

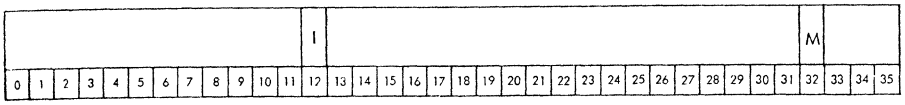
4.3.4.3.1 Indexing (Bits 13 thru 15)

The Index field (bits 13-15) specifies one of seven index registers (scratchpad registers 1-7) or the Index field (000) specifies that indexing is not to be performed. If indexing is required, the contents of the specified index register is added to the contents of bits 16-31 of the instruction word. The sum replaces the bits 16-31 of the instruction word.

													INDEX																						
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

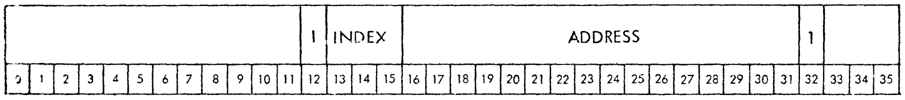
4.3.4.3.2 Indirect Addressing (Bit 12)

The process of indirect addressing entails replacing bits 12-32 of the original instruction word. Bit 12 of the original instruction, when a one, specifies that indirect addressing is to be performed. When bit 12 is a 0, no indirect addressing is performed. (See Section 5.3.13 for indirect addressing in regard to Dimension Words)



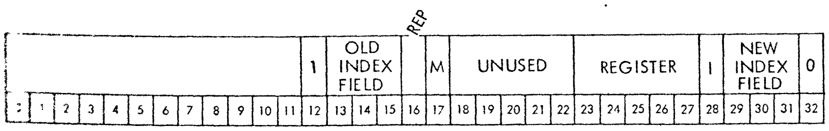
When bit 12 is a one, bit 32 specifies one of two indirect modes: Memory or Register Indirect Addressing.

4.3.4.3.2.1 <u>Memory Indirect Addressing (Bit 32 is ONE)</u>



When bit 32 is one, and indirect addressing is to be performed (bit 12 is one), bits 16-31 are used to directly specify an indirect word. The fetch of this indirect word is accomplished as in Section 4.3.4.4. The contents of bits 12-32 of the fetched indirect word replace bits 12-32 of the original instruction word. The remaining bits of the indirect word are ignored.

4.3.4.3.2.2 <u>Register Indirect Addressing (Bit 32 is zero)</u>



When bit 32 is zero, and indirect addressing is to be performed (bit 12 is one), bits 12-32 of the original instruction word are replaced as follows:

Bits 28-31 replace bits 12-15

Bit 17 replaces bit 32

The contents (bits 0-15) of the scratchpad register specified by bits 23-27 replace bits 16-31

In addition, bit 16, if a one, indicates a Replacement operation (see Section 4.3.4.3.3)

4.3.4.3.3 Chaining of Address Modification

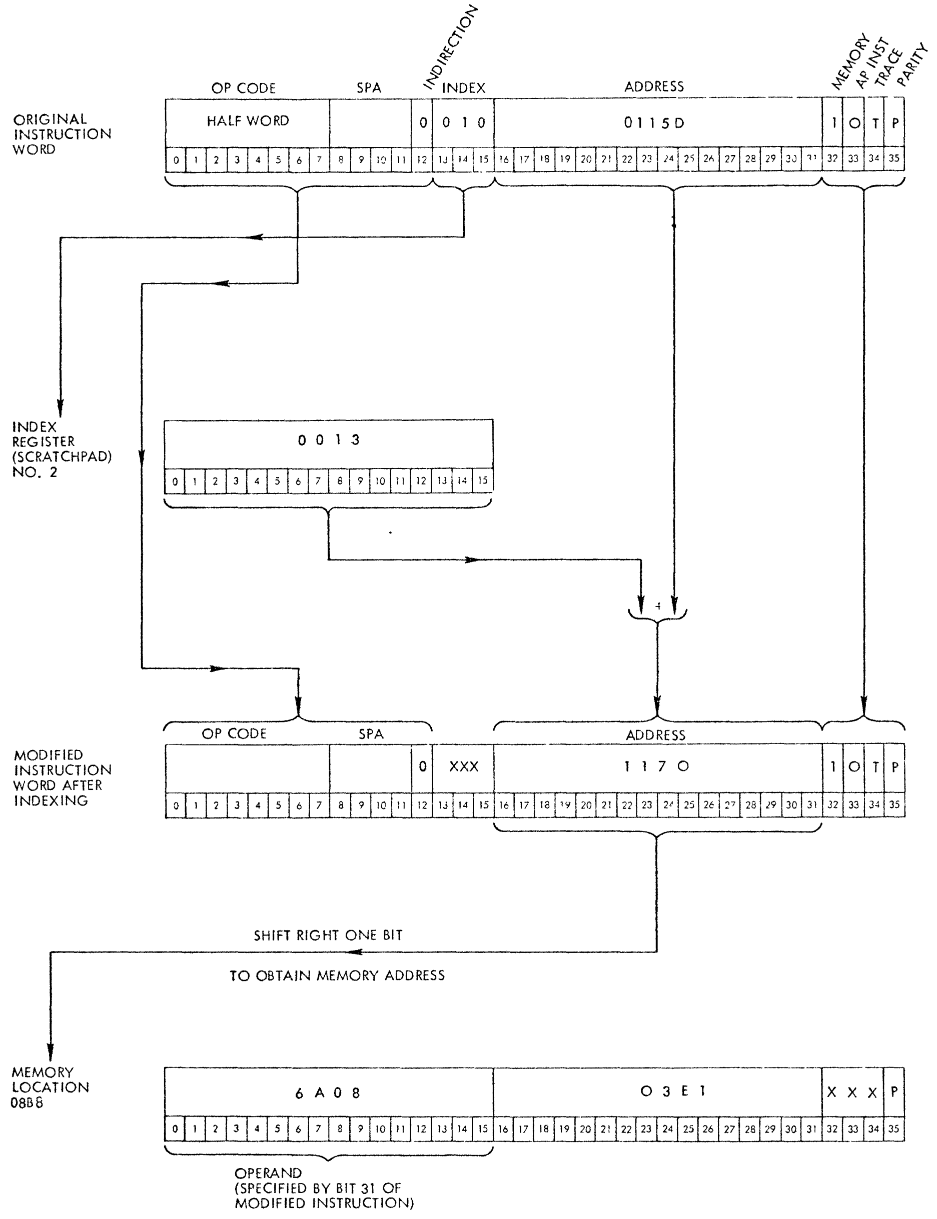
If indirect addressing was required, the process of indexing is again performed and indirect addressing may be repeated. If the Replacement operation was specified from the previous register indirection, bits 16-31 of the modified instruction word (after indexing) are written back into the register specified by the preceding register indirect cycle (4.3.4.3.2.2). In all cases, bit 12 of the modified instruction word is then checked. If bit 12 is a one, the process of indirect addressing is repeated as before. If bit 12 is a zero, the operand cycle will commence.

4.3.4.3.4 Examples of Address Modification

Figures 7 through 9 are illustrative examples of the various forms of address modification discussed in section 4.3.4.3. The digits used in these examples are either binary or hexadecimal and may be discerned by the field length when the digit(s) are used. Although the normal addressing mode is virtual, it has been assumed for simplicity that the addressing mode in these examples is absolute.

4.3.4.3.4.1 Example No. 1 (Figure 7)

This example indicates the steps involved during address modification of a half word PMU instruction (bit 33 = 0) using indexing and direct memory fetch of the operand. In the original instruction word the index field (bits 13 - 15) indicates that the contents of Index Register No. 2 are to be added to the contents of the address field (bits 16-31). This addition results in a new address of 1170 for the modified instruction word. No indirection is required (bit 12 is zero) and because bit 32 is a ONE, a memory access is to be made for the operand. Because this instruction contains a half word op code the address field of the modified instruction word is shifted right one bit and a leading zero added to the high order bit position to obtain the actual memory location (08B8) of the operand (see section 4.3.4.4.1). Bit 31 of the modified instruction word was a zero which specifies that the left half of the 32 bit operand read from memory is to be used for the execution of this instruction.



CONDITIONS - PMU INSTRUCTION (BIT 33 IS 0)
HALF WORD OP CODE
INDEXING, MEMORY OPERAND
FETCH NO INDIRECTION

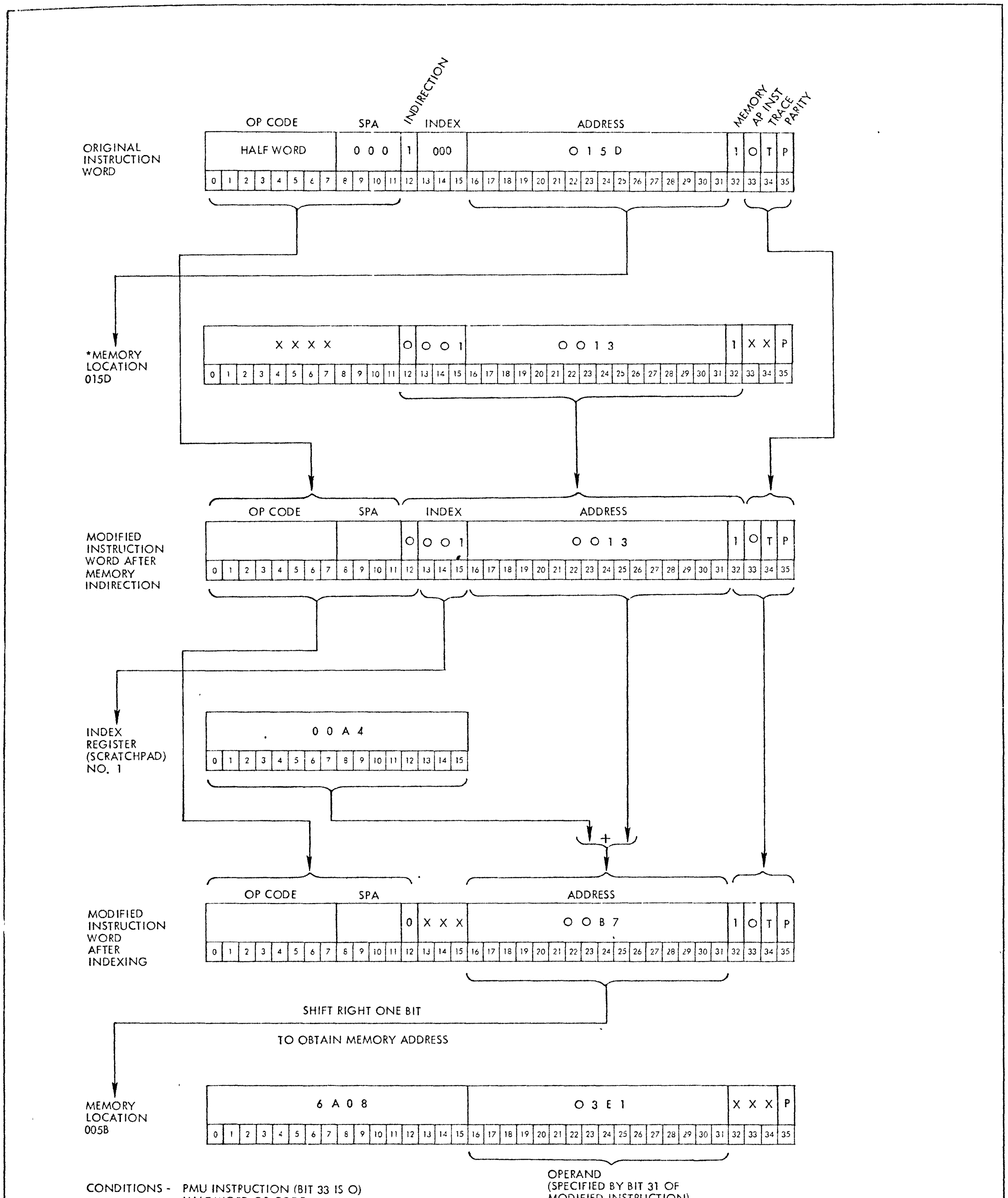
FIGURE 7 Address Modification - Example 1

4.3.4.3.4.2 Example No. 2 (Figure 8)

In this example, a half word PMU instruction (bit 33 = 0) is modified by memory indirection and indexing prior to a memory fetch of the operand. Indexing always precedes indirection, however, in this example the index field of the original instruction word is zero resulting in no index modification. Bit 12, being one, and bit 32 set to one, indicates that memory indirection occurs. The indirection process results in bits 12-32 of the original instruction word being replaced by bits 12-32 of the contents of the memory location (015D) as specified by the address field of the original instruction. The resulting modified instruction now specifies that the contents of Index Register No. 1 are to be added to the contents of the address field in order to obtain the new operand address. Again, as in Example No. 1, this instruction involves a half word op code. However, because bit 31 = 1, the right half of the 32 bit operand read from memory is used for the execution of this instruction. (See Section 4.3.4.4.1)

4.3.4.3.4.3 Example No. 3 (Figure 9)

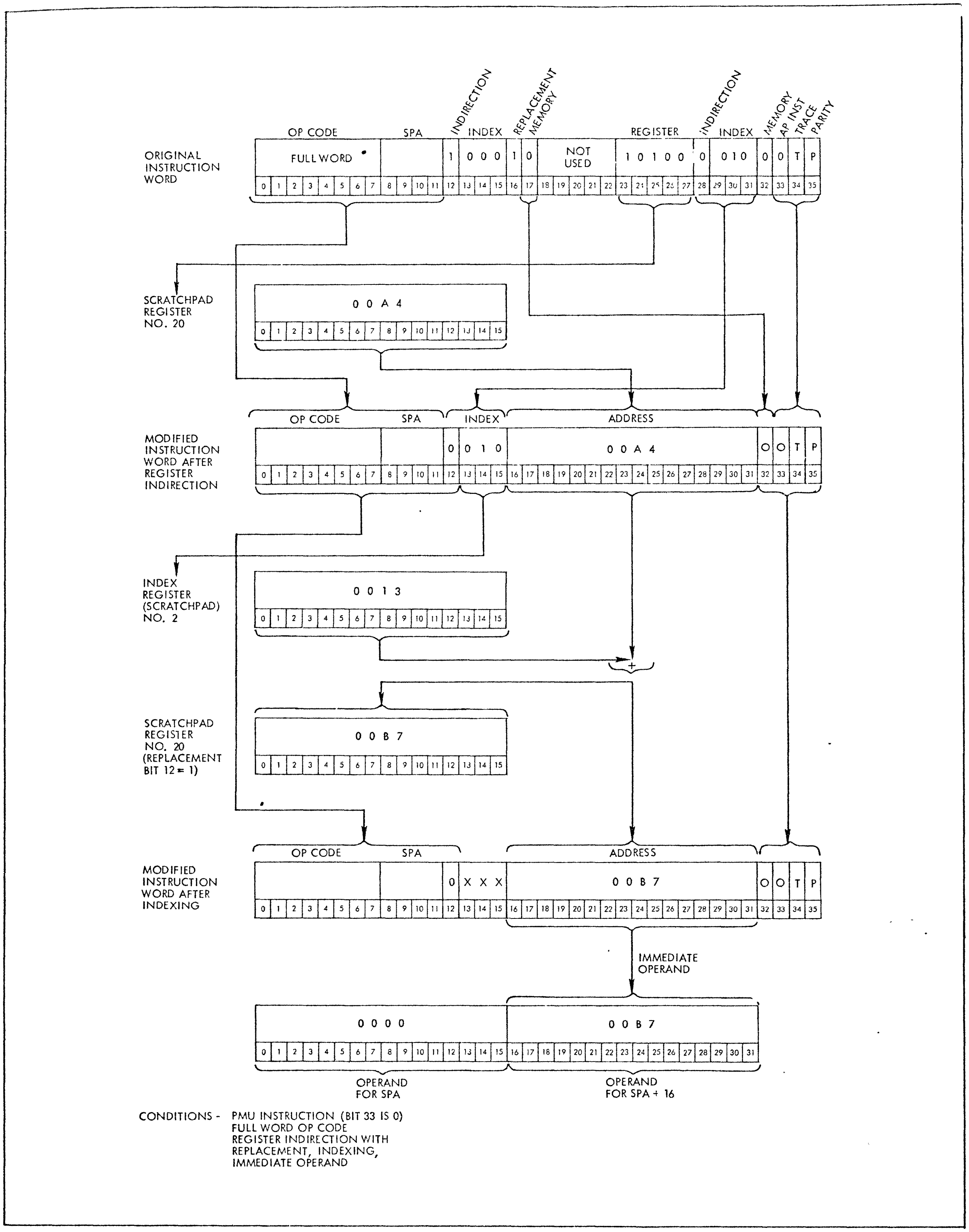
For this example it is assumed that the instruction is a full word PMU instruction calling for register indirection, indexing and an immediate operand. Bit 12, of the original instruction word, being set to one with bit 32 set to zero specifies register indirection, with bits 23-27 specifying Register No. 20. The contents of this register



CONDITIONS - PMU INSTPUCTION (BIT 33 IS O)
HALF WORD OP CODE
MEMORY INDIRECTION, INDEXING,
MEMORY OPERAND FETCH

*INDIRECTION IS TO A FULL WORD
EVEN IF OP CODE IS FOR 1/2 WORD

FIGURE 8 Address Modification - Example 2



CONDITIONS - PMU INSTRUCTION (BIT 33 IS 0)
FULL WORD OP CODE
REGISTER INDIRECTION WITH
REPLACEMENT, INDEXING,
IMMEDIATE OPERAND

FIGURE 9 Address Modification - Example 3

are then used for the address field of the modified instruction. At this point indexing is called for and proceeds in the same manner as the previous examples. However, bit 16 of the original instruction was set to one, calling for replacement, so the contents of Register 20 are replaced by the results of the indexing operation. The modified instruction after indexing specifies no indirection, bit 12 = 0, and no memory access, bit 32 = 0, resulting in an immediate operand. Because the instruction calls for a full word operand the the immediate operand is only 16 bits, the high order portion of the final operand is filled with zeros. (See Section 4.3.4.4.2)

4.3.4.4 Operand Cycle

The operand cycle determines the operand for use with bits 0-11 of the original instruction word. The state of bit 32 of the modified instruction word (after all indexing and indirect addressing operations have been accomplished) determines whether the operand is contained in memory (bit 32 is 1) or is the actual contents of bits 16-31 of the modified instruction word (bit 32 is 0). If the operand is to come from memory, the contents of bits 16-31 of the modified instruction word are used to address memory to obtain a 36 bit operand. Bits 0-15 (left half) or bits 16-31 (right half) of the addressed location are selectable, should a half word be required.

All operands for use by external devices are full 36 bit operands. For these cases, the 16 bit address is used to access up to 65K of memory.

Operands for use by a PMU are half word operands (16 bits) or full word operands (32 bits). The word size is determined by the op code field (bits 0-7) of the instruction. In cases where a 16 bit operand is required, the least significant bit of the address for the half word instructions specified the left half (bit 31 of the modified instruction word equals zero) or the right half (bit 31 of the modified instruction word equals one) of the full word addressed by the remaining 15 address bits. A 16 bit address with the most significant bit being zero is sent to memory to retrieve the desired operand.

For those PMU instructions which require full word operands all 16 address bits of the instruction word are used for accessing full word operands from memory.

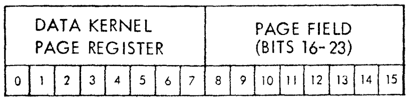
Memories always read and deliver full words. The PMU selects the referenced half word (if appropriate) when the operand is received.

4.3.4.4.1 Memory Operand Fetch

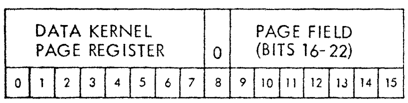
4.3.4.4.1.1 Virtual Address Mode

The normal address mode for DPE operations is virtual, which is indicated when bit 23 of the Set Task Parameter Instruction (Op Code 29) is set to one. The first step in calculating the absolute address of the referenced operand is to access the entry in the kernel indicated by the Page Field of the effective address of the modified instruction. The local memory location of this kernel entry word is formed by appending the Data Kernel Page Register to the Page field. The resulting 16 bit address has the following format.

If the addressed operand is full word (32 bits), the address of the kernel entry is:

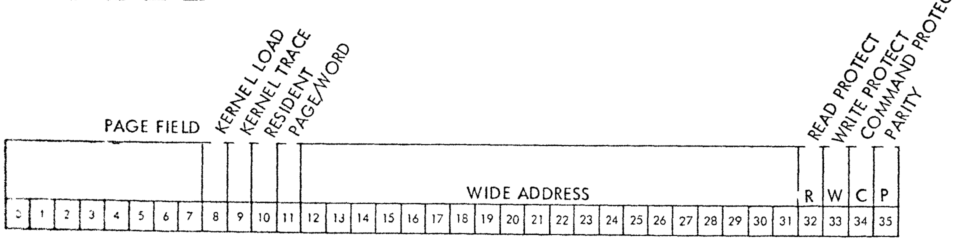


If the instruction addresses a half word (16 bits) the address of the kernel entry is:

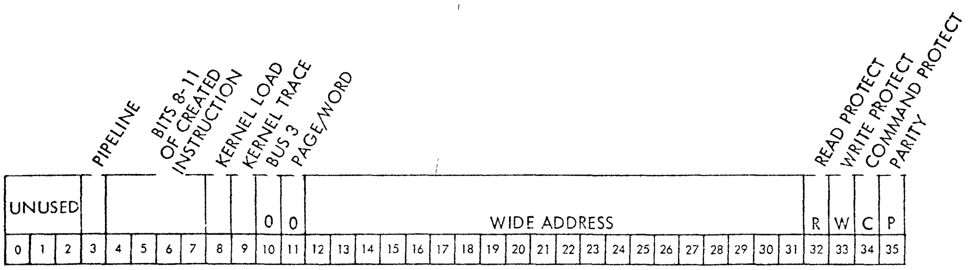


The kernel entry word has the format as indicated below and depending upon whether the data is paged or word oriented.

PAGED DATA



WORD DATA



If the kernel entry indicates word data (bit 11 = 0), bits 24-31 of the kernel entry are added to the displacement field of the effective address of the instruction and the carry from bit 24 to bit 23 inhibited. The resulting 20 bit wide address (bits 12-23 of the kernel and the 8 bit sum just generated) is the address field of the instruction that is created and sent to the channel. The channel instruction generated is a function of the type of instruction that is being executed, Read or Write.

For word data, bit 10 of the kernel word indicates which set of primary buses are used. With bit 10 equal 0, Bus 1 or 2 is used. When bit 10 equals 1, Bus 3 is used. It should be noted that the availability of the 3 possible primary buses is controlled by the Set System Parameter instruction (Op Code 25). Should bit 10 be 1 and the DPE is not allowed to use Bus , the DPE will hang up.

For all created transmission intructions, bits 4-7 of the referenced kernel becomes bits 8-11 of the created Input/ Output bus instruction.

For word data, bit 3 denotes whether the pipeline (see Section 4.3.4.4.5) is to be entered for the next sequential instruction. If bit 3 is one, the pipeline is entered. If bit 3 is zero, the pipeline is exited or terminated. Bit 3 must be specified as zero when arrays or complex data type is to be encountered. By definition, paged data halts the pipeline and therefore the stopping of the pipeline need only be determined for word data.

If the kernel entry indicates Paged Data and not Resident, a Read Page to Output instruction (Op Code 06) is created and sent to the channel. This is true even if the instruction is a Store type. The displacement field of the instruction is not added to the 20 bit wide address when data is page oriented. Thus the 20 bit wide address of the Read Page to Output instruction is bits 12-31 of the kernel entry. The PMU, while awaiting the return of the addressed page, performs a page replacement sequence. The sequence for bringing in a page is identical to that described for procedure (see Section 4.3.3.3). Once the referenced page is loaded, the virtual addressing mechanism is re-entered to resolve addressing to an operand.

If the kernel entry indicates paged data, resident, and command bit (bit 34) is 0, the Procedure Page register is loaded with bits 0-7 of the kernel entry. Then the operand located at the local memory location indicated by appending the data page register to the Displacement field of the effective virtual address is fetched and operated upon.

In all the above cases, if an operand is read, the Read Protect bit must be 0. If an operand is written into, the Write Protect bit must be 0, and if an operand is executed the Command Protect bit must be 0. If any of the above conditions are not true, the appropriate violation is detected and a trap occurs.

4.3.4.4.1.2 Absolute Address Mode

When bit 23 of the Set Task Parameter instruction (Op Code 29) is zero, the operand addressing mode is absolute i.e., local memory is directly addressed. It should be noted that absolute addressing applies only to operand fetch and not to procedure fetch which is always done in the virtual mode.

For a full word operand, the contents of bits 16-31 of the modified instruction are used to directly address local memory. Bits 0-31 of the contents of the memory location are used as the operand.

For a half word operand, the contents of bits 16-30 of the modified instruction are shifted right one position and a leading zero appended to the high order position to form a 16 bit direct address in local memory. Bits 0-31 of the contents of the memory location so addressed contain the desired half word operand. If bit 31 of the modified instruction is zero, bits 0-15 of the memory word are used as the half word operand. If bit 31 of the modified instruction is one, bits 16-31 of the memory word are used as the half word operand.

4.3.4.4.2 Non Memory Operand Fetch

When bit 32 of the modified instruction word is zero, the operand is immediate and is the contents of bits 16 thru 31 of the modified instruction word. If a half word operand is required by the op code (bits 0-7) of the instruction word, the contents of bits 16-31 is that operand. If a full word operand is required by the op code (bits 0-7) and if the contents of bits 16-31 are not the contents of a scratchpad register obtained by a register indirection addressing operation, then the contents of bits 16-31 form the low order portion (bits 16-31) of the full word operand and sixteen zeroes are used as the high order portion (bits 0-15) of the full word operand.

If a full word operand is required and the contents of bits 16-31 of the modified instruction word are the contents of a scratchpad register obtained by a register indirect addressing operation, then these contents will form the high order portion of the full word operand. The contents of the corresponding scratchpad register, whether in the high (SP [16] - [31]) or low (SP [0] - SP [15]) bank, will form the low order portion of the full word operand.

It should be noted that register to register operations are accomplished, whether with half or full word operands, through the use of register indirect addressing with immediate access (bit 32 is zero).

4.3.4.4.3 Instruction Trace (Bit 34 is One)

Bit 34 of an internal instruction word, when set to one, causes a Trace Interrupt to be generated at the conclusion of PMU involvement with the instruction. The instruction trace trap will also be generated whenever bit 34 of the second instruction word of the Two Word I/O with Indexing instruction (Op Code 7B) is one.

4.3.4.4.4 Example of a Non Resident Full Word Operand Fetch (Fig. 10)

In this example it is assumed the PMU is operating in the virtual addressing mode so that the address field (bits 16-31) of the instruction is partioned into a page field (bits 16-23) and the displacement field (bits 24-31). The first step in obtaining the operand is to catenate the contents of the Data Kernel Page Register with the contents of the page field to determine the local memory address (in this case, 0301) of the kernel word for the page which contains the operand. In this example, the kernel word indicates word data (bit 11 is zero).

Therefore, a "Read Operand to Output instruction (Op Code 04) is created and transmitted over the primary bus to the RAMM. In this example, the address of the RAMM is assumed to be 02 and location within the RAMM of the desired operand is assumed as 3EC.

4.3.4.4.5 Pipeline

The DPE overlaps data requests to RAMM with instruction fetching, and provides up to 13 levels of instruction look ahead.

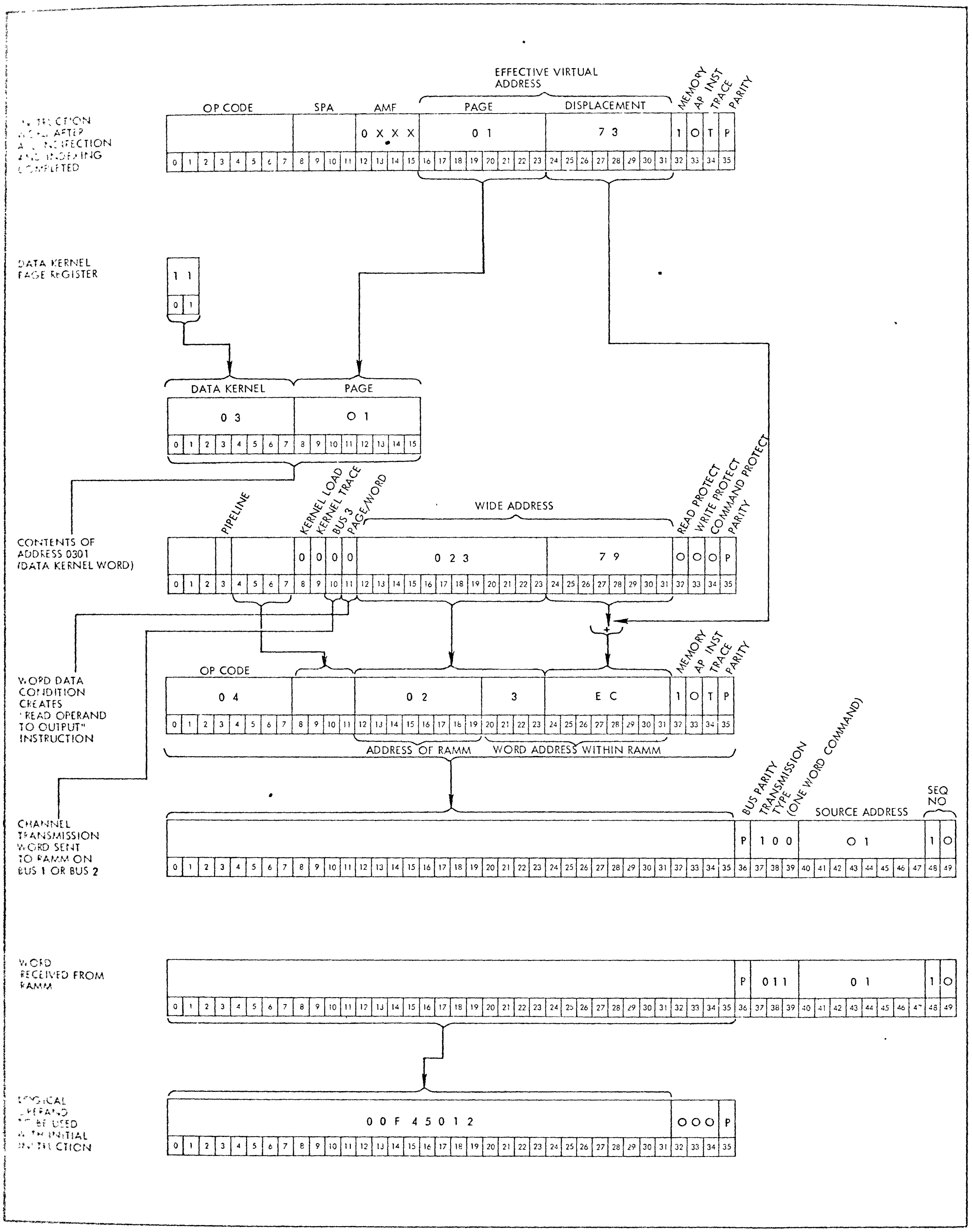


FIGURE 10 Virtual Addressing - Non-Resident Full Word Operand

The facilities which provide these capabilities are the APQ and the AADC channel.

The APQ is a FIFO stack which holds scalar, real operands and instructions awaiting execution by the AP. The APQ is configured in a manner that allows the queue to act as two independent stacks, for the control part (bits 0-11 of an instruction) and the referenced operand. Information is also placed in the operand queue to designate the precision of the loaded operand. In the absence of any address modification, the control part of the APQ is loaded every two internal cycles.

The fetch cycle pipeline utilizes the APQ and channel functions in a basic two step sequence loop. Step 1 of the sequence decodes the instruction that has been read out on the previous step, reads the data kernel location associated with the virtual address of the decoded instruction, sets up various internal indicators, and, in the absence of any address modification specifications or pipeline obstruction conditions, (to be described), advánces to step 2 of the basic pipeline sequence.

On step 2 of the pipeline sequence, security interrogation is performed on the data kernel word just read out, and in the absence of security violations, but in the presence of word declared data, a Read Operand request is placed in the Channels input queue, the control half of the present instruction is placed in the APQ, the next sequential instruction is read, and a transfer to step 1 of the basic two step sequence is executed.

On step 1 of this sequence, in addition to those actions described above, the input secondary bus is monitored for data. When data is present, it is routed to the operand half of the APQ thereby completing the fetch of the referenced operand and providing sufficient information for the AP to execute the instruction with its operand from the APQ.

The conditions that interrupt this two step sequence can be divided into two classes: address modification specification and pipeline obstructions.

Address modification specifications involves instructions which declare indexing, indirection, and literal operands. When indexing is specified, step 2 of the basic pipeline sequence cannot be entered until the indexing is performed. Indirection involves entering a sequence which obtains an indirect word. This involves transmitting a Read Indirect Word to Output to a RAMM if word data is declared in the data kernel word referenced by the virtual address of the instructions, or referencing Task Memory if the data is paged and resident.

In either case, the pipeline is halted until indirection is no longer specified, and a final operand reference is obtained, which results in re-entering the basic pipeline sequence. In the case of a word declared indirection, the fetch cycle loops awaiting the return of the indirect word referenced by the Read Indirect Word to Output instruction transmitted to the RAMM. During this looping process, the operand part of the APQ is being loaded with the operand requests made before the indirection was specified.

If a literal (immediate) is encountered during the instruction decoding on step 1, a test is made to determine the state of the pipeline. If no data requests are outstanding, the APQ control part and operand part are simultaneously loaded and step 1 re-entered. If the pipeline is not empty, the APQ control half is loaded, and the specified literal is placed in the channels input queue. The channel sequence number mechanism maintains the proper association between literal and instruction control part.

Pipeline obstructions involve the decoding of an instruction whose execution must be delayed until some pipeline condition is satisfied. Otherwise an ambiguous association of operand to op code or some machine state change may occur that would not be anticipated.

The DPE pipeline obstructions are: An AP transfer in the APQ and a second AP transfer detected, or a PMU instruction detected. When an AP transfer is detected, program counter sequencing progresses as if <u>no</u> transfer was taken. To ensure recoverability of the machine state, when a second AP transfer or a PMU instruction is detected, the fetch cycle loops awaiting the completion of the pending conditional transfers by the AP.

An AP Store and Halt instruction (Op Code E8) is detected. The fetch cycle loops awaiting the completion of the pending store by the AP.

A PMU instruction or AP instruction executed within the PMU is decoded. The fetch cycle must await the return of the word requested by the decoded PMU instruction. Simultaneously, the AP fetch cycle is loading the data part of the APQ. Once the word requested by the PMU instruction is obtained, the instruction is executed, and the basic fetch cycle re-entered. It should be noted, that PMU and AP instructions can be executed simultaneously.

Resident Data (as indicated by the reference Data Kernal Word) is decoded and the pipeline is not empty. The fetch cycle loops awaiting the pipeline to clear to maintain the proper association of data with instructions.

Block Mode Operations - The fetch cycle ceases to function in an overlapped manner when non-scalar, or complex operands are manipulated. This condition is signalled by bit 3 of the reference data kernel word being 0, when word structured data is encountered.

4.3.5 Instruction Execution

The operation code, determined by bits 0-7 of the original instruction word, determine the specific operation to be performed by the PMU. Hexadecimal notation is used to designate the contents of the operation code field, with one hexadecimal digit representing bits 0-3 and another, bits 4-7.

4.3.5.1 Operand Destination

Bit 33 of the original instruction word, which was unaffected by indirect replacement, specifies whether or not this instruction is to be sent to the AP.

4.3.5.1.1 Destination is AP

When bit 33 is a ONE, the Arithmetic Processor is defined as the recipient of the instruction and the operand. In the case of an operand fetched from memory, bits 0-11 of the original instruction word and the 36 bit memory operand are sent to the AP as follows:

OPERAND FROM MEMORY																																DATA TYPE		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

OP CODE								PF			
0	1	2	3	4	5	6	7	8	9	10	11

The AP will interpret bits 0-7 of the original instruction word as an operation code, and bits 8-11 as the Parenthetical Field. Bits 32-34 of the memory operand are interpreted as a data type tag, defining bits 0-31 of the operand. These fields are defined in detail in conjunction with AP operations (see Section 5.0).

If an AP destined operand was immediate, bits 0-11 of the original instruction word, interpreted as shown above, are sent to the AP. In addition, bits 16-31 of the modified instruction word are sent to the AP where they will be interpreted as a 32 bit logical operand with ZEROs in bits 0-15 and the immediate value in bits 16-31.

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

OP CODE								PF			
0	1	2	3	4	5	6	7	8	9	10	11

4.3.5.1.2 Destination is PMU

If bit 33 is zero, the instruction will be executed by the PMU. Bits 0-7 of the original instruction word specify an operation that the PMU is to perform, and bits 8-11 typically specify a scratchpad register (except for TIMER instructions). Unless the operation specifies a high-bank register, bits 8-11 specify SP [0] - SP [15] . (If full word operations are used, the corresponding register for full word operations, as specified in Section 4.1.1.1, is also implied by this field.) If the operation does specify a high-bank register, these bits specify SP [16] - SP [31] . In most cases, a 16 bit operand, received during operand fetch, is also used for this operation. Thus, typical PMU computer operations will use the following formats:

OP CODE								SPA			
0	1	2	3	4	5	6	7	8	9	10	11

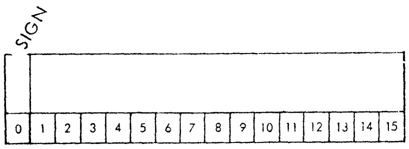
OPERAND															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The computer then executes the instruction.

4.3.5.2 PMU Word Formats

4.3.5.2.1 Half Word Arithmetic Format

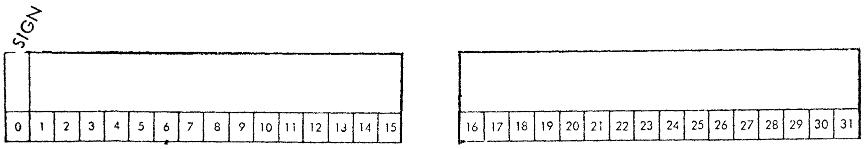
This is a 16 bit format held in a single 16 bit register. It represents a sign and magnitude integer number.



Bit 0 is the only unweighted bit; it carries the sign (0=+, 1=-). All other bits are weighted as $+2^{15-j}$ where j is the bit position. The numbers +0 and -0 are algebraic equivalents

4.3.5.2.2 Full Word Arithmetic Format

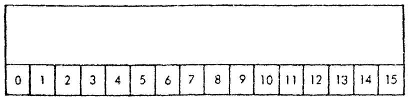
This is a 32 bit format held in two 16 bit registers. It represents a sign and magnitude integer number.



Again, bit 0 is unweighted and treated as a sign bit (0=+, 1=-). All other bits (1-31) are weighted as $+2^{31-j}$ where j is the bit position. The numbers +0 and -0 are algebraic equivalents

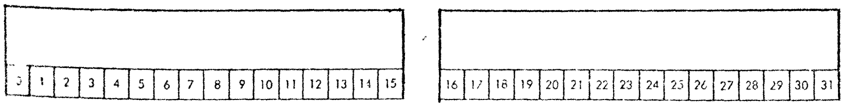
4.3.5.2.3 Half Word Logical Format

This is a 16 bit format held in a single register. All bits are unweighted.



4.3.5.2.4 Full Word Logical Format

This is a 32 bit format held in two 16 bit registers. All bits are unweighted.



4.3.5.2.5 Shift Count Format

In shift operations, an 8 bit N field (least significant 8 bits of the operand) is defined as the SHIFT COUNT.

								N							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

These bits are executed as a positive integer value, with bits 8-15 of the operand weighted $+2^{15-j}$ where j is the bit position.

4.3.5.2.6 Data Transmission Word Format

Instructions that call for data to be transmitted to an External Subsystem format a 50 bit Data Transmission Word with the format:

OPERAND FROM MEMORY																																				P	TRANSMISSION CODE			ACTIVE SOURCE								SEQ NO	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21.	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49

Bits 0-35 are the operand fetched from memory. Bit 36 is odd parity for bits 0 - 47 (even parity for bits 36-47). Bits 37-39 are the transmission code. For data, this code is weighted 011 (single word or end of block) or

010 (another word to follow). Bits 40-47 are the 8 bit contents of the active source register. Bits 48-49 are the sequence number which is set to be identical to bits 48-49 of the incoming instruction transmission word that requested the data transmission.

4.3.5.2.7 Instruction Transmission Word Format

In operations that call for the PMU to read or write data to or from an external subsystem, the PMU informs the external subsystem by formatting a standard instruction transmission word. The format of this word is shown in Figure 3 as the Command Word format. Bits 37-39 designate the transmission code which, for an instruction transmission word may be either 100 or 101. Code 100 designates a single word command. Code 101 designates a two word command.

4.3.6 Operand Types

When the PMU responds to an "operand" instruction, i.e., "Read Operand to Output" (Op Code 04) or "Write Operand From Input" (Op Code 05), the actual number of memory words to be transmitted or received is determined primarily by a three bit field, bits 32-34, of the first operand word. (See section 5.3 for a complete description of this field.) If this three bit field is 000, 010, 011, 100 or 111, a single word is involved. If the field is 101, or 110, two memory locations are to be read or written. If the code is 001 (dimension word - see section 5.3.12 for a complete description), the number of words to be read or written is determined by the word associated with this code.

The meaning of the dimension word data tag (bits 8-11) is:

0000 - Single Precision
0001 - Double Precision
0100 - Complex
1000 - Packed Binary (1 bit)
1001 - Packed Quaternary (2 bits)

1010 - Packed Hexadecimal (4 bits)
1011 - Packed Byte (8 bits)
1100 - Packed Half-Word (16 bits)

The number of words to be transferred is a function of the rank of the array, the number of elements in each dimension, and the data type of each element.

The total number of elements in an array is either equal to the low order dimension if the rank is 0000 (vector) or the product of the low order and high order dimensions if the rank is 0001 (matrix). Once the number of elements is determined, the number of words must be calculated. If the elements are single precision, the number of words transferred is equal to the number of elements. If the elements are double precision or complex, the number of words is equal to twice the number of elements. If the elements are packed, the number of words is equal to the quotient obtained by dividing the number of elements by the number of operands that can fit into a 32 bit word. The quotient obtained is then rounded up to the next highest integer. Thus, a 4 x 4 packed binary matrix occupies one word.

4.3.7 PMU Fetch Cycle (Overlapped Fetch Cycle)

One of the features to the fetch cycle is that a hierarchy of action is evaluated after each prior action to determine whether to wait for some external event to occur, i.e., honor an instruction interrupt, honor a trap, etc.

Once an action has been decided upon, it will be performed to completion or until it reaches a point where it must wait before returning to this decision logic.

There are 16 levels of action defined for the DPE. They are presented in hierarchal order. Lower actions will only be performed if no higher action has been selected.

1) The control panel indicates its desire to take control of the DPE. Control is relinquished to it.

2) An instruction interrupt has been received and validated and is in the input instruction queue. The DPE data pipeline is empty or the instruction is declared an emergency. This causes the instruction interrupt to be honored.

3) An AP deferral overflow is detected. A transfer to a control sequence which empties the deferral is executed.

4) An AP deferral underflow is detected. A transfer to a control sequence which re-establishes a deferral identity is executed.

5) A trap of higher priority than that of the presently running program is pending and the DPE data pipeline is empty. This causes the trap to be honored.

6) An AP interrupt is pending. This is caused by either a store or transfer instruction being executed. The appropriate control sequence is entered.

7) The fetch cycle is halted due to the previous decode of an AP store instruction (Op Code A-E8) or an attempt to execute a PMU instruction or an AP transfer when an AP transfer is in the APQ.

8) The existance of pipeline status conditions: one or two pending operands that have not been received and one data entry remaining in the pipeline and data present on the input bus, or no pending operands and no data entries remaining in the pipeline. The occurrence of any of these conditions results in the entering of a control sequence determined by a pipeline return indicator.

9) The existance of none or one pending operand instruction interrupt, or trap interrupt indicators without their other satisfiable conditions. A control sequence is entered which performs the necessary pipelining functions so that a satisfiable condition is obtained.

10) A page carry is detected. A transfer to the contents of the program counter is performed.

11) The Halt indicator is set. A control sequence which performs the next function is entered. Exit from this sequence occurs when one of the higher priority next actions occur.

12) A non-scalar or complex operand is decoded when an attempt to load the APQ data part is performed. A sequence is entered which manipulates this type of operand.

13) The DPE is in block mode and the pipeline is empty. Step 1 of the basic two step fetch cycle is entered.

14) The DPE is in Block mode and the pipeline is not empty. A control sequence is entered which mointors the input bus for the next data entry.

15) The APQ data or operand half is full. A control sequence is entered which awaits the absence of the full indication.

16) Step 1 of the basic two step fetch cycle is entered.

4.4 PMU Instruction Definitions

For the purposes of instruction definition, the following designations are used.

D — Represents the effective address or a half word operand after all address modifications and/or operand fetching.

DD — Represents a full word operand after all address modifications and operand fetching.

R — Represents low scratchpad register (SP [0] through SP [15]) determined by the SPA field of instruction word.

(R) — Represents contents of R.

S — Represents low order part of full word register pair when required for full word operations. This register shall be determined as described in 3.3.2.1.

(S) — Represents contents of S.

RS — Represents register R and S treated as a single full word register.

(RS) — Represents contents of RS.

H — Represents high scratchpad register (SP [16] through [31]) determined by SPA field of instruction word.

(H) — Represents contents of H.

E — Represents external subsystem specified by contents of active source register.

M_D — Represents the memory location referenced by the effective address.

(M_D) — Represents contents of M_D.

P — Represents program counter.

(P) — Represents contents of P.

SPA — Field of bits 8 through 11 of instruction word.

OP CODE — Bits 0 through 7 of instruction word, designated in hexadecimal notation.

Wherever, in an instruction definition, the memory is "read", "referenced", written", or "stored", unless otherwise noted, the memory operation is performed in the data addressing mode of the DPE, that is, absolute or virtual as determined by the Set Task Parameter instruction.

4.4.1 PMU Arithmetic Instructions

INSTRUCTION NAME: Add

OP CODE: BO

FUNCTION: (R)+D → R

MACHINE FORMAT:

B				O				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The algebraic sum of the contents of the scratchpad register specified by the SPA field of the instruction and the contents of the memory word location specified by the effective address replaces the contents of the scratchpad register. The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand.

If bit 32 of the instruction word is zero, then the operand is immediate, i.e., the contents of bits 16-31 of the instruction word are added to the contents of the scratchpad register specified by the SPA field of the instruction.

If an overflow occurs, the Overflow Interrupt (Trap No. 5) is enabled.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	271C	0D244A13
CONTENTS AFTER EXECUTION	3440	0D244A13

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	4A13
CONTENTS AFTER EXECUTION	712F	4A13

INSTRUCTION NAME: ADD Full

OP CODE: B4

FUNCTION: (RS)+DD → RS

MACHINE FORMAT:

B				4				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The algebraic sum of the contents of the full word scratchpad register pair specified by the SPA field of the instruction and the contents of the memory word location specified by the effective address replaces the contents of the scratchpad register pair. The contents of memory remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction are added to the contents of the low order portion of the register pair with carry, if necessary, to the high order portion of the register pair.

If an overflow occurs the Overflow Interrupt (Trap No. 5) is enabled.

EXAMPLE 1 Bit 32 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	271C	D351	0D244A13
CONTENTS AFTER EXECUTION	3441	1D64	0D244A13

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	D351	4A13
CONTENTS AFTER EXECUTION	271D	1D64	4A13

INSTRUCTION NAME: Subtract

OP CODE: B3

FUNCTION: (R)-D → R

MACHINE FORMAT:

<table>
<tr><td colspan="4">B</td><td colspan="4">3</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: The algebraic difference between the contents of the scratchpad register specified by the SPA field of the instruction minus the contents of the memory word location specified by the effective address replaces the contents of the scratchpad register. The contents of memory remains unchanged. When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand.

If bit 32 of the instruction word is zero, then the operand is immediate, i.e., the contents of bits 16-31 of the instruction word are subtracted from the contents of the scratchpad register specified by the SPA field of the instruction.

If an overflow occurs, the Overflow Interrupt (Trap No. 5) is enabled.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	271C	119C2108
CONTENTS AFTER EXECUTION	0614	119C2108

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	2108
CONTENTS AFTER EXECUTION	0614	2108

INSTRUCTION NAME: Subtract Full

OP CODE: B7

FUNCTION: (RS)-DD → RS

MACHINE FORMAT:

B				7				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The algebraic difference between the contents of the full word scratchpad register pair specified by the SPA field of the instruction minus the contents of the memory word location specified by the effective address replaces the contents of the scratchpad register pair. The contents of memory remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction are subtracted from the contents of the low order position of the register pair.

If an overflow occurs, the Overflow Interrupt (Trap No. 5) is enabled.

EXAMPLE 1 Bit 32 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	7653	ABCD	0031987A
CONTENTS AFTER EXECUTION	7622	1353	0031987A

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	7653	ABCD	987A
CONTENTS AFTER EXECUTION	7653	1353	987A

INSTRUCTION NAME: Multiply Half to Half

OP CODE: C0

FUNCTION: Dx(R) → R

MACHINE FORMAT:

<table>
<tr><td colspan="4">C</td><td colspan="4">0</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: This multiply instruction utilizes a 16 bit multiplier contained in the scratchpad register specified by the SPA field of the instruction and a 16 bit multiplicand located in the memory word location specified by the effective address of the instruction. Execution of this instruction produces a 16 bit lower order algebraic product in the scratchpad register specified by the SPA field of the instruction. The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of (M_D) are used as the multiplicand, and when bit 31 is one, bits 16-31 of (M_D) are used as the multiplicand.

If bit 32 of the instruction word is zero then the operand is immediate. The contents of bits 16-31 of the instruction are used as the multiplicand.

If an overflow occurs, the Overflow Interrupt (Trap No. 5) is enabled.

EXAMPLE 1   Bit 32 = 1, Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	000A	0011Q0A0
CONTENTS AFTER EXECUTION	0640	001100A0

EXAMPLE 2   Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	000A	000A
CONTENTS AFTER EXECUTION	0064	000A

INSTRUCTION NAME: Multiply Half to Full

OP CODE: C3

FUNCTION:Dx(R) → RS

MACHINE FORMAT:

C				3				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This multiply instruction utilizes a 16 bit multiplier contained in the scratchpad register specified by the SPA field of the instruction and a 16 bit multiplicand located in the memory word location specified by the effective address of the instruction. Execution of this instruction produces a 32 bit algebraic product. The high order portion of the product (bits 0-15) is placed in the scratchpad register specified by the SPA field of the instruction. The low order portion of the product (bits 16-31) is placed in the corresponding scratchpad register of the full word register pair (R+16). The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of (M_D)are used as the multiplicand, and when bit 31 is one, bits 16-31 of (M_D) are used as the multiplicand.

If bit 32 of the instruction word is zero then the operand is immediate. The contents of bits 16-31 of the instruction are used as the multiplicand.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	03E8	XXXX	006457B2
CONTENTS AFTER EXECUTION	0001	86A0	006457B2

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	03E8	XXXX	0064
CONTENTS AFTER EXECUTION	0001	86A0	0064

INSTRUCTION NAME: Divide Half by Half

OP CODE: D3

FUNCTION: (R)÷ D → R, Remainder → S

MACHINE FORMAT:

<table>
<tr><td colspan="4">D</td><td colspan="4">3</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: This divide instruction utilizes a 16 bit dividend contained in the scratchpad register specified by the SPA field of the instruction and a 16 bit divisor located in the memory word location specified by the effective address of the instruction. Execution of this instruction performs an algebraic division and produces a 16 bit quotient in the scratchpad register specified by the SPA field of the instruction. The remainder is stored in the corresponding scratchpad register of the full word pair (R+16) with the sign of the remainder identical to that of the dividend. The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of(M_D) are used as the divisor, and when bit 31 is one, bits 16-31 of(M_D) are used as the divisor.

If bit 32 of the instruction word is zero then the operand is immediate. The contents of bits 16-31 of the instruction are used as the divisor.

If an overflow occurs, the Overflow Interrupt (Trap No. 5) is enabled and the result is not stored. SPR and SPS will remain unaltered.

Overflow occurs if the divisor equals zero.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	800A	XXXX	7A128003
CONTENTS AFTER EXECUTION	0003	8001	7A128003

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	800A	XXXX	0003
CONTENTS AFTER EXECUTION	8003	8001	0003

INSTRUCTION NAME: Divide Full by Half

OP CODE: D2

FUNCTION: (RS) ÷ D → R, Remainder → S

MACHINE FORMAT:

D				2				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This divide instruction utilizes a 32 bit dividend contained in the full word scratchpad register pair specified by the SPA field of the instruction and a 16 bit divisor located in the memory word location specified by the effective address of the instruction. Execution of this instruction performs an algebraic division and produces a 16 bit quotient in the scratchpad register specified by the SPA field of the instruction. The remainder is stored in the corresponding scratchpad register of the full word pair (R+16) with the sign of the remainder identical to that of the dividend. The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of (M_D) are used as the divisor, and when bit 31 is one, bits 16-31 of (M_D) are used as the divisor.

If bit 32 of the instruction word is zero then the operand is immediate. The contents of bits 16-31 of the instruction are used as the divisor.

If an overflow occurs, the Overflow Interrupt (Trap No. 5) is enabled and the result is not stored. SPR and SPS will remain unchanged.

Overflow occurs if the divisor equals zero or if the quotient exceeds the capacity of SPR.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	0001	86A0	006457B2
CONTENTS AFTER EXECUTION	03E8	0000	006457B2

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	8001	86A0	8064
CONTENTS AFTER EXECUTION	03E8	8000	8064

4.4.2 PMU Logical Instructions

INSTRUCTION NAME: AND

OP CODE: AO

FUNCTION: $D \cdot (R) \longrightarrow R$

MACHINE FORMAT:

<table>
<tr><td colspan="4">A</td><td colspan="4">0</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

The contents of the scratchpad register specified by the SPA field of the instruction are logically combined bit by bit, according to the truth table below, with the contents of the memory word location specified by the effective address. If both bits are ONE, the corresponding bit of the scratchpad register is set to ONE. If not, the corresponding bit is cleared to ZERO.

D	0011
(R)	0101
Result	0001

The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction word are logically "anded" to the contents of the scratchpad register specified by the SPA field of the instruction.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	271C	0D244A13
CONTENTS AFTER EXECUTION	0504	0D244A13

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	4A13
CONTENTS AFTER EXECUTION	0210	4A13

INSTRUCTION NAME: AND Full

OP CODE: A4

FUNCTION: DD·(RS) → RS

MACHINE FORMAT:

A				4				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The contents of the full word scratchpad register pair specified by the SPA field of the instruction are logically combined bit by bit, according to the truth table below, with the contents of the memory word location specified by the effective address. If both bits are ONE, the corresponding bit of the scratchpad registers is set to ONE. If not, the corresponding bit is cleared to ZERO.

DD	0011
· (RS)	0101
Result	0001

The contents of memory remain unchanged. If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction are logically "anded" to the contents of the lower order portion of the register pair. The contents of the higher order portion of the register pair are cleared to ZERO.

EXAMPLE 1 Bit 32 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	271C	58A2	0D244A13
CONTENTS AFTER EXECUTION	0504	4802	0D244A13

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	58A2	4A13
CONTENTS AFTER EXECUTION	0000	4802	4A13

INSTRUCTION NAME: $\bar{D}$ AND R

OP CODE: 90

FUNCTION: $\bar{D}\cdot(R) \longrightarrow R$

MACHINE FORMAT:

<table>
<tr><td colspan="4">9</td><td colspan="4">0</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

The contents of the scratchpad register specified by the SPA field of the instruction are logically combined bit by bit with the logical complement of the contents of the memory word location specified by the effective address, according to the truth table below. If the complemented bit of D is ONE and the bit of (R) is ONE, the corresponding bit of R is set to ONE. If not, the corresponding bit of R is cleared to ZERO.

D	0011
(R)	0101
Result	0100

The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction word are logically combined, as above, to the contents of the scratchpad register specified by the SPA field of the instruction.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	271C	0D244A13
CONTENTS AFTER EXECUTION	250C	0D244A13

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	4A13
CONTENTS AFTER EXECUTION	250C	4A13

INSTRUCTION NAME: $\bar{D}$ And R Full

OP CODE: 94

FUNCTION: $\overline{DD} \cdot (RS) \longrightarrow RS$

MACHINE FORMAT:

9				4				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The contents of the full word scratchpad register pair specified by the SPA field of the instruction are logically combined bit by bit with the logical complement of the contents of the memory word location specified by the effective address, according to the truth table below. If the complemented bit of DD is ONE and the bit of (RS) is ONE, the corresponding bit of (RS) is set to ONE. If not, the corresponding bit of (RS) is cleared to ZERO.

DD	0011
(RS)	0101
Result	0100

The contents of memory remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction are logically combined, as above, to the contents of the lower order portion of the register pair. The contents of the higher order portion of the register pair remain unchanged.

EXAMPLE 1   Bit 32 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	271C	58A2	0D244A13
CONTENTS AFTER EXECUTION	2218	10A0	0D244A13

EXAMPLE 2   Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	58A2	4A13
CONTENTS AFTER EXECUTION	271C	10A0	4A13

INSTRUCTION NAME: D And $\overline{R}$

OP CODE: C2

FUNCTION: $D \cdot (\overline{R}) \longrightarrow R$

MACHINE FORMAT:

<table>
<tr><td colspan="4">C</td><td colspan="4">2</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

The logical complement of the contents of the scratchpad register specified by the SPA field of the instruction are logically combined bit by bit with the contents of the memory word location specified by the effective address, according to the truth table below. If the complemented bit of (R) is ONE and the bit of D is ONE, the corresponding bit of R is set to ONE. If not, the corresponding bit of R . is cleared to ZERO.

D	0011
(R)	<u>0101</u>
Result	0010

The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction word are logically combined, as above, to the contents of the scratchpad register specified by the SPA field of the instruction.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	271C	0D244A13
CONTENTS AFTER EXECUTION	4803	0D244A13

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	4A13
CONTENTS AFTER EXECUTION	4803	4A13

INSTRUCTION NAME: D And $\overline{R}$ Full

OP CODE: C6

FUNCTION: $DD \cdot (\overline{RS}) \longrightarrow RS$

MACHINE FORMAT:

C				6				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The logical complement of the contents of the full word scratchpad register pair specified by the SPA field of the instruction are logically combined bit by bit with the contents of the memory word location specified by the effective address, according to the truth table below. If the complemented bit of (RS) is ONE and the bit of D is ONE, the corresponding bit of RS is set to ONE. If not, the corresponding bit of RS is cleared to zero.

DD	0011
(RS)	0101
Result	0010

The contents of memory remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction are logically combined, as above, to the contents of the lower order portion of the register pair. The contents of the higher order portion of the register pair are cleared to zeros.

EXAMPLE 1 Bit 32 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	271C	58A2	0D244A13
CONTENTS AFTER EXECUTION	0820	0211	0D244A13

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	58A2	4A13
CONTENTS AFTER EXECUTION	0000	0211	4A13

INSTRUCTION NAME: OR

OP CODE: B2

FUNCTION: $D \vee (R) \longrightarrow R$

MACHINE FORMAT:

<table>
<tr><td colspan="4">B</td><td colspan="4">2</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

The contents of the scratchpad register specified by the SPA field of the instruction are logically combined bit by bit, according to the truth table below, with the contents of the memory word location specified by the effective address. If both bits are ZERO, the corresponding bit of R is cleared to zero. If not, the corresponding bit of R is set to ONE.

D	0011
(R)	0101
Result	0111

The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction word are logically combined, as above, with the contents of the scratchpad register specified by the SPA field of the instruction.

EXAMPLE 1   Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	271C.	0D244A13
CONTENTS AFTER EXECUTION	2F3C	0D244A13

EXAMPLE 2   Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	4A13
CONTENTS AFTER EXECUTION	6F1F	4A13

INSTRUCTION NAME: OR Full

OP CODE: B6

FUNCTION: DD∨(RS) ⟶ RS

MACHINE FORMAT:

<table>
<tr><td colspan="4">B</td><td colspan="4">6</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

The contents of the full word scratchpad register pair specified by the SPA field of the instruction are logically combined bit by bit, according to the truth table below, with the contents of the memory word location specified by the effective address. If both bits are ZERO, the corresponding bit of RS is cleared to ZERO. If not, the corresponding bits of RS are set to ONE.

DD	0011
(RS)	0101
Result	0111

The contents of memory remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction are logically combined, as above, with the contents of the lower order portion of the register pair. The contents of the higher order portion remain unchanged.

EXAMPLE 1 Bit 32 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	271C	58A2	0D244A13
CONTENTS AFTER EXECUTION	2F3C	5AB3	0D244A13

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	58A2	4A13
CONTENTS AFTER EXECUTION	271C	5AB3	4A13

INSTRUCTION NAME: $\bar{D}$ OR R

OP CODE: B1

FUNCTION: $\bar{D} \vee (R) \longrightarrow R$

MACHINE FORMAT:

B				1				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The contents of the scratchpad register specified by the SPA field of the instruction are logically combined bit by bit with the logical complement of the contents of the memory word location specified by the effective address, according to the truth table below. If the complemented bit of D is ZERO and the bit of (R) is ZERO, the corresponding bit of R is cleared to ZERO. If not, the corresponding bit of R is set to ONE.

D	0011
(R)	0101
Result	1101

The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction word are logically combined, as above, to the contents of the scratchpad register specified by the SPA field of the instruction.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	271C	0D244A13
CONTENTS AFTER EXECUTION	B7FC	0D244A13

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	4A13
CONTENTS AFTER EXECUTION	B7FC	4A13

INSTRUCTION NAME: $\bar{D}$ OR R Full

OP CODE: B5

FUNCTION: $D\bar{D}\vee(RS) \longrightarrow RS$

MACHINE FORMAT:

<table>
<tr><td colspan="4">B</td><td colspan="4">5</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

The contents of the full word scratchpad register pair specified by the SPA field of the instruction are logically combined bit by bit with the logical complement of the contents of the memory word location specified by the effective address, according to the truth table below. If the complemented bit of DD is ZERO and the bit of (RS) is ZERO, the corresponding bit of RS is cleared to ZERO. If not, the corresponding bit of RS is set to ONE.

DD	0011
(RS)	0101
Result	1101

The contents of memory remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction word are logically combined, as above, to the contents of the lower order portion of the register pair. The contents of the higher order portion of the register pair are set to ones.

EXAMPLE 1 Bit 32 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	271C	58A2	0D244A13
CONTENTS AFTER EXECUTION	F7DF	FDEF	0D244A13

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	58A2	4A13
CONTENTS AFTER EXECUTION	FFFF	FDEF	4A13

INSTRUCTION NAME: D OR $\overline{R}$

OP CODE: A3

FUNCTION: $D \vee (\overline{R}) \longrightarrow R$.

MACHINE FORMAT:

A				3				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The logical complement of the contents of the scratchpad register specified by the SPA field of the instruction are logically combined bit by bit with the contents of the memory word location specified by the effective address, according to the truth table below. If the complemented bit of (R) is ZERO and the bit of D is ZERO, the corresponding bit of R is cleared to ZERO. If not, the corresponding bit of R is set to ONE.

D	0011
(R)	0101
Result	1011

The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand.

If bit 32 of the instruction word is zero, the the operand is immediate. The contents of bits 16-31 of the instruction word are logically combined, as above, to the contents of the scratchpad register specified by the SPA field of the instruction.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	271C	0D244A13
CONTENTS AFTER EXECUTION	DAF3	0D244A13

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	4A13
CONTENTS AFTER EXECUTION	DAF3	4A13

INSTRUCTION NAME: D OR $\overline{R}$ Full

OP CODE: A7

FUNCTION: $DD \vee (\overline{RS}) \longrightarrow RS$

MACHINE FORMAT:

A				7				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The logical complement of the contents of the full word scratchpad register pair specified by the SPA field of the instruction are logically combined bit by bit with the contents of the memory word location specified by the effective address, according to the truth table below. If the complemented bit of (RS) is ZERO and the bit of DD is ZERO, the corresponding bit of RS is cleared to ZERO. If not, the corresponding bit of RS is set to ONE.

DD	0011
(RS)	0101
Result	1011

The contents of memory remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction are logically combined, as above, to the contents of the lower order portion of the register pair. The contents of the higher order portion of the register pair are logically complemented.

EXAMPLE 1   Bit 32 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	271C	58A2	0D244A13
CONTENTS AFTER EXECUTION	DDE7	EF7F	0D244A13

EXAMPLE 2   Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	58A2	4A13
CONTENTS AFTER EXECUTION	D8E3	EF5F	4A13

INSTRUCTION NAME: NAND

OP CODE: 93

FUNCTION: $\bar{D}\vee\bar{R}) \longrightarrow R$

MACHINE FORMAT:

<table>
<tr><td colspan="4">9</td><td colspan="4">3</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

The logical complement of the contents of the scratchpad register specified by the SPA field of the instruction are logically combined bit by bit with the logical complement of the contents of the memory word location specified by the effective address, according to the truth table below. If the complemented bits of both D and (R) are ZERO, the corresponding bit of R is cleared to ZERO. If not, the corresponding bit of R is set to ONE.

D	0011
(R)	0101
Result	1110

The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction word are logically combined, as above, to the contents of the scratchpad register specified by the SPA field of the instruction.

EXAMPLE 1   Bit 32 = 1, Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	271C	0D244A13
CONTENTS AFTER EXECUTION	FDEF	0D244A13

EXAMPLE 2   Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	4A13
CONTENTS AFTER EXECUTION	FDEF	4A13

INSTRUCTION NAME: NAND Full

OP CODE: 97

FUNCTION: $\overline{DD} \vee (\overline{RS}) \longrightarrow$ RS

MACHINE FORMAT:

9				7				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The logical complement of the contents of the full word scratchpad register pair specified by the SPA field of the instruction are logically combined bit by bit with the logical complement of the contents of the memory word location specified by the effective address, according to the truth table below. If the complemented bits of both DD and $(\overline{RS})$ are zero the corresponding bit of RS is cleared to ZERO. If not, the corresponding bit of RS is set to ONE.

DD	0011
(RS)	0101
Result	1110

The contents of memory remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction are logically combined, as above, to the contents of the lower order portion of the register pair. The contents of the higher order portion of the register pair are set to ones.

EXAMPLE 1   Bit 32 = 1

	(RS)		
	(R)	(S).	(M_D)
CONTENTS BEFORE EXECUTION	271C	58A2	0D244A13
CONTENTS AFTER EXECUTION	FAFB	B7FD	0D244A13

EXAMPLE 2   Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	58A2	4A13
CONTENTS AFTER EXECUTION	FFFF	B7FD	4A13

INSTRUCTION NAME: NOR

OP CODE: C1

FUNCTION: $\overline{D}\cdot(\overline{R}) \longrightarrow R$

MACHINE FORMAT:

C				1				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The logical complement of the contents of the scratchpad register specified by the SPA field of the instruction are logically combined bit by bit with the logical complement of the contents of the memory word location specified by the effective address, according to the truth table below. If the complemented bits of both D and (R) are ONE, the corresponding bit of R is set to ONE. If not, the corresponding bit of R is cleared to ZERO.

D	0011
R	0101
Result	1000

The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of M_D are used as the operand, and when bit 31 is one, bits 16-31 of M_D are used as the operand.

If bit 32 of the instruction word is zero then the operand is immediate. The contents of bits 16-31 of the instruction word are logically combined, as above, to the contents of the scratchpad register specified by the SPA field of the instruction.

EXAMPLE 1   Bit 32 = 1; Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	271C	0D244A13
CONTENTS AFTER EXECUTION	90E0	0D244A13

EXAMPLE 2   Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	4A13
CONTENTS AFTER EXECUTION	90E0	4A13

INSTRUCTION NAME: NOR Full

OP CODE: C5

FUNCTION: $\overline{DD}\cdot(\overline{RS}) \longrightarrow$ RS

MACHINE FORMAT:

<table>
<tr><td colspan="4">C</td><td colspan="4">5</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

The logical complement of the contents of the full word scratchpad register pair specified by the SPA field of the instruction are logically combined bit by bit with the logical complement of the contents of the memory word location specified by the effective address, according to the truth table below. If the complemented bits of both DD and (RS) are ONE, the corresponding bit of RS is set to ONE. If not, the corresponding bit of RS is cleared to ZERO.

DD	0011
(RS)	0101
Result	1000

The contents of memory remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction are logically combined, as above, to the contents of the lower order portion of the register pair. The contents of the higher order portion of the register pair are complemented.

EXAMPLE 1 Bit 32 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	271C	58A2	0D244A13
CONTENTS AFTER EXECUTION	D0C3	A54C	0D244A13

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	58A2	4A13
CONTENTS AFTER EXECUTION	D8E3	A54C	4A13

INSTRUCTION NAME: XOR

OP CODE: 92

FUNCTION: $D \oplus (R) \longrightarrow R$

MACHINE FORMAT:

9				2				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The contents of the scratchpad register specified by the SPA field of the instruction are logically combined bit by bit with the contents of the memory word location specified by the effective address, according to the truth table below. If both bits are ONE, or if both bits are ZERO, the corresponding bit of R is cleared to ZERO. If not, the corresponding bit of R is set to ONE.

D	0011
(R)	0101
Result	0110

The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of M_D are used as the operand, and when bit 31 is one, bits 16-31 are used as the operand.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction word are logically combined, as above, to the contents of the scratchpad register specified by the SPA field of the instruction.

EXAMPLE 1   Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	271C	0D244A13
CONTENTS AFTER EXECUTION	2A38	0D244A13

EXAMPLE 2   Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	4A13
CONTENTS AFTER EXECUTION	6D0F	4A13

INSTRUCTION NAME: XOR Full

OP CODE: 96

FUNCTION: DD⊕(RS) → RS

MACHINE FORMAT:

9				6				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The contents of the full word scratchpad register pair specified by the SPA field of the instruction are logically combined bit by bit with the contents of the memory word location specified by the effective address, according to the truth table below. If both bits are ONE, or if both bits are ZERO, the corresponding bit of RS is cleared to ZERO. If not, the corresponding bit of RS is set to ONE.

DD	0011
(RS)	0101
Result	0110

The contents of memory remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction are logically combined, as above, with the contents of the lower order portion of the register pair. The contents of the higher order portion remain unchanged.

EXAMPLE 1 Bit 32 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	271C	58A2	0D244A13
CONTENTS AFTER EXECUTION	2A38	12B1	0D244A13

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	58A2	4A13
CONTENTS AFTER EXECUTION	271C	12B1	4A13

INSTRUCTION NAME: XNOR

OP CODE: A1

FUNCTION: $D \overline{\oplus} (R) \longrightarrow R$

MACHINE FORMAT:

<table>
<tr><td colspan="4">A</td><td colspan="4">1</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

The contents of the scratchpad register specified by the SPA field of the instruction are logically combined bit by bit with the contents of the memory word location specified by the effective address, according to the truth table below. If both bits are ONE, or if both bits are ZERO, the corresponding bit of R is set to ONE. If not, the corresponding bit of R is cleared to zero.

D	0011
(R)	0101
Result	1001

The contents of memory remain unchanged. When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction word are logically combined, as above, to the contents of the scratchpad register specified by the SPA field of the instruction.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	271C	0D244A13
CONTENTS AFTER EXECUTION	D5C7	0D244A13

EXAMPLE2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	4A13
CONTENTS AFTER EXECUTION	92F0	4A13

INSTRUCTION NAME: XNOR Full

OP CODE: A5

FUNCTION: $DD\overline{\oplus}(RS) \longrightarrow RS$

MACHINE FORMAT:

A				5				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The contents of the full word scratchpad register pair specified by the SPA field of the instruction are logically combined bit by bit with the contents of the memory word location specified by the effective address, according to the truth table below. If both bits are ONE, or if both bits are ZERO, the corresponding bit of SPRS is set to ONE. If not, the corresponding bit of SPRS is cleared to ZERO.

DD	0011
(RS)	0101
Result	1001

The contents of memory remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate. The contents of bits 16-31 of the instruction word are logically combined, as above with the contents of the lower order portion of the register pair. The contents of the higher order portion are complemented.

EXAMPLE 1   Bit 32 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	271C	58A2	0D244A13
CONTENTS AFTER EXECUTION	D5C7	ED4E	0D244A13

EXAMPLE 2   Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	271C	58A2	4A13
CONTENTS AFTER EXECUTION	D8E3	ED4E	4A13

4.4.3 PMU Shift Instructions

INSTRUCTION NAME: Shift ARHO

OP CODE: E0

FUNCTION: $(R)x2^{-N} \longrightarrow R$

MACHINE FORMAT:

E				0				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

Bits 1-15 of the scratchpad register specified by the SPA field of the instruction are algebraically shifted right N places, where N is determined by bits 8-15 of the operand specified by the effective address. The result is placed in the specified scratchpad register. Bits shifted beyond bit position 15 are lost. Vacated bit positions are filled by zeroes. The contents of memory remain unchanged. N is treated as an integer value.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand (N = bits 8-15), and when bit 31 is one, bits 16-31 of (M_D) are used as the operand (N = bits 24-31).

If bit 32 of the instruction word is zero, then the operand is immediate, and N is determined by bit positions 24-31 of the instruction word.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	E3D7	5103 F202
CONTENTS AFTER EXECUTION	8C7A	5103 F202

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	E3D7	F202
CONTENTS AFTER EXECUTION	98F5	F202

INSTRUCTION NAME: Shift ARFO

OP CODE: E1

FUNCTION: $(RS)x2^{-N} \longrightarrow RS$

MACHINE FORMAT:

E				1				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

Bits 1-31 of the full word scratchpad register pair specified by the SPA field of the instruction are algebraically shifted right N places, where N is determined by bits 8-15 of the operand specified by the effective address. The result is placed in the specified register pair. Bits shifted beyond bit position 31 are lost. Vacated bit positions are filled by zeroes. The contents of memory remain unchanged. N is treated as an integer value.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand (N = bits 8-15), and when bit 31 is one, bits 16-31 of (M_D) are used as the operand (N = bits 24-31).

If bit 32 of the instruction word is zero, then the operand is immediate and N is determined by bit positions 24-31 of the instruction word.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	E3D7	0A42	5103 F202
CONTENTS AFTER EXECUTION	98F5	C290	5103 F202

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	E3D7	0A42	5103
CONTENTS AFTER EXECUTION	8C74	E148	5103

INSTRUCTION NAME: Shift ALHO

OP CODE: E2

FUNCTION: $(R)x2^N \longrightarrow R$

MACHINE FORMAT:

<table>
<tr><td colspan="4">E</td><td colspan="4">2</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

Bits 1-15 of the scratchpad register specified by the SPA field of the instruction are algebraically shifted left N places, where N is determined by bits 8-15 of the operand specified by the effective address. The result is placed in the specified scratchpad register. Bits shifted beyond bit position 1 are lost. Vacated positions are filled by zeroes. The contents of memory remain unchanged. N is treated as an integer value.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand (N = bits 8-15), and when bit 31 is one, bits 16-31 of (M_D) are used as the operand (N = bits 24-31).

If bit 32 of the instruction word is zero, then the operand is immediate, and N is determined by bit positions 24-31 of the instruction word.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	E3D7	5103 F202
CONTENTS AFTER EXECUTION	9EB8	5103 F202

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	E3D7	F202
CONTENTS AFTER EXECUTION	8F5C	F202

INSTRUCTION NAME: Shift ALFO

OP CODE: E3

FUNCTION: $(RS)x2^{N} \longrightarrow RS$

MACHINE FORMAT:

E				3				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

Bits 1-31 of the full word scratchpad register pair specified by the SPA field of the instruction are algebraically shifted left N places, where N is determined by bits 8-15 of the operand specified by the effective address. The result is placed in the specified register pair. Bits shifted beyond bit position 1 are lost. Vacated bit positions are filled by zeroes. The contents of memory remain unchanged. N is treated as an integer value.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand (N = bits 8-15), and when bit 31 is one, bits 16-31 of (M_D) are used as the operand (N = bits 24-31).

If bit 32 of the instruction word is zero, then the operand is immediate and N is determined by bit positions 24-31 of the instruction word.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 1

	(RS)			
	(R)	(S)	(M_D)	
CONTENTS BEFORE EXECUTION	E3D7	0A42	5103	F202
CONTENTS AFTER EXECUTION	8F5C	2908	5103	F202

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	E3D7	0A42	5103
CONTENTS AFTER EXECUTION	9EB8	5210	5103

INSTRUCTION NAME: Shift LRHO

OP CODE: F0

FUNCTION: (R) x $2^{-N} \longrightarrow$ R

MACHINE FORMAT:

<table>
<tr><td colspan="4">F</td><td colspan="4">0</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

The contents of the scratchpad register specified by the SPA field of the instruction are logically shifted right N places, where N is determined by bits 8-15 of the operand specified by the effective address. The result is placed in the specified scratchpad register. Bits shifted beyond bit position 15 are lost. Vacated positions are filled with zeroes. The contents of memory remain unchanged. N is treated as an integer value.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand (N = bits 8-15), and when bit 31 is one, bits 16-31 of (M_D) are used as the operand (N = bits 24-31).

If bit 32 of the instruction word is zero, then the operand is immediate, and N is determined by bit positions 24-31 of the instruction word.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	E3D7	5103 F202
CONTENTS AFTER EXECUTION	1C7A	5103 F202

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	E3D7	F202
CONTENTS AFTER EXECUTION	38F5	F202

INSTRUCTION NAME: Shift LRFO

OP CODE: F1

FUNCTION: (RS) x 2^{-N} → RS

MACHINE FORMAT:

F	1	SPA	I	INDEX	ADDRESS	M	O	T	P
0 1 2 3	4 5 6 7	8 9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

DEFINITION:

The contents of the full word scratchpad register pair specified by the SPA field of the instruction are logically shifted right N places, where N is determined by bits 8-15 of the operand specified by the effective address. The result is placed in the specified register pair. Bits shifted beyond bit position 31 are lost. Vacated bit positions are filled by zeroes. The contents of memory remain unchanged. N is treated as an integer value.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand (N = bits 8-15), and when bit 31 is one, bits 16-31 of (M_D) are used as the operand (N = bits 24-31).

If bit 32 of the instruction word is zero, then the operand is immediate and N is determined by bit positions 24-31 of the instruction word.

EXAMPLE 1   Bit 32 = 1, Bit 31 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	E3D7	0A42	5103 F202
CONTENTS AFTER EXECUTION	38F5	C 290	5103 F202

EXAMPLE 2   Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	E3D7	0A42	5103
CONTENTS AFTER EXECUTION	1C7A	E148	5103

INSTRUCTION NAME: Shift LLHO

OP CODE: F2

FUNCTION: (R) x $2^N \longrightarrow$ R

MACHINE FORMAT:

F				2				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The contents of the scratchpad register specified by the SPA field of the instruction are logically shifted left N places, where N is determined by bits 8-15 of the operand specified by the effective address. The result is placed in the specified scratchpad register. Bits shifted beyond bit position 0 are lost. Vacated positions are filled with zeroes. The contents of memory remain unchanged. N is treated as an integer value.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand (N = bits 8-15), and when bit 31 is one, bits 16-31 of (M_D) are used as the operand (N = bits 24-31).

If bit 32 of the instruction word is zero, then the operand is immediate, and N is determined by bit positions 24-31 of the instruction word.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	E3D7	5103 F202
CONTENTS AFTER EXECUTION	1EB8	5103 F202

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	E3D7	F202
CONTENTS AFTER EXECUTION	8F5C	F202

INSTRUCTION NAME: Shift LLFO

OP CODE: F3

FUNCTION: (RS) x $2^N \longrightarrow$ RS

MACHINE FORMAT:

<table>
<tr><td colspan="4">F</td><td colspan="4">3</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

The contents of the full word scratchpad register pair specified by the SPA field of the instruction are logically shifted left N places, where N is determined by bits 8-15 of the operand specified by the effective address. The result is placed in the specified register pair. Bits shifted beyond bit position 0 are lost. Vacated bit positions are filled by zeroes. The contents of memory remain unchanged. N is treated as an integer value.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand (N = bits 8-15), and when bit 31 is one, bits 16-31 of (M_D) are used as the operand (N = bits 24-31).

If bit 32 of the instruction word is zero, then the operand is immediate and N is determined by bit positions 24-31 of the instruction word.

EXAMPLE 1   Bit 32 = 1, Bit 31 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	E3D7	0A42	5103 F202
CONTENTS AFTER EXECUTION	8F5C	2908	5103 F202

EXAMPLE 2   Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	E3D7	0A42	5103
CONTENTS AFTER EXECUTION	1EB8	5210	5103

INSTRUCTION NAME: Shift LRHC

OP CODE: D0

FUNCTION: (R) x $2^{-N} \longrightarrow R$

MACHINE FORMAT:

D				0				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The contents of the scratchpad register specified by the SPA field of the instruction are logically shifted right N places, where N is determined by bits 8-15 of the operand specified by the effective address. Bits shifted beyond bit position 15 are used to fill corresponding vacated positions. The contents of memory remain unchanged. N is treated as an integer value. The result is placed in the specified scratchpad register.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand (N = bits 8-15), and when bit 31 is one, bits 16-31 of (M_D) are used as the operand (N = bits 24-31).

If bit 32 of the instruction word is zero, then the operand is immediate, and N is determined by bit positions 24-31 of the instruction word.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	E3D7	5103 F202
CONTENTS AFTER EXECUTION	FC7A	5103 F202

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	E3D7	F202
CONTENTS AFTER EXECUTION	F8F5	F202

INSTRUCTION NAME: Shift LRFC

OP CODE: D1

FUNCTION: (RS) x $2^{-N} \longrightarrow$ RS

MACHINE FORMAT:

<table>
<tr><td colspan="4">D</td><td colspan="4">1</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

The contents of the full word scratchpad register pair specified by the SPA field of the instruction are logically shifted right N places, where N is determined by bits 8-15 of the operand specified by the effective address. Bits shifted beyond bit position 31 are used to fill corresponding vacated positions. The contents of memory remain unchanged. N is treated as an integer value. The result is placed in the specified scratchpad register.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand (N = bits 8-15), and when bit 31 is one, bits 16-31 of (M_D) are used as the operand (N = bits 24-31).

If bit 32 of the instruction word is zero, then the operand is immediate, and N is determined by bit positions 24-31 of the instruction word.

EXAMPLE 1   Bit 32 = 1, Bit 31 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	E3D7	0A42	5103 F202
CONTENTS AFTER EXECUTION	B8F5	C290	5103 F202

EXAMPLE 2   Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	E3D7	0A42	5103
CONTENTS AFTER EXECUTION	5C7A	E148	5103

4.4.4 PMU Skip Instructions

INSTRUCTION NAME: Skip If Equal To

OP CODE: 62

FUNCTION: If D =(R), (P)+1 → P

MACHINE FORMAT:

6				2				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

If the contents of the memory word location specified by the effective address are equal to the contents of the scratchpad register specified by the SPA field of the instruction, the next instruction in sequence is skipped; otherwise, the next instruction in sequence is executed. The contents of memory and the scratchpad register remain unchanged.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand. If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)	(P)
CONTENTS BEFORE EXECUTION	15A3	029A15A3	0 100
CONTENTS AFTER EXECUTION	15A3	029A15A3	0101

EXAMPLE 2 Bit 32 - 0

	(R)	BITS 16-31 OF INSTRUCTION	(P)
CONTENTS BEFORE EXECUTION	15A3	15A3	0100
CONTENTS AFTER EXECUTION	15A3	15A3	0102

INSTRUCTION NAME: Skip If Equal To Full

OP CODE: 66

FUNCTION: If DD=(RS), (P)+1 → P

MACHINE FORMAT:

6				6				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

If the contents of the memory word location specified by the effective address are equal to the contents of the full word scratchpad register pair specified by the SPA field of the instruction, the next instruction in sequence is skipped; otherwise, the next instruction in sequence is executed. The contents of memory and the scratchpad register pair remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1

	(RS)			
	(R)	(S)	(M_D)	(P)
CONTENTS BEFORE EXECUTION	00A3	2100	00A32100	0100
CONTENTS AFTER EXECUTION	00A3	2100	00A32100	0102

EXAMPLE 2 Bit 32 = 0

	(RS)			
	(R)	(S)	BITS 16-31 OF INSTRUCTION	(P)
CONTENTS BEFORE EXECUTION	00A3	2100	2100	0100
CONTENTS AFTER EXECUTION	00A3	2100	2100	0101

INSTRUCTION NAME: Skip If Not Equal To

OP CODE: 71

FUNCTION: If D $\neq$(R), (P)+1 $\longrightarrow$ P

MACHINE FORMAT:

7				1				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

If the contents of the memory word location specified by the effective address are not equal to the contents of the scratchpad register specified by the SPA field of the instruction, the next instruction in sequence is skipped; otherwise, the next instruction in sequence is executed. The contents of memory and the scratchpad register remain unchanged.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand. If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 1

	(R)	(M_D)	(P)
CONTENTS BEFORE EXECUTION	15A3	029A 15A3	0200
CONTENTS AFTER EXECUTION	15A3	029A 15A3	0201

EXAMPLE 2   Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION	(P)
CONTENTS BEFORE EXECUTION	15A3	029A	0200
CONTENTS AFTER EXECUTION	15A3	029A	0202

INSTRUCTION NAME: Skip If Not Equal To Full

OP CODE: 75

FUNCTION: IfDD ≠ (RS), (P)+1 → P

MACHINE FORMAT:

7				5				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

If the contents of the memory word location specified by the effective address are not equal to the contents of the full word scratchpad register pair specified by the SPA field of the instruction, the next instruction in sequence is skipped; otherwise, the next instruction in sequence is executed. The contents of memory and the scratchpad register pair remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1

	(RS)			
	(R)	(S)	(M_D)	(P)
CONTENTS BEFORE EXECUTION	0012	3456	00133456	0200
CONTENTS AFTER EXECUTION	0012	3456	00133456	0202

EXAMPLE 2 Bit 32 = 0

	(RS)			
	(R)	(S)	BITS 16-31 OF INSTRUCTION	(P)
CONTENTS BEFORE EXECUTION	0000	57A2	57A1	0200
CONTENTS AFTER EXECUTION	0000	57A2	57A1	0202

INSTRUCTION NAME: Skip If Greater Than

OP CODE: 61

FUNCTION: If $D>(R)$, $(P)+1 \longrightarrow P$

MACHINE FORMAT:

<table>
<tr><td colspan="4">6</td><td colspan="4">1</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

If the contents of the memory word location specified by the effective address are greater than the contents of the scratchpad register specified by the SPA field of the instruction, the next instruction in sequence is skipped; otherwise, the next instruction in sequence is executed. The contents of memory and the scratchpad register remains unchanged.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand. If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 1

	(R)	(M_D)	(P)
CONTENTS BEFORE EXECUTION	4321	54321098	0200
CONTENTS AFTER EXECUTION	4321	54321098	0201

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION	(P)
CONTENTS BEFORE EXECUTION	4321	ABCD	0200
CONTENTS AFTER EXECUTION	4321	ABCD	0202

INSTRUCTION NAME: Skip If Greater Than Full

OP CODE: 65

FUNCTION: IfDD>(RS), (P)+1 → P

MACHINE FORMAT:

6				5				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

If the contents of the memory word location specified by the effective address are greater than the contents of the full word scratchpad register pair specified by the SPA field of the instruction, the next instruction in sequence is skipped; otherwise, the next instruction in sequence is executed. The contents of memory and the scratchpad register pair remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1

	(RS)			
	(R)	(S)	(M_D)	(P)
CONTENTS BEFORE EXECUTION	1234	5678	12348765	0200
CONTENTS AFTER EXECUTION	1234	5678	12348765	0202

EXAMPLE 2   Bit 32 = 0

	(RS)			
	(R)	(S)	BITS 16-31 OF INSTRUCTION	(P)
CONTENTS BEFORE EXECUTION	1234	5678	8765	0200
CONTENTS AFTER EXECUTION	1234	5678	8765	0201

INSTRUCTION NAME: Skip If Not Greater

OP CODE: 72

FUNCTION: If $D \leq (R)$, $(P)+1 \longrightarrow P$

MACHINE FORMAT:

7				2				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

If the contents of the memory word location specified by the effective address are not greater than the contents of the scratchpad register specified by the SPA field of the instruction, the next instruction sequence is skipped; otherwise, the next instruction in sequence is executed. The contents of memory and the scratchpad register remain unchanged.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand. If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)	(P)
CONTENTS BEFORE EXECUTION	3456	3446DFEG	0200
CONTENTS AFTER EXECUTION	3456	3446DFEG	0202

EXAMPLE 2   Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION	(P)
CONTENTS BEFORE EXECUTION	3456	4563	0200
CONTENTS AFTER EXECUTION	3456	4563	0201

INSTRUCTION NAME: Skip If Not Greater Than Full

OP CODE: 76

FUNCTION: IfDD ≤(RS), (P)+1 ⟶ P

MACHINE FORMAT:

<table>
<tr><td colspan="4">7</td><td colspan="4">6</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

If the contents of the memory word location specified by the effective address are not greater than the contents of the full word scratchpad register pair specified by the SPA field of the instruction, the next instruction in sequence is skipped; otherwise, the next instruction in sequence is executed. The contents of memory and the scratchpad register pair remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1

	(RS)			
	(R)	(S)	(M_D)	(P)
CONTENTS BEFORE EXECUTION	1234	5678	12345678	0200
CONTENTS AFTER EXECUTION	1234	5678	12345678	0202

EXAMPLE 2   Bit 32 = 0

	(RS)			
	(R)	(S)	BITS 16-31 OF INSTRUCTION	(P)
CONTENTS BEFORE EXECUTION	1234	5678	5678	0200
CONTENTS AFTER EXECUTION	1234	5678	5678	0202

INSTRUCTION NAME: Skip If Less Than

OP CODE: 70

FUNCTION: If $D<(R)$, $(P)+1 \longrightarrow P$

MACHINE FORMAT:

7				0				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

If the contents of the memory word location specified by the effective address are less than the contents of the scratchpad register specified by the SPA field of the instruction, the next instruction in sequence is skipped; otherwise, the next instruction in sequence is executed. The contents of memory and the scratchpad register remain unchanged.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand. If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE1 Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)	(P)
CONTENTS BEFORE EXECUTION	1234	12345678	0200
CONTENTS AFTER EXECUTION	1234	12345678	0201

EXAMPLE 2   Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION	(P)
CONTENTS BEFORE EXECUTION	1234	0123	0200
CONTENTS AFTER EXECUTION	1234	0123	0202

INSTRUCTION NAME: Skip If Less Than Full

OP CODE: 74

FUNCTION: If DD< (RS), (P)+1 → P

MACHINE FORMAT:

7				4				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

If the contents of the memory word location specified by the effective address are less than the contents of the full word scratchpad register pair specified by the SPA field of the instruction, the next instruction in sequence is skipped; otherwise, the next instruction in sequence is executed. The contents of memory and the scratchpad register pair remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1

	(RS)			
	(R)	(S)	(M_D)	(P)
CONTENTS BEFORE EXECUTION	1234	5678	12348765	0200
CONTENTS AFTER EXECUTION	1234	5678	12348765	0201

EXAMPLE 2 Bit 32 = 0

	(RS)			
	(R)	(S)	BITS 16-31 OF INSTRUCTION	(P)
CONTENTS BEFORE EXECUTION	1234	5678	1234	0200
CONTENTS AFTER EXECUTION	1234	5678	1234	0202

INSTRUCTION NAME: Skip If Not Less Than

OP CODE: 63

FUNCTION: If $D \geq (R)$, $(P)+1 \longrightarrow P$

MACHINE FORMAT:

6				3				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

If the contents of the memory word location specified by the effective address are not less than the contents of the scratchpad register specified by the SPA field of the instruction, the next instruction in sequence is skipped; otherwise, the next instruction in sequence is executed. The contents of memory and the scratchpad register remain unchanged.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when 31 is one, bits 16-31 of (M_D) are used as the operand. If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)	P
CONTENTS BEFORE EXECUTION	1234	12345678	0200
CONTENTS AFTER EXECUTION	1234	12345678	0202

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION	(P)
CONTENTS BEFORE EXECUTION	1234	0123	0200
CONTENTS AFTER EXECUTION	1234	0123	0201

INSTRUCTION NAME: Skip If Not Less Than Full

OP CODE: 67

FUNCTION: IfDD≥(RS), (P)+1 ⟶ P

MACHINE FORMAT:

<table>
<tr><td colspan="4">6</td><td colspan="4">7</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>1</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

If the contents of the memory word location specified by the effective address are not less than the contents of the full word scratchpad register pair specified by the SPA field of the instruction, the next instruction in sequence is skipped; otherwise, the next instruction in sequence is executed. The contents of memory and the scratchpad register pair remain unchanged.

If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1

	(RS)			
	(R)	(S)	(M_D)	P
CONTENTS BEFORE EXECUTION	1234	5678	12348765	0200
CONTENTS AFTER EXECUTION	1234	5678	12348765	0202

EXAMPLE 2   Bit 32 = 0

	(RS)			
	(R)	(S)	BITS 16-31 OF INSTRUCTION	(P)
CONTENTS BEFORE EXECUTION	1234	5678	1234	0200
CONTENTS AFTER EXECUTION	1234	5678	1234	0201

INSTRUCTION NAME: Skip On Bit N

OP CODE: 83

FUNCTION: If $\mathrm{Bit}_i = 1$ where i =0,----,15; (P)+1 → P

MACHINE FORMAT:

8				3				BIT NO				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The selected bit within the half word specified by the effective address is tested. If it is 0, the next instruction in sequence is executed. If it is a 1, the next instruction in sequence is skipped. The decimal value of bits 8-11 of the instruction specify the bit to be tested. The contents of memory remain unchanged.

When bit 31 of the instruction is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand. If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	BITS 8 - 11 OF INSTRUCTION	(M_D)	(P)
CONTENTS BEFORE EXECUTION	A	029A15A3	0200
CONTENTS AFTER EXECUTION	A	029A15A3	0202

EXAMPLE 2 Bit 32 = 0

	BITS 8 - 11 OF INSTRUCTION	BITS 16 - 31 OF INSTRUCTION	(P)
CONTENTS BEFORE EXECUTION	A	2345	0200
CONTENTS AFTER EXECUTION	A	2345	0201

4.4.5 PMU Data Instructions

INSTRUCTION NAME: Convert 2's to SM

OP CODE: 91

FUNCTION: $(\overline{D - 1}) \longrightarrow R$

MACHINE FORMAT:

9				1				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The half word specified by the effective address is converted from two's complement to sign and magnitude and placed in the scratchpad register specified by the SPA field of the instruction. The Overflow Interrupt (Trap No. 5) is enabled if the smallest negative number (-2^{+15}) is converted.

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand. If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	XXXX	EDCBA987
CONTENTS AFTER EXECUTION	9235	EDCBA987

EXAMPLE 2   Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	XXXX	A987
CONTENTS AFTER EXECUTION	D679	A987

INSTRUCTION NAME: Convert 2's to SM Full

OP CODE: 95

FUNCTION: $(\overline{DD - 1}) \longrightarrow$ 'RS

MACHINE FORMAT:

9				5				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The full word specified by the effective address is converted from two's complement to sign and magnitude and placed in the full word scratchpad register pair specified by the SPA field of the instruction. The Overflow Interrupt (Trap No. 5) is enabled if the smallest negative number (-2^{+31}) is converted.

If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	XXXX	XXXX	EDCBA987
CONTENTS AFTER EXECUTION	9234	5679	EDCBA987

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	XXXX	XXXX	A987
CONTENTS AFTER EXECUTION	0000	A987	A987

INSTRUCTION NAME: Convert SM to 2's

OP CODE: A2

FUNCTION: $(\overline{D} + 1) \longrightarrow R$

MACHINE FORMAT:

A				2				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The half word specified by the effective address is converted from a sign and magnitude representation to a two's. complement number and placed in the scratchpad register specified by the SPA field of the instruction.

This instruction can be used to load a scratched register with a negative 2's complement number for loop control using the Transfer On Incremented SP instruction (Op Code 50).

When bit 31 is zero, bits 0-15 of (M_D) are used as the operand, and when bit 31 is one, bits 16-31 of (M_D) are used as the operand. If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1, Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	XXXX	12345678
CONTENTS AFTER EXECUTION	1234	12345678

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	XXXX	5678
CONTENTS AFTER EXECUTION	5678	5678

INSTRUCTION NAME: Convert SM to 2's Full

OP CODE: A6

FUNCTION: $(\overline{DD}+ 1) \longrightarrow$ RS

MACHINE FORMAT:

A				6				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The full word specified by the effective address is converted from a sign and magnitude representation to a two's complement number and placed in the full word scratchpad register pair specified by the SPA field of the instruction.

If bit 32 of the instruction word is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	XXXX	XXXX	12345678
CONTENTS AFTER EXECUTION	1234	5678	12345678

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	XXXX	XXXX	5678
CONTENTS AFTER EXECUTION	0000	5678	5678

INSTRUCTION NAME: Round

OP CODE: D6

FUNCTION: $(R) \rightarrow R_{ROUND}$, $O \rightarrow S$

MACHINE FORMAT:

D				6				SPA				X X X X X																				X	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The contents of the full word scratchpad register pair specified by the SPA field of the instruction are algebraically rounded to 16 bits. The result is placed in the high order part of the scratchpad register pair (SPR) and the low order part of the register pair (SPS) is cleared to zero. If an overflow occurs, the Overflow Interrupt (Trap No. 5) is enabled.

EXAMPLE 1

	(RS)	
	(R)	(S)
CONTENTS BEFORE EXECUTION	1234	5678
CONTENTS AFTER EXECUTION	1234	0000

EXAMPLE 2

	(RS)	
	(R)	(S)
CONTENTS BEFORE EXECUTION	FEDC	BA98
CONTENTS AFTER EXECUTION	FEDD	0000

INSTRUCTION NAME: Binary Normalize

OP CODE: D7

$$\left.\begin{array}{l}\text{FUNCTION: IF (RS) Pos, (RS)} \rightarrow \text{RS}_{(2^{30})} \\ \qquad\qquad\quad \text{IF (RS) Neg, (RS)} \rightarrow \text{RS}_{(-2^{30})}\end{array}\right\} \quad \text{Shift Count} \rightarrow M_D$$

MACHINE FORMAT: IF (RS) Zero. 0 → M_D

D				7				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The non zero contents (RS) of the full word scratchpad register pair specified by the SPA field of the instruction is scaled to the range $2^{30} \leq$ RS if RS is positive or RS < -2^{30} if RS is negative. The result is placed in the same scratchpad register pair. The count of the number of shifts required to bring RS into range is placed in the memory word location specified by the effective address. The shift count is stored in bit positions 24-31 with bit positions 0-23 cleared to zero and a data tag of 010 (integer). The maximum shift count that will be stored is 30. If (RS) was ZERO, a count of ZERO is placed in memory and no shifting occurs.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

EXAMPLE 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	0024	68AC	XXXX XXXX
CONTENTS AFTER EXECUTION	48D1	5800	0000 0009

EXAMPLE 2

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	8024	68AC	XXXX XXXX
CONTENTS AFTER EXECUTION	C8D1	5800	0000 0009

4.4.6 PMU Transfer Instructions

INSTRUCTION NAME: Transfer Unconditional

OP CODE: 40

FUNCTION: D ⟶ P

MACHINE FORMAT:

<table>
<tr><td colspan="4">4</td><td colspan="4">0</td><td colspan="4">XXXX</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: The effective address unconditionally replaces the contents of the Program Counter. Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

EXAMPLE

	D	(P)
CONTENTS BEFORE EXECUTION	0200	0010
CONTENTS AFTER EXECUTION	0200	0200

INSTRUCTION NAME: Transfer If R Is Zero

OP CODE: 42

FUNCTION: If (R) = 0, D ⟶ P

MACHINE FORMAT:

4				2				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: If the contents of the scratchpad register specified by the SPA field of the instruction is equal to zero, then the effective address replaces the contents of the program counter. If the contents of the scratchpad register are not equal to zero, normal operation continues with the fetch of the next sequantial instruction. Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

EXAMPLE 1

	(R)	D	(P)
CONTENTS BEFORE EXECUTION	0000	0200	0010
CONTENTS AFTER EXECUTION	0000	0200	0200

EXAMPLE 2

	(R)	D	(P)
CONTENTS BEFORE EXECUTION	0001	0200	0010
CONTENTS AFTER EXECUTION	0001	0200	0011

INSTRUCTION NAME: Transfer If R Zero Full

OP CODE: 46

FUNCTION: If (RS) = 0, D → P

MACHINE FORMAT:

4				6				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: If the contents of the full word scratchpad register pair specified by the SPA field of the instruction is equal to zero, then the effective address replaces the contents of the program counter. If the contents of the full word scratchpad register pair does not equal zero, normal operation continues with the fetch of the next sequential instruction.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

EXAMPLE 1

	(RS)			
	(R)	(S)	D	(P)
CONTENTS BEFORE EXECUTION	0000	0000	0200	0010
CONTENTS AFTER EXECUTION	0000	0000	0200	0200

EXAMPLE 2

	(RS)			
	(R)	(S)	D	(P)
CONTENTS BEFORE EXECUTION	0001	0000	0200	0010
CONTENTS AFTER EXECUTION	0001	0000	0200	0011

INSTRUCTION NAME: Transfer If R Negative

OP CODE: 41

FUNCTION: If (R) < 0, D ⟶ P

MACHINE FORMAT:

<table>
<tr><td colspan="4">4</td><td colspan="4">1</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: If the contents of the scratchpad register specified by the SPA field of the instruction is negative, then the effective address replaces the contents of the program counter. If the contents of the scratchpad register is not negative, normal operation continues with the fetch of the next sequential instruction. The number, negative 0, is treated as a positive 0, that is, non-negative for this instruction. Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

EXAMPLE 1

	(R)	D	(P)
CONTENTS BEFORE EXECUTION	E123	0200	0010
CONTENTS AFTER EXECUTION	E123	0200	0200

EXAMPLE 2

	(R)	D	(P)
CONTENTS BEFORE EXECUTION	7123	0200	0010
CONTENTS AFTER EXECUTION	7123	0200	0011

INSTRUCTION NAME: Transfer If R NEG Full

OP CODE: 45

FUNCTION: If $(RS) < 0$, $D \rightarrow P$

MACHINE FORMAT:

4				5				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: If the contents of the full word scratchpad register pair specified by the SPA field of the instruction is negative, then the effective address replaces the contents of the program counter. If the contents of the full word scratchpad register pair is not negative, normal operation continues with the fetch of the next sequential instruction. The number, negative 0, is treated as a positive 0, that is non-negative for this instruction.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

EXAMPLE 1

	(RS)			
	(R)	(S)	D	(P)
CONTENTS BEFORE EXECUTION	E123	4567	0200	0010
CONTENTS AFTER EXECUTION	E123	4567	0200	0200

EXAMPLE 2

	(RS)			
	(R)	(S)	D	(P)
CONTENTS BEFORE EXECUTION	7123	4567	0200	0010
CONTENTS AFTER EXECUTION	7123	4567	0200	0011

INSTRUCTION NAME: Transfer If Not Equal

OP CODE: 43

FUNCTION: If $D \neq (R), (S) \longrightarrow P$

<table>
<tr><td colspan="4">4</td><td colspan="4">3</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: The contents of the scratchpad register specified by the SPA field of the instruction are compared with the half word specified by the effective address for equality. If they are not equal, the contents of the low order part of the full word scratchpad register pair specified by the SPA field replace the contents of the program counter. If equality exists, normal operation continues with the fetch of the next sequential instruction. When bit 31 of the instruction word, after address modification, is zero, bits 0-15 of the memory word location are used as the operand, and when bit 31 is one, bits 16-31 of the memory location are used as the operand.

If bit 32 of the instruction word is zero, then the operand is immediate, i.e., the contents of bits 16-31 of the instruction word are used for the equality comparison with the contents of the scratchpad register specified by the SPA field.

When used in conjunction with register indirection mode, register replacement and non-addressable (immediate) operand, this instruction acts as a Transfer on Incremented Scratchpad Register instruction where the increment is specified in the register used to index the immediate operand.

EXAMPLE 1 Bit 32 = 1 Bit 31 = 1

	(R)	(S)	(M_D)	(P)
CONTENTS BEFORE EXECUTION	1234	0200	1234 5678	0010
CONTENTS AFTER EXECUTION	1234	0200	1234 5678	0200

EXAMPLE 2 Bit 32 = 0

	(R)	(S)	BITS 16-31 OF INSTRUCTION	(P)
CONTENTS BEFORE EXECUTION	1234	0200	1234	0010
CONTENTS AFTER EXECUTION	1234	0200	1234	0011

INSTRUCTION NAME: Transfer on Incremented SP

OP CODE: 50

FUNCTION: If (R) + 1 $\neq$ 0, D $\longrightarrow$ P

MACHINE FORMAT:

5				0				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The contents of the scratchpad register specified by the SPA field of the instruction are incremented by one. If the incremented contents are equal to zero, the next sequential instruction is executed. If the incremented contents are not equal to zero, the effective address replaces the contents of the program counter. The contents of the scratchpad register are incremented as a 16 bit unsigned binary number. Negative numbers should be specified in two's complement notation in order to facilitate loop control. Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

EXAMPLE 1

	(R)	D	(P)
CONTENTS BEFORE EXECUTION	8002	0200	0010
CONTENTS AFTER EXECUTION	8003	0200	0200

EXAMPLE 2

	(R)	D	(P)
CONTENTS BEFORE EXECUTION	FFFF	0200	0010
CONTENTS AFTER EXECUTION	0000	0200	0011

INSTRUCTION NAME: Transfer to Executive

OP CODE: 37

FUNCTION: Control of Interval Timer, Program Control ——► EXEC

3				7				TIMER CONTROL				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction may be received by the PMU as an external instruction or as an internal instruction from the program counter. When received as an external instruction no indexing, indirection, nor trace (bit 34) will be performed. When executed from a program counter fetch (internal instruction) indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

This instruction loads a two word command to the channel and performs control functions of the PMU's interval timer.

Upon completion of execution of this instruction the Timer Control field (bits 8-11) are used to initiate the following actions of the Interval Timer.

Bit 8 - If zero, the interval timer is unchanged. If one, the interval timer is loaded with bits 16-31 of the final modified Transfer To Executive instruction word after all address modification has taken place.

Bits 9-10 - If both bits are zero, there is no effect on the Interval Timer and it will continue to count if it had been counting or remain stopped if it had been stopped. If bits 9 and 10 are "01," the Interval Timer will continue to count. If bits 9 and 10 are "10," the Interval Timer will stop counting. If bits 9 and 10 are "11," the Interval Timer will resume counting if it had been stopped, or will stop counting if it had been previously counting.

Bit 11 - If zero, the next sequential instruction will be executed. If one, the HALT Indicator is set and the computer will respond only to interrupts.

The first word loaded by this instruction to the channel is a "Transfer and Stack Kernel 0" instruction and has the following format.

5				4				1 1 1 1				F F F O O																				O	O	O	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Bits 0-7 contain the op code number of the Transfer and Stack Kernel 0 instruction.

Bits 8-11 contain the scratchpad register name used as the parameter stack pointer.

Bits 12-31 are the systems wide address to the executive and its entry point. The entry point to the Executive is Word 0 of page number 255 using procedure kernel 0. The procedure kernel page register is updated to 0 upon execution of the "Transfer and Stack Kernel 0" instruction. The previous contents of the procedure kernel page register is saved in the parameter stack.

The Executive must execute a "Write Word From Input" instruction in order to load the second word into a memory location. This second word, loaded in the channel by this "Transfer to Executive" instruction, has the following format.

RESOURCE NAME								0 0								EEFECTIVE ADDRESS OF TRANSFER TO EXEC INST																011			P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Bits 0-7 contain the resource name of the PMU executing the "Transfer To Executive" instruction. This field allows the Executive to determine which resource is calling it.

Bits 8-15 are all zeroes.

Bits 16-31 contain bits 16-31 of the final modified Transfer To Executive instruction word after all address modification has taken place. This field is intended to be used by the Executive to determine the reason for the transfer. For this purpose the programmer must code this field according to the system wide conventions established by the Executive design. The programmer must also remember that should he have set bit 8 of the original Transfer To Executive instruction this same field will be used to load the interval timer.

Bits 32-34 are set to "011" as the data tag indicating logical type data.

INSTRUCTION NAME: Transfer and Stack

OP CODE: 51

FUNCTION:

MACHINE FORMAT:

5				1				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction writes two words into memory and causes a program branch. These words contain the status of the PMU at the time the Transfer and Stack instruction is executed. In the performance of this instruction, the contents of the scratchpad register specified by the SPA field of the instruction is incremented by 1. WORD 1, as defined below, is stored in the memory location specified by the incremented contents of the scratchpad register. The contents of the scratchpad register is then reincremented by 1. WORD 2, as defined below, is then stored in the memory location specified by the reincremented contents of the scratchpad register. The reincremented contents of the scratchpad register specified by the SPA field of the instruction are placed in Scratchpad Register 31 (L-Register). The Halt Indicator is cleared to zero and the Data Kernel Procedure Page Registers remain unchanged. The effective address, after address modification, is then loaded into the program counter.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

If this instruction is the first instruction executed in response to an internal interrupt, the Trap Level Register (Trap #) is appropriately updated to reflect the level of the Trap being honored.

WORD 1 FORMAT

TRAP								O	M	O	H	DK		PK		L-REG																011			P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Bits 0-7 contain the TRAP LEVEL OF THE PROGRAM being branched from.

Bit 8 is cleared to ZERO (unused).

Bit 9 contains the present state of the MODE indicator.

Bit 10 is cleared to ZERO (unused).

Bit 11 contains the present state of the HALT indicator.

Bits 12-13 contain the present state of the Data Kernel Page Register.

Bits 14-15 contain the present state of the Procedure Kernel Page Register.

Bits 16-31 contain the contents of Scratchpad Register 31 (L-Register).

Bits 32-34 contain the field 011 (logical data type).

Bit 35 is adjusted for ODD Parity.

WORD 2 FORMAT

<table>
<tr><td colspan="8">P-SOURCE</td><td colspan="8">0 0</td><td colspan="16">P-COUNTER</td><td colspan="3">011</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

Bits 0-7 are set the present contents of the P-SOURCE Register.

Bits 8-15 are cleared to ZERO (unused).

Bits 16-31 contain the present contents of the Program Counter. These contents indicate the next sequential instruction that was to have been executed.

Bits 32-34 contain the field 011 (logical data type).

Bit 35 is adjústed for odd parity.

INSTRUCTION NAME: Transfer And Stack Kernel N
(N = 0, 1, 2, 3)

OP CODE: 54 (N=0)
55 (N=1)
56 (N-2)
57 (N=3)

FUNCTION:

MACHINE FORMAT:

OP CODE								SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: These instructions perform a PMU-Channel Transfer and Stack instruction as defined in Op Code 51 and sets the procedure kernel page register to N as dictated by the op code. The data kernel register remains unchanged.

The procedure kernel page register specified by the instruction must be within the task memory lower bound as set by the Set Task Parameter (Op Code 29) instruction. If not, an out of bound trap interrupt is generated and the instruction is not executed.

This instruction may be received by the PMU as an external instruction or as an internal instruction from the program counter. When received as an external instruction, no address modification nor trace will be performed.

INSTRUCTION NAME: Escape Number N
(N = 0, 1,--- 7)

OP CODE:	E4 (No. 0)	F4 (No. 4)
	E5 (No. 1)	F5 (No. 5)
	E6 (No. 2)	F6 (No. 6)
	E7 (No. 3)	F7 (No. 7)

FUNCTION: (P) → 33 + 2 x (No. of Escape), DD → S
32 + 2 x (No. of Escape) → P

MACHINE FORMAT:

OP CODE								SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The contents of the memory word location specified by the effective address are loaded into the full word scratchpad register pair specified by the SPA field of the instruction. The current value of the program counter is stored in memory location 33 + 2 x (No. of Escape). The right half (bits 16-31) of the word in memory location 32 + 2 x (No. of Escape) then replaces the contents of the program counter.

Returning from a routine entered by use of this instruction would be accomplished by executing a Transfer Unconditional (Op Code 40) indirecting to location 33 + 2 x (No. of Original Escape).

EXAMPLE OP CODE E6

	(RS)					
	(R)	(S)	(M_D)	LOC 36	LOC 37	(P)
CONTENTS BEFORE EXECUTION	XXXX	XXXX	12345678	1234 0200	XXXX XXXX	0010
CONTENTS AFTER EXECUTION	1234	5678	12345678	1234 0200	0000 0010	0200

4.4.7 PMU Load/Store Instructions

INSTRUCTION NAME: Load SP

OP CODE: 12

FUNCTION: D ⟶ R

MACHINE FORMAT:

1				2				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The contents of the memory word location specified by the effective address of the instruction are placed in the scratchpad register specified by the SPA field of the instruction. The contents of memory remain unchanged. When bit 31 of the modified instruction, after address modification, is zero, bits 0-15 of the memory word are used as the operand. When bit 31 is one, bits 16-31 of the memory word are used as the operand. If bit 32 of the instruction is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1 Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	FEDC	12345678
CONTENTS AFTER EXECUTION	5678	12345678

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	FEDC	1234
CONTENTS AFTER EXECUTION	1234	1234

INSTRUCTION NAME: Load High SP

OP CODE: 10

FUNCTION: D ⟶ H

MACHINE FORMAT:

<table>
<tr><td colspan="4">1</td><td colspan="4">0</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: The contents of the memory word location specified by the effective address of the instruction are placed in the high scratchpad register (SP [16] -SP [31]) as determined by the SPA field of the instruction. For this instruction, the SPA field is interpreted by the computer as being the low order part of the full word register pair specified by the SPA field. The contents of memory remain unchanged. When bit 31 of the modified instruction, after address modification, is zero, bits 0-15 of the memory word are used as the operand. When bit 31 is one, bits 16-31 of the memory word are used as the operand. If bit 32 of the instruction is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1 Bit 31 = 0 SPA = 3

	(H) SP [19]	(M_D)
CONTENTS BEFORE EXECUTION	FEDC	12345678
CONTENTS AFTER EXECUTION	1234	12345678

EXAMPLE 2 Bit 32 = 0 SPA = 8

	(H) SP [24]	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	FEDC	5678
CONTENTS AFTER EXECUTION	5678	5678

INSTRUCTION NAME: Load SP Full

OP CODE: 16

FUNCTION: DD ⟶ RS

MACHINE FORMAT:

1				6				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The contents of the memory word location specified by the effective address of the instruction are placed in the full word scratchpad register pair specified by the SPA field of the instruction. The contents of memory remain unchanged. Bits 0-15 of the memory word are placed in the scratchpad register specified by the SPA field while bits 16-31 are placed in the corresponding scratchpad register of the full word pair (R+16).

If bit 32 of the instruction word is zero, then the operand is immediate. In this case, zeroes are loaded into the scratchpad register specified by the SPA field, while bits 16-31 of the modified instruction word, after address modification, are loaded in the corresponding scratchpad register of the full word pair (R+16).

EXAMPLE 1 Bit 32 = 1

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	FEDC	BA98	1234 5678
CONTENTS AFTER EXECUTION	1234	5678	1234 5678

EXAMPLE 2   Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	FEDC	BA98	5678
CONTENTS AFTER EXECUTION	0000	5678	5678

INSTRUCTION NAME: Load Left Byte

OP CODE: 20

FUNCTION: $D_{0-7} \longrightarrow R_{0-7}$, $0 \longrightarrow R_{8-15}$

MACHINE FORMAT:

2				0				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The left byte (bits 0-7) of the operand specified by the effective address of the instruction is placed into the left byte of the scratchpad register specified by the SPA field of the instruction. Bits 8-15 of the scratchpad register are cleared to zero. The contents of memory remain unchanged. When bit 31 of the modified instruction, after address modification, is zero, bits 0-15 of the memory word are used as the operand. When bit 31 is one, bits 16-31 of the memory word are used as the operand. If bit 32 of the instruction is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1 Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	FEDC	1234 5678
CONTENTS AFTER EXECUTION	5600	1234 5678

EXAMPLE 2   Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	FEDC	1234
CONTENTS AFTER EXECUTION	1200	1234

INSTRUCTION NAME: Load Right Byte

OP CODE: 22

FUNCTION: $D_{8-15} \longrightarrow R_{0-7}, 0 \longrightarrow R_{8-15}$

MACHINE FORMAT:

2				2				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The right byte (bits 8-15) of the operand specified by the effective address of the instruction is placed into the left byte of the scratchpad register specified by the SPA field of the instruction. Bits 8-15 of the scratchpad register are cleared to zero. The contents of memory remain unchanged. When bit 31 of the modified instruction, after address modification, is zero, bits 0-15 of the memory word are used as the operand. When bit 31 is one, bits 16-31 of the memory word are used as the operand. If bit 32 of the instruction is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1 Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	FEDC	1234 5678
CONTENTS AFTER EXECUTION	3400	1234 5678

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	FEDC	5678
CONTENTS AFTER EXECUTION	7800	5678

INSTRUCTION NAME: Load Absolute Value

OP CODE 30

FUNCTION: |D| ⟶ R

MACHINE FORMAT:

3				0				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The absolute value of the half word operand specified by the effective address of the instruction is placed in the scratchpad register specified by the SPA field of the instruction. If the operand is already positive, it is loaded directly. If the operand is negative, the sign bit is made zero and the result is placed in the specified scratchpad register. The contents of memory remain unchanged. When bit 31 of the modified instruction, after address modification, is zero, bits 0-15 of the memory word are used as the operand. When bit 31 is one, bits 16-31 of the memory word are used as the operand. If bit 32 of the instruction is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1 Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	FEDC	1234 5678
CONTENTS AFTER EXECUTION	5678	1234 5678

EXAMPLE 2   Bit 32 =·0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	FEDC	8765
CONTENTS AFTER EXECUTION	0765	8765

INSTRUCTION NAME: Load Absolute Full

OP CODE: 34

FUNCTION: |DD| → RS

MACHINE FORMAT:

3				4				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The absolute value of the full word operand specified by the effective address of the instruction is placed in the full word scratchpad register pair specified by the SPA field of the instruction. If the operand is already positive, it is loaded directly. If the operand is negative, the sign bit is made zero and the result is placed in the specified scratchpad registers. The contents of memory remain unchanged. If bit 32 of the instruction is zero, then the operand is immediate. Zeroes are loaded into the scratchpad register specified by the SPA field while bits 16-31 of the modified instruction word, after address modification, are loaded in the corresponding scratchpad register of the full word pair (R+16).

EXAMPLE 1 Bit 32 = 1

	(RS)			
	(R)	(S)	(M_D)	
CONTENTS BEFORE EXECUTION	FEDC	BA98	9234	5678
CONTENTS AFTER EXECUTION	1234	5678	9234	5678

EXAMPLE 2 Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	FEDC	BA98	9234
CONTENTS AFTER EXECUTION	0000	9234	9234

INSTRUCTION NAME: Load Negative

OP CODE: 31

FUNCTION: -D ⟶ R

MACHINE FORMAT:

3				1				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The half word operand specified by the effective address of the instruction with the sign bit complemented, is loaded in the scratchpad register specified by the SPA field of the instruction. The contents of memory remain unchanged. When bit 31 of the modified instruction, after address modification, is zero, bits 0-15 of the memory word are used as the operand. When bit 31 is one, bits 16-31 of the memory word are used as the operand. If bit 32 of the instruction is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1 Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	FEDC	1234 5678
CONTENTS AFTER EXECUTION	D678	1234 5678

EXAMPLE 2 Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	1234	FEDC
CONTENTS AFTER EXECUTION	7EDC	FEDC

INSTRUCTION NAME: Load Negative Full

OP CODE: 35

FUNCTION: -DD ——► RS

MACHINE FORMAT:

3				5				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The full word operand specified by the effective address of the instruction, with the sign bit complemented, is loaded in the full word scratchpad register pair specified by the SPA field of the instruction. The contents of memory remain unchanged. If bit 32 of the instruction is zero, then the operand is immediate. Zeroes are loaded into the scratchpad register specified by the SPA field, except bit 0 is set to one, while bits 16-31 of the modified instruction word after address modification, are loaded in the corresponding scratchpad register of the full word pair (R+16).

EXAMPLE 1 Bit 32 = 1

	(RS)		(M_D)	
	(R)	(S)		
CONTENTS BEFORE EXECUTION	FEDC	BA98	1234	5678
CONTENTS AFTER EXECUTION	9234	5678	1234	5678

EXAMPLE 2   Bit 32 = 0

	(RS)		
	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	FEDC	BA98	5678
CONTENTS AFTER EXECUTION	8000	5678	5678

INSTRUCTION NAME: Mask Load

OP CODE: 80

FUNCTION: $D\cdot(S)\vee(R)\cdot(\overline{S}) \longrightarrow R$

MACHINE FORMAT:

<table>
<tr><td colspan="4">8</td><td colspan="4">0</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: The half word operand specified by the effective address of the instruction and the contents (S) of the low order part of the full word scratchpad register pair specified by the SPA field of the instruction are logically combined, bit by bit. If the bit of (S) is ONE, the corresponding bit of the contents (R) of the scratchpad register specified by the SPA field is made to equal the bit of D. If not, the corresponding bit of R is unaltered. The contents of memory remain unchanged. When bit 31 of the modified instruction, after address modification is zero, bits 0-15 of the memory word are used as the operand. When bit 31 is one, bits 16-31 of the memory word are used as the operand. If bit 32 of the instruction is zero, then the operand is immediate.

EXAMPLE 1 Bit 32 = 1 Bit 31 = 0

	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	FEDC	3456	1234 5678
CONTENTS AFTER EXECUTION	DA9C	3456	1234 5678

EXAMPLE 2 Bit 32 = 0

	(R)	(S)	BITS 16-31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	FEDC	1234	5678
CONTENTS AFTER EXECUTION	FEF8	1234	5678

INSTRUCTION NAME: Load Control Bits

OP CODE: 84

FUNCTION: $(M_D)_{(32-35)} \longrightarrow R_{(12-15)}$, $0 \longrightarrow R_{(0-11)}$

MACHINE FORMAT:

<table>
<tr><td colspan="4">8</td><td colspan="4">4</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: The control bits (bits 32-35) of the memory word location specified by the effective address of the instruction are placed in bits 12-15 of the scratchpad register specified by the SPA field of the instruction. The remaining bits of the scratchpad register (bits 0-11) are cleared to zero. The contents of memory remain unchanged. If bit 32 of the instruction is zero, then the control bits of this instruction are placed in bits 12-15 of the scratchpad register.

EXAMPLE Bit 32 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	FEDC	70001234B
CONTENTS AFTER EXECUTION	000B	70001234B

INSTRUCTION NAME: Load Multiple

OP CODE: 86

FUNCTION: $(M_D) \longrightarrow RS, (M_{D+1}) \longrightarrow RS+1, \cdot\cdot\cdot\cdot\cdot, (M_{D+15}) \longrightarrow RS + 15$

MACHINE FORMAT:

<table>
<tr><td colspan="4">8</td><td colspan="4">6</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: Sixteen full length words are loaded into the set of scratchpad registers in full word format. The contents of the memory word location specified by the effective address of the instruction are placed in the full word scratchpad register pair specified by the SPA field of the instruction. The effective address of the instruction is then incremented by one and the contents of this new memory location are loaded into the scratchpad register pair specified by SPA+1. The process is repeated until all sixteen scratchpad register pairs are loaded. Normally the SPA field of this instruction should be 0000, designating SP [0] and SP [16] as the first register pair to be loaded. However, if a different register pair is specified, the load operation will continue to SP [15] and SP [31] and then wrap around to SP [0] and SP [16]. The contents of memory remain unchanged. Bit 32 of the modified instruction word, after address modification, has no functionality in the execution of this instruction.

INSTRUCTION NAME: Load Data Kernel

OP CODE: 44

FUNCTION:

MACHINE FORMAT:

<table>
<tr><td colspan="16"></td><td colspan="16">|← ADDRESS →|</td><td colspan="4"></td></tr>
<tr><td colspan="4">4</td><td colspan="4">4</td><td colspan="4">CONTROL</td><td>I</td><td colspan="3">INDEX</td><td colspan="8">PAGE</td><td colspan="8">DISPLACEMENT</td><td>O</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: This instruction is primarily intended to be used for loading a page of data kernel words in a previously reserved kernel area of task memory. This instruction may be executed as an external instruction or from the program counter.

When bit 8 of this instruction is set to 1, the current contents of the Data Kernel Page Register are replaced by the current contents of the Procedure Kernel Page Register. The use of this instruction in this manner will allow all procedure and data accesses through one kernel page.

When bit 8 of this instruction is zero, the current contents of the Data Kernel Page Register are appended to the page field (bits 16-23) of the address field of this instruction to obtain the address of the Kernel Word for the page of data kernel words to be loaded. The residency bit

(bit 10) of this Kernel Word is examined to determine if the page is resident or not. If bit 10 is a ONE, indicating a resident page, the Data Kernel Page Register is updated with the contents of bits 6 and 7 of the Kernel Word. If bit 10 is a zero, indicating a nonresident page, the page is fetched from the remote memory device using the Wide Address field of the Kernel Word. The fetched page is loaded in the kernel area of local memory defined by bits 6 and 7 of the data kernel word and the Data Kernel Page Register is updated with the contents of the same two bit positions.

Bits 9, 10, 11 and the displacement field (bits 24-31) of the address field for this instruction are "don't care" bits and are ignored in the execution of the instruction. The memory bit (bit 32) of this instruction must always be zero, if not, a Read Protect or Kernel Protect Trap Interrupt may be generated since a legal data kernel word entry is read protected.

The Data Kernel Word must indicate paged data (bit 11 set to one) with Kernel Load allowed (bit 8 a zero) and the read, write, and command protect (bits 32, 33 and 34 respectively) bits set to ones. If not, a Kernel Protect security violation exists and execution of this instruction is terminated.

If a non-resident page is fetched, exactly 256 words must be received or a Page Error Trap is initiated. Before a page is requested, a check is performed to see if the referenced data kernel area is within the lower bound area. If not, no page request is issued and a kernel out of bounds trap occurs.

All updates of the Data Kernel Page Register are performed after proper security validation and receipt of 256 words (if necessary). The value to be loaded into the Data Kernel Register must be within the lower bound area defined by the Set Task Parameter instruction otherwise a kernel out of bounds violation occurs.

INSTRUCTION NAME: Load Page

OP CODE: 64

FUNCTION:

MACHINE FORMAT:

																ADDRESS																			
6				4				XXXX				I	INDEX			PAGE								DISPLACEMENT								O	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction is used to load a page of data into local memory. This instruction may be executed as an external instruction or from the program counter.

The contents of the Data Kernel Page Register are appended to the page field (bits 16-23) of the address field of this instruction to obtain the address of the Kernel Word for the page to be loaded. This Kernel word must indicate paged data (bit 11 set to one) with read access (bit 32 a zero) allowed, otherwise if both error conditions or a read protect exists, a Read Protect Violation occurs. If word declared data exists, a Page Violation occurs.

If the Kernel Word indicates the paged data is resident (bit 10 set to one), no further execution is performed and the next sequential instruction is initiated.

If the Kernel word indicates the paged data is non resident (bit 10 is a zero), the page is fetched from the remote memory device using the Wide Address field (bits 12-31) of the Kernel Word. The fetched page (256 words) is loaded into the local memory page indicated by the Task Memory Page field (bits 0-7) of the Kernel Word. If upon reception of the referenced non resident page, exactly 256 words are not received, a Page Error Trap is initiated.

The Load Page instruction temporarily overrides the replacement algorithm specified by the Set Task Parameter instruction.

This instruction is normally non addressable. The memory access bit (bit 32) of the modified instruction, after all address modification is complete, should be zero.

INSTRUCTION NAME: Store SP

OP CODE: 13

FUNCTION: $(R) \rightarrow M_D$

MACHINE FORMAT:

1	3	SPA	I	INDEX	ADDRESS	M	O	T	P
0 3	4 7	8 11	12	13 15	16 31	32	33	34	35

DEFINITION: The contents of the scratchpad register specified by the SPA field of the instruction are placed in the memory location specified by the effective address of the instruction. When bit 31 of the modified instruction, after address modification, is zero, bits 0-15 of the memory location are used to store the scratchpad register contents. When bit 31 is one, bits 16-31 of the memory location are used to store the contents of the register. The remaining contents of the memory location are not affected except that parity is adjusted as necessary.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

EXAMPLE 1 Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	ABCD	1234 5678 6
CONTENTS AFTER EXECUTION	ABCD	ABCD 5678 7

EXAMPLE 2 Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	ABCD	1234 5678 6
CONTENTS AFTER EXECUTION	ABCD	1234 ABCD 6

INSTRUCTION NAME: Store High SP

OP CODE: 11

FUNCTION: $(H) \longrightarrow M_D$

MACHINE FORMAT:

<table>
<tr><td colspan="4">1</td><td colspan="4">$\overline{1}$</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>$\overline{M}$</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: The contents of the high scratchpad register (SP [16] - SP [31]) as specified by the SPA field of the instruction are placed in the memory location specified by the effective address of the instruction. For this instruction, the SPA field is interpreted by the computer as being the low order part of the full word register pair specified by the SPA field. When bit 31 of the modified instruction, after address modification, is zero, bits 0-15 of the memory location are used to store the scratchpad register contents. When bit 31 is one, bits bits 16-31 of the memory location are used to store the contents of the register. The remaining contents of the memory location are not affected except that parity is adjusted as necessary.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

EXAMPLE 1 Bit 31 = 0

	(H)	(M_D)
CONTENTS BEFORE EXECUTION	ABCD	1234 5678 6
CONTENTS AFTER EXECUTION	ABCD	ABCD 5678 7

EXAMPLE 2 Bit 31 = 1

	(H)	(M_D)
CONTENTS BEFORE EXECUTION	ABCD	1234 5678 6
CONTENTS AFTER EXECUTION	ABCD	1234 ABCD 6

INSTRUCTION NAME: Store SP Full

OP CODE: 17

FUNCTION: (RS)⟶ M_D

MACHINE FORMAT:

1				7				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The contents of the full word scratchpad register pair specified by the SPA field of the instruction are placed in the memory location specified by the effective address of the instruction. The data tag (bits 32-34) of the stored word are set to 010 and the parity is adjusted as necessary.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

EXAMPLE

	(RS)		
	(R)	(S)	(M_D)
CONTENTS BEFORE EXECUTION	ABCD	EF01	1234 5678 6
CONTENTS AFTER EXECUTION	ABCD	EF01	ABCD EF01 4

INSTRUCTION NAME: Store Left Byte

OP CODE: 21

FUNCTION: $R_{(0-7)} \longrightarrow M_{D(0-7)}$ or $M_{D(16-23)}$

MACHINE FORMAT:

2				1				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The most significant byte (bits 0-7) of the contents of the scratchpad register specified by the SPA field of the instruction is placed in the left byte position of the half word specified by the effective address of the instruction. When bit 31 of the modified instruction, after address modification, is zero, bits 0-15 of the memory location is the addressed half word with the specified byte being placed in bits 0-7 of the memory word. When bit 31 is one, bits 16-31 of the memory location is the addressed half word with the specified byte being placed in bits 16-23 of the memory word. The remaining contents of the memory location are not affected, except that parity is adjusted as necessary.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

EXAMPLE 1 Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	ABCD	1234 5678 6
CONTENTS AFTER EXECUTION	ABCD	AB34 5678 7

EXAMPLE 2 Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	ABCD	1234 5678 6
CONTENTS AFTER EXECUTION	ABCD	1234 AB78 7

INSTRUCTION NAME: Store Right Byte

OP CODE: 23

FUNCTION: $R_{(0-7)} \rightarrow M_{D(8-15)}$ or $M_{D(24-31)}$

MACHINE FORMAT:

2				3				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The most significant byte (bit 0-7) of the contents of the scratchpad register specified by the SPA field of the instruction is placed in the right byte position of the halfword specified by the effective address of the instruction. When bit 31 of the modified instruction, after address modification, is zero, bits 0-15 of the memory location is the addressed halfword with the specified byte being placed in bits 8-15 of the memory word. When bit 31 is one, bits 16-31 of the memory location is the addressed halfword with the specified byte being placed in bits 24-31 of the memory word. The remaining contents of the memory location are not affected, except that parity is adjusted as necessary.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

EXAMPLE 1   Bit 31 = 0

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	ABCD	1234 5678 6
CONTENTS AFTER EXFCUTION	ABCD	12AB 5678 6

EXAMPLE 2   Bit 31 = 1

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	ABCD	1234 5678 6
CONTENTS AFTER EXECUTION	ABCD	1234 56AB 7

INSTRUCTION NAME: Store Control Bits

OP CODE: 85

FUNCTION: $(R)_{(12-14)} \longrightarrow M_D$ (32-34)

MACHINE FORMAT:

8				5				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The contents of bit positions 12-14 of the scratchpad register specified by the SPA field of this instruction are placed in bit positions 32-34 of the memory location specified by the effective address of this instruction. The remainder of the memory location is unaltered except that parity is adjusted as necessary.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

EXAMPLE

	(R)	(M_D)
CONTENTS BEFORE EXECUTION	0002	1234 5678 6
CONTENTS AFTER EXECUTION	0002	1234 5678 3

INSTRUCTION NAME: Store Multiple

OP CODE: 87

FUNCTION: $(RS) \rightarrow M_D, (RS+1) \rightarrow M_{D+1}, \cdots, (RS+15) \rightarrow M_{D+15}$

MACHINE FORMAT:

8				7				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The sixteen full length words contained in the set of scratchpad registers in full word format are stored in consecutive locations of memory starting with the effective address given in the instruction. The first full length word to be stored is that word contained in the full word scratchpad register pair specified by the SPA field of the instruction. Normally, the SPA field of this instruction should be 0000, designating SP [0] and SP [16] as the first register pair to be stored. However, if a different register pair is specified, the store operation will continue to SP [15] and SP [31] and then wrap around to SP [0] and SP [16].

Bits 32 to 34 of the words stored in memory are set to 011 indicating a logical data tag.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

INSTRUCTION NAME: Store Page

OP CODE: 77

FUNCTI ´

MACHINE FORMAT:

<table>
<tr><td colspan="16"></td><td colspan="16">ADDRESS</td><td colspan="4"></td></tr>
<tr><td colspan="4">7</td><td colspan="4">7</td><td colspan="4">XXXX</td><td>I</td><td colspan="3">INDEX</td><td colspan="8">PAGE</td><td colspan="8">DISPLACEMENT</td><td>O</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: This instruction is used to store a page (256 words) of data in an external resource. This instruction may be executed as an external instruction or from the program counter.

The contents of the Data Kernel Page Register are appended to the page field (bits 16-23) of the address field of this instruction to obtain the address of the Kernel Word for the page to be stored in the external resource. This Kernel word must indicate paged data (bit 11 set to one) with write access (bit 33 is zero) allowed, otherwise if both error conditions or a write protect exists, a Write Protect Violation occurs. If word declared data exists, a Page Violation occurs.

If the Kernel Word indicates the paged data is non resident (bit 10 is a zero), no further execution is performed and the next sequential action is initiated.

If the kernel word indicates the paged data is resident (bit 10 set to one), the page to be stored is fetched from local memory and sent to the remote device using the Wide Address field (bits 12-31) of the Kernel Word. The fetched page is located in local memory using the Task Memory Page field (bits 0-7) of the Kernel Word.

This instruction is normally non addressable. The memory access bit (bit 32) of the modified instruction, after all address modification is complete, should be zero.

INSTRUCTION NAME: Move Half to Half

OP CODE: 19

FUNCTION: $(M_D) \longrightarrow M_{(R)}$

MACHINE FORMAT:

1				9				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction is used to move a half word from one memory location to another. The memory half word addressed by the effective address of the instruction is stored into the half word memory location pointed to by the contents of the scratchpad register specified by the SPA field of the instruction. The remaining half word of memory which as stored into remains unchanged. When bit 31 of the modified instruction, after address modification, is zero, bits 0-15 of the read memory location contain the half word that is to be moved. When bit 31 is one, bits 16-31 of the read memory location contains the half word that is to be moved.

When bit 15 of the designated scratchpad register is zero, the half word being moved is placed in bits 0-15 of the store memory location. When bit 15 of the designated scratchpad register is one, the half word being moved is placed in bits 16-31 of the store memory location.

When bit 32 of the instruction word is zero, bits 16 to 31 of the instruction, after address modification, are the operand to be stored in the memory location specified by the contents of the scratchpad register.

EXAMPLE 1   Bit 32 = 1   Bit 31 = 0   SPR = 03A5

	(M_D)	BITS 16 - 31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	1234 5678 5	FEDC BA98 2
CONTENTS AFTER EXECUTION	1234 5678 5	FEDC 1234 2

EXAMPLE 2   Bit 32 = 0   PSR = 03A4

	BITS 8 - 11 OF INSTRUCTION	BITS 16 - 31 OF INSTRUCTION
CONTENTS BEFORE EXECUTION	1234	FEDC BA98 2
CONTENTS AFTER EXECUTION	1234	1234 BA98 2

INSTRUCTION NAME: Move Full to Full

OP CODE: 1D

FUNCTION: $(M_D) \longrightarrow M_{(R)}$

MACHINE FORMAT:

1				D				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction is used to move a full word from one memory location to another. The word addressed by the effective address of the instruction is stored into the memory location pointed to by the contents of the scratchpad register specified by the SPA field of the instruction. The data tag field (bits 32-34) of the stored word are the data tag bits of the fetched word.

When bit 32 of the instruction word is zero, bits 16-31 of the instruction, after address modification, are the operand to be stored in bit positions 16 to 31 of the store memory location. In this case, bits 0-15 of the store memory location are reset to zeroes.

EXAMPLE 1 Bit 32 = 1

	(M_D)	MEMORY LOCATION POINTED TO BY (R)
CONTENTS BEFORE EXECUTION	1234 5678 6	FEDC BA98 2
CONTENTS AFTER EXECUTION	1234 5678 6	1234 5678 6

EXAMPLE 2 Bit 32 = 0

	BITS 16-31 OF INSTRUCTION	MEMORY LOCATION POINTED TO BY (R)
CONTENTS BEFORE EXECUTION	1234	FEDC BA98 2
CONTENTS AFTER EXECUTION	1234	0000 1234 5

INSTRUCTION NAME: Move Full and Stack

OP CODE: 1F

FUNCTION: $(M_D) \longrightarrow M_{(R)+1}, (R)+1 \longrightarrow R$

MACHINE FORMAT:

I 0 3 I4

1				F				SPA				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction is used to move a full word from one memory location to a full word stack in memory. The memory word addressed by the effective address of the instruction is stored into the full word memory location indicated by the incremented value of the contents of the scratchpad register specified by the SPA field of the instruction. The incremented value of SPR is restored to SPR. The data tag field (bits 32-34) of the stored word are the data tag bits of the fetched word.

When bit 32 of the instruction word is zero, bits 16-31 of the instruction, after address modification, are the operand to be stored in bit positions 16 to 31 of the store memory location. In this case, bits 0-15 of the store memory location are reset to zeroes.

EXAMPLE 1   Bit 32 = 1

	(R)	(M_D)	MEMORY LOCATION POINTED TO BY (R)
CONTENTS BEFORE EXECUTION	03A5	1234 5678 5	FEDC BA98 2
CONTENTS AFTER EXECUTION	03A6	1234 5678 5	1234 5678 5

EXAMPLE 2   Bit 32 = 0

	(R)	BITS 16-31 OF INSTRUCTION	MEMORY LOCATION POINTED TO BY (R)
CONTENTS BEFORE EXECUTION	03A5	1234	FEDC BA98 2
CONTENTS AFTER EXECUTION	03A6	1234	0000 1234 5

4.4.8 PMU Control Instructions

INSTRUCTION NAME: Proceed

OP CODE: 01

FUNCTION: Zero → HALT

MACHINE FORMAT:

0				1				XXXX				I	INDEX			ADDRESS																$\overline{M}$	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: If the HALT INDICATOR is set, it is reset at the conclusion of this instruction and normal processing is resumed.

If the HALT INDICATOR is not set, this instruction performs no operation, unless register indirection with indexing and replacement is the addressing mode.

This instruction is normally received as an external instruction and would therefore be used to resume a previously halted operation.

INSTRUCTION NAME: Execute

OP CODE: 27

FUNCTION:

MACHINE FORMAT:

2				7				XXXX				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The contents of the memory location specified by the effective address of the instruction are treated as an instruction. Instruction sequencing will return to normal operation at the conclusion of this instruction unless the addressed instruction specifies a program control change.

Virtual addressing requires the read protect bit (bit 32) of the kernel word to be equal to zero for security purposes.

The memory access bit (bit 32) of this instruction should be set to 0 after all address modification has taken place.

The memory location treated as an instruction will always be interpreted as a PMU instruction. This condition occurs regardless of the state of bit 33 of the referenced memory location when treated as an instruction.

See AP instruction OP-27, EXECUTE, for the execution of a data word as on AP or PMU instruction.

INSTRUCTION NAME: Interval Timer Control/HALT

OP CODE: 32

FUNCTION:

MACHINE FORMAT:

<table>
<tr><td colspan="4">3</td><td colspan="4">2</td><td>L</td><td>C</td><td>R</td><td>H</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

Bit 8 = Load
Bit 9 = Cease
Bit 10 = Resume
Bit 11 = Halt

DEFINITION: This instruction is used for loading and/or controlling the interval timer.

This instruction treats bits 8-11 of the instruction word as follows:

Bit 8 - If zero, the interval timer contents remain unchanged. If one, the interval timer is loaded with the contents of the memory location specified by the effective address of this instruction.

Bits 9 and 10 - If both bits are zero, there is no effect on the interval timer and it will continue to count if it had been counting or remain stopped if it had been stopped. If bits 9 and 10 are "01," the interval timer will continue to count. If bits 9 and 10 are "10," the interval timer will stop counting. If bits 9 and 10 are "11," the interval timer will resume counting if it had been stopped, or will stop counting if it had been previously counting.

Bit 11 - If set, the computer will suspend program operation at the conclusion of this instruction, set the HALT INDICATOR, and respond only to interrupts. Bit 11 when 0 causes no action.

If bit 32 of the modified instruction word, after address modification is complete, is set to one and bit 31 is zero, bits 0-15 of the addressed memory location are used as the operand to load the interval timer (when bit 8 is a one). When bit 31 is a one, bits 16-31 of the addressed memory location are used as the operand to load the interval timer.

When bit 32 is a zero, the operand is immediate.

INSTRUCTION NAME: Store Interval Timer

OP CODE: 33

FUNCTION:

MACHINE FORMAT:

3				3				XXX				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction causes the contents of the interval timer to be placed in the half word location of memory specified by the effective address of the instruction. The counting operation of the interval timer is unaffected.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

If bit 31 of this instruction, after address modification, is zero, the timer contents will be stored in bits 0-15 of the addressed memory location. If bit 31 is set to one, bits 16-31 will be used to store the timer contents. The remaining bit positions of the memory word remain unchanged except that parity is adjusted as necessary.

INSTRUCTION NAME: Return Stack to P

OP CODE: 52

FUNCTION:

MACHINE FORMAT:

5				2				SPA				0000																				M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction returns the computer to the state it was in preceding the execution of the most recent Transfer and Stack (OP CODE 51) instruction. It should be noted that if the HALT indicator was not set prior to executing the most recent Transfer and Stack instruction then normal computer operation will automatically resume upon completion of the execution of this instruction. If the HALT indicator was set prior to executing the most recent Transfer and Stack instruction, then the Proceed instruction (OP CODE 01) must follow this instruction when normal computer operation is to be resumed.

The contents of the memory location specified by the contents of Scratchpad Register 31 are placed as follows (this word corresponds to Word 2 of the Transfer and Stack Instruction, OP CODE 51):

Bits 0-7 are placed in the P-SOURCE register.

Bits 16-31 replace the contents of the Program Counter.

The contents of Scratchpad Register 31 are now decremented by 1. The contents of the memory location specified by the decremented contents of Scratchpad Register 31 are placed as follows (this word corresponds to Word 1 of the Transfer and Stack Instruction, OP CODE 51):

Bits 0-7 are placed in the TRAP LEVEL indicator.

Bit 9 is placed in the MODE indicator, except that if the MODE presently is 0 and bit 9 is ONE the MODE indicator is left as 0.

Bit 11 is placed in the HALT indicator.

Bits 12 and 13 are placed in the Data Kernel Page Register.

Bits 14 and 15 are placed in the Procedure Kernel Page Register.

Bits 16-31 are placed in the Scratchpad Register 31 and SPA.

INSTRUCTION NAME: Return Stack to P and Proceed

OP CODE: 53

FUNCTION:

MACHINE FORMAT:

5				3				SPA				0000																				M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction returns the computer to the state it was in preceding the execution of the most recent Transfer and Stack (OP CODE 51) instruction and causes normal computer operation to resume. This instruction differs from the Return Stack to P Instruction (OP CODE 52) by clearing the HALT indicator to zero thereby allowing processing to continue regardless of the condition of the HALT indicator prior to the execution of the most recent Transfer and Stack (OP CODE 51).

The contents of the memory location specified by the contents of Scratchpad Register 31 are placed as follows (this word corresponds to Word 2 of the Transfer and Stack Instruction OP CODE 51):

Bits 0-7 are placed in the P-SOURCE register.

Bits 16-31 replace the contents of the Program Counter.

The contents of Scratchpad Register 31 are now decremented by 1. The contents of the memory location specified by the decremented contents of Scratchpad Register 31 are placed as follows (this word corresponds to Word 1 of the Transfer and Stack Instruction, OP CODE 51):

Bits 0-7 are placed in the TRAP LEVEL indicator.

Bit 9 is placed in the MODE indicator, except that if the MODE presently is 0 and bit 9 is ONE, the MODE indicator is left as 0.

Bits 12 and 13 are placed in the Data Kernel Page Register.

Bits 14 and 15 are placed in the Procedure Kernel Page Register.

Bits 16-31 are placed in Scratchpad Register 31 and SPA.

The HALT indicator is cleared to ZERO.

INSTRUCTION NAME: Reset Bit N

OP CODE: 81

FUNCTION: $0 \rightarrow M_{D(\text{BIT N})}$, N=0, 1,····,15

MACHINE FORMAT:

8				1				BIT NO.				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The selected bit within the half word addressed by the effective address of the instruction is cleared to zero. The decimal value of bits 8 and 11 of the instruction specify the bit that is to be cleared. After the specified bit is cleared to zero, the addressed half word is returned to memory in the same memory location. This instruction should always have bit 32 set to zero after all address modification has been completed.

If bit 31 of the instruction, after address modification, is zero, bits 0-15 of the addressed memory location are used as the operand. When bit 31 is a one, bits 16-31 of the addressed memory location are used as the operand.

If an operand to be manipulated is contained in a scratchpad register, the logical operation Not D And R (OP CODE 90) should be used, where R is the contents of the scratchpad register and D is a literal contained in the instruction.

EXAMPLE Bit 31 = 1

	BITS 8 - 11 OF INSTRUCTION	(M_D)
CONTENTS BEFORE EXECUTION	B	1234 5678
CONTENTS AFTER EXECUTION	B	1234 5668

INSTRUCTION NAME: Set Bit N

OP CODE: 82

FUNCTION: $1 \rightarrow M_{D(BIT\ N)}$, N = 0,1,....,15

<table>
<tr><td colspan="4">8</td><td colspan="4">2</td><td colspan="4">BIT NO.</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: The selected bit within the half word addressed by the effective address of the instruction is set to one. The decimal value of bits 8 to 11 of the instruction specify the bit that is to be set. After the specified bit is set to one, the addressed half word is returned to memory in the same memory location. This instruction should always have bit 32 set to zero after all address modification has been completed.

If bit 31 of the instruction, after address modification, is zero, bits 0-15 of the addressed memory location are used as the operand. When bit 31 is a one, bits 16-31 of the addressed memory location are used as the operand.

If an operand to be manipulated is contained in a scratchpad register, the logical operation D or R (OP CODE B2) should be used, where R is the contents of the scratchpad register and D is a literal contained in the instruction.

EXAMPLE Bit 31 = 0

	BITS 8 - 11 OF INSTRUCTION	(M_D)
CONTENTS BEFORE EXECUTION	5	1234 5678
CONTENTS AFTER EXECUTION	5	1634 5678

INSTRUCTION NAME: Set Task Parameters

OP CODE: 29

FUNCTION:

MACHINE FORMAT:

<table>
<tr><td colspan="18"></td><td colspan="2">REPLACEMENT ALGORITHM</td><td colspan="3"></td><td>MODE</td><td colspan="12"></td></tr>
<tr><td colspan="4">2</td><td colspan="4">9</td><td colspan="4">UPPER BOUND</td><td colspan="4">LOWER BOUND</td><td colspan="2">X X</td><td colspan="2"></td><td colspan="3">XXX</td><td></td><td colspan="8">X X</td><td>X</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: This instruction is used to define the upper and lower bounds in memory (see section 4.3.3.3 Virtual Addressing), specify the replacement algorithm to be used (see section 4.3.3.3.4 Replacement Algorithm), and to specify whether virtual or direct addressing will be used. This instruction is illegal when generated from a Program Counter fetch.

The format of the instruction is as follows:

The Upper Bound, Lower Bound, and Replacement Algorithm fields are used to initialize internal registers. Bit 23, the Addressing Mode bit, when set to one, specifies that further data references will utilize virtual addressing. When bit 23 is zero, all further data references will utilize direct addressing.

The decimal value contained in the lower bound field (bits 12-15) of the instruction should never exceed three. If the lower bound indicated is loaded with a value less than the present value of the Data and Procedure Kernel Page Registers, a Kernel Protect trap occurs.

The Replacement Algorithm is specified by bits 18 and 19 of the Set Task Parameter Instruction. The algorithms are:

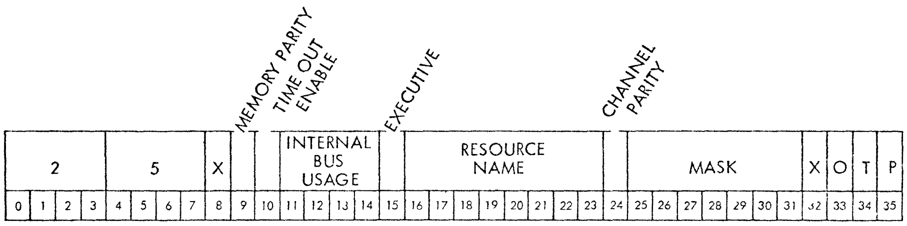
18	19	
0	0	- Programmer Specified
0	1	- FIFO
1	0	- Random
1	1	- Sequantial Fill/Random

INSTRUCTION NAME: Set System Parameters

OP CODE: 25

FUNCTION:

MACHINE FORMAT:



DEFINITION: This instruction is used to disable memory or channel parity and to initialize the channel. This instruction must be received as an external instruction rather than through the Program Counter, otherwise, an illegal instruction trap will be generated and no operation will occur. The channel sensing this instruction will interpret all fields except the memory parity designation (bit 9). Within the PMU every bit except bit 9 is a don't care.

The bit fields of the instruction have the following meaning:

Bit 9 - Memory Parity

When bit 9 is one, the detection of bad parity on a local memory read is masked out and no parity trap interrupt is generated. When bit 9 is zero, the detection of incorrect parity on a local memory read will will result in a parity trap interrupt.

Bit 10 - Time Out Enable

When bit 10 is one, the Time Out Counter is enabled. When bit 10 is zero, the Time Out Counter is disabled. An enabled Time Out Counter results in a timer being initiated every time a new word is placed on the secondary input bus by the channel. If after 4096 time slots, no acknowledge is received, the channel resets its secondary input bus controller, notifies the Executive of this occurrence and activates the reset line of the secondary bus.

Bit 11 - Permit Enable

When bit 11 is one, the channel will be allowed to keep the internal bus until all transfers are completed. When bit 11 is zero, the channel can keep the bus for the transmission of only one word or a two word command. The channel must then contend for further internal bus accesses.

Bit 12 - Bus 1 Control

When bit 12 is zero, further use of bus 1 is disallowed. When bit 12 is one, further use of bus 1 is allowed.

Bit 13 - Bus 2 Control

When bit 13 is zero, further use of bus 2 is disallowed. When bit 13 is one, further use of bus 2 is allowed.

Bit 14 - Bus 3 Control

When bit 14 is zero, further use of bus 3 is disallowed. When bit 14 is one, further use of bus 3 is allowed.

Bit 15 - Executive

When bit 15 is one, the channel will recognize the destination code hex "FF," designating the Executive, in addition to the code indicated by the resource name and mask fields. When bit 15 is zero, the channel will only recognize the destination code indicated by the resource name and mask fields. Care must be taken when this instruction is sent to a previously designated Executive. If Bit 15 is zero, the mode indicator is reset.

Bits 16-23 - Resource Name

This field designates the resource name of the channel. Bit 16 is the most significant bit of the name and bit 23 is the least significant bit.

Bit 24 -˙Channel Parity

When bit 24 is one, the parity checking on all channel receptions is overridden and the data received is considered validated. When bit 24 is zero, the channel performs parity checking on all transmissions received and responds to incorrect parity.

Bits 25-31 - The resource mask is used by the channel to determine the number of bits of the resource name to be masked out when the channel is determining if a transmission is addressed to it. When a mask bit is one, the checking of the corresponding bit of the destination address contained on the internal bus against the bit of the channel name field (both physical and assigned) is disabled. When a mask bit is zero, the checking of the corresponding bit of the destination address contained on the internal bus against the bit of the channel name is enabled.

Bit 0, the most significant bit of the destination address is always checked.

As an example, a mask of 000 0001_2 and an assigned resource name of 100_{10} allows a channel to accept information for resource names 100_{10} and 101_{10}. A mask of $100\ 0000_2$ and an assigned resource name of 100_{10} allows a channel to accept information for resource names 36_{10} and 100_{10}.

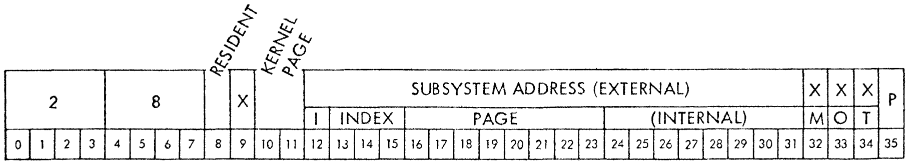
Note: Set System Parameter (SSP). The instruction MUST always be preceded by a COMS immediate instruction. An SSP instruction received in any other fashion will result in the channel NOT executing the instruction.

INSTRUCTION NAME: Initiate New Task

OP CODE: 28

FUNCTIOŃ:

MACHINE FORMAT:



DEFINITION: This instruction is used to initialize the PMU to begin a new task. The instruction may be received as an external instruction (input) or be fetched from local memory and executed from the Program Counter (internal).

When executed as an internal instruction, this instruction is normally non addressable, bit 32 is a zero after all address modification is completed.

When the instruction is fetched from the input instruction queue (external), the following actions occur. If bit 8 is 1, indicating resident, the procedure and data kernel page registers are loaded with bits 10-11, the Program Counter cleared to all zeroes, the P-source register loaded with the I-source register, and a transfer to the instruction located at page 0, location 0 is executed.

If bit 8.is 0, indicating non resident, the procedure and data kernel page registers are loaded with bits 10-11, the Program Counter cleared to all 0's, the P-source register loaded with the I-source register and a Read Page to Output instruction (OP CODE 06) is generated to the channel. The wide address of the Read Page to Output instruction is bits 12-31 of the Initiate New Task instruction. After successful reception of the addressed kernel page, a transfer to the instruction located a page 0, location 0 is executed.

If the Initiate New Task instruction is fetched from local memory, the kernel entry word (in the Data Kernel) is accessed. If no security violation exists and the page is resident, the Procedure and Data Kernel Page registers are updated to bits 6-7 of the data kernel word and a transfer to the instruction located at page 0, location 0, is executed.

If no security violation exists and the page is non resident, a Read Page to Output is issued to the channel with the wide address field being bits 12-31 of the accessed kernel entry word. The Procedure and Data Kernal registers are updated with bits 6 & 7 of the data kernel word, and upon successful reception of the referenced page, a transfer to the instruction located as page 0, location 0 is executed.

The Halt flop is reset independent of both internal and external executions.

Bit 8 is a don't care when the Initiate New Task instruction is fetched from local memory.

If the instruction is fetched from local memory, a kernel security check is performed. The kernel entry word must indicate paged data, and bits 8, 32, 33, 34 must be 0111. If this is not true, a kernel protect trap is initiated.

Before a kernel security check is made on internal instructions, a lower bound check is performed. The indicated Procedure and Data Kernel pages must be within the area defined by the lower bound. If not, a Kernel Out of Bounds Trap is initiated.

If a page is received due to an issuance of a Read Page to Output instruction, exactly 256 data words must be received. If not, a page error trap is initiated. The Procedure Kernel Register is updated with data kernel word bits 6 and 7 prior to the initiation of a Page Error Trap should one occur. The data kernel register is not altered if a page error trap occurs.

INSTRUCTION NAME: Test And Reset

OP CODE: 36

FUNCTION: $(M_D)_{16-31} \rightarrow R$, ZERO $\rightarrow M_D$

MACHINE FORMAT:

<table>
<tr><td colspan="4">3</td><td colspan="4">6</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: This instruction is normally used to control access to queued access data bases.

The execution of this instruction will store bits 16-31 of the contents of the effective address into the scratchpad register specified by the SPA field of the instruction. Bits 0-31 of the contents of the effective address are then cleared to zero.

If the data addressing mode is absolute, the local memory location is cleared.

If the data addressing mode is virtual, the data must be word oriented, if not, an illegal instruction trap will be generated.

The accessed, Data Kernel Word must indicate Read and Write access allowed. If not, the appropriate violation occurs. (Both violations can occur simultaneously.)

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

INSTRUCTION NAME: Command Subsystem/Address Modification

OP CODE: 47

FUNCTION:

MACHINE FORMAT:

<table>
<tr><td colspan="8"></td><td>EMERGENCY</td><td colspan="7"></td><td colspan="16">ADDRESS</td><td colspan="4"></td></tr>
<tr><td colspan="4">4</td><td colspan="4">7</td><td></td><td colspan="3">XXX</td><td>I</td><td colspan="3">INDEX</td><td colspan="8">PAGE</td><td colspan="8">DISPLACEMENT</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: This instruction is used to cause the transmission of another instruction to a remote device for execution in that device when this instruction is fetched from local memory. When this instruction is received as an external instruction, transmitted by a remote device, the remote device is causing the execution of another instruction to occur in this DPE.

As implied above, this instruction may be received as an external instruction or from the Program Counter.

The Command Subsystem/Address Modification is essentially a two word instruction. When the instruction is executed from local memory through the Program Counter, the instruction word is used to set up a bus command word for routing the second word. The bus command word has the following format:

Bits 0-7 contain the value hex "47," the op code for this instruction.

Bits 8-11 contain the bits 4-7 of the kernel word referenced by the effective address of this instruction.

Bits 12-23 contain the bits 12-23 of the kernel word referenced by the effective address of this instruction.

Bits 24-31 contain the result of adding bits 24-31 of the effective address of this instruction to bits 24-31 of the kernel word.

Note that bits 12-31 form the displaced wide address.

Bit 32-34 are cleared to zero indicating immediate addressing, PMU instruction and no tracing.

Bit 35 is set to indicate odd parity for bits 0-35.

Bit 36 is set for bus parity.

Bits 37-39 is set to indicate a transmission tag for a two word instruction. If bit 8, Emergency bit, of this instruction is zero, this field is set to 101. If bit 8 is a one and the PMU is in Executive Mode, this field will be set to 111 indicating a two word emergency. If bit 8 of this instruction is set to 1 and the PMU is not in the Executive Mode, an illegal instruction trap will occur.

Bits 40-47 will contain the source address indicating the channel transmitting this word.

The second word of this two word instruction set, which will be transmitted over the bus, is the contents of the memory location indicated by the next sequential instruction address as indicated by the Program Counter. This second word will be transmitted with a transmission tag of 011 indicating "Data and End of Block." When the Program Counter is incremented, page wraparound will occur if the Command Subsystem instruction was located at word 255. In this case, the second word will be fetched from word 0 of the present procedure page.

In the above case, whenever the kernel word is accessed, bit 8 (Kernel Load) must be one, and bit 34 (Command Protect) must be zero; otherwise, a Command Protect occurs.

When fetched through the Program Counter, indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

If this instruction, Command Subsystem, is fetched from the input queue, the second word of the two word command is fetched from the input instruction queue and executed. The execution of the second word is performed as if it was fetched from memory rather than the input queue. Therefore, address modification and tracing capabilities are allowed, as appropriate, even though the instruction was actually fetched as an external instruction.

INSTRUCTION NAME: Command Subsystem/Immediate Execution

OP CODE: 4F

FUNCTION:

MACHINE FORMAT:

		EMERGENCY				ADDRESS					
4	F		XXX	I	INDEX	PAGE	DISPLACEMENT	M	O	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31	32	33	34	35

DEFINITION: The use of this instruction is similar to the Command Subsystem/Address Modification (OP CODE 47) except that this instruction would typically be used to transmit Set System Parameter (OP CODE 25), Set Task Parameter (OP CODE 29), Initiate New Task (OP CODE 28), or other instructions whose bit positions 8-15 are control fields and not SPA and/or AMF fields.

The Command Subsystem/Immediate Execution is essentially a two word instruction. When the instruction is executed from local memory through the Program Counter, the instruction word is used to set up a bus command word for routing the second word. The bus command word has the following format:

Bits 0-7 contain the value hex "4F," the op code for this instruction.

Bits 8-11 contain 1111.

Bits 12-23 contain the bits 12-23 of the kernel word referenced by the effective address of this instruction.

Bits 24-31 contain the result of adding bits 24-31 of the effective address of this instruction to bits 24-31 of the kernel word.

Note that Bits 12-31 form the displaced wide address.

Bit 32-34 are cleared to zero indicating immediate addressing, PMU instruction and no tracing.

Bit 34 is set to indicate odd parity for bits 0-35.

Bit 36 is set for bus parity.

Bits 37-39 is set to indicate a transmission tag for a two word instruction. If bit 8, Emergency bit, of this instruction is zero, this field is set to 101. If bit 8 is a one and the PMU is in Executive Mode, this field will be set to 111 indicating a two word emergency. If bit 8 of this instruction is set to 1 and the PMU is not in the Executive Mode, an illegal instruction trap will occur.

Bits 40-47 will contain the source address indicating the channel transmitting this word.

The second word of this two word instruction set, which will be transmitted over the bus, has the format described as follows:

Bits 0-15 are bits 0-15 of the next instruction word fetched by the incremented Program Counter.

Bits 16-31 are bits 16-31 of the kernel entry word referenced by bits 16-23 of the second word fetched by the Program Counter.

Bits 32-34 are cleared to zero indicating immediate addressing, PMU instruction and no instruction tracing.

Bit 35 is set to indicate odd parity for bits 0-35.

Bit 36 is set for bus parity.

Bits 37-39 is set to 011, the transmission tag indicating "Data and End of Block."

Bits 40-47 contain the source address indicating the channel transmitting this word.

When the second instruction word is fetched as a result of incrementing the Program Counter, page wraparound will occur if the Command Subsystem instruction was located at word 255. In this case, the second word will be fetched from word 0 of the present procedure page.

In the above case, whenever the kernel word is accessed, bit 8 (Kernel Load) must be one, and bit 35 (Command Protect) must be zero; otherwise, a Command Protect will occur.

If a security violation occurs when the second Kernel entry is accessed, a PMU "OR" instruction (OP CODE B2) with an address field of all 0's with immediate addressing specified is created and associated with the first bus word transmitted.

If this instruction, Command Subsystem, is fetched from the input queue, the second word of the two word command is fetched from the input instruction queue and is executed immediately as an external instruction.

4.4.9 PMU Input/Output Instructions

INSTRUCTION NAME: Test and Reset to Output

OP CODE: 00

FUNCTION $(M_D) \rightarrow E$, ZERO $\rightarrow M_D$

MACHINE FORMAT:

0				0				XXXX				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction is received over the Primary Bus and causes the contents of the effective address to be sent to the external subsystem specified by the contents of the Active Source Register, which generated this instruction, using a standard data transmission word (transmission code is 011). Bits 0-31 of the contents of the effective address are reset to ZERO.

This instruction, received as an external instruction interrupt, is not subject to address modification except if it is received via the COMMAND SUBSYSTEM Address Modification Instruction (OP CODE 47).

This instruction is illegal if generated by a program counter operation, and its appearance enables the Illegal Instruction Interrupt (Trap No. 9). No operation of this instruction is performed in this situation.

INSTRUCTION NAME: Read Word to Output

OP CODE: 02

FUNCTION $(M_D) \rightarrow E$

MACHINE FORMAT:

<table>
<tr><td colspan="4">0</td><td colspan="4">2</td><td colspan="4">XXXX</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>1</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: This instruction is received over the Primary Bus and causes the contents of the effective address to be sent to the external subsystem specified by the contents of the Active Source Register, which generated this instruction, using a standard data transmission word (transmission code is 011). The contents of memory remain unchanged.

This instruction, received as an external instruction interrupt, is not subject to address modification except if it is received via the COMMAND SUBSYSTEM/ Address Modification Instruction (OP CODE 47).

This instruction is illegal if generated by a program counter operation, and its appearance enables the Illegal Instruction Interrupt (Trap No. 9). No operation of this instruction is performed in this situation.

INSTRUCTION NAME: Write Word From Input

OP CODE: 03

FUNCTION: $E \rightarrow M_D$

MACHINE FORMAT:

0				3				XXXX				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction is received over the Primary Bus and causes the DPE to receive a second 36 bit word over the primary bus which the PMU will store into the local memory location directly addressed by the address field of this instruction.

If this instruction is received as an external instruction interrupt it is not subject to address modification except if it is received via the Command Subsystem/Address Modification instruction (OP CODE 47) in a PMU with channel configuration.

This instruction is legal if generated by a Program Counter operation. The bus data word is stored into the memory location directly addressed by the effective address field of the instruction.

INSTRUCTION NAME: Single Word I/O Command

OP CODE: 60

FUNCTION: $(M_D) \rightarrow E$

MACHINE FORMAT:

<table>
<tr><td colspan="4">6</td><td colspan="4">0</td><td>SN</td><td colspan="3">XXX</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>O</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: This instruction is used to send a word to an external resource which is to interpret the transmitted word as a single word instruction.

This instruction will cause the contents of the memory location specified by the effective address of this instruction, plus a parity bit, a transmission tag of 100 (indicating Single Word Command) and a source code to be placed on the secondary bus. The source code is hardwired within the PMU. The destination of the transmitted word is contained within the transmitted word itself.

If the addressing mode is virtual, a security check is performed on the referenced kernel word before reading the addressed word. Kernel word bits 8 (Kernel Protect) and 34 (Command Protect) must be zero. If a security violation exists, further execution of the instruction is aborted and a command protect trap occurs. Although not required, normal procedure should have the kernel word bit 33 set to one. A user program should not be allowed to modify the data that is to be interpreted as an instruction.

Bit 8 of the instruction controls the sequence number of the transmitted command. If bit 8 is 1, a reply is expected and the channel applies the next available sequence number to the command. If bit 8 is 0, the present sequence number applies.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

INSTRUCTION NAME: Two Word I/O Command

OP CODE: 73

FUNCTION:

MACHINE FORMAT:

7	3	SN	XXX	I	INDEX	ADDRESS	M	O	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

DEFINITION: This instruction is used to send two words to an external resource. The first word is a command and the second word is data. The second word is never a command. The Command Subsystem instruction (OP CODE 47 or 4F) is used to send commands in the second word that are to be interpreted as commands.

This instruction is a two word instruction where the first word has the format given above and is fetched by the Program Counter. The second word of the instruction is fetched from the location Program Counter plus one and has the following format.

X X X X	ADDRESS	XXX	P
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32 33 34	35

The execution of the first word of the instruction causes a standard data transmission word to be placed on the bus. This word consists of the full 36 bit contents of the memory location specified by the effective address of the Two Word I/O Command instruction, a bus parity bit, the transmission code 101, and the 8 bit source code of this PMU. Then, a second data transmission word consisting of the full 36 bit contents of the memory location specified by the unmodified address of the second word of this instruction, a bus parity bit, the transmission code 011, and the 8 bit source code of this PMU is placed on the output bus.

If the addressing mode in the PMU is virtual, a security check is performed on the kernel word referenced by the address of the first word of this instruction before reading the first word to be transmitted. Kernel word bits 8 (Kernel Protect) and 34 (Command Protect) must be zero. If a security violation occurs, further execution of the instruction is aborted and a command protect trap occurs. If no security violation exists, the first word to be transmitted is read and the bus word is formatted. The second word of the instruction is obtained and the address (bits 16-31) contained in this second word is used to obtain the kernel word for the second word to be transmitted. Bit 32 (Read Protect) of this kernel word must be zero, otherwise, an instruction which causes no operation is placed on the secondary bus. If no security violation occurs, the second word to be transmitted is obtained and a bus word is formatted.

With virtual addressing mode, word on of this instructiòn should be on an even boundary or have an address value less than 254 in the lower order 8 bits (bits 24-31 of the instruction). If the lower order value is .255, word 2 of this instruction set will be read from location zero of the same page that the initial instruction was read. The next sequential instruction will be read from location one of the next sequential procedure page.

Bit 8 of the instruction controls the sequence number of the transmitted command. If Bit 8 is 1, a reply is expected and the channel applies the next available sequence number to the command. If bit 8 is 0, the present sequence number applies.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

INSTRUCTION NAME: Two Word I/O With Indexing

OP CODE: 7B

FUNCTION:

MACHINE FORMAT:

7	B	XXXX	I	INDEX	ADDRESS	M	O	T	P
0 1 2 3	4 5 6 7	8 9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

DEFINITION: This instruction is similar to the TWO WORD I/O COMMAND instruction (OP CODE 73) except that the second word of the instruction allows indexing in obtaining the second word to be transmitted on the bus.

This instruction is a two word instruction where the first word has the format given above and is fetched by the Program Counter. The second word of the instruction is fetched from the location program counter plus one and has the following format.

X	X	SPA	XXXX	ADDRESS	XX	T	P
0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32 33	34	35

The execution of the first word of the instruction causes a standard data transmission word to be placed on the bus. This word consists of the full 36 bit contents of the memory location specified by the effective address of the Two Word I/O Command instruction, a bus parity bit, the transmission code 101, and the 8 bit source code of this PMU. Then, a second data transmission word consisting of the full 36 bit contents of the memory location specified by the indexed address of the second word of this instruction, a bus parity bit, the transmission code 011, and the 8 bit source code of this PMU is placed on the output bus. The indexed address of the second word to be transmitted is obtained by adding the contents of the scratchpad register specified by the SPA field of the second word of this instruction to the address field (bits 16-31) of the second word of this instruction.

If the procedure addressing mode in the PMU is virtual, a security check is performed on the kernal word referenced by the address of the first word of this instruction before reading the first word to be transmitted. Kernel word bits 8 (Kernel Protect) and 34 (Command Protect) must be zero. If a security violation occurs, further execution of the instruction is aborted and a command protect trap occurs. If no security violation exists, the first word to be transmitted is read and the bus word is formatted. The second word of the instruction is obtained and the indexed address of this second word is used to obtain the kernel word for the second word to be transmitted. Bit 32 (Read Protect) of this kernel word must be zero, otherwise, an instruction which causes no operation is placed on the secondary bus. If no security violation occurs, the second word to be transmitted is obtained and a bus word is formatted.

With virtual addressing mode, word one of this instruction should be on an even boundary or have an address value less than 254 in the lower order 8 bits (bits 24-31 of the instruction). If the lower order value is 255, word 2 of this instruction set will be read from location zero of the same page that the initial instruction was read. The next sequential instruction will be read from location one of the next sequential procedure page.

Indexing and indirection are allowed with this instruction. However, the final modified instruction word after all address modification has taken place should have its memory access bit (bit 32) reset to a zero, indicating an immediate address. Should bit 32 be set to one, a non required extra clock cycle will be expended to read the contents of the memory location specified by the effective address.

INSTRUCTION NAME: Read Operand to Output

OP CODE: 04

FUNCTION: $(M_D) \longrightarrow E$

MACHINE FORMAT:

<table>
<tr><td colspan="4">0</td><td colspan="4">4</td><td colspan="4">XXXX</td><td colspan="4">XXXX</td><td colspan="16">ADDRESS</td><td>1</td><td>0</td><td>X</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: This instruction is received over the primary bus and causes the PMU to output a word that is sent to the resource initiating this instruction.

The data word to be transmitted is contained in the memory location specified by the effective address of this instruction and is placed in the output buffer. The destination of the data word is contained in the Active Source Register. If the operand is two word or complex, a second word is read and placed in the output buffer. If the addressed operand is an array, all operands associated with the array are read and placed in the output buffer.

This instruction, received as an external instruction interrupt, is not subject to address modifications. This instruction is illegal if generated by a Program Counter fetch.

The appropriate transmission tag is used in the formatting of the word to be transmitted on the internal bus according to paragraph 3.1.1.

INSTRUCTION NAME: Read Page to Output

OP CODE: 06

FUNCTION:

MACHINE FORMAT:

0				6				XXXX				XXXX				ADDRESS																1	0	X	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction is received over the primary bus and causes the PMU to output a page (256 words), that is sent to the resource initiating this instruction.

The 256 words beginning with the word directly specified by the effective address of this instruction are placed into the output buffer using the contents of the Active Source Register as the destination. If the 255th word of a page is read, the next operand read will be the 0th word of the same page (Page Wraparound).

This instruction, received as an external instruction interrupt, is not subject to address modification. This instruction is illegal if generated by a Program Counter fetch.

The appropriate transmission tag is used in formatting the words to be transmitted on the internal bus, according to paragraph 3.1.1.

INSTRUCTION NAME: Read Array to Output

OP CODE: 0C

FUNCTION:

MACHINE FORMAT:

0				C				XXXX				X	XXX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction is a two word instruction. This command is issued by the PMU as a result of a dimension word (see paragraph 5.3.12) being accessed during an indirection step for all AP addressable instructions. It is issued by the PMU to an external source in order to obtain the array operands for processing by the AP.

The first word of this two word instruction has the format given above. The second word is the dimension word (see paragraph 5.3.12) of the array) to be read.

The first word of the array to be read is contained in the location specified by the effective address of the first word of this instruction. The structure of the array operand is described by the dimension word contained in the second word of this command.

This instruction is executed as an external instruction interrupt. This instruction is illegal if generated by a program counter operation, and its appearance enables the Illegal Instruction Trap (No. 9).

INSTRUCTION NAME: Read Indirect Word to Output

OP CODE: 0E

FUNCTION:

MACHINE FORMAT:

0				E				XXXX				XXXX				ADDRESS																001			P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction is received over the primary bus and causes the PMU to output two words that are sent to the resource initiating this instruction.

The contents of the memory location specified by the effective address of this instruction are transmitted to the destination specified by the contents of the Active Source Register. If bits 32-34 of this instruction have the bit pattern 001, the contents of the next sequential memory location are also transmitted to the specified destination. Both transfers use a standard data transmission word.

This instruction, received as an external instruction, interrupt, is not subject to address modification. This instruction is illegal if generated by a Program Counter fetch.

INSTRUCTION NAME: Write Operand From Input

OP CODE: 05

FUNCTION: $E \rightarrow M_D$

MACHINE FORMAT:

<table>
<tr><td colspan="4">0</td><td colspan="4">5</td><td colspan="4">XXXX</td><td colspan="4">XXXX</td><td colspan="16">ADDRESS</td><td colspan="3">XXX</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: This instruction is received over the primary bus and causes the PMU to input an operand that has been sent by the remote resource which initiated this instruction.

The operand, located in the instruction input queue, is stored in the memory location specified by the effective address of this instruction. If this data word is not tagged with an "End of Block," a continue command is sent to the device indicated by the active resource register. The data that is subsequently returned by the resource addressed is read from the input data queue and stored in consecutive memory locations until the EOB is received. If an EOB is not received and the least significant 8 bits of the address just written into were 255, the next word is stored in an address whose least significant 8 bits are all zeroes (page wraparound) and whose most significant bits are unchanged.

This instruction, received as an external instruction interrupt is not subject to address modification. This instruction is illegal if generated by a Program Counter fetch.

The appropriate transmission tags used in the sending of the operands are explained in paragraph 3.1.1.

INSTRUCTION NAME: Write Page From Input

OP CODE: 07

FUNCTION:

MACHINE FORMAT:

0				7				XXXX				XXXX				ADDRESS																XXX			P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction is received over the primary bus and causes the PMU to input a page (256 words) that has been transmitted by the remote resource which initiated this instruction.

When the computer has received this instruction and is ready to accept the page, a continue command is sent to the resource specified by the contents of the active source register. The next 256 words from the input data queue are written into local memory starting with the memory location specified by the effective address of this instruction. After writing a word into location 255 of a page, the next word received is written into location 0 of the same page (page wraparound). If an EOB is received before 256 words are received or if the EOB is received after exactly 256 words are written, an error condition is signalled.

This instruction received as an external instruction interrupt, is not subject to address modification. This instruction is illegal if generated by a Program Counter fetch.

The appropriate transmission tags used in the sending of the page are explained in paragraph 3.1.1.

INSTRUCTION NAME: Write Array From Input

OP CODE: 0D

FUNCTION:

MACHINE FORMAT:

0				D				XXXX				I	INDEX			ADDRESS																M	O	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This word is a single word command and is generated as a result of a storage operation involving an array, whose starting address was obtained as a result of an indirect dimension word (see paragraph 5.3.13).

This instruction is used to write an array into local memory. The starting address is indicated by the effective address of this instruction. When the computer is ready to accept the operand, a continue is sent to the active source. Subsequently received words are written into local memory until an End of Block is received.

5. AADC ARITHMETIC PROCESSOR (AP)

5.0 The AADC Arithmetic Processor (AP)

The Arithmetic Processor (AP) has the capability to process data in a conventional manner and also has the added features of processing data arrays with single operators that may otherwise be represented by subroutines in conventional processors. The "built-in" operators include a matrix algebra facility that utilizes memory in an efficient manner by saving program instruction steps.

The architecture of the AP is characterized by "pipeline" techniques in which the instructions and operands are buffered on input to the AP and by an internal processor stack that can hold up to thirteen operator/operand pairs that are waiting for execution.

Deferral operations involving the accumulator stack can be extended into the PMU task memory as required. The stack mechanism eliminates the need for many of the LOAD/STORE instructions required in conventional digital processor applications and thus saves program memory and time.

Figure 11 represents the AP architecture.

5.1 AP System Components

The AP has several major components that are referenced extensively in the functional description of the instruction set. They are:

- The AP floating-point Fanout Box (FB)
- The AP Input Instruction/Data Queue or APQ

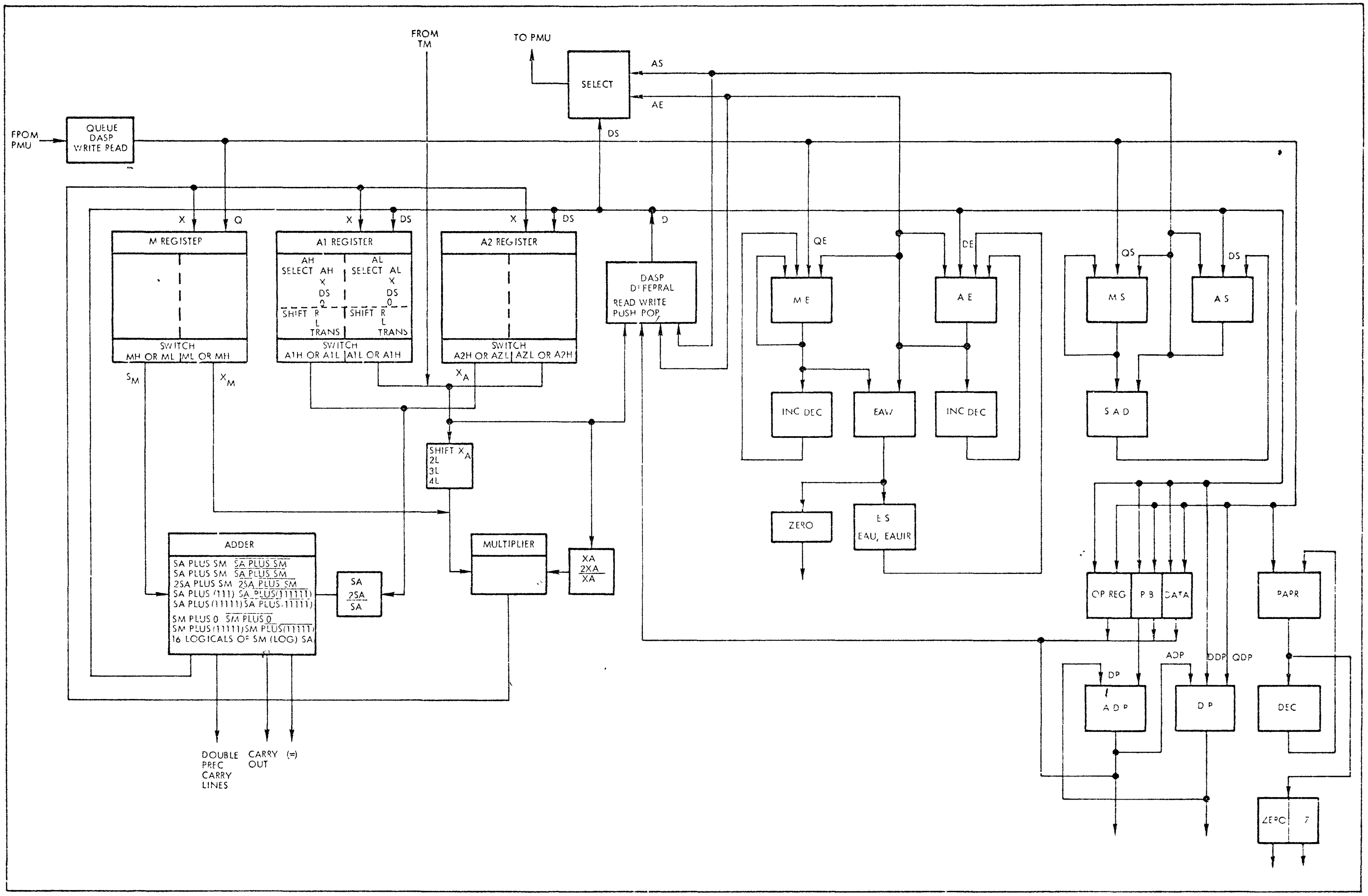


Figure 11. AP Block Diagram

- The AP Arithmetic and Control Unit (ACU) that includes:
    - The M Register
    - The A Register (Accumulator)
    - The Deferral Unit (DU) or Stack
    - The Array Control Logic

5.1.1 The AP Fanout Box (FB)

The FB converts data received from the Program Management Unit (PMU) into the desired internal AP format. After the conversion, the data is placed in the APQ. The FB retains the data tags in order to identify a double precision or complex operand process required for the conversion as contrasted with the single precision data.

The input to the FB is a thirty five (35) bit word with the right most three (3) bits being data tags identifying the data type (DT). The type codes in binary are as follows:

000	- logical variable
001	- array dimension control word
010	- integer variable
011	- logical variable
100	- single precision floating point variable
101	- double precision floating point variable
110	- complex floating point variable
111	- second word of two word variable

When data is passed from the AP to the PMU memory, it is reformatted with appropriate data tags appended. This occurs with the AP Store instructions as discussed below. The reformat device is called a "Select" box and is shown in Figure 11.

The contents of the AP Accumulator are stored in the effective address. If the Accumulator is complex or double precision, the operand stored is complex or double precision. If the Accumulator is single precision, the following actions occur:

If the Accumulator operand is positive, and the exponent is all ZERO's, the operand is stored as a logical with a data tag of 011.

If the Accumulator operand is negative, the exponent is all ZERO's, the sign is negative and the most significant Accumulator bit is ZERO, the operand is stored as an integer with a data tag of 010.

If the Accumulator exponent is +1 and the 5 most significant Accumulator bits are ZERO, the operand is shifted left one hex digit. The shifted operand will either be stored as a logical or integer as a function of the sign bit.

If the Accumulator exponent is not 0 or +1, and the 8 most significant Accumulator bits are all ZERO's, the Accumulator is stored as a single precision floating point number with a data tag of 100.

If the Accumulator exponent is not 0 or +1, and the 8 most significant Accumulator bits are not all ZERO's, the Accumulator is shifted right one hex digit and the exponent incremented. Then, the conditions stated above are applied.

When the Accumulator is stored in a floating point format, the 8 bit Accumulator exponent is converted into the 7 bit memory exponent format. A Store Error Trap occurs if the conversion is not exact. This occurs if the most significant two bits of the Accumulator exponent are not equal. The AP expands the 7 bit memory word floating point exponent by duplicating the most significant bit upon fetching the operand from the APQ. This eight bit exponent is used for all subsequent internal scaler/real manupulations.

The PMU may enter a store halt cycle (Store and Halt Instruction) after placing the store instruction in the APQ. When the AP completes the execution of the store instruction, the PMU store halt cycle is exited and normal instruction fetching is resumed.

5.1.2 AP INPUT FORMATS

The following diagrams represent the input formats to the FB:

Logical Variable:

DATA																																0 0 0			P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

or

DATA																																0 1 1			P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Where: The DATA is a logical variable.

Array Dimension Control Word:

SDL	SDH	UNUSED					S	DT				RANK				ROW								COL								0 0 1			P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Where: SDL and SDH are sign bits for the dimension word used for Take and Drop operators
S is a scaler bit for use in indirect dimension words only
DT is the array data type (see para. 5.3.12)
RANK is equal to one (1) for matrix and zero for vector
ROW is the Row index (0...255)
COL is the Column index (0...255)
P is the word parity

Integer Variable·

S	DATA																															0 1 0			P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Where: S is the sign bit. S = 1 is negative.
S = 0 is positive. Data is a 31 bit integer justified right in the field. The numbers +0 and -0 are equivalent in all algebraic manipulations.

Floating-Point Single Precision:

S	EXP							MANTISSA																								1	0	0	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Where: S is the mantissa sign bit. S = 1 is negative. S = 0 is positive
EXP is a two complement exponent
MANTISSA is an integer justified right in the field.

Floating-Point Double Precision:

S	EXP							MANTISSA																								1	0	1	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

MANTISSA 2nd PART																																1	1	1	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Where: The first word is the same as in single precision except for the data tags and the MANTISSA in the second word is thirty-two (32) bits long. The binary point is between the Mantissa of the double length operand.

Complex Variable:

S	EXP							MANTISSA																								1	1	0	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

S	EXP							MANTISSA																								1	1	1	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

Where: The first variable is REAL and the second variable is IMAGINARY. The data field are the same as Single Precision Floating-Point.

5.1.3 The AP Input Instruction/Data Queue (APQ)

The APQ is physically a dual stack of sixteen (16) words of FIFO (first in, first out) instructions and operands. The figure below represents the APQ format for a pair of entries in the dual stack:

W
0
•
•
•
15

OPERATION CODE (SAME AS IN PMU)												DATA TAGS (DT)				OPERAND FB OUTPUT TO APQ																																
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	49	55

For computational purposes, the APQ can look ahead up to thirteen (13) instruction/operand pairs.

The Data Tags have the following meaning:

0000 = single precision operand (includes logical, integer and single precision memory data tags)

0001 - double precision operand

0010 - complex operand

0100 to 0111 = not used

1000 = packed bianry with each bit in Mantissa representing one array word

1001 = packed quatenary with each two bits in Mantissa representing one array word

1010 = packed hexadecimal with eight characters in the Mantissa

1011 = four bytes in Mantissa

1100 = two sixteen (16) bit half words in Mantissa

1101 to 1111 not used

The APQ is loaded by the PMU as two separate parts of the dual entry: (1) the operation code and data tags in the left half of the queue and (2) the FB converted operands in the right half. Each half has a stack pointer to indicate where the bottom of the stack is located for PMU loading and a top of stack is located for AP unloading. The data tags tell the AP whether one or two operands are required for the operation code. The following diagram represents the APQ structure.

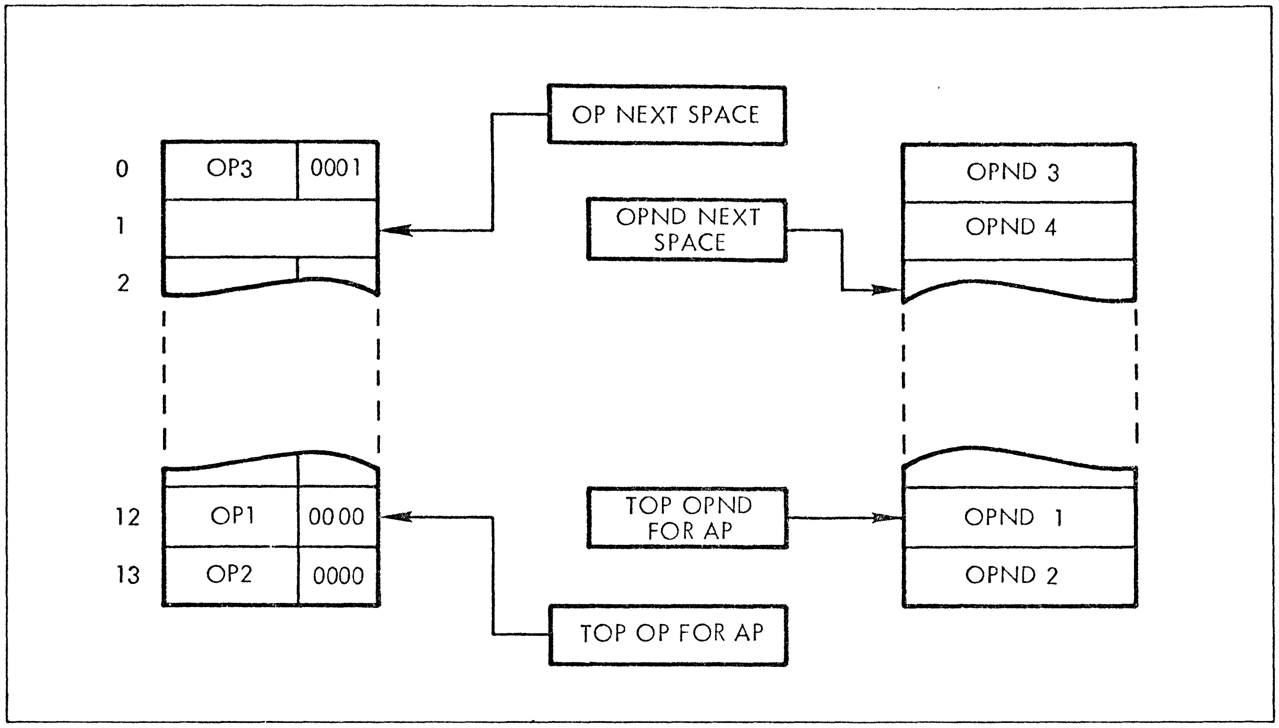


FIGURE 12 Instruction Look-Ahead Mechanism (APQ)

As the PMU loads the APQ a test is made for APQ full. If full, the PMU is interrupted and notified to halt APQ loading. After the top operand (OPND 1) is dequeued by the AP, the PMU is enabled to continue filling the APQ.

5.1.4 The AP Arithmetic Unit (APAU)

The AP Arithmetic Unit performs the arithmetic and logical operations on data furnished it by the PMU. The data furnished the AP passes through the APQ mechanism for the conventional instructions and directly to the AP for the array processing instructions.

Elements in the APAU include the M Register (Mantissa), the A Register (Accumulator), the Deferral Unit (DU) or Stack, and array control logic.

The M Register, A Register and Stack can be thought of as an eighteen register LIFO (last in, first out) stack of registers that contain the operators and associated operands to be executed. Putting information onto the stack and removing it from the Stack is controlled by bits 9-11 in the operator fetched from the APQ. These control bits are the "Parenthetical Field" bits as observed by the programmer in his application text. The Stack control bits 9-11 have the following meaning:

000 = Execute the operator immediately

001 = Execute (EX) the operator immediately, pop (P) an operator/operand from the Stack, and execute (EX) that operator/operand.

010 = (EX), P, (EX), P, (EX) - (two Pops) + (EX)

011 = (EX), P, (EX), P, (EX), P (EX), - (three Pops) + (EX)

100 = (EX), P, (EX), P, (EX), P, (EX), P, (EX) - (four Pops) + (EX)

101 = ... - (five Pops) + (EX)

110 = ... - (six Pops) + (EX)

111 = Push operator and operand onto Stack from A register. The M Register is copied to the A Register.

The deferral mechanism is represented by the following illustration showing the registers and the Stack pointers.

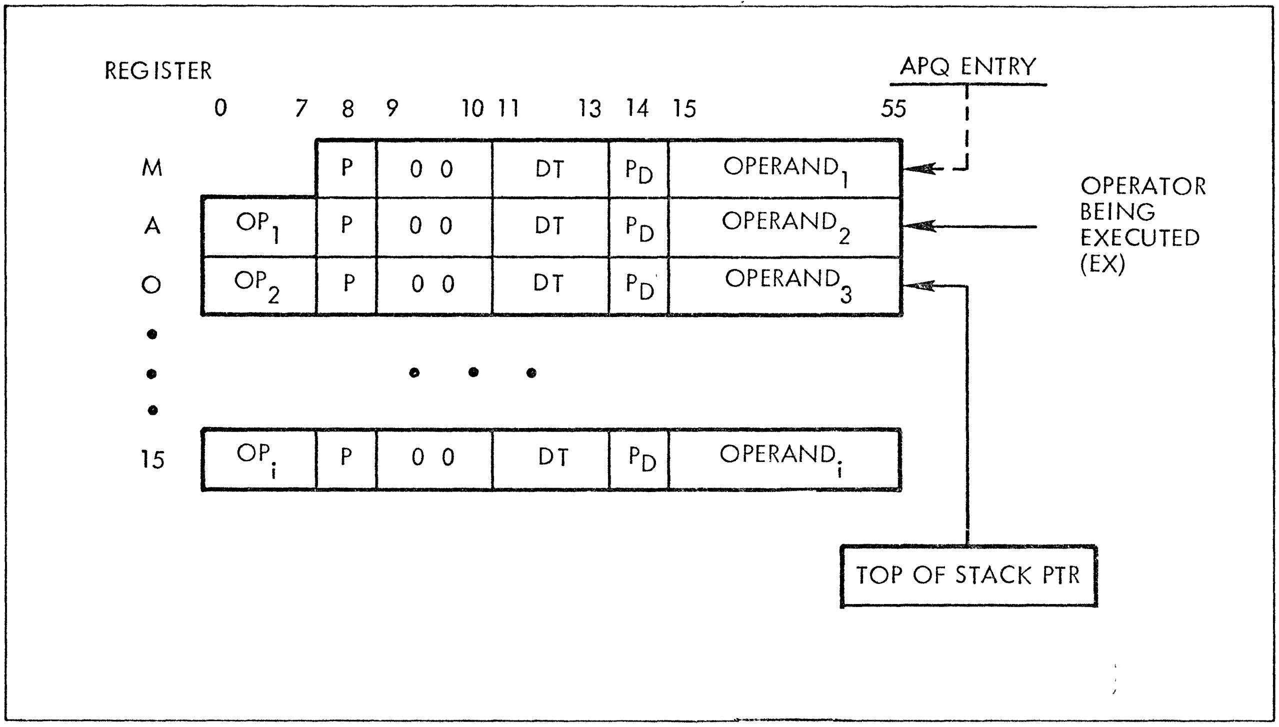


FIGURE 13 The Deferral Mechanism

where:

OPERAND is the data (1 bit sign, 8 bit exponent, 32 bit mantissa, if double mantissa is 64 bits)

DT is the DATA TYPE CODE where:

Bit 11 CP - Deferral contains Complex Operand

Bit 12 AR - Deferral contains Array

Bit 13 FA - First Array in Deferral

P is the precision of the operand resultant

P_D is the precision of the data

OP_1 is the operator to be executed

POP is moving contents of Top of Stack to A Register

Limit checks are made every time the Stack is accessed for over-flow or under-flow. When an over-flow condition exists, the AP stores the deferral stack in memory and signals this occurrence (see Appendix F, Note 8 - Trap Level 17). An under-flow (empty stack) or too many "Pops" causes the AP to retrieve the last contents of the deferral stack from memory, reload the deferral and execute the pop. A trap level 16 is initiated to signal the occurrence of this condition.

When the operator is actually being executed, it can be thought of as being in the A Register part of the Stack. In the hardware implementation, the operator is placed in the OP CODE register as shown in Figure 11 and the other control fields are gated to their respective registers for decoding in the execution process. The operand data in the A Register and M Register are then combined according to the OP CODE specified with the results saved in the A Register or Accumulator.

The data field of the Stack is increased from forty (40) bits in length to forty one (41) upon execution by extending (copying the sign bit) the exponent one bit to the left. This permits a multiple operator sequence to occur while reducing the possibility of exponent over-flow/under-flow before completing the OP CODE execution. When over-flow/under-flow of the exponent has occurred, the AP signals the PMU, completes the function execution, and continues.

When real double precision data is pushed, bits 0-23 of the first push is repeated. The most significant part of the mantissa is pushed first. When a complex or array operand is pushed, a single precision operand is pushed into the stack. The actual operand is pushed into the deferral memory stack maintained by scratchpad 26. In this manner, deferral underflow and overflow checks are maintained for complex or array operands.

The A and M registers are the arguments for most of the operators executed in the AP. The M Register is a double length register of seventy-three (73) bits, while the A Register is also double length with seventy-three (73) bits to accommodate double precision arithmetic. The A/M Register data format is:

<table>
<tr><td colspan="8">EXP</td><td>S</td><td colspan="32">OPERAND PART 1</td><td colspan="7">OPERAND PART 2</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td><td>36</td><td>37</td><td>38</td><td>39</td><td>40</td><td>41</td><td>42</td><td>43</td><td>44</td><td>45</td><td>46</td><td>72</td></tr>
</table>

where:

EXP is the eight bit exponent

S is the sign for the double length operand (OPND)

5.2 Basic AP Instruction Sequence

Each basic AP instruction which enters the APAU contains an operator code (OP Code), a parenthetical field (PF) and an operand (OPND). The operand is moved from the APQ to the M Register in the first step. Then the DT bits are examined for the double precision flag (bits 12-15 of APQ are 0001). The PF Stack functions are then executed as per the PF bits 9-11 in the OP CODE in the next and subsequent steps. Simultaneously with the last step of the execution sequence, the AP control fetches the next instruction from the APQ.

5.3 Basic AP Instruction Set

The basic AP instruction set has operators that use two arguments and are called dyadic functions. Instructions that have operators using one argument are called monadic functions. The instruction set may be thought of being in seven (7) groups or categories:

1. Arithmetic
2. Load/Store
3. Compare
4. Transfer
5. Shift
6. Boolean/Logical
7. Array

AP Instructions

The AP Instructions have two formats. The first format is in the PMU memory. The second format is a result of being operated on by the PMU and the AP Fanout Box (FB). In the description of the instruction set, both formats will be illustrated. For the examples given the following notation will be used:

M = M Register
(M) = Contents of M Register
A = A Register (Accumulator)
(A) = Contents of A Register
D = Effective Address

AP Machine Format represents structure of the instruction and data as is appears in the AP Queue.

Wherever in an instruction definition, the memory is "read", "referenced", "written", or "stored", unless otherwise noted, the memory operation is performed in the data addressing mode of the DPE, that is, absolute or virtual as determined by the PMU Set Task Parameter instruction.

The OP CODE field will be hexadecimal notation.

Examples in the use of the instruction will use the common arithmetic operator symbols and those of APL in array functions. Care will be taken to avoid using hardware register names in sample definitions of data and programs.

The PMU instruction format is discussed in the sections devoted to the PMU.

5.3.1 AP Arithmetic Instructions

INSTRUCTION NAME: Addition

OP CODE: C1

FUNCTION: (M) + (A) ⟶ A

PMU MACHINE FORMAT:

C				1				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

C				1				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	37	38	40	41	42	43	44	45	46	55

DEFINITION: The contents of the M Register are algebraically added to the contents of the A Register. The result is placed in A. Interrupts to the PMU may occur with exponent overflow/ underflow.

If bit 8 (P) is a one (1), the function executed results in a double precision operand. Otherwise, the low order resultant mantissa is cleared to zero for single precision.

The execution of the Addition operator is as follows: The exponents (with non-zero A or M operands) are compared and the operand with the larger exponent value is shifted left 4 bits at a time until a non-zero value appears in the left most four bits of the Mantissa, or the two exponents become equal by decrementing the larger exponent value by one for every four (4) bits shifted. In the event that the larger value becomes "normalized" to the left in the Mantissa before the two operands can be added, then, the second operand is shifted right four bits at a time, while adding one to its exponent for each shift of four bits, until the exponents are equal in value. Then, the addition is performed on the Mantissa with the result remaining in the Accumulator. If the shifting results in clearing any operand register or either operand was initially zero, the sum is the remaining operand.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	66	0	12345678	66	0	12345678
CONTENTS AFTER EXECUTION		0		66	0	2468ACF0

INSTRUCTION NAME: Reverse Subtract

OP CODE: C4

FUNCTION: (M) - (A)→ A

PMU MACHINE FORMAT:

C	4	P	PF	I	INDEX	ADDRESS	M	1	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

AP MACHINE FORMAT:

C	4	P	PF	DT	OPERAND
0 1 2 3	4 5 6 7	8	9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 … 55

DEFINITION: The contents of the A Register are algebraically subtracted from the contents of the M Register. The difference is placed in the A Register. Interrupts to the PMU may occur with exponent overflow/underflow.

If bit 8 (P) is a one (1), the resultant operand is double precision. Otherwise, the lower order mantissa is cleared to zero for single precision. The execution of the Reverse Subtract operator is as follows: The exponents, if any, are compared and the operand with the larger exponent value is shifted left four bits at a time until a non zero value appears in the left most four (4) bits of the Mantissa, or the two exponents become equal by decrementing the larger exponent value by one (1) for every four (4) bits shifted. In the event that the larger value becomes "normalized" to the left in the Mantissa before the two operands can be combined, then the second operand is shifted right four (4) bits at a time, while adding one to its exponent for each shift of four (4) bits, until the exponents are equal in value. Then, the function is performed with the result being placed in the Accumulator.

If shifting results in clearing any operand register or either operand was initially zero, the result is the remaining operand.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	66	0	2468ACF0	66	0	12345678
CONTENTS AFTER EXECUTION		0		66	0	12345678

INSTRUCTION: Subtract

OP CODE: C2

FUNCTION: (A) - (M)⟶A

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">C</td><td colspan="4">2</td><td>P</td><td colspan="3">PF</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>I</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

AP MACHINE FORMAT:

<table>
<tr><td colspan="4">C</td><td colspan="4">2</td><td>P</td><td colspan="3">PF</td><td colspan="4">DT</td><td colspan="31">OPERAND</td><td></td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td><td>36</td><td>37</td><td>38</td><td>39</td><td>40</td><td>41</td><td>42</td><td>43</td><td>44</td><td>45</td><td>46</td><td>55</td></tr>
</table>

DEFINITION: The contents of the M Register are algebraically subtracted from the contents of the A Register. The difference is placed in the A Register. Interrupts to the PMU may occur with exponent overflow/underflow.

If bit 8 (P) is a one (1), a double precision operand will result. Otherwise, the low order mantissa is cleared to zero and results in a single precision operand.

The execution of the Subtract operator is as follows: The exponents, if any, are compared and the operand with the larger exponent value is shifted left four (4) bits at a time until a non zero value appears in the left most four (4) bits of the Mantissa, or the two exponents become equal by decrementing the larger exponent value by one (1) for every four (4) bits shifted. In the event that the larger value becomes "normalized" to the left in the Mantissa before the two operands can be combined, then the second operand is shifted right four (4) bits at a time while adding one (1) to its exponent for each shift of four (4) bits until the exponents are equal in value. Then the function is performed with the result being placed in the Accumulator. If shifting results in clearing any operand register or either operand was initially zero, the result is the remaining operand.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	66	0	12345678	66	0	2468ACF0
CONTENTS AFTER EXECUTION		0		66	0	12345678

INSTRUCTION NAME: Multiply

OP CODE: E0

FUNCTION: (M) x (A)⟶A

PMU MACHINE FORMAT:

E				0				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

E				0				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the A Register are multiplied by the contents of the M Register. The product is placed in the A Register. Interrupts to the PMU may occur with exponent overflow/underflow. If bit 8 (P) is a one (1), the resultant operand is double precision. Otherwise, the lower order mantissa is cleared to zero and the resultant operand is single precision.

The execution of the Multiply operator is as follows: The exponents are algebraically added and placed in the A Register. The Mantissa are then algebraically multiplied four bits at a time until all multipliers and Multiplicand bits are processed with the result being placed in the A Register.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	04	0	00000002	04	0	00000002
CONTENTS AFTER EXECUTION				08	0	00000004

INSTRUCTION NAME: Reverse Divide

OP CODE: D8

FUNCTION: (M) ÷ (A) ⟶ A

PMU MACHINE FORMAT:

D				8				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

D				8				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the M Register are divided by the contents of the A Register. The quotient is placed in the Accumulator.

If bit 8 (P) is a one (1), the resultant operand is double precision. Otherwise, the lower order mantissa is cleared to zero and the resultant is single precision.

The execution of the Reverse Divide function may result in exponent overflow/underflow or division by zero and cause an interrupt to be sent to the PMU.

The Reverse Divide operator is executed as follows: The exponent of the A Register is algebraically subtracted from the exponent of the M Register with the result being placed in the Accumulator. The Mantissa of the M Register is divided by the Mantissa of the A Register with the quotient being placed in the Accumulator.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	08	0	00000004	04	0	00000002
CONTENTS AFTER EXECUTION				FD	0	20000000

INSTRUCTION NAME: Divide

OP CODE: D0

FUNCTION: (A) ÷ (M) → A

PMU MACHINE FORMAT:

D	0	P	PF	I	INDEX	ADDRESS	M	1	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

AP MACHINE FORMAT:

D	0	P	PF	DT	OPERAND
0 1 2 3	4 5 6 7	8	9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 ... 55

DEFINITION: The contents of the dividend in the A Register are divided by the divisor in the M Register. The quotient is placed in the Accumulator. If bit 8 (P) is a one (1), the resultant operand is double precision. Otherwise, the lower mantissa is cleared and the resultant is single precision. The execution of the Divide operator may cause an interrupt to be sent to the PMU for exponent overflow or an attempt to divide by zero.

The execution of the Divide operator is as follows: The exponent of the M Register is algebraically subtracted from the exponent of the A Register with the result being placed in the Accumulator. The Mantissa of the A Register is divided by the Mantissa of the M Register with the result placed in the Accumulator.

EXAMPLE

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	04	0	00000002	08	0	00000004
CONTENTS AFTER EXECUTION		0		FD	0	20000000

INSTRUCTION NAME: Divide Residue

OP CODE: D4

FUNCTION: $(A) \div (M);\ Q_R \longrightarrow A$

PMU MACHINE FORMAT:

D				4				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

D				4				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the A Register, the dividend, are divided by the contents of the M Register, the divisor. Interrupts are sent to the PMU if an attempt is made to divide by zero and A is negative or there is an exponent overflow/underflow. If bit 8 (P) is a one (1), the resultant operand is double precision. Otherwise, the lower order mantissa is cleared to zero and the resultant is single precision.

The execution of the Divide Residue operator is as follows: The exponent of the M Register is algebraically subtracted from the exponent of the A Register. The result is placed in the Accumulator. The Mantissa of the A Register is divided by the Mantissa of the M Register.

The residue (remainder) is normalized with corrected exponent in the Accumulator. The residue produced will always be a positive number or zero.
If A is negative, the divisor is added to the remainder to produce a positive residue.

EXAMPLES:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	22	0	00000005	22	0	00000008
CONTENTS AFTER EXECUTION				FD	0	30000000

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	22	0	00000005	22	1	00000008
CONTENTS AFTER EXECUTION				FD	0	20000000

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	22	1	00000005	22	1	00000008
CONTENTS AFTER EXECUTION				FD	0	20000000

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	22	1	00000005	22	0	00000008
CONTENTS AFTER EXECUTION				FD	0	30000000

INSTRUCTION NAME: Reverse Divide Residue

OP CODE: DC

FUNCTION: $(M) \div (A);\ Q_R \longrightarrow A$

PMU MACHINE FORMAT:

D				C				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

D				C				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the M Register, the dividend, are divided by the contents of the A Register, the divisor. The fraction part of the double length quotient, the residue, is placed in the Accumulator. Interrupts are sent to the PMU if an attempt is made to divide by zero or there is an exponent over/underflow.
If bit 8 (P) is a one (1), the resultant operand is double precision. Otherwise, the lower order mantissa is cleared to zero and the resultant is single precision.
The execution of the Reverse Divide Residue operator is as follows: The exponent of the A Register is algebraically subtracted from the exponent of the M Register. The Mantissa of the M Register is divided by the Mantissa of the A Register. The residue (remainder) is normalized with corrected exponent in the Accumulator. The residue produced will always be a positive number or zero. If M is negative, the divisor is added to the remainder to produce a positive number.

EXAMPLES

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	22	0	00000008	22	0	00000005
CONTENTS AFTER EXECUTION				FD	0	30000000

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	22	1	00000008	22	0	00000005
CONTENTS AFTER EXECUTION				FD	0	20000000

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	22	1	00000008	22	1	00000005
CONTENTS AFTER EXECUTION				FD	0	20000000

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	22	0	00000008	22	1	00000005
CONTENTS AFTER EXECUTION				FD	0	30000000

INSTRUCTION NAME: Divide Short

OP CODE: D2

FUNCTION: (A) ÷ (M); $Q_I \longrightarrow A$

PMU MACHINE FORMAT:

D	2	P	PF	I	INDEX	ADDRESS	M	1	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

AP MACHINE FORMAT:

D	2	P	PF	DT	OPERAND	
0 1 2 3	4 5 6 7	8	9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46	55

DEFINITION: The contents of the A Register are divided by the contents of the M Register with the integer portion of the quotient being placed in the Accumulator.

If bit 8 (P) is a one (1), the resultant operand is double precision. Otherwise, the lower order mantissa is cleared to zero for a single precision resultant operand. An interrupt is sent to the PMU if an attempt to divide by zero is made or there is an exponent overflow/ underflow. The execution of the Divide Short operator is as follows: The exponent of the M Register is algebraically subtracted from the exponent of the A Register and the result is placed in the Accumulator. The

A Register Mantissa is divided by the M Register Mantissa with the result being placed in the Accumulator. The quotient is then adjusted such that the exponent is equal to zero by shifting the Mantissa left or right four (4) bits at a time and decrementing or incrementing the exponent for each shift.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	22	0	00000008	22	0	00000010
CONTENTS AFTER EXECUTION				00	0	00000002

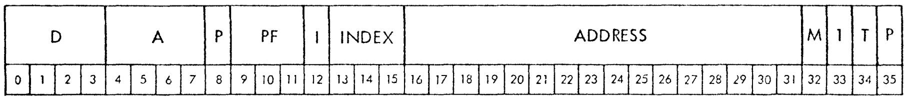
NOTE: If a divide by zero is attempted, an exponent overflow condition interrupt is sent to the PMU. The contents of the Accumulator Mantissa is unchanged, and the exponent of the Accumulator becomes the difference between the original exponents. The resulting sign bit is formed by the exclusive OR between the dividend and the divisor sign bits.

INSTRUCTION NAME: Reverse Divide Short

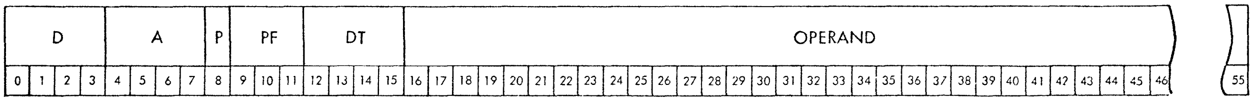
OP CODE: DA

FUNCTION: $(M) \div (A);\ Q_I \longrightarrow A$

PMU MACHINE FORMAT:



AP MACHINE FORMAT:



DEFINITION: The contents of the M Register are divided by the contents of the A Register with the integer portion of the quotient being placed in the Accumulator.

If bit 8 (P) is a one (1), the resultant operand is double precision. Otherwise, the lower order mantissa is cleared to zero and the resultant is single precision. An interrupt is sent to the PMU if an attempt to divide by zero is made or there is an exponent overflow/underflow.

The execution of the Reverse Divide Short operator is as follows: The exponent of the A Register is algebraically subtracted from the exponent of the M Register and the result is placed in the Accumulator.

The M Register Mantissa is divided by the A Register Mantissa with the result being placed in the Accumulator. The quotient is then adjusted such that the exponent is equal to zero by shifting the Mantissa left or right four (4) bits at a time and decrementing or incrementing the exponent for each shift.

EXAMPLE

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	22	0	00000010	22	0	00000008
CONTENTS AFTER EXECUTION				00	0	00000002

NOTE: If a divide by zero is attempted, an exponent overlfow condition interrupt is sent to the PMU. The contents of the Accumulator Mantissa become the dividend Mantissa and the exponent of the Accumulator contains the difference between the dividend the divisor exponents. The resulting sign bit is the exclusive OR between the dividend the divisor sign bits.

INSTRUCTION NAME: Load Accumulator

OP CODE: C5

FUNCTION: $(M_D) \rightarrow M$, then $(M) \rightarrow A$

PMU MACHINE FORMAT:

C	5	P	PF	I	INDEX	ADDRESS	M	1	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

AP MACHINE FORMAT:

C	5	P	PF	DT	OPERAND
0 1 2 3	4 5 6 7	8	9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 55

DEFINITION: The contents of the M Register are copied into the Accumulator.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	BB	1	76543210	FF	1	FFFF0000
CONTENTS AFTER EXECUTION	BB	1	76543210	BB	1	76543210

INSTRUCTION NAME: Load Negative

OP CODE: CA

FUNCTION: $(M_D) \longrightarrow M$, then $-(M) \longrightarrow A$

PMU MACHINE FORMAT:

C				A				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

C				A				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The algebraic negative of the contents of the M Register replaces the contents of the Accumulator.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	88	0	12345678	66	0	87654321
CONTENTS AFTER EXECUTION	88	0	12345678	88	1	12345678

INSTRUCTION NAME: Negation

OP CODE: CC

FUNCTION: -(A) ⟶ A

PMU MACHINE FORMAT:

C				C				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

C				C				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The algebraic negative of the contents of the A Register replaces the contents of the Accumulator.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	BB	0	AABBCCDD	33	0	33221100
CONTENTS AFTER EXECUTION	BB	0	AABBCCDD	33	1	33221100

INSTRUCTION NAME: Absolute Value

OP CODE: CF

FUNCTION: $|(A)| \longrightarrow A$

PMU MACHINE FORMAT:

C				F				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

C				F				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the Accumulator are set to a positive value.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	66	0	11223344	44	1	AABBCCDD
CONTENTS AFTER EXECUTION	66	0	11223344	44	0	AABBCCDD

INSTRUCTION NAME: Signum

OP CODE: B0

FUNCTION: If (A) < 0; -1 → A. If (A) = 0; 0 → A

If (A) > 0; 1 → A.

PMU MACHINE FORMAT:

B				0				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

B				0				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the contents of the Accumulator is less than zero, the Accumulator is set to minus one (-1). If the contents of the Accumulator is equal to zero, the Accumulator remains unchanged. If the contents of the Accumulator is greater than zero, the Accumulator is set to plus one (+1).

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	06	0	12345678	11	1	01234567
CONTENTS AFTER EXECUTION	C6	0	12345678	00	1	00000001

INSTRUCTION NAME: Floor

OP CODE: E4

FUNCTION: If $A_I - 1 < (A) < A_I$; $A_I - 1 \longrightarrow A$

If $(A) = (A_I)$; $A_I \longrightarrow A$

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">E</td><td colspan="4">4</td><td>P</td><td colspan="3">PF</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>1</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

AP MACHINE FORMAT:

<table>
<tr><td colspan="4">E</td><td colspan="4">4</td><td>P</td><td colspan="3">PF</td><td colspan="4">DT</td><td colspan="32">OPERAND</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td><td>36</td><td>37</td><td>38</td><td>39</td><td>40</td><td>41</td><td>42</td><td>43</td><td>44</td><td>45</td><td>46</td><td>55</td></tr>
</table>

DEFINITION: If the contents of the A Register are less than the integerized value of the Accumulator; then, the integerized value minus one (1) is placed in the Accumulator.

If the (A) is already integerized, there is no change to the Accumulator.

EXAMPLE:

<table>
<tr><td></td><td colspan="3">M REGISTER</td><td colspan="3">A REGISTER</td></tr>
<tr><td>CONTENTS BEFORE EXECUTION</td><td>AA</td><td>0</td><td>BBCCDDEE</td><td>FF</td><td>0</td><td>00000041</td></tr>
<tr><td>CONTENTS AFTER EXECUTION</td><td>AA</td><td>0</td><td>BBCCDDEE</td><td>00</td><td>0</td><td>00000004</td></tr>
</table>

INSTRUCTION NAME: Ceiling

OP CODE: E5

FUNCTION: If $A_I + 1 > (A) > A_I$; $A_I + 1 \longrightarrow A$

If $(A) = (A_I)$; $A_I \longrightarrow A$

PMU MACHINE FORMAT:

E	5	P	PF	I	INDEX	ADDRESS	M	1	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

AP MACHINE FORMAT:

E	5	P	PF	DT	OPERAND	
0 1 2 3	4 5 6 7	8	9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46	55

DEFINITION: If the contents of the A Register are greater than the integerized value of the Accumulator; then, the integerized value plus one (1) is placed in the Accumulator.

If the (A) is already integerized, there is no change to the Accumulator.

EXAMPLE

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	AA	0	BBCCDDEE	FF	0	00000041
CONTENTS AFTER EXECUTION	AA	0	BBCCDDEE	00	0	00000005

INSTRUCTION NAME: Square Root

OP CODE: E2

FUNCTION: $\sqrt{|(A)|} \longrightarrow A$

PMU MACHINE FORMAT:

E				2				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

E				2				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The square root of the absolute value of the contents of the A Register replaces the contents of the Accumulator. The resultant sign is zero (positive).

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	11	0	EFAB1212	00	0	00000009
CONTENTS AFTER EXECUTION	11	0	EFAB1212	00	0	00000003

INSTRUCTION NAME: Normalize

OP CODE: F4

FUNCTION: (A) Normalized ⟶ A

PMU MACHINE FORMAT:

F				4				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

F				4				P	PF			DT				OPERAND																																
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	49	55

DEFINITION: The contents of the Accumulator Mantissa is shifted left four bits at a time, while decrementing the exponent, until the high order four bits contains at least one non zero bit. If the Accumulator is zero, no normalization occurs. An interrupt may be sent to the PMU if the exponent underflows.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	32	1	87654321	21	0	00445566
CONTENTS AFTER EXECUTION	32	1	87654321	IF	0	44556600

5.3.2 LOAD/STORE INSTRUCTIONS

INSTRUCTION NAME: Store and Halt

OP CODE: E8

FUNCTION $(A) \rightarrow M_D$

PMU MACHINE FORMAT:

E	8	P	PF	I	INDEX	ADDRESS	M	1	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

AP MACHINE FORMAT:

E	8	P	PF	DT	OPERAND
0 1 2 3	4 5 6 7	8	9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 49 55

DEFINITION: The contents of the A Register are formatted for the PMU and stored at the effective address held in the M Register.

The formatting rules for the PMU are discussed in section 5.1.1. The PMU may be sent an interrupt if the formatting results in an error. The PMU is temporarily halted while the AP performs the store function.

EXAMPLE

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	0	00001FFF	00	0	12345678
CONTENTS AFTER EXECUTION	00	0	00001FFF	00	0	12345678

- When the Store and Halt instruction is executed with an Array or complex accumulator, the precision bit (instruction bit 8) has no effect. Thus, the accumulator is unchanged.

INSTRUCTION NAME: Store and Proceed

OP CODE: EA

FUNCTION: $(A) \longrightarrow M_D$

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">E</td><td colspan="4">A</td><td>P</td><td colspan="3">PF</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>1</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

AP MACHINE FORMAT:

<table>
<tr><td colspan="4">E</td><td colspan="4">A</td><td>P</td><td colspan="3">PF</td><td colspan="4">DT</td><td colspan="33">OPERAND</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td><td>36</td><td>37</td><td>38</td><td>39</td><td>40</td><td>41</td><td>42</td><td>43</td><td>44</td><td>45</td><td>46</td><td>49</td><td>55</td></tr>
</table>

DEFINITION: The contents of the A Register are formatted for the PMU and stored at the effective address held in the M Register.

The formatting rules for the PMU are discussed in section 5.1.1. The PMU may be sent an interrupt if the formatting results in an error. The PMU continues to process information while the AP performs the store function.

EXAMPLE:

<table>
<tr><td></td><td colspan="3">M REGISTER</td><td colspan="3">A REGISTER</td></tr>
<tr><td>CONTENTS BEFORE EXECUTION</td><td>00</td><td>0</td><td>00001FFF</td><td>00</td><td>0</td><td>12345678</td></tr>
<tr><td>CONTENTS AFTER EXECUTION</td><td>00</td><td>0</td><td>00001FFF</td><td>00</td><td>0</td><td>12345678</td></tr>
</table>

- When the Store and Proceed instruction is executed with an Array or complex accumulator, the precision bit (instruction bit 8) has no effect. Thus, the accumulator is unchanged.

INSTRUCTION NAME: Load Memory Word

OP CODE: 31

FUNCTION: $M_D \longrightarrow A$

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">3</td><td colspan="4">1</td><td>P</td><td colspan="3">PF</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>1</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: This instruction is normally addressable. The first word of the operand addressed by the effective address in the PMU format is treated as a logical operand and is loaded into the AP Accumulator. During the execution of this instruction in the PMU, a Load Word To Accumulator instruction (Op Code B9) is created and placed in the APQ with the operand for final execution by the AP.

The data tag of the operand is not interpreted. Therefore, the first word of a double precision number or a dimension word can be treated as a single precision logical operand.

Note: The previous accumulator is destroyed. The new accumulator contains a logical real operand.

INSTRUCTION NAME: Store Packed

OP CODE: 32

FUNCTION: $(A) \longrightarrow M_D$

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">3</td><td colspan="4">2</td><td colspan="4">DATA TAG</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>1</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: A dimension word indicating packed data is stored away into the location indicated by the effective address of the instruction.

The accumulator is reshaped into a matrix whose number of rows is a function of the packed array operand. The number of columns is:

2	-	packed	halfword
4	-	"	byte
8	-	"	hex digit
16	-	"	2 bit
32	-	"	binary

After completion of the new form of Store Packed, the instructions transpose, inner product, and then Store would be executed to actually pack the accumulator and store the packed array away.

The purpose of the inner product is to scale the accumulator array. For example, if packed byte is required, the memory operand is specified as a 4 element vector, with elements, (2*0), (2*8), (2*16), (2*24), the opcodes are Add and Multiply. The multiplication shifts the desired operand and the Add places that operand in the correct space in the accumulator.

It should be noted that via this approach all packed arrays subsequently accessed must use indirect dimension words. The correct dimension word was stored via the storepacked instruction in the indirect dimension word location.

It should also be noted that via this technique nonhomogeneous operands can be packed into a word. Thus a table can be constructed with packed 6, 10, 4, and 12 bit operand in a word. The scaling vector of the inner product would supply the appropriate powers of 2. Ths array would however, have to be unpacked via some software program.

INSTRUCTION NAME: Load Deferral

OP CODE: 33

FUNCTION: $M_D \longrightarrow$ Deferral Stack

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">3</td><td colspan="4">3</td><td colspan="4">SPA</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>1̄</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: The contents of the scratchpad location referenced by the SPA field are decremented by one. The contents of the 32 memory locations starting with this new value and working backwards are stored into the AP deferral stack. The sequence of loading is deferral location 15 high order part (24 bits taken from bits 8-31 of the memory word) deferral 15 lower order (32 bits), down to deferral 0 low order part.

The scratchpad location referenced by the SPA will be at a value 32 less than when it started upon completion of the instruction. This instruction is normally non addressable.

Note: The starting deferral location written into is the present value of the AP deferral address pointer. This address pointer is decremented 16 times. Wraparound occurs when the pointer goes from 0 to 15. Thus, the final value is the initial deferral value.

INSTRUCTION NAME: Store Deferral

OP CODE: 34

FUNCTION: (Deferred Stack) → M_D

PMU MACHINE FORMAT:

3				4					SPA			I	INDEX			ADDRESS																$\bar{M}$	1	T	$\bar{P}$
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The contents of the AP deferral are stored into 32 sequential memory locations beginning with the address pointed to by the scratchpad location referenced by the SPA FIELD. The first word stored is deferral location 0 low order part (32 bits), deferral location 0 high order part (24 bits placed in bits 8-31), up to defeyral 15 high order part. Deferral low order part represents the mantissa. Deferral high order part represents deferral opcode, control, exponent & sign.

The scratchpad location referenced by the SPA FIELD will have a value 32 higher than when it started.

INSTRUCTION NAME: Unpack

OP CODE: B8

FUNCTION: Unpack Accumulator

PMU MACHINE FORMAT:

B				8				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

B				8				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: This instruction is generated internally by the PMU to the AP when packed data is encountered. It causes the AP to take the M mantissa (32 bits) and return these bits as collections of 32-bit words with a specified number of bits of the M mantissa in the lowest order bit positions and all ZERO's in the unused positions. Depending on control information from the PMU, the AP will put 1,2,4,8, or 16 bits of the M mantissa in each 32-bit word. It treats these groups from the left and places the left-most group of M bits into a word which it then returns to the PMU in the form of an internal control sequence.

If this instruction is generated due to a program counter fetch, the AP will halt to await PMU servicing resulting in both a PMU and AP halt.

INSTRUCTION NAME: LOAD WORD TO ACCUMULATOR

OP CODE: B9

FUNCTION: $M_D \longrightarrow A$

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">B</td><td colspan="4">9</td><td>P</td><td colspan="3">PF</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>1</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

AP MACHINE FORMAT:

<table>
<tr><td colspan="4">B</td><td colspan="4">9</td><td>P</td><td colspan="3">PF</td><td colspan="4">DT</td><td colspan="32">OPERAND</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td><td>36</td><td>37</td><td>38</td><td>39</td><td>40</td><td>41</td><td>42</td><td>43</td><td>44</td><td>45</td><td>46</td><td>55</td></tr>
</table>

DEFINITION: This instruction causes a Logical Load (M mantissa to A mantissa, A exponent and Sign cleared to zero) of M to A.

It is generated by Load Memory Word instruction (see Op Code 31) execution in the PMU.

If generated by program or interrupt, this instruction will cause a Logical Load, identical to Op Code 85.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	0	12345678	32	1	87654321
CONTENTS AFTER EXECUTION	00	0	12345678	00	0	12345678

- This instruction should never be generated by a program or interrupt when the DPE is processing array or complex operands.

INSTRUCTION NAME: Push Data

OP CODE: 35

FUNCTION: $(A) \longrightarrow M_D$

PMU MACHINE FORMAT:

1				4				SPA				I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The contents of the scratchpad referenced by bits 8-11 are incremented by one. The full word operand referenced by the effective address is stored in the memory location indicated by the incremented scratchpad register. If the effective address does not reference local memory, a Read Operand instruction is formatted to the channel. The incremented value of R is restored to R. If more than one operand is fetched, as indicated by the data tag field, R is reincremented and an operand stored in memory for each additional word accessed. The operand fetch of this instruction is sensitive to the data tag (bits 32-34) of the operand.

INSTRUCTION NAME: Store Operand

OP CODE: 36

FUNCTION: $(A) \longrightarrow M_D$

PMU MACHINE FORMAT:

1				5				SPA				I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The contents of the scratchpad indicated by bits 8-11 of the instruction is used as an address to reference a full word operand. The source operand is stored in the memory location indicated by the effective address. If the data tag of the source operand indicates that more than one word is to be stored, the additional words indicated are stored in consecutive memory locations.

If the effective address is not local, a WRITE OPERAND FROM INPUT is transmitted to the bus. If more than one word is to be stored, the originating PMU waits for the receipt of a Continue. Upon receipt of the Continue, the additional words to be stored are transmitted to the channel with the tag 010. The last word to be transmitted is tagged 011.

The destination of the words to be transmitted is kept in the ISOURCE register. The ISOURCE register is loaded with bits 12-19 of the original WRITE OPERAND FROM INPUT INSTRUCTION.

5.3.3 COMPARE AND TVD INSTRUCTIONS

INSTRUCTION NAME: Compare Less Than Destructive

OP CODE: 92

FUNCTION: If (M) $<$ (A); +1 $\longrightarrow$ A, TVD Set
If (M) $\geq$ (A); 0 $\longrightarrow$ A, TVD Reset

PMU MACHINE FORMAT:

9				2				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				2				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the Accumulator are algebraically compared to the contents of the M Register. If the contents of the M Register is less than the Accumulator, the Accumulator is set to the integer one (+1) and the Test Valid (TVD) flip-flop is set. Otherwise, the Accumulator is cleared to zero and the TVD flip-flop is reset.

EXAMPLE

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	22	0	12345678	11	0	12345678
CONTENTS AFTER EXECUTION	22	0	12345678	00	0	00000000

INSTRUCTION NAME: Compare Equal Non Destructive

OP CODE: 91

FUNCTION: If (M) = (A); TVD Set

If (M) ≠ (A); TVD Reset

PMU MACHINE FORMAT:

9				1				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				1				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the M Register are algebraically compared to the contents of the Accumulator. If the contents of the M Register are equal to the Accumulator the Test Valid (TVD) flip-flop is set. Otherwise, the TVD flip-flop is reset.

INSTRUCTION NAME: Compare Less Than or Equal Non Destructive

OP CODE: 93

FUNCTION: If (M) $\leq$ (A); TVD Set

If (M) > (A); TVD Reset

PMU MACHINE FORMAT:

9				3				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				3				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the M Register are algebraically compared to the contents of the Accumulator. If the contents of the M Register are less than or equal to the Accumulator, the Test Valid (TVD) flip-flop is set. Otherwise, the TVD flip-flop is reset.

INSTRUCTION NAME: Compare Greater Than Destructive

OP CODE: 94

FUNCTION: If (M) > (A); +1 → A, TVD Set

If (M) $\leq$ (A); 0 → A, TVD Reset

PMU MACHINE FORMAT:

9				4				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				4				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the Accumulator are algebraically compared to the contents of the M Register. If the contents of the M Register are greater than the Accumulator, the contents of the Accumulator are set to one (+1) and the Test Valid (TVD) flip-flop is set. Otherwise, the Accumulator is cleared to zero and the TVD flip-flop is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	0	12345678	00	0	00112233
CONTENTS AFTER EXECUTION	00	0	12345678	00	0	00000001

INSTRUCTION NAME: Compare Greater Than or Equal Non Destructive

OP CODE: 95

FUNCTION: If (M) $\geq$ (A); TVD Set

If (M) $<$ (A); TVD Reset

PMU MACHINE FORMAT:

9				5				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				5				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the M Register are compared algebraically with the contents of the Accumulator. If the contents of the M Register are greater or equal to the Accumulator, the Test Valid (TVD) flip-flop is set. Otherwise, the TVD flip-flop is reset.

INSTRUCTION NAME: Compare Not Equal Destructive

OP CODE: 96

FUNCTION: If (M) ≠ (A); +1 → A, TVD Set

If (M) = (A); 0 → A, TVD Reset

PMU MACHINE FORMAT:

9				6				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				6				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the Accumulator are algebraically compared to the contents of the M Register. If the contents of the M Register is not equal to the Accumulator, the contents of the Accumulator is set to one (+1) and the Test Valid (TVD) flip-flop is set. Otherwise, the Accumulator is cleared to zero and the TVD flip-flop is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	0	12345678	00	0	87654321
CONTENTS AFTER EXECUTION	00	0	12345678	00	0	00000001

INSTRUCTION NAME: Set TVD Non Destructive

OP CODE: 97

FUNCTION: TVD Set

PMU MACHINE FORMAT:

9				7				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				7				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The Test Valid (TVD) flip-flop is set.

INSTRUCTION NAME: Reset TVD Non Destructive

OP CODE: 98

FUNCTION: TVD Reset

PMU MACHINE FORMAT:

9				8				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				8				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The Test Valid flip-flop is reset.

INSTRUCTION NAME: Compare Equal Destructive

OP CODE: 99

FUNCTION: If (M) = (A); +1 ⟶ A, TVD Set

If (M) ≠ (A); 0 ⟶ A, TVD Reset

PMU MACHINE FORMAT:

9				9				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				9				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the Accumulator are algebraically compared with the contents of the M Register. If the contents of the M Register is equal to the Accumulator, the contents of the Accumulator are set to one (+1) and the Test Valid (TVD) flip-flop is set. Otherwise, the Accumulator is cleared to zero and the TVD flip-flop is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	0	12345678	00	0	12345678
CONTENTS AFTER EXECUTION	00	0	12345678	00	0	00000001

INSTRUCTION NAME: Compare Less Than Non Destructive

OP CODE: 9A

FUNCTION: If (M) $<$ (A); TVD Set

If (M) $\geq$ (A); TVD Reset

PMU MACHINE FORMAT:

9				A				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				A				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the M Register are compared algebraically with the contents of the Accumulator. If the contents of the M Register is less than the Accumulator, the Test Valid (TVD) flip-flop is set. Otherwise, the TVD flip-flop is reset.

INSTRUCTION NAME: Compare Less Than or Equal Destructivc

OP CODE: 9B

FUNCTION: If (M) $\leq$ (A); +1 ⟶ A, TVD Set

If (M) $>$ (A); 0 ⟶ A, TVD Reset

PMU MACHINE FORMAT:

9	B	P	PF	I	INDEX	ADDRESS	M	1	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

AP MACHINE FORMAT:

9	B	P	PF	DT	OPERAND
0 1 2 3	4 5 6 7	8	9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 55

DEFINITION: The contents of the Accumulator are compared algebraically with the contents of the M Register. If the contents of the M Register is less than or equal to the Accumulator, the contents of the Accumulator are set to one (+1) and the Test Valid (TVD) flip-flop is set. Otherwise, the Accumulator is cleared to zero and the TVD flip-flop is reset.

EXAMPLE :

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	1	01234567	00	0	01234567
CONTENTS AFTER EXECUTION	00	1	01234567	00	0	00000001

INSTRUCTION NAME: Compare Greater Than Non Destructive

OP CODE: 9C

FUNCTION: If (M) > (A); TVD Set

If (M) ≤ (A); TVD Reset

PMU MACHINE FORMAT:

9				C				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				C				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the M Register are compared algebraically with the contents of the Accumulator. If the contents of the M Register are greater than the Accumulator, the Test Valid (TVD) flip-flop is set. Otherwise, the TVD flip-flop is reset.

INSTRUCTION NAME: Compare Greater Than or Equal Destructive

OP CODE: 9D

FUNCTION: If (M) $\geq$ (A); +1 → A, TVD Set

If (M) $<$ (A); 0 → A, TVD Reset

PMU MACHINE FORMAT:

9				D				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				D				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the Accumulator are algebraically compared to the contents of the M Register. If the contents of the M Register is greater than or equal to the Accumulator, the contents of the Accumulator are set to one (+1) and the Test Valid (TVD) flip flop is set. Otherwise, the Accumulator is cleared to zero and the TVD flip-flop is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	0	12345678	00	0	87654321
CONTENTS AFTER EXECUTION	00	0	12345678	00	0	0000000

INSTRUCTION NAME: Compare Not Equal Non Destructive

OP CODE: 9E

FUNCTION: If (M) $\neq$ (A); TVD Set

If (M) = (A); TVD Reset

PMU MACHINE FORMAT:

9				E				P	PF			I	INDEX			ADDRESS																M	I	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				E				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the M Register are compared algebraically with the contents of the Accumulator. If the contents of the M Register are not equal to the Accumulator, the Test Valid (TVD) flip-flop is set. Otherwise, the TVD flip-flop is reset.

INSTRUCTION NAME: Set TVD Destructive

OP CODE: 9F

FUNCTION: TVD Set, +1 ⟶ A

PMU MACHINE FORMAT:

9				F				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				F				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the Accumulator are set to one and the Test Valid (TVD) flip-flop is set.

INSTRUCTION NAME: Reset TVD Destructive

OP CODE: 90

FUNCTION: TVD Reset, 0 ⟶ A

PMU MACHINE FORMAT:

9				0				P	PF			I	INDEX																			M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

9				0				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the Accumulator are cleared to zero and the Test Valid (TVD) flip-flop is reset.

INSTRUCTION NAME: Minimum

OP CODE: B2

FUNCTION: If $(A) > (M)$; $(M) \longrightarrow A$

PMU MACHINE FORMAT:

B				2				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

B				2				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the M Register and the Accumulator are compared algebraically. If the contents of the Accumulator are greater than the M Register, the contents of the Accumulator are replaced with the contents of the M Register. Otherwise, the Accumulator remains unchanged.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	0	12345678	00	0	87654321
CONTENTS AFTER EXECUTION	00	0	12345678	00	0	12345678

INSTRUCTION NAME: Maximum

OP CODE: B4

FUNCTION: If (M) $>$ (A); (M) $\longrightarrow$ A

PMU MACHINE FORMAT:

B				4				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

B				4				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the M Register are compared algebraically with the contents of the Accumulator. If the contents of the M Register are greater than the Accumulator, the contents of the Accumulator are replaced by the contents of the M Register. Otherwise, the Accumulator remains unchanged.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	0	87654321	00	0	12345678
CONTENTS AFTER EXECUTION	00	0	87654321	00	0	87654321

5.3.4 TRANSFER INSTRUCTIONS

INSTRUCTION NAME: No Transfer

OP CODE: A0 or A8

FUNCTION: NOP

PMU MACHINE FORMAT:

A				0				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

A				0				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: No operation. The PMU recognizes that this AP "transfer" instruction is present in the APQ. Subsequent fetchcycle sequencing may be affected.

INSTRUCTION NAME: Transfer on Equal to Zero

OP CODE: A1 or A9

FUNCTION: If (A) = 0; (M) ⟶ P

If (A) ≠ 0; (P) +1 ⟶ P

PMU MACHINE FORMAT:

A				1				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

A				1				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the contents of the Accumulator are equal to zero, the contents of the least significant 16 bits of the M Register are placed in the program counter (P). Otherwise, no operation is performed.

INSTRUCTION NAME: Transfer on Greater Than Zero

OP CODE: A2 or AA

FUNCTION: If $(A) > 0;\ (M) \longrightarrow P$

If $(A) \leq 0;\ (P) +1 \longrightarrow P$

PMU MACHINE FORMAT:

A				2				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

A				2				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the contents of the Accumulator are greater than zero, the contents of the least significant 16 bits of the M Register are placed in the program counter (P). Otherwise, no operation is performed.

INSTRUCTION NAME: Transfer on Greater Than or Equal to Zero

OP CODE: A3 or AB

FUNCTION: If (A) ≥ 0; (M) ⟶ P

If (A) < 0; (P) +1 ⟶ P

PMU MACHINE FORMAT:

A				3				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

A				3				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the contents of the Accumulator are greater than or equal to zero, the contents of the least significant 16 bits of the M Register are placed in the program counter (P). Otherwise, no operation is performed.

INSTRUCTION NAME: Transfer on Less Than Zero

OP CODE: A4 or AC

FUNCTION: If (A) $<$ 0; (M) $\longrightarrow$ P

If (A) $\geq$ 0; (P) +1 $\longrightarrow$ P

PMU MACHINE FORMAT:

A	4	P	PF	I	INDEX	ADDRESS	M	1	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

AP MACHINE FORMAT:

A	4	P	PF	DT	OPERAND	
0 1 2 3	4 5 6 7	8	9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46	55

DEFINITION: If the contents of the Accumulator are less than zero, the contents of the least significant 16 bits of the M Register are placed in the program counter (P). Otherwise, no operation is performed.

INSTRUCTION NAME: Transfer on Less Than or Equal to Zero

OP CODE: A5 or AD

FUNCTION: If $(A) \leq 0;\ (M) \longrightarrow P$

$\qquad\qquad$ If $(A) > 0;\ (P) + 1 \longrightarrow P$

PMU MACHINE FORMAT:

A	5	P	PF	I	INDEX	ADDRESS	M	1	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

AP MACHINE FORMAT:

A	5	P	PF	DT	OPERAND
0 1 2 3	4 5 6 7	8	9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46) (55

DEFINITION: If the contents of the Accumulator are less than or equal to zero, the contents of the least significant 16 bits of the M Register are placed in the program counter (P). Otherwise, no operation is performed.

INSTRUCTION NAME: Transfer on Not Equal to Zero

OP CODE: A6 or AE

FUNCTION: $\text{If } (A) \neq 0;\ (M) \longrightarrow P$

$\text{If } (A) = 0;\ (P) + 1 \longrightarrow P$

PMU MACHINE FORMAT:

A	6	P	PF	I	INDEX	ADDRESS	M	1	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

AP MACHINE FORMAT:

A	6	P	PF	DT	OPERAND	
0 1 2 3	4 5 6 7	8	9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46	55

DEFINITION: If the contents of the Accumulator are not equal to zero, the contents of the least significant 16 bits of the M Register are placed in the program counter (P). Otherwise, no operation is performed.

INSTRUCTION NAME: Unconditional Transfer

OP CODE: A7 or AF

FUNCTION: (M) ⟶ P

PMU MACHINE FORMAT:

A				7				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

A				7				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the least significant 16 bits of the M Register are placed in the program counter (P). Control is passed to the instruction address specified in P.

INSTRUCTION NAME: Transfer on Test Valid Set

OP CODE: BC

FUNCTION: If TVD Set; (M)⟶P

If TVD Reset; (P) +1 ⟶P

PMU MACHINE FORMAT:

B				C				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

B				C				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the Test Valid flip-flop is set, the contents of the least significant 16 bits of the M Register are placed in the program counter (P). Otherwise, P is incremented by one and control is passed to the instruction specified by P.

INSTRUCTION NAME: Execute

OP CODE: 27

FUNCTION:

PMU MACHINE FORMAT:

2				7				XXXX				I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The contents of the memory location specified by the effective address of the instruction are treated as an instruction. Instruction sequencing will return to normal operation at the conclusion of this instruction unless the addressed instruction specifies a program control change.

Virtual addressing requires the read protect bit (bit 32) of the kernel word to be equal to zero for security purposes.

The memory access bit (bit 32) of this instruction should be set to 0 after all address modification has taken place.

The memory location treated as an instruction will be interpreted according to its bit position 33. If bit 33 of this memory location is 0, a PMU instruction is executed. If bit 33 of this memory location is 1, an AP instruction is executed.

5.3.5 SHIFT INSTRUCTIONS

INSTRUCTION NAME: Shift Open

OP CODE: F0

FUNCTION: If (M) > 0; (A) Left Shifted by Four X $|M| \longrightarrow A$

If (M) < 0; (A) Right Shifted by Four X $|M| \longrightarrow A$

PMU MACHINE FORMAT:

F				0				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

F				0				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the A Register are shifted left if the M Register is positive and right if the M Register is negative by four bit positions for each shift as specified by the shift count in the M Register. Positions vacated by data are filled with zeroes. Data shifted out of the Accumulator is lost. The sign bit and exponent of the Accumulator are cleared to zero.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	0	00000004	44	1	054376AB
CONTENTS AFTER EXECUTION	00	0	00000004	00	0	76AB0000

INSTRUCTION NAME: Shift Cyclic

OP CODE: F1

FUNCTION: If $(M) \neq 0$; $(A)_{0-31}$ are rotated L/R → A
or 0-63

If $(M) = 0$; $(A) \rightarrow A$

PMU MACHINE FORMAT:

F				1				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

F				1				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the contents of the M Register are positive, the Accumulator is rotated left; or, if the contents of the M Register are negative, the contents of the Accumulator are shifted right four(4) bits for each shift count of (M). Bits shifted out of the Accumulator are placed end-around in positions vacated by the shifting. Double precision shift to the right does not provide end-around bit rotation and positions vacated by the shift are set to zero. If (M) are zero, no rotation takes place. The sign bit and exponent of the Accumulator are cleared to zero.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	1	00000004	12	1	12345678
CONTENTS AFTER EXECUTION	00	1	00000004	00	0	56781234

INSTRUCTION NAME: Shift Single Open

OP CODE: F8

FUNCTION: If $(M)_{30-31} \neq 0$; $(A)_{0-31}$ are shifted left X $(M)_{30-31} \longrightarrow A$

PMU MACHINE FORMAT:

F				8				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

F				8				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the A Register are shifted left the number of bits specified in Bits 30-31 of the M Register. Zeros are placed in vacated bit positions. Bits shifted past Bit 0 are lost. The sign bit and exponent of the Accumulator are cleared to zero.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	0	00000003	66	1	12345678
CONTENTS AFTER EXECUTION	00	0	00000003	00	0	91A2B3C0

INSTRUCTION NAME: Shift Single Closed

OP CODE: F9

FUNCTION: If $(M)_{30-31} \neq 0$; $(A)_{0-31}$ are shifted left X $(M)_{30-31} \longrightarrow$ A

PMU MACHINE FORMAT:

F				9				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

F				9				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the A Register are shifted left the number of bits specified in Bits 30-31 of the M Register. Bits shifted out of the Accumulator are placed end-around in positions vacated by the shifting. The sign bit and exponent of the Accumulator are cleared to zero.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	0	00000003	66	1	F2345678
CONTENTS AFTER EXECUTION	00	0	00000003	00	0	91A2B3C7

5.3.6 BOOLEAN AND LOGICAL INSTRUCTIONS

INSTRUCTION NAME: Boolean Zero

OP CODE: 70

FUNCTION: 0 → A ; 0 → TVD

PMU MACHINE FORMAT:

7				0				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

7				0				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the Accumulator are set to zero. The Test Valid (TVD) flip-flop is reset to zero.

EXAMPLE :

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	11	0	12345678	22	1	87654321
CONTENTS AFTER EXECUTION	11	0	12345678	00	0	00000000

INSTRUCTION NAME: Boolean AND

OP CODE: 71

FUNCTION: $(A)_{31} \bullet (M)_{31} \longrightarrow (A)_{31} \longrightarrow \text{TVD}$

PMU MACHINE FORMAT:

7				1				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

7				1				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of Bit 31 of the A Register and Bit 31 of the M Register are logically ANDed with the result being placed in Bit 31 of the Accumulator and the Test Valid (TVD) flip-flop. The other bits in the Accumulator are cleared to zero.

EXAMPLE :

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	11	0	00000001	22	0	12345678
CONTENTS AFTER EXECUTION	11	0	00000001	00	0	00000000

INSTRUCTION NAME: Boolean Less Than

OP CODE: 72

FUNCTION: If $(M)_{31} = 0$ and $(A)_{31} = 1$; $0 \longrightarrow A$, $1 \longrightarrow A_{31}$ TVD Set

If $(M)_{31} = 1$; $0 \longrightarrow A$, TVD Reset

PMU MACHINE FORMAT:

7				2				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

7				2				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of Bit 31 of the M Register is zero and Bit 31 of the A Register is one (1), the Accumulator is cleared to zero and Bit 31 of the Accumulator is set to one (1). The Test Valid (TVD) flip flop is set. Otherwise, the Accumulator is cleared to zero and TVD is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	44	1	12345678	33	0	12345671
CONTENTS AFTER EXECUTION	44	1	12345678	00	0	00000001

INSTRUCTION NAME: Boolean Odd Even

OP CODE: 73

FUNCTION: If $(A)_{31} = 1$; $0 \longrightarrow A$, $1 \longrightarrow A_{31}$, $1 \longrightarrow TVD$

If $(A)_{31} = 0$; $0 \longrightarrow A$; TVD Reset

PMU MACHINE FORMAT:

7				3				P	PF			I	INDEX			ADDRESS																M	I	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

7				3				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of $(A)_{31}$ is a one (1), the Accumulator is cleared to zero and Bit 31 of the Accumulator is set to one (1) and the Test Valid (TVD) flip flop is set. Otherwise, the Accumulator is cleared to zero and TVD is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	21	1	12345678	66	0	12345671
CONTENTS AFTER EXECUTION	21	1	12345678	00	0	00000001

INSTRUCTION NAME: Boolean Greater Than

OP CODE: 74

FUNCTION: If $(M)_{31} > (A)_{31}$; $0 \longrightarrow A$, $1 \longrightarrow A_{31}$; TVD Set

If $(M)_{31} = 0$; $0 \longrightarrow A$; TVD Reset

PMU MACHINE FORMAT:

7				4				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

7				4				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of Bit 31 of the M Register is a one (1), and the value of Bit 31 of the A Register is a zero, the Accumulator is cleared to zero, Bit 31 of the Accumulator is set to one, and the Test Valid (TVD) flip flop is set. Otherwise, the Accumulator is cleared to zero and TVD is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	11	1	12345671	11	1	12345678
CONTENTS AFTER EXECUTION	11	1	12345671	00	0	00000001

INSTRUCTION NAME: Boolean Load

OP CODE: 75

FUNCTION: If $(M)_{31}$ = 1; 0 ⟶ A, 1 ⟶ A_{31}, TVD Set

If $(M)_{31}$ = 0; 0 ⟶ A, TVD Reset

PMU MACHINE FORMAT:

7				5				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

7				5				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of Bit 31 of the M Register is one (1), the Accumulator is cleared, Bit 31 of the Accumulator is set to one (1), and the Test Valid (TVD) flip flop is set. Otherwise, the Accumulator is cleared to zero and TVD is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	22	1	12345671	55	0	12345678
CONTENTS AFTER EXECUTION	22	1	12345671	00	0	00000001

INSTRUCTION NAME: Boolean Not Equal

OP CODE: 76

FUNCTION: If $(M)_{31} \neq (A)_{31}$; $0 \rightarrow A$, $1 \rightarrow A_{31}$, TVD Set

If $(M)_{31} = (A)_{31}$; $0 \rightarrow A$, TVD Reset

PMU MACHINE FORMAT:

7	6	P	PF	I	INDEX	ADDRESS	M	1	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

AP MACHINE FORMAT:

7	6	P	PF	DT	OPERAND
0 1 2 3	4 5 6 7	8	9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 … 55

DEFINITION: If the values of Bit 31 of both the M Register and A Register are not equal, then the Accumulator is cleared to zero, Bit 31 of the Accumulator is set to one (1), and the Test Valid (TVD) flip flop is set. Otherwsie, the Accumulator is cleared to zero and TVD is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	13	0	12345678	11	1	12345671
CONTENTS AFTER EXECUTION	13	0	12345678	00	0	00000001

INSTRUCTION NAME: Boolean Inclusive OR

OP CODE: 77

FUNCTION: If $(M)_{31}$ or $(A)_{31} = 1$, $0 \longrightarrow A$, $1 \longrightarrow A_{31}$, TVD Set

If $(M)_{31}$ and $(A)_{31} \neq 1$, $0 \longrightarrow A$, TVD Reset

PMU MACHINE FORMAT:

7				7				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

7				7				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of Bit 31 of either the M Register or A Register is a one (1), the Accumulator is cleared to zero, Bit 31 of the Accumulator is set to one (1), and the Test Valid (TVD) flip flop is set. Otherwise, the Accumulator is cleared to zero and TVD is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	41	0	12345678	14	1	12345678
CONTENTS AFTER EXECUTION	41	0	12345678	00	0	00000000

INSTRUCTION NAME: Boolean NOR

OP CODE: 78

FUNCTION: If $(M)_{31}$ and $(A)_{31} = 0$; $0 \longrightarrow A$, $1 \longrightarrow A_{31}$, TVD Set

If $(M)_{31}$ or $(A)_{31} \neq 0$; $0 \longrightarrow A$, TVD Reset

PMU MACHINE FORMAT:

7	8	P	PF	I	INDEX	ADDRESS	M	1	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

AP MACHINE FORMAT:

7	8	P	PF	DT	OPERAND
0 1 2 3	4 5 6 7	8	9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 55

DEFINITION: If the value of Bit 31 in both the M Register and A Register is zero; then, the Accumulator is cleared to zero, Bit 31 of the Accumulator is set to one (1), and the Test Valid (TVD) flip flop is set. Otherwise, the Accumulator is cleared to zero and TVD is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	05	0	12345678	50	1	12345678
CONTENTS AFTER EXECUTION	05	0	12345678	00	0	00000001

INSTRUCTION NAME: Boolean Equals

OP CODE: 79

FUNCTION: If $(M)_{31} = (A)_{31}$; $0 \longrightarrow A$, $1 \longrightarrow A_{31}$, TVD Set

If $(M)_{31} \neq (A)_{31}$; $0 \longrightarrow A$, TVD Reset

PMU MACHINE FORMAT:

7				9				P	PF			I	INDEX			ADDRESS																M	I	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

7				9				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of Bit 31 of both the M Register and A Register are equal; then, the Accumulator is cleared to zero, Bit 31 of the Accumulator is set to one (1), and the Test Valid (TVD) flip flop is set. Otherwise, the Accumulator is cleared to zero and TVD is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	33	0	12345678	22	0	87654324
CONTENTS AFTER EXECUTION	33	0	12345678	00	0	00000001

INSTRUCTION NAME: Boolean Load Complement

OP CODE: 7A

FUNCTION: If $(M)_{31} = 0$; $0 \longrightarrow A$, $1 \longrightarrow A_{31}$, TVD Set

If $(M)_{31} = 1$; $0 \longrightarrow A$, TVD Reset

PMU MACHINE FORMAT:

7				A				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

7				A				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of Bit 31 of the M Register is zero; then, the Accumulator is cleared to zero, Bit 31 of the Accumulator is set to one (1), and the Test Valid (TVD) flip flop is set. Otherwise, the Accumulator is cleared to zero and TVD is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	0	00000008	16	0	12345678
CONTENTS AFTER EXECUTION	00	0	00000008	00	0	00000001

INSTRUCTION NAME: Boolean Less Than or Equal

OP CODE: 7B

FUNCTION: If $(M)_{31} = 0$ or $(A)_{31} = 1$; $0 \longrightarrow A$, $1 \longrightarrow A_{31}$, TVD Set

If $(M)_{31} \neq 0$ or $(A)_{31} \neq 1$; $0 \longrightarrow A$, TVD Reset

PMU MACHINE FORMAT:

7				B				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

7				B				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of Bit 31 of the M Register is equal to zero or Bit 31 of the A Register is equal to one (1); then, the Accumulator is cleared to zero, Bit 31 of the Accumulator is set to one (1), and the Test Valid (TVD) flip flop is set. Otherwise, the Accumulator is cleared to zero and TVD is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	12	0	3456789A	12	0	3456789A
CONTENTS AFTER EXECUTION	12	0	3456789A	00	0	00000001

INSTRUCTION NAME: Boolean NOT

OP CODE: 7C

FUNCTION: If $(A)_{31} = 0;\ 0 \longrightarrow A,\ 1 \longrightarrow A_{31}$, TVD Set

If $(A)_{31} = 1;\ 0 \longrightarrow A$, TVD Reset

PMU MACHINE FORMAT:

7				C				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

7				C				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of Bit 31 of the Accumulator is zero; then, the Accumulator is cleared to zero, Bit 31 of the Accumulator is set to one (1), and the Test Valid (TVD) flip flop is set. Otherwise, the Accumulator is cleared to zero and TVD is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	22	0	12345678	11	1	67812345
CONTENTS AFTER EXECUTION	22	0	12345678	00	0	00000001

INSTRUCTION NAME: Boolean Greater or Equal

OP CODE: 7D

FUNCTION: If $(M)_{31} = 1$ or $(A)_{31} = 0$; $0 \longrightarrow A$, $1 \longrightarrow A_{31}$, TVD Set
If $(M)_{31} = 0$ or $(A)_{31} = 1$; $0 \longrightarrow A$, TVD Reset

PMU MACHINE FORMAT:

7				D				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

7				D				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of Bit 31 of the M Register is a one (1), or the value of Bit 31 of the A Register is a zero; then, the Accumulator is cleared to zero, Bit 31 of the Accumulator is set to one (1), and the Test Valid (TVD) flip flop is set. Otherwise, the Accumulator is cleared to zero and TVD is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	09	0	87654321	08	0	12345678
CONTENTS AFTER EXECUTION	09	0	87654321	00	0	00000001

INSTRUCTION NAME: Boolean NAND

OP CODE: 7E

FUNCTION: If $(M)_{31} = 0$ or $(A)_{31} = 0$; $0 \rightarrow A$, $1 \rightarrow A_{31}$, TVD Set

If $(M)_{31} = 1$ and $(A)_{31} = 1$; $0 \rightarrow A$, TVD Reset

PMU MACHINE FORMAT:

7				E				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

7				E				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of Bit 31 of the M Register is zero or the value of Bit 31 of the A Register is zero; then, the Accumulator is celared to zero, Bit 31 of the Accumulator is set to one, and the Test Valid (TVD) flip-flop is set. Otherwise, the Accumulator is cleared to zero and TVD is reset.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	23	0	12345678	11	0	12345678
CONTENTS AFTER EXECUTION	23	0	12345678	00	0	00000001

INSTRUCTION NAME: Boolean One

OP CODE: 7F

FUNCTION: 0 ⟶ A; 1 ⟶ A_{31}, TVD Set

PMU MACHINE FORMAT:

7				F				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

7				F				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The Accumulator is cleared to zero, Bit 31 of the Accumulator is set to one (1), and the Test Valid (TVD) flip-flop is set.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	AA	1	23456789	12	0	12345678
CONTENTS AFTER EXECUTION	AA	1	23456789	00	0	00000001

INSTRUCTION NAME: Logical Zero

OP CODE: 80

FUNCTION $0 \longrightarrow A_{0-31}$

PMU MACHINE FORMAT:

8				0				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

8				0				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The Mantissa of the Accumulator is cleared to zero.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	01	0	12345678	19	1	01234567
CONTENTS AFTER EXECUTION	01	0	12345678	19	1	00000000

INSTRUCTION NAME: Logical AND

OP CODE: 81

FUNCTION: $(M)_{0-31} \bullet (A)_{0-31} \longrightarrow A_{0-31}$

PMU MACHINE FORMAT:

8				1				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

8				1				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the M Register are logically ANDed bit for bit with the contents of the A Register. The result is placed in the Accumulator.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	04	1	12345678	03	0	12345678
CONTENTS AFTER EXECUTION	04	1	12345678	03	0	12345678

INSTRUCTION NAME: Logical Less Than

OP CODE: 82

FUNCTION: $(A)_{0-31} > (M)_{0-31} \longrightarrow A_{0-31}$

PMU MACHINE FORMAT:

8				2				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

8				2				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of each bit position in the A Register is greater than the corresponding bit position of the M Register; then a one (1) is placed in that position in the Accumulator. Otherwise, the corresponding bit position of the Accumulator is set to zero.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	44	0	12345678	55	1	87654321
CONTENTS AFTER EXECUTION	44	0	12345678	55	1	85410101

INSTRUCTION NAME: No Operation

OP CODE: 83

FUNCTION:

PMU MACHINE FORMAT:

8	3	P	PF	I	INDEX	ADDRESS	M	1	T	P
0 1 2 3	4 5 6 7	8	9 10 11	12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	32	33	34	35

AP MACHINE FORMAT:

8	3	P	PF	DT	OPERAND
0 1 2 3	4 5 6 7	8	9 10 11	12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 … 55

DEFINITION: Idle processor for one instruction cycle time.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	AA	0	12345678	BB	0	12345678
CONTENTS AFTER EXECUTION	AA	0	12345678	BB	0	12345678

INSTRUCTION NAME: Logical Greater Than

OP CODE: 84

FUNCTION: $(M)_{0-31} > (A)_{0-31} \longrightarrow A_{0-31}$

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">8</td><td colspan="4">4</td><td>P</td><td colspan="3">PF</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>I</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

AP MACHINE FORMAT:

<table>
<tr><td colspan="4">8</td><td colspan="4">4</td><td>P</td><td colspan="3">PF</td><td colspan="4">DT</td><td colspan="32">OPERAND</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td><td>36</td><td>37</td><td>38</td><td>39</td><td>40</td><td>41</td><td>42</td><td>43</td><td>44</td><td>45</td><td>46</td><td>55</td></tr>
</table>

DEFINITION: If the value of each bit position in the M Register is greater than the corresponding bit position in the A Register; then, a one (1) is placed in that position in the Accumulator. Otherwise, the corresponding bit position of the Accumulator is set to zero.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	44	0	12345678	55	1	87654321
CONTENTS AFTER EXECUTION	44	0	12345678	55	1	10101458

INSTRUCTION NAME: Logical Load

OP CODE: 85

FUNCTION: $(M)_{0-31} \rightarrow A_{0-31}$

PMU MACHINE FORMAT:

8				5				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

8				5				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the M Register are copied into the A Register.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	FE	0	12345678	1B	0	87654321
CONTENTS AFTER EXECUTION	FE	0	12345678	1B	0	12345678

INSTRUCTION NAME: Logical Not Equal

OP CODE: 86

FUNCTION: If $(M)_{0-31} \neq (A)_{0-31}$; $1 \rightarrow A_{0-31}$

If $(M)_{0-31} = (A)_{0-31}$; $0 \rightarrow A_{0-31}$

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">8</td><td colspan="4">6</td><td>P</td><td colspan="3">PF</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>1</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

AP MACHINE FORMAT:

<table>
<tr><td colspan="4">8</td><td colspan="4">6</td><td>P</td><td colspan="3">PF</td><td colspan="4">DT</td><td colspan="31">OPERAND</td><td></td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td><td>36</td><td>37</td><td>38</td><td>39</td><td>40</td><td>41</td><td>42</td><td>43</td><td>44</td><td>45</td><td>46</td><td>55</td></tr>
</table>

DEFINITION: If the contents of the M Register are not equal bit for bit with the A Register, then the corresponding bit position in the Accumulator is set to one (1). Otherwise, the corresponding bit position is set to zero.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	11	0	10101078	22	1	98765432
CONTENTS AFTER EXECUTION	11	0	10101078	22	1	8866444A

INSTRUCTION NAME: Logical Inclusive OR

OP CODE: 87

FUNCTION: If $(M)_{0-31} = 1$ or $(A)_{0-31} = 1$; $1 \rightarrow A_{0-31}$

If $(M)_{0-31} = 0$ and $(A)_{0-31} = 0$; $0 \rightarrow A_{0-31}$

PMU MACHINE FORMAT:

8				7				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

8				7				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the M Register are logically mapped onto the contents of the A Register bit for bit without binary overflow and the result is placed in the Accumulator.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	CC	0	2468ABCD	33	1	3579BADE
CONTENTS AFTER EXECUTION	CC	0	2468ABCD	33	1	3579BBDF

INSTRUCTION NAME: Logical Equals

OP CODE: 89

FUNCTION: If $(M)_{0-31} = (A)_{0-31}$; $1 \rightarrow A_{0-31}$

If $(M)_{0-31} \neq (A)_{0-31}$; $0 \rightarrow A_{0-31}$

PMU MACHINE FORMAT:

8				9				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

8				9				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the Accumulator are logically compared with the contents of the M Register bit for bit. If the bit of the M Register is equal to the bit the Accumulator the corresponding bit of the accumulator is set to one. Otherwise the Accumulator bit is cleared to zero.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	0	12345678	00	0	12345678
CONTENTS AFTER EXECUTION	00	0	12345678	00	0	FFFFFFFF

INSTRUCTION NAME: Logical NOR

OP CODE: 88

FUNCTION: If $(M)_{0-31}$ and $(A)_{0-31} = 0;\ 1 \rightarrow A_{0-31}$

If $(M)_{0-31}$ or $(A)_{0-31} \neq 0;\ 0 \rightarrow A_{0-31}$

PMU MACHINE FORMAT:

8				8				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

8				8				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of the M Register corresponding bit for bit with the A Register is a zero, that corresponding bit position is set to one (1) in the Accumulator. Otherwise, the corresponding bit position is set to zero.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	66	1	98765432	22	0	12345678
CONTENTS AFTER EXECUTION	66	1	98765432	22	0	6589A983

INSTRUCTION NAME: Load Complement

OP CODE: 8A

FUNCTION: $(\overline{M}) \longrightarrow A$

PMU MACHINE FORMAT:

8				A				P	PF			I	INDEX			ADDRESS																$\overline{M}$	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

8				A				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The ones complement of the M Register is loaded into the Accumulator.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	55	0	12345678	33	1	98765432
CONTENTS AFTER EXECUTION	55	0	12345678	33	1	EDC9A987

INSTRUCTION NAME: Logical Less Than or Equal

OP CODE: 8B

FUNCTION: If $(M)_{0-31} = 0$ or $(A)_{0-31} = 1$; $1 \rightarrow A_{0-31}$

If $(M)_{0-31} = 1$ and $(A)_{0-31} = 0$; $0 \rightarrow A$

PMU MACHINE FORMAT:

8				B				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

8				B				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of the M Register, bit for bit, is a zero or the corresponding bit of the A Register is a one (1), then the corresponding bit of the Accumulator is set to one (1). Otherwise, the corresponding bit is set to zero.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	01	0	FEDCBA98	F0	1	23456789
CONTENTS AFTER EXECUTION	01	0	FEDCBA98	F0	1	2D4767EF

INSTRUCTION NAME: Logical NOT

OP CODE: 8C

FUNCTION: $(\overline{A}) \longrightarrow A$

PMU MACHINE FORMAT:

8				C				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

8				C				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The ones complement of the contents of the Accumulator replaces the contents of the Accumulator.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	FF	1	76543210	FE	1	01A98765
CONTENTS AFTER EXECUTION	FF	1	76543210	FE	1	FE56789A

INSTRUCTION NAME: Logical Greater or Equals

OP CODE: 8D

FUNCTION: If $(M)_{0-31} = 1$ or $(A)_{0-31} = 0$; $1 \longrightarrow A_{0-31}$

If $(M)_{0-31} = 0$ and $(A)_{0-31} = 1$; $0 \longrightarrow A_{0-31}$

PMU MACHINE FORMAT:

8				D				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

8				D				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of the M Register, bit for bit, is a one or the corresponding bit of the A Register is a zero; then the corresponding bit of the Accumulator is set to one (1). Otherwise, the value of the corresponding bit is set to zero in the Accumulator.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	FF	0	67543210	10	1	01010101
CONTENTS AFTER EXECUTION	FF	0	67543210	10	1	FEFEFEFE

INSTRUCTION NAME: Logical NAND

OP CODE: 8E

FUNCTION: If $(M)_{0-31} = 0$ or $(A)_{0-31} = 0$; $1 \longrightarrow A_{0-31}$

If $(M)_{0-31} = 1$ and $(A)_{0-31} = 1$; $0 \longrightarrow A_{0-31}$

PMU MACHINE FORMAT:

8				E				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

8				E				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: If the value of the M Register, bit for bit, is a zero or the corresponding bit of the A Register is a zero; then the corresponding bit of the Accumulator is set to one (1). Otherwise, the corresponding bit is set to zero.

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	1	12233445	FF	0	0DDCCBBA
CONTENTS AFTER EXECUTION	00	1	12233445	FF	0	FFFFFFFF

INSTRUCTION NAME: Logical Set

OP CODE: 8F

FUNCTION: $1 \longrightarrow A_{0-31}$

PMU MACHINE FORMAT:

8				F				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

AP MACHINE FORMAT:

8				F				P	PF			DT				OPERAND																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	55

DEFINITION: The contents of the Accumulator, bit for bit, is set to one (1).

EXAMPLE:

	M REGISTER			A REGISTER		
CONTENTS BEFORE EXECUTION	00	1	12233445	FE	1	23456789
CONTENTS AFTER EXECUTION	00	1	12233445	FE	1	FFFFFFFF

5.3.7 Array Storage

The number of words used by an array, in its unpacked form, must be equal to or less than 255. A Dimension Word plus 255 words make up a page and a page is the maximum size of any array. That is, a double precision array can only have 127 elements in it (127x2 words + 1 Dimension Word = 255): a packed binary array is limited to 9 words in storage (7x32 + 1x31) bits + 1 Dimension Word = 256).

Arrays are stored backward in the memories with the dimension word first followed by the last word element of array going to the first element. For example, if an array is created in memory of consecutive integers beginning with Ø and going to A-1 it would appear as follows:

Location		Contents
0	Dimension Word	
1	Last Word	A-1
.		A-2
.		A-3
.		.
N	First Word	Ø

As previously stated, the shape of the array in storage is defined by the Dimension Word. The dimension high field contains the number of rows in the matrix. The dimension low field contains the number of elements in a row or in the case of a vector the number of elements in the vector. Both of these fields may contain a value of 0-255, but there must be less than 256 elements in the array.

Packed Data is stored in memory in the following format (8 bit data, with 6 elements is used in this example).

location				
0	1	0	$(00)_{16}$	$(00)_{16}$
1	5	4	3	2
2	DIMENSION WORD			

When packed data is referenced, the array controller issues an UNPACK instruction to the AP. The AP unpacks the data to full word logical operands for subsequent processing. The STORE PACKED instruction is used to convert full word data to a packed form.

Arrays with more than 255 elements must be handled with a software subroutine.

5.3.8 Array Operations

Any AP instruction which is addressable may access an array as an operand. If it does so, and the Accumulator is scalar, each element of the array will operate against the Accumulator separately for the case of arithmetic instructions, generating an array of the same shape as the original array fetched. This array will become the new AP Accumulator.

If the Accumulator is an array, and the operand is also an array, they must have conformal shapes (as a function of the operator) in order to allow processing. Any deviation is signalled as a length error. If they do have the correct shape, each element of the operand is combined with the corresponding Accumulator element forming a new Accumulator.

Once the Accumulator is defined as an array, even if the memory operand is scalar, an array operation is performed. In this case, operation is analogous to the case where the operand was an array and the Accumulator a scalar.

5.3.9 Parenthetical Control

The following parenthetical control actions occur during array processing:

PUSH (Open Parenthesis) - If the Accumulator is an array, the array controller accesses scratchpad register 26, uses it as a virtual address for storage of the Accumulator array, then increments the higher order byte (bits 0-7) of register 26.

If the operand is an array, it is loaded into the Accumulator array.

The AP is sent a scalar push instruction in order to save the operation code of the instruction, or to load a scalar operand, if this is specified.

For the purposes of this operation, a complex scalar is considered an array.

POP (Close Parenthesis) - When a POP occurs in which the Accumulator to be popped is an array, the array controller accesses scratchpad register 26, decrements its higher order byte (bits 0-7), and fetches the array stored at that location. This array is placed into the Accumulator array, and the popped operation may be performed.

The AP has two flip-flops which indicate whether the present accumulator is complex, real, or scalar array. These are pushed and popped in the AP in order to maintain memory of whether the value stored in the AP stack is a value or an image of a value stored in memory.

5.3.10 Arrays

Array handling procedures are implemented in hardware in an AADC. Both the AP and PMU are controlled by the Array Controller during array operations. Array operations are set into operation when one of the following events occur:

1) The Accumulator is already an array from a previous operation.

2) One of the set of instructions that are array instructions is fetched. These instructions will act in array mode regardless of the shape of either of the operands. Even with two scalars, array mode will be called into effect. (i.e. catenation)

3) When an arithmetic instruction addresses an operand, and the operand has a data tag of complex or is a Dimension word.

5.3.11 Array Storage Area

Reference has been made to an Array Accumulator, and a working Array Accumulator. These are the last three pages of Task Memory (pages 13, 14, 15 of a 4K word memory) which are reserved for array operations. When an array enters the Accumulator for the first time, it is placed in page 13 . The next array operation generates a new Accumulator in page 14. The next is assigned to page 13 and so on.

When a packed array enters the DPE, it is first unpacked and placed in page 15, then operated upon. When a "close parenthesis" occurs, as described before, the array to be popped is placed in page 15. Page 15 is termed the working Accumulator area. Pages 13 and 14 are termed the Array Accumulators, though only one will be the Array Accumulator at a given time.

5.3.12 Dimension Word

The dimension word format is:

<table>
<tr><td>SDL</td><td>SDH</td><td colspan="5">UNUSED</td><td>S</td><td colspan="4">DATA TYPE</td><td colspan="4">RANK</td><td colspan="8">DIM (HIGH)</td><td colspan="8">DIM (LOW)</td><td colspan="3">001</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

For all operations, except TAKE and DROP, the sign bits (SDL/SDH) of the dimension word are not interpreted. For TAKE and DROP, the sign bits have meaning. Refer to the definition of of TAKE and DROP for these meanings.

Bit 7 - Scalar bit flag for indirect dimension words only.

Bits 8-11 describe the data type of the array. The possible types are:

0 - Single precision floating, integer or logical
1 - Double precision floating
2 - Complex

8 - Packed binary (1 bit)
9 - Packed quaternary (2 bits)
A - Packed hexadecimal (4 bits)
B - Packed byte (8 bits)
C - Packed half-word (16 bits)

Packed arrays are unpacked upon receipt by a DPE for use in an instruction.

The rank field (bits 12-15) indicates whether this is a matrix or vector. A rank field of 0 indicates a vector, 1, a matrix. If the rank field is greater than 1, odd number ranks will be treated as an array, even number ranks as a vector.

If the operand is a vector, bits 24-31 enumerate the number of elements in the vector. There are a maximum of 255 single precision operands, or 127 double precision or complex operands. If bits 24-31 of a vector are all ZERO, the vector is said to be NULL. Null vectors have significance in certain operations.

If the operand is a matrix, bits 16-23 enumerate the number of row elements, bits 24-31 enumerate the number of column elements. There are a maximum of 255 single precision matrix elements, or 127 double precision or complex elements.

If bits 16-23 or 24-31 of a matrix are all zero, the matrix is said to be NULL. If the number of matrix elements specified exceeds the stated limits, the desired answer will not result. What does occur is a function of the operation being executed.

Bit 35 is parity.

Note that scalars can be accessed via an indirect dimension word with bit 7 being set to one.

5.3.13 Indirect Dimension Words

Normally, a dimension word is adjacent to the block of data it describes. Indirect Dimension Words provide a means to describe a block of data with a non-adjacent dimension word.

When indirection is specified in data virtual addressing mode (the indirect dimension word capability is not provided when the data addressing mode is absolute), the data tag of the referenced indirect word is examined. If the data tag is 001, the address of the indirect dimension word is stored in the scratchpad register 28, the dimension word is stored in the Task Memory location whose address is all 1's and the next sequential indirect word is read and interpreted as a normal indirect word.

Before proceeding with subsequent handling of the indirect dimension words, the following remarks are noteworthy:

- If multiple indirect dimension words are accessed, the last indirect dimension word controls the final operand fetch.
- An indirect dimension word may be marked as "SCALAR." This means that during the course of computation on an array referenced by an indirect word, the array has been converted to a scalar. Bit 7 (being 1) of an indirect dimension word indicates this condition.

The final binding of the referenced operand is a function of the normal addressability condition of the present instruction.

If the instruction is normally addressable, one of two actions occurs. If a non scalar indirect dimension word was referenced, the operand is referenced by a Read Array to Output Instruction if word oriented data. If paged data, internal indicators are used to maintain the proper association of dimension word with data.

If a scalar indirect dimension word was referenced, the operand is referenced through the normal operand accessing mechanism (i.e., as if no indirect dimension word was accessed). This means that a non-scalar operand may be accessed if the word referenced by the scalar dimension word is a dimension word.

If the instruction is normally non addressable (e.g., STORE), the final operand produced is the address field of the last indirect word accessed. Thus, during the execution of the Store instruction, the dimension word of the array is stored in the location of the last indirect dimension word received, and the array is stored (less dimension word) in the memory area defined by the address field of the last indirect word. Thus, the same indirect loop can be used for obtaining and storing operands.

5.3.14 Complex Numbers

A Complex number is represented by two adjacent single precision floating point numbers. The first element is treated as the real part and the second element as the imaginary part. A complex number is interpreted when the data tag is 110. Most real operators have meaning when applied to complex operands. Thus, using the example used in array operations, evaluating the expression A BXC+D where A,B,C,&D are complex, one uses the same DPE code.

```
  LOAD D
     +    C
     X    B
STORE A
```

All operations involving complex operand must specify the result of the operation to be double precision (instruction bite).

There are, however, some exceptions in the handling of complex operands when compared to real operands. They are:

A true complex divide is not performed. A ÷ B where A and B are complex results in the real part divided by the real and the imaginary divided by the imaginary.

A complex compare signals a domain error.

Mixed mode arithmetics are signalled as domain errors. Thus, the statement 5+B where B is complex results in a domain error. The constant 5 must be specified as 5+i0 or 5+i5 depending on the meaning intended.

Mixed mode non-arithmetics of the form A op B where A is a control specification is allowed. For example 1 2 ϕ B, where B is a complex matrix.

In cases where a multiplication is specified, a true complex multiply is performed except for the following instructions where operation 2 (operation specified in bits 16-23) is multiply.

-61 - Polynomial

-62 - Outer Product Reduction

-6C - Inner Product

-6D - Outer Product
-6E - Reduction Along Row

5.3.15 Array Instructions

The following instructions cause the array controller to take control of the DPE.

INSTRUCTION NAME: Load Op Code

OP CODE: 37

FUNCTION:

PMU MACHINE FORMAT:

3				7				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: This instruction is intended to define the op codes to be used for the AP array operations.

(61) Polynomial
(62) Outer Product Reduction
(6C) Inner Product
(6D) Outer Product
(6E) Reduction Along Row

If bits 9, 10 and 11 of this instruction indicate immediate operation (000), the effective address field is used to load the AP Op Code registers.

If bits 9-11 indicate a push operation (111), the effective address field is pushed into the AP accumulator. This action implies that the above listed AP instructions also specify a deferral push.

If bit 9-10 indicate a pop, it will be treated as if an immediate operation was specified, no pop will actually occur.

This instruction is normally non-addressable. Bits 16-23 specify operation 2 (OP 2) and bits 24-31 specify operation 1 (OP 1). See the definitions of the above listed array instructions for the particular manipulations performed by these operations.

If one of the above array instructions is popped, OP 1 and OP 2 are contained in the Accumulator and the operand which is to be arithmetically manipulated is contained in the first level of deferral. The control sequence for the above op codes will adjust the deferral and obtain the Op Codes from the accumulator if necessary.

NOTE: In the instruction which pops the deferred array operand, the parenthetical field must specify at least two pops. One for the deferred array instruction. One for the deferred Load opcode instruction.

With this op code, the following sequences are necessary to perform the three basic modes of AP instruction execution (immediate, pop, push). The instruction, Outer Product, is used for illustrative purposes.

IMMEDIATE

Instruction Location	Instruction	
X	LOAD OP CODE	OP2, OP1
X+1	OUTER PRODUCT	⟨Address⟩

POP

Instruction Location	Instruction	
X	LOAD OP CODE	OP2, OP1
X+1	OUTER PRODUCT)...)	⟨Address⟩

PUSH

Instruction Location	Instruction	
X	LOAD OP CODE(	OP2, OP1
X+1	OUTER PRODUCT(	⟨Address⟩

where a "(" denotes a push and a ")...)" denotes one or more pops.

(See above note concerning instruction which pops deferred instructions in this example.)

INSTRUCTION NAME: Polynomial

OP CODE: 61

FUNCTION:

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">6</td><td colspan="4">1</td><td>P</td><td colspan="3">PF</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>1</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: This instruction interprets bits 16-31 of the AP Accumulator as 2 operation codes (bits 16-23 is operator 2 and bits 24-31 is operator 1). The addressed operand is combined according to a power series expansion with the Accumulator operand. The two op codes are previously loaded by the Load Op Code instruction (Op Code 37).

The power series expansion used to evaluate polynomial is:

$$((A[N]op_1\ M[0]op_2\ A[N-1])op_1\ M[N-1]op_2\ A[N-2])\ldots)op_1\ M[N-1]op_2\ A[0]$$

If op_1 is multiply and op_2 is addition, the typical power series expansion is generated. The characteristics of the expansion for the allowed data types are:

M and A are scalars: The accumulator is loaded with the accumulator. (This is subject to precision control).

M is a scalar and A is a vector: M is effectively expanded to equal the length of A.

M is a vector and A is a scalar: A is effectively expanded to equal the length of M.

M and A are vectors: Alternate values of M and A are used for coefficients and bases. The first value of M (M[0]) is ignored.

If A or M is a null vector, the accumulator is loaded with ZERO.

Error Condition:

A Domain Error occurs if A or M are matrices or one real and one complex.

A Length Error occurs if A and M are vectors of unequal length.

Notes:

- A vector M of length 1 is NOT treated as a scalar.
- In the case of M vector, the Polynomial instruction can be made equivalent to the APL statement (where op_1 is X, and op_2 is +).

(¯1⌽ ⌽M)⊥A

- A vector A of length 1 is treated as a scalar.
- If no error condition exists the rank of the result is 0 (scalar).

INSTRUCTION NAME: Dimension

OP CODE: 51

FUNCTION: $\rho A \longrightarrow A$

PMU MACHINE FORMAT:

5				1				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The Accumulator is loaded with a logical real scalar whose bit configuration is that of the dimension word of the array that was in the Accumulator. Bit 7 in the Accumulator is set to 1 to indicate Scalar A. This differentiates the results of this operation from an Accumulator of all zeros which is generated for null vectors.

INSTRUCTION NAME: Index Generator

OP CODE: 60

FUNCTION: 0, 1, 2, ..., $(A)_{24-31} - 1$

PMU MACHINE FORMAT:

6				0				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The Mantissa of the Accumulator (bits 24-31) is used as an integer and creates a vector of consecutive integers beginning with Ø and continuing to one less than the initial Accumulator value. An appropriate dimension word is assigned to the array of numbers and the array is stored in the Task Memory (TM) Array Accumulator.

A non scalar or negative scaler Accumulator results in a domain error interrupt being sent to the PMU.

The accumulator is assumed to be integerized.

INSTRUCTION NAME: Ravel

OP CODE: 50

FUNCTION: ,A → A

PMU MACHINE FORMAT:

5				0				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The Accumulator array is made into a vector. If a scalar, a vector of length 1 is formed. If a vector already, no operation is performed. If the Accumulator is a matrix, its dimensions are changed to those of a vector with an identical number of elements.

INSTRUCTION NAME: Outer Product Reduction

OP CODE: 62

FUNCTION:: $OP_2/M\circ.OP_1, A \longrightarrow A$

PMU MACHINE FORMAT:

6				2				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The Load Op Code instruction (Op Code 37) is used first to load the two operation codes to be used with this instruction.

The operand in the AP accumulator is raveled (i.e., treated as a vector.) Each memory operand is combined (operator 1) with every AP operand (Outer Product). The resulting vector whose length is equal to the AP accumulator operand is reduced to a single value by performing an element-by-element combination using operator 2. This sequence is repeated until all memory operands are used to perform an outer product. The accumulated results of the repeated reductions become the contents of the accumulator.

Both operands must be complex or real. Otherwise, a domain error occurs. If the operator 2 is Boolean Inclusive OR and operator 1 is equal, the APL primitive 'membership' is performed. Various high-low searches are performed by specifying the proper compares as operators 1 and 2, and specifying the bounds of the compare as the AP operand.

The shape of the result is the shape of M. See the description of Outer Product and the Reduction instructions for examples involving these two operators.

It should be noted that a potential temporary a accumulator wraparound exists. If M represents the total number of M operands and A the total number of A operands, on the last iteration of the algorithm the array accumulator uses the A+M-1 location, even though the final number of locations used is M. If this condition exists, operands occupying locations in excess of 255 will be written into the origin of the accumulator page destroying the initial result calculations.

INSTRUCTION NAME: Expand Along Column

OP CODE: 63

FUNCTION: (M)⍀(A) ⟶ A

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">6</td><td colspan="4">3</td><td>P</td><td colspan="3">PF</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>1</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: The contents of the M Register is a Boolean vector. The number of non zero elements in the M Register must be equal to the number of rows in the Accumulator array if the Accumulator is a matrix, or the number of elements if the Accumulator is a vector.

For each zero element in the "M" vector a new row of Matrix Accumulator or element of Vector Accumulator which is 0 will be placed into the new Accumulator. For each non zero element of the operand, the row or element of the Accumulator will be duplicated. If the Accumulator is a scalar, the value will be used again for each non zero element of the M Register.

If the M element is complex, double precision, or a matrix, a domain error occurs.

EXAMPLE:

M	ACC	Result	
11101	1,2,3,4	1,2,3,0,4	Vector
101	1,2,3	1,2,3	Matrix
	4,5,6	0,0,0	
		4,5,6	
0	Null Matrix Shape 03	0 0 0	Matrix

The following table summarizes the Expand operator.

M	A	Result
Scalar	Scalar	Vector
Vector	Scalar	Vector
Matrix	Scalar	Domain Error
Scalar	Vector	Length Error if length of A not equal to 1: Vector result.
Vector	Vector	Length Error if the number of 1 in M is not equal to the length of A. If A is empty M must be 0(s) or empty. Vector result.
Matrix	Vector	Domain Error.
Scalar	Matrix	Length Error if the number of rows in A is not equal to the value of the scalar. Matrix result.
Vector	Matrix	Length Error if the number of 1 in M is not equal to the number of rows in A. Matrix result.
Matrix	Matrix	Domain Error.

INSTRUCTION NAME: Catenate Rows

OP CODE: 64

FUNCTION: A,M → A

PMU MACHINE FORMAT:

6				4				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The Array A is appended to the Array M. If M and A are both vectors, the result is a vector of length M+ A. If M and A are both matrices, they must conform in that number of columns (i.e., row length) are the same. The number of columns in the result matrix is identical to the number of columns in the original Array A. The number of rows is the sum of the rows of the initial A and M arrays. A length error is signalled for non conforming array dimensions.

If either A or M is complex, then both A and M must be complex; otherwise, a domain error occurs.

The following table shows the dimensions resulting from the operation. Numbers in parentheses represent dimensions.

A Array	M Array	Result
Scalar	Scalar	Vector (2)
Scalar	Vector ( )	Vector (p+1)

A Array	M Array	Result
Scalar	Matrix (p,r)	Matrix (p+l,r)
Vector (n)	Scalar	Vector (n+l)
Vector (n)	Vector (p)	Vector (n+p)
Vector (n)	Matrix (p,n)	Matrix (p+l,n)
Matrix (m,n)	Scalar	Matrix (m+l,n)
Matrix (m,n)	Vector (n)	Matrix (m+l,n)
Matrix (m,n)	Matrix (p,n)	Matrix (m+p,n)

INSTRUCTION NAME: Transpose

OP CODE: 65

FUNCTION: ⍉A → A

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">6</td><td colspan="4">5</td><td>P</td><td colspan="3">PF</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>1</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION:

The contents of a matrix Accumulator A is reshaped such that each row of the initial Accumulator becomes a column, and each column becomes a row. If A is a vector or scalar, no operation is performed.

EXAMPLE:

ACC	Result
1,2,3	1,4
4,5,6	2,5
	3,6

INSTRUCTION NAME: Reversal Along Rows

OP CODE: 66

FUNCTION: ⌽A → A

PMU MACHINE FORMAT:

6				6				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION:

The elements of each row vector of A are rearranged in the reverse index sequence. A may be a vector. or matrix. No operation is performed if A is scalar.

EXAMPLE:

ACC	Result	
1,2,3,4	4,3,2,1	Vector
1,2,3	3,2,1	Matrix
4,5,6	6,5,4	

INSTRUCTION NAME: Laminate Rows

OP CODE: 67

FUNCTION: (A), [-5] (M) → A

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">6</td><td colspan="4">7</td><td>P</td><td colspan="3">PF</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>1</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: Vector A, whose length is equal to Vector M, is appended to Vector M to form a matrix with the row length being equal to the length of the original vectors, and whose column length is equal to 2.

If A or M are scalar, they are extended to the appropriate length. If both are scalar, a catenate operation is performed.

Vectors of unequal lengths will cause a length error. Matrix arguments will cause a domain error.

If A or M is complex, then both A & M must be complex, otherwise a domain error occurs.

EXAMPLE:

M	ACC	Result	Dimension Result	
1,2,3	4,5,6	123 456	2	3
5	4,5,6	555 456	2	3

INSTRUCTION NAME: Rotate Row

OP CODE: 68

FUNCTION: A←MØA

PMU MACHINE FORMAT:

6				8				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: Rotate Accumulator Array M places.
Each row vector of Matrix A is rotated left the number of component positions specified by the corresponding scalar component of Vector M, and the result is placed in A. If M is a scalar, it is extended to equal the column length of Matrix A. A may be a vector and considered as a single row matrix. M must conform in length with the length of column of A when A is a matrix.

Scalar A produces no operation, regardless of the shape of M. Vector A requires scalar M.

The elements of M are treated as single precision integers whose absolute values are less than or equal to the number of columns if matrix or elements if vector in A. If M is complex or double precision a domain error occurs.

EXAMPLE:

M	ACC	Result
4	1,2,3,4,5	5,1,2,3,4
-2	1,2,3,4,5	4,5,1,2,3

M	ACC	Result
-1,2,1	1,2,3	3,1,2
	4,5,6	6,4,5
	7,8,9	8,9,7
2	1,2,3	3,1,2
	4,5,6	6,4,5
	7,8,9	9,8,7

AØP

The elements of M are treated as single precision integers. Whole absolute values are less than or equal to the number of columns of A (if matrix) or element if A is a vector.

A domain error occurs if M is complex.

The least significant 8 bits of M are used to determine the rotation specification. Significance in bits 1 thru 23 of the elements of M are ignored.

INSTRUCTION NAME: Reshape

OP CODE: 69

FUNCTION: MρA⟶A

PMU MACHINE FORMAT:

6				9				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: M must be a scalar integer, logical, or single precision operand, otherwise a domain error occurs. M has the format of an indirect dimension word. That is if bit 7 is a "1", the result of the Reshape operator is a scalar. If bit 7 is a "0", the dimension word of the result is the dimension word in M.

Elements of the Accumulator are raveled and used to form the new array (scalar) up to the number of elements required. If the accumulator has more elements than the new array, excess elements are discarded. If it has less, the accumulator will repeat as often as necessary.

A length error occurs if the accumulator is null and the dimension word specified by M indicates a result which is not null.

EXAMPLE:

M	ACC	Result
5	1,2,3	1,2,3,1,2
2	1,2,3	1,2
2 3	1,2,3,4	1,2,3 4,1,2
4 0	1,2,3,4	empty matrix with shape 4 0

INSTRUCTION NAME: Take

OP CODE: 6A

FUNCTION: $(M) \uparrow (A) \rightarrow A$

PMU MACHINE FORMAT:

6				A				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: M must be a scalar integer, logical, or single precision operand; otherwise a Domain Error occurs. M has the format of an indirect dimension word. That is, if bit 7 is a "1", the result of the Take operator is a scalar if the accumulator is presently a scalar. If bit 7 is a "0", the dimension word of the result is the dimension word in M.

Bits 0 and 1 of M act as the sign bits (0 = plus, 1 = minus) for the array dimensions in bits 24-31 and bits 16-23 respectively. The magnitude of M acts as a new dimension for A. If a value of M is greater than the corresponding dimension of A, the rows (or columns) of A are extended to the length of the M dimension by catenating zeros, to the left if M is negative, or to the right if M is positive.

If a value of M is less than or equal to the corresponding dimension of A, the first or last M elements of each row (or column) are taken depending upon the sign of M being positive or negative. If a new dimension is zero, the result is an empty array.

The following tables enumerates the actions taken by combining the shapes of A and the result dimension specified by M.

A (present accumulator)	M (result accumulator	Result
Scalar	Scalar	Scalar
Vector	Scalar	Length Error
Matrix	Scalar	Length Error
Scalar	Vector	Vector
Vector	Vector	Vector
Matrix	Vector	Length Error
Scalar	Matrix	Matrix
Vector	Matrix	Length Error
Matrix	Matrix	Matrix

EXAMPLE:

M	ACC	Result
3	3,4,5,6	3,4,5
-3	3,4,5,6	4,5,6
6	3,4,5,6	3,4,5,6,0,0
-6	3,4,5,6	0,0,3,4,5,6
1,1	3,4	3
	5,6	
3,3	3,4	3,4,0
	5,6	5,6,0
		0,0,0
(-3), 3	3,4	0,0,0
	5,6	3,4,0
		5,6,0
3,3	NULL	0,0,0
		0,0,0
		0,0,0
3,0	3,4	NULL WITH SHAPE 3,0
	5,6	
3,3	1 SCALAR	1,0,0
		0,0,0
		0,0,0

INSTRUCTION NAME: Drop

OP CODE: 6B

FUNCTION: $(M)\downarrow(A)\longrightarrow A$

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">6</td><td colspan="4">B</td><td>P</td><td colspan="3">PF</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>1</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: The M operand has the same form as in TAKE. Each dimension of A is reduced by the value of the corresponding magnitude of M. If any resulting dimension is zero or negative, the result is an empty array. The first M elements of each row (or column) of A are dropped if M is negative. The last M elements of each row (or column) of A are dropped if M is positive.

EXAMPLE:

M	ACC	Result
2	3,4,5,6	5,6
-2	3,4,5,6	3,4
1,1	3,4	6
	5,6	
3,3	3,4	NULL MATRIX
	5,6	WITH SHAPE 0,0
1,-1	3,4	5
	5,6	

INSTRUCTION NAME: Inner Product
OP CODE: 6C
FUNCTION: (M) $OP_1 \cdot OP_2$ (A)—→ A
PMU MACHINE FORMAT:

6				C				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: The Load Op Code instruction (Op Code 37) is used first to load the two operation codes to be used with this instruction.

Each element of the result A i;j is formed by a combination of the elements of the ith row of M with the jth column of the initial A.

A [1;1] A [1;2] ... A [1;N] M [1;1] M [1;2] ... M [1;P] A[1;1] A[1;2] ... A[1;N]
A [2;1] A [2;2] ... A [2;N] M [2;1] M [2;2] ... M [2;P] A[2;1] A[2;2] ... A[2;N]
... ...
... ... ← ... ...$OP_1 \cdot OP_2$. ... ...
... ...
[illegible];1] A [M;2] ... A [M;N] M [M;1] M [m;2] ... M [M;P] ... ...
A[P;1] A[P;2] ... A[P;N]

Each element of row i of M is paired with each element of column j of A. Dyadic operation op2 is placed between the elements of each pair. Dyadic operation opl is placed between the pairs.

If A or M are vectors, M will be treated as a row vector and A as a column vector.

If A and M are matrices, a matrix will result. If one is a vector, a vector will result.

If both are vectors, a scalar will result.

A [i;j]←M[i;l] op2 A [l;j] opl M [i;2] op2 A [2;j] opl M [i;3] op2 A [3;j] opl...opl M [i;k] op2 A [k;j]

opl and op2 can be any of the dyadic operation codes. The combination +.x represents the ordinary matrix product.

Within the processing element, for M and A arrays, each Consecutive element of M is expanded to the length of a row of A, operation 2 is executed with each expanded M and each element of the row of A. The results are stored in a set of partial answer registers. The second element of M is expanded and paired to a second row of A. Operation 2 is executed with each of these pairs of elements. The results of this set of operations are now combined with the results of the first set of operations by use of operation 1. The process is repeated until a complete row of M has been applied to the entire matrix A. This yields row 1 of the final matrix. The entire operation repeats for each row of M, until the completed answer matrix results.

The following table summarizes the Inner Product operation.

INNER PRODUCTS

A	M	Result
Scalar	Scalar	Scalar
Scalar	Vector	Scalar
Scalar	Matrix	Vector
Vector	Scalar	Scalar
Vector	Vector	Scalar ①
Vector	Matrix	Vector ②
Matrix	Scalar	Vector
Matrix	Vector	Vector ③
Matrix	Matrix	Matrix ④

① The Vectors must be of equal length. This operation is equivalent to a vector dot product with op_2 as add and op_1 as multiply.

② The length of the vector must be equal to the number of matrix columns, otherwise a length error.

③ The length of the vector must be equal to the number of matrix rows, otherwise a length error.

④ The number of M matrix columns must be equal to the number of A matrix rows, otherwise a length error.

⑤ A vector of length 1 is not treated as a scalar.

INSTRUCTION NAME: Outer Product
OP CODE: 6D
FUNCTION: (M)∘ . OP (A)→ A
PMU MACHINE FORMAT:

6				D				P	PF			I	INDEX			ADDRESS																M	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: Each element of M is algebraically combined, according to the specified opcode, with every element of A. (The opcode is specified by bits 24-31 of the Load Opcode instruction). The rank of the result is the sum of the A and M ranks (scalars have rank 0). The following table summarizes the Outer Product operation.

A	M	Result	Shape
Scalar	Scalar	Scalar	
Scalar	Vector (N)	Vector	N
Scalar	Matrix (M,N)	Matrix	M,N
Vector (M)	Scalar	Vector	M
Vector (M)	Vector (M)	Matrix	M,N
Vector (M)	Matrix	Domain Error, Vector	M
Matrix (M,N)	Scalar	Matrix	M,N
Matrix (M,N)	Vector	Domain Error, Matrix	M,N
Matrix (M,N)	Matrix	Domain Error, Matrix	M,N

If either A or M is complex, both must be complex, otherwise a Domain Error occurs.

```
A [1;1]←M [1] op A [1],A [1;2]←M [1] op A [2],...A [1;j]←M [1] op A [j]
A [2;1]←M [2] op A [1],A [2;2]←M [2] op A [2],...A [2;j]←M [2] op A [j]
A [3;1]←M [3] op A [1],A [3;2]←M [3] op A [2],...A [3;j]←M [3] op A [j]
            .
            .
            .
A [i;1]←M [i] op A [1],A [i;2]←M [i] op A [2],...A [i;j]←M [i] op A [j]
```

Each element of vector M is treated as a scalar and extended to equal the length of vector A. The specified dyadic operation is executed for each repeated M against the vector A. Each element of M produces a row vector when combined with A. The resulting matrix has the column dimensions of the length of M and the row dimension of the length of A. Any of the dyadic operation codes can be used in conjunction with the outer product instruction (i.e., preset into the array accumulator).

If either A or M are scalar, the operation code is executed as in generalized array operations previously discussed.

If the sum of ranks of A and M exceeds two (scalars have rank of 0) an interrupt will be sent to the PMU.

INSTRUCTION NAME: Reduction Along Row

OP CODE: 6E

FUNCTION: OP⌿(A)→ A

PMU MACHINE FORMAT:

6				E				P	PF			X	XXX			XXXX																X	1	T	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

DEFINITION: An expression is formed for each row of the array contained in A by placing the dyadic operator associated with reduction between each element. The evaluation of these expressions is placed in the Accumulator. A matrix reduced in this manner results in a vector. A vector reduced in this manner results in a scalar. A scalar reduced remains a scalar.

The dyadic operation code associated with this reduction instruction is operation 2 as loaded by the Load Op Code instruction (Op Code 37).

The order or reduction is from right to left. The subtract SBR operation produces an alternating sum. The divide DVR operation produces an alternating product.

Any interrupt that would normally be produced by by execution of the dyadic operation provided with the reduction instruction may be sent to the PMU.

```
A [1]←A [1;1] op A [1;2] op A [1;3] op...op A [1;j]
A [2]←A [2;1] op A [2;2] op A [2;3] op...op A [2;j]
A [3]←A [3;1] op A [3;1] op A [3;3] op...op A [3;j]
   .     .          .          .
   .     .          .          .
   .     .          .          .
A [i]←A [i;1] op A [i;2] op A [i;3] op...op A [i;j]
```

The contents of A_1 are moved from A to M. The first element is placed in the Accumulator. The dyadic operation result replaces A. The process repeats through the entire row. If the reduction is a matrix, the first row result is placed in the Accumulator Answer register area so that the next row can be similarly reduced. Again, the process continues until the entire array has been processed.

EXAMPLE:

Op Code	ACC	Result
+	1,2,3	6
-	1,2,3,4	-2
+	1,2,3,4 5,6,7,8	10, 26
Any Op	2,0	0
	NULL 30	0,0,0
	NULL 03	- NULL VECTOR
	MEMORY	1
	SHAPE	2
	3,1 1 2 3	2

Note: If A is complex and op2 is multiply, a true complex multiply is not performed. If A is matrix and has 1 column or a vector with one element, the accumulator remains unchanged (operands still subject to precision control as specified by bit 8 of the instruction).

INSTRUCTION NAME: Compression Along Columns

OP CODE: 6F

FUNCTION: M/A → A

PMU MACHINE FORMAT:

<table>
<tr><td colspan="4">6</td><td colspan="4">F</td><td>P</td><td colspan="3">PF</td><td>I</td><td colspan="3">INDEX</td><td colspan="16">ADDRESS</td><td>M</td><td>1</td><td>T</td><td>P</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td></tr>
</table>

DEFINITION: The value of each M element is tested for Zero. If zero, the corresponding A element is deleted from the result accumulator. If the M element is non-zero, the corresponding A element is included in the result accumulator.

If the M element is complex or matrix, or double precision, a domain error occurs. The following summarizes the Compress operator.

M	A	Result
Scalar	Scalar	Vector
Vector	Scalar	Vector
Matrix	Scalar	Domain Error
Scalar	Vector	Vector
Vector	Vector	Vector A&M must have the same length. Otherwise a length error is enabled.
Matrix	Vector	Domain Error
Scalar	Matrix	Matrix

M	A	Result
Vector	Matrix	Matrix. The Length of M must be equal to the number of rows of A.
Matrix	Matrix	Domain

Note: Length and Domain Errors leave the accumulator unchanged.

EXAMPLE:

M	ACC	Result	
0,0,1,0,1	1,2,3,4,5	3,5	Vector
1,0,1	1,2,3	1,2,3	Matrix
	4,5,6	7,8,9	
	7,8,9		
1,0,1	NULL SHAPE 30	NULL SHAPE 20	

5.3.16 Generalized Array Operations

The Array Mode is entered when AP scalar instructions listed in Section 5.3 encounter non-scalar or complex operands. This form of array processing is broken into the following forms:

- ° AP Dyadic Arithmetic
- ° AP Monadic Arithmetic
- ° AP Compares
- ° AP Stores
- ° AP Transfers
- ° AP Load

° AP Dyadic Arithmetic

The generalized array operation, M op A, is performed. If one operand is complex, both must be complex; otherwise, a domain error is signaled. The A and M operand must be conformal as summarized by the following table:

A	M	Result
Scalar	Scalar	Scalar
Scalar	Vector (M)	Vector (M)
Scalar	Matrix (M,N)	Matrix (M,N)
Vector (J)	Scalar	Vector (J)
Vector (J)	Vector (M)	Vector M - M&N must be equal; otherwise a length error occurs.
Vector (J)	Matrix (M,N)	A domain error occurs.
Matrix (J,K)	Scalar	Matrix (J,K)
Matrix (J,K)	Vector (M)	Domain Error
Matrix (J,K)	Matrix (M,N)	Matrix (J,K) - If J≠M or K≠N, a length error occurs.

If either M or A is null, the result is null. Domain errors are signalled before length errors. Boolean operations on complex operands are done pairwise.

° AP Monadic Arithmetics (e.g., ABS)

The specified AP Monadic is applied to the contents of the accumulator. The shape of the accumulator is unchanged.

Note: The square root of a complex produces the square root of the real part and then the square root of the imaginary part.

° AP Compares

The specified AP Compare is performed with A and M. The rules of AP Dyadic Arithmetic holds with the additional rule that neither A or M are complex.

° AP Stores

The accumulator operand is stored according to the effective address specified.

° AP Transfers

The operand produced by performing a 1↑A (TAKE) operation (or 11↑A for matrices) on the accumulator replaces the accumulator. This accumulator is used to determine whether a transfer is performed or not.

° AP Load

The previous accumulator is replaced by the contents of the effective address. No length or domain checking occurs.

5.3.17 Reductions on Null Elements

Three instructions perform, as part of their execution, a reduction. These instructions are:

- 6C - Inner Product
- 62 - Outer Product Reduction
- 6E - Reduction

For Inner Product and Reduction, when the reduction is applied to a null element, zeros are always produced. (In Outer Production Reduction, this does not apply since the shape of the answer is the shape of M). Thus, the reduction of a 3 by 0 matrix is three zeros. This is a differentiation from what would be expected with APL. In APL, zeros are not always produced; the identity operand is produced. Thus, a + reduction on a 3 by 0 matrix is three zeros, a X reduction is three 1's, and max reduction is three -∞'s (infinity).

6. TRAP STRUCTURE

6.0 TRAP STRUCTURE

The DPE possesses an instruction lookahead fetch-cycle. In this structure more than one trap can occur at any given time. To properly process this kind of trap system, the PMU computer stratifies traps into levels which are vectored directly into a trap location of memory. This process is based on priorities and is completely mechanized in the hardware.

Normal execution of a program is performed on level Ø (lowest priority) and traps occurring on higher levels will interrupt the current level and be serviced immediately. A comparison between trap levels (trap being honored and present trap) is performed by the hardware to determine the appropriate level to service. Lower priority, previous trap levels are placed in a stack to be executed when the higher levels have been completed. For example:

Present Program Trap Level	Parameter Stack	Trap
0	Prog. flow	Instruction Trace Occurs
1	0	Instruction Trace Routine
1	0	PMU Overflow Trap Occurs
5	1 0	Overflow Routine
5	1 0	Finish Overflow Routine, execute "Return Stack to P and Proceed" to return
1	0	Returned to Instruction Trace routine, execute "Return Stack to P and Proceed"
0	Prog. flow	Return to program flow

6.1 Trap Levels

There are 19 internal traps generated in the DPE configuration. Table 6.1 below lists the DPE trap interrupts and their priority level.

	Priority Level
Parity Error	19
Interval Timer	18
AP Deferral Overflow	17
AP Deferral Underflow	16
Kernel Protect	15
Read Protect	14
Write Protect	13
Command Protect	12
Page Error	11
Kernel Out of Bounds	10
Illegal Instruction	9
AP Underflow	8
AP Overflow	7
AP Store Error	6
PMU Arithmetic Overflow	5
AP Domain Error	4
AP Length Error	3
Instruction Trace	2
Kernel Trace	1

DPE Trap Interrupts

Table 6.1

6.2 Trap Mechanism

All traps are processed in the PMU and are honored when all of the following conditions are true.

1) No other instruction is in process.

2) No externally generated instruction is pending.

3) The priority of the trap attempting to be honored is greater than the present priority of the trap being honored. The absence of any traps is a trap priority level 0 and indicates normal program execution.

4) The priority of the trap attempting to be honored is greater than the priority of any other trap that is trying to be honored. That is, two or more traps can occur simultaneously and only one can be honored at a given time.

When a trap occurs and is to be honored, the trap mechanism vectors to a given location in memory specific to the trap. The specific memory location is determined by using the Procedure Page referenced by Word 255 of the Kernel Page indicated by the Procedure Kernel Register. Within this page, the word to be executed is in the word location equivalent to the priority level of the trap as given in Table 6.1. For example, a parity error trap would vector to the word in location 19 of the virtual page.

The instruction executed should either be a TRANSFER AND STACK (Op Code 51), TRANSFER AND STACK KERNEL N (Op Codes 54-57), or TRANSFER TO EXECUTIVE (Op Code 37). A TRANSFER AND STACK type instruction results in a transfer to a user provided trap routine dedicated to the trap and adjustment of the program trap level to the priority of the trap honored. A TRANSFER TO EXECUTIVE results in the generation of a two word interrupt to the executive and the PMU entering the HALT state. The address field (bits 16-31) of the TRANSFER TO EXECUTIVE instruction is a programmer defined mask that by convention should indicate the type of internal trap generated. If any other instruction is executed, the trap is masked and the program level remains unchanged. It should be noted that regardless of the instruction executed, the trap indication causing the trap to be honored is reset. If the instruction executed does not result in a change in the program counter, then upon completion of the executed instruction, the next sequential instruction is fetched and intepreted.

The traps which can be generated within a DPE and the conditions which generate them are given in the following paragraphs.

6.3 Parity Error Trap (No. 19)

This trap is generated whenever a read operation discloses a parity error, unless the Parity Error Inhibit Indicator is set. (See Section 4.1.1.7)

6.4 Interval Timer Trap (No. 18)

This trap is generated when the contents of the interval timer register has decremented to zero.

6.5 AP Deferral Overflow Trap (No. 17)

At the completion of the PMU servicing of the present instruction, when the next action function is performed, the AP Deferral Overflow trap is honored. The AP Deferral Overflow occurs as a result of an attempt to perform a deferral push when the deferral is full. The hardware will already have executed appropriate store deferral instructions. This trap is generated to indicate that an overflow had occurred.

6.6 AP Deferral Underflow Trap (No. 16)

At the completion of the PMU servicing of the present instruction, when the next action function is performed, the Ap Deferral Underflow trap is honored. The AP Deferral Underflow occurs as a result of an attempt to perform a deferral pop when the deferral is empty. The hardware will already have executed appropriate load deferral instructions. This trap is generated to indicate that an underflow had occurred.

6.7 Kernel Protect Trap (No. 15)

When a Kernel Protect violation is detected, the remaining execution sequence of the instruction being executed is aborted and the kernel protect trap honored. The kernel protect trap results from any direct addressing (read or write) of the kernel area defined by the lower bound indicator with the mode being 0, indicating problem state (not executive mode), or an invalid attempt to perform a Load Data Kernel or Initiate New Task instruction. There are two exceptions to this condition. If the instruction executed is either an Initiate New Task or Load Data Kernel and the addressed kernel entry allows a load of the Kernel area, no violation occurs. This violation cannot occur in executive mode or if the active source is I-Source and it indicates executive (EF). When this violation is detected, the attempted Read or Write is inhibited.

6.8 Read Protect Trap (No. 14)

When a Read Protect violation is detected, the remaining execution sequence of the instruction being executed is aborted and the Read Protect trap honored. The Read Protect trap results from an attempt to perform a read operation and Bit 32 of the kernel entry associated with the virtual address being 1. This violation cannot occur in executive mode. (Read-modify-write instructions are considered to be write operations.)

6.9 Write Protect Trap (No. 13)

When a Write Protect violation is detected, the remaining execution sequence of the instruction being executed is aborted and the Write Protect trap honored. The Write Protect trap results from an attempt to perform a write operation and Bit 33 of the kernel entry associated with the virtual address being 1. This violation cannot occur in executive mode.

6.10 Command Protect Trap (No. 12)

When a Command Protect violation is detected, the remaining execution sequence of the instruction being executed is aborted and the Command Protect trap honored. The Command Protect trap results from an attempt to perform a Command Subsystem instruction, a Transfer instruction or a Page Carry Sequence, and Bit 34 of the kernel entry associated with the virtual address being 1. This violation cannot occur in executive mode.

6.11 Page Error Trap (No. 11)

When a page of data is requested and exactly 256 words are not received, or an instruction which specifically references a page is executed and the referenced operand is not page oriented, the Page Error trap is honored. When this violation is detected, the remaining execution sequence of the instruction being executed is aborted and the trap is honored.

6.12 Kernel Out of Bounds Trap (No. 10)

When Kernel Out of Bounds violation is detected, the remaining execution sequence of the instruction being executed is aborted and the kernel out of bounds trap honored. The kernel out of bounds trap is caused when an attempt is made to set, or when the value of, the Data Kernel Register or the Procedure Kernel Register is greater than the Lower Bound Register value. The values of the kernel registers will not be changed. If the out of bounds condition occurs as a result of a Set Task Parameter instruction, the kernel registers will be incorrect.

6.13 Illegal Instruction Trap (No. 9)

When an illegal PMU instruction is detected, the remaining execution sequence of that instruction is aborted and the illegal instruction trap honored. When an illegal AP instruction is detected, the remaining execution sequence of that instruction is treated as 'no operation,' an illegal instruction trap is transmitted to the PMU, and the AP next action sequence performed. At the completion of the PMU servicing of the present instruction, when the PMU next action function is performed, the Illegal Instruction trap is honored.

6.14 AP Underflow Trap (No. 8)

When an exponent Underflow is detected (the exponent cannot express the proper negative value), an AP Underflow trap is transmitted to the PMU, and the AP next action sequence performed. At the completion of the PMU servicing of the present instruction, when the PMU next action is performed, the AP Underflow trap is honored.

6.15 AP Overflow Trap (No. 7)

When an exponent Overflow is detected (the exponent cannot express the proper positive value), or a divide by zero, an AP Overflow trap is transmitted to the PMU, and the AP next action sequence performed. At the completion of the PMU servicing of the present instruction, when the PMU next action is performed, the AP Overflow trap is honored.

6.16 AP Store Error Trap (No. 6)

When the AP accumulator exponent is converted to a memory exponent (8 bits to 7 bits), and the conversion process is not exact (the higher order two bits of the accumulator exponent are not equal), an AP Store Error trap is transmitted to the PMU, and the AP next action sequence performed. At the completion of the PMU servicing of the present inst uction, when the PMU next action is performed, the AP Store Error trap is honored.

6.17 PMU Arithmetic Overflow Trap (No. 5)

When the overflow condition is detected, the overflow trap is honored when the next action function is performed.

6.18 AP Domain Error Trap (No. 4)

When a Domain Error is detected (during the execution sequence of an array operation), the remaining execution sequence is aborted, and the AP Domain Error trap is transmitted to the PMU. When the PMU next action function is performed, the AP Domain Error is honored. A Domain Error occurs when the specified operation is not defined for the given values of the arguments. The value of arguments refers to data type and structure (scalar, array).

6.19 AP Length Error Trap (No. 3)

When a Length Error is detected (during the execution sequence of an array operation), the remaining execution sequence is aborted, and the AP Length Error trap is transmitted to the PMU. When the PMU next action function is performed, the AP Length Error trap is honored. A Length Error occurs when an operation is not defined due to the absence of conformable arrays. Arrays are non conformable because combining coordinates have unequal lengths.

6.20 Instruction Trace Trap (No. 2)

At the completion of the PMU servicing of the present instruction, when the next action function is performed, the instruction trace trap is honored. The instruction trace trap is indicated by Bit 34 of the internal instruction, or the second instruction word of a Two Word I/O with Indexing instruction, being one.

6.21 Kernel Trace Trap (No. 1)

At the completion of the PMU servicing of the present instruction, when the next action function is performed, the kernel trace trap is honored. The kernel trace trap is indicated by Bit 9 of the kernel entry accessed being 1.

6.22 Instruction Abort

The PMU, under certain circumstances, will cause the remaining processing of the current instruction being executed to be aborted. These circumstances are any of the following conditions:

1) An interval of 153.6 microseconds have passed since the generation of an Interval Timer Trap, during which interval the trap was not honored.

2) A Kernel Protect, Read Protect, Write Protect, Command Protect, Page Error, or Kernel Out of Bounds trap is generated.

3) An illegal instruction has been fetched.

6.23 Computer Failure Signal

The PMU, under certain circumstances, will raise a signal to the external environment indicating a Computer Failure. Once generated, this signal will remain on until a reset condition is received.

The circumstances which will cause a Computer Failure signal are any of the following conditions.

1) A Parity Error Trap occurs during the processing of a previous Parity Error Trap.

2) An interval of 153.6 microseconds have passed since an instruction abort caused by an Interval Timer Trap not being honored and the trap still has not been honored. This amounts to a total interval of 307.2 microseconds since the occurrence of the Interval Timer Trap, during which time the trap was not honored.

The computer failure signal is reset when the reset line of the DPE is enabled.

BLANK

APPENDIX A

GLOSSARY

A	The arithmetic processor A register or accumulator
(A)	The contents of the arithmetic processor A registers
AADC	All Application Digital Computer System
AE	Arithmetic processor accumulator exponent
A_I	Integerized value of the AP accumulator
AP	Arithmetic Processor
APQ	Arithmetic Processor Queue
Array	Is a non sealer operand that can have either one dimension (vector) or two dimensions (matrix)
BORAM	Block Oriented Random Access Memory
C	The command protect bit
D	The effective address or a half word operand after all address modifications and/or operand fetching
DCM	Data Communicator Module
DD	Represents a full word operand after all address modifications and operand fetching
DPE	Data Processing Element
DT	Data Type
DU	Deferral Unit or Stack
E	External subsystem specified by the contents of the active source register
EOB	End of Block
FB	Fanout Box
FIFO	First In/First Out
H	A high scratchpad register (SP [16] through [31]) determined by the SPA field of the instruction word
(H)	Represents the contents of H
I	Indirection Bit
I-Source	Interrupt Source Register
LIFO	Last In/First Out
M	Memory reference bit or the AP M Register
(M)	Contents of the AP M Register
M_D	The memory location referenced by the effective address
(M_D)	The contents of M_D

OP CODE	Bits 0 through 7 of the instruction word, designated in hexadecimal notation
OPND	Operand
P	Program Counter or Parity bit
(P)	The contents of the Program Counter
PEI	Parity Error Inhibit Indicator
PF	Parenthetical Field
PMU	Program Management Unit
P-Source	Program Source Register
Q_I	Integerized Quotient
Q_R	Quotient Residue (Remainder)
R	A low scratchpad register (SP [0] through SP [15]) determined by the SPA field of the instruction word or the Read Protect Bit
(R)	The contents of the low scratchpad register
RAMM	Random Access Main Memory
Real	Any operand, sealer or array, that is not complex
REP	Replacement
RS	Represents scratchpad registers R and S treated as a single full word register
(RS)	Contents of RS
S	Represents the low order part of full word register pair when required for full word operations
(S)	Contents of S
SP	Scratchpad
SPA	Scratchpad Register Address indicated by bits 8 through 11 of the instruction word
SP [n]	Scratchpad Register No. n
T	Trace Bit
TM	Task Memory or local memory
TVD	Test Valid Flip Flop
W	Write Protect Bit

APPENDIX B

Numerical Listing of PMU Op Codes

Op Code	Instruction	Page
00	Test and Reset To Output	4-276
01	Proceed	4-243
02	Read Word To Output	4-277
03	Write Word From Input	4-278
04	Read Operand To Output	4-287
05	Write Operand From Input	4-292
06	Read Page To Output	4-289
07	Write Page From Input	4-294
0C	Read Array To Output	4-290
0D	Write Array From Input	4-296
0E	Read Indirect Word to Output	4-291
10	Load High SP	4-200
11	Store High SP	4-225
12	Load SP	4-198
13	Store SP	4-223
16	Load SP Full	4-202
17	Store SP Full	4-227
19	Move Half To Half	4-237
1D	Move Full To Full	4-239
1F	Move Full And Stack	4-241
20	Load Left Byte	4-204
21	Store Left Byte	4-228
22	Load Right Byte	4-206
23	Store Right Byte	4-230
25	Set System Parameters	4-258
27	Execute	4-244
28	Initiate New Task	4-263
29	Set Task Parameters	4-256
30	Load Absolute Value	4-207
31	Load Negative	4-211
32	Interval Timer Control/Halt	4-245
33	Store Interval Timer	4-247
34	Load Absolute Full	4-209
35	Load Negative Full	4-212
36	Test and Reset	4-266
37	Transfer To Executive	4-188

40	Transfer Unconditional	4-175
41	Transfer If R Negative	4-180
42	Transfer If R Is Zero	4-176
43	Transfer If Not Equal	4-184
44	Load Data Kernel	4-218
45	Transfer If R Neg Full	4-182
46	Transfer If R Zero Full	4-178
47	Command Subsystem/Address Modification	4-268
4F	Command Subsystem/Immediate Execution	4-272
50	Transfer On Incremented SP	4-186
51	Transfer And Stack	4-192
52	Return Stack To P	4-248
53	Return Stack to P and Proceed	4-250
54	Transfer and Stack Kernel 0	4-196
55	Transfer and Stack Kernel 1	4-196
56	Transfer and Stack Kernel 2	4-196
57	Transfer and Stack Kernel 3	4-196
60	Single Word I/O Command	4-279
61	Skip If Greater Than	4-146
62	Skip If Equal To	4-138
63	Skip If Not Less Than	4-158
64	Load Page	4-221
65	Skip If Greater Than Full	4-148
66	Skip If Equal To Full	4-140
67	Skip If Not Less Than Full	4-160
70	Skip If Less Than	4-154
71	Skip If Not Equal To	4-142
72	Skip If Not Greater	4-150
73	Two Word I/O Command	4-281
74	Skip If Less Than Full	4-156
75	Skip If Not Equal To Full	4-144
76	Skip If Not Greater Than Full	4-152
77	Store Page	4-235
7B	Two Word I/O With Indexing	4-284
80	Mask Load	4-214
81	Reset Bit N	4-252
82	Set Bit N	4-254
83	Skip On Bit N	4-162
84	Load Control Bits	4-216
85	Store Control Bits	4-232
86	Load Multiple	4-217
87	Store Multiple	4-233

90	$\overline{D}$ And R	4-82
91	Convert 2's To SM	4-164
92	XOR	4-110
93	NAND	4-102
94	$\overline{D}$ And R Full	4-84
95	Convert 2's To SM Full	4-166
96	XOR Full	4-112
97	NAND Full	4-104
A0	And	4-78
A1	XNOR	4-114
A2	Convert SM To 2's	4-168
A3	D Or $\overline{R}$	4-98
A4	AND Full	4-80
A5	XNOR Full	4-116
A6	Convert SM To 2's Full	4-170
A7	D or $\overline{R}$ Full	4-100
B0	Add	4-62
B1	$\overline{D}$ Or R	4-94
B2	Or	4-90
B3	Subtract	4-66
B4	Add Full	4-64
B5	$\overline{D}$ Or R Full	4-96
B6	Or Full	4-92
B7	Subtract Full	4-68
C0	Multiply Half To Half	4-70
C1	NOR	4-106
C2	D And $\overline{R}$	4-86
C3	Multiply Half To Full	4-72
C5	NOR Full	4-108
C6	D and $\overline{R}$ Full	4-88
D0	Shift LRHC	4-134
D1	Shift LRFC	4-136
D2	Divide Full By Half	4-76
D3	Divide Half By Half	4-74
D6	Round	4-172
D7	Binary Normalize	4-173
E0	Shift ARHO	4-118
E1	Shift ARFO	4-120
E2	Shift ALHO	4-122
E3	Shift ALFO	4-124
E4	Escape 0	4-197
E5	Escape 1	4-197
E6	Escape 2	4-197
E7	Escape 3	4-197

F0	Shift LRHO	4-126
Fl	Shift LRFO	4-128
F2	Shift LLHO	4-130
F3	Shift LLFO	4-132
F4	Escape 4	4-197
F5	Escape 5	4-197
F6	Escape 6	4-197
F7	Escape 7	4-197

APPENDIX C

Numerical Listing of AP Op Codes

Op Code	Instruction	Page
27	Execute	5-85
31	Load Memory Word	5-48
32	Store Packed	5-49
33	Load Deferral	5-50
34	Store Deferral	5-51
35	Push Data	5-55
36	Store Operand	5-56
37	Load Op Code	5-133
50	Ravel	5-139
51	Dimension	5-137
60	Index Generator	5-138
61	Polynomial	5-135
62	Outer Product Reduction	5-140
63	Expand Along Column	5-141
64	Catenate Rows	5-142
65	Transpose	5-145
66	Reversal Along Rows	5-146
67	Laminate Rows	5-147
68	Rotate Row	5-149
69	Reshape	5-151
6A	Take	5-152
6B	Drop	5-154
6C	Inner Product	5-155
6D	Outer Product	5-158
6E	Reduction Along Row	5-160
6F	Compression Along Column	5-162
70	Boolean Zero	5-90
71	Boolean AND	5-91
72	Boolean Less Than	5-92
73	Boolean Odd Even	5-93
74	Boolean Greater Than	5-94
75	Boolean Load	5-95
76	Boolean Not Equal	5-96
77	Boolean Inclusive OR	5-97
78	Boolean NOR	5-98
79	Boolean Equals	5-99
7A	Boolean Load Complement	5-100
7B	Boolean Less Than or Equal To	5-101
7C	Boolean Not	5-102
7D	Boolean Greater Or Equal	5-103
7E	Boolean NAND	5-104
7F	Boolean One	5-105

80	Logical Zero	5-106
81	Logical AND	5-107
82	Logical Less Than	5-108
83	No Operation	5-109
84	Logical Greater Than	5-110
85	Logical Load	5-111
86	Logical Not Equal	5-112
87	Logical Inclusive Or	5-113
88	Logical NOR	5-115
89	Logical Equals	5-114
8A	Load Complement	5-116
8B	Logical Less Than Or Equal	5-117
8C	Logical Not	5-118
8D	Logical Greater or Equals	5-119
8E	Logical NAND	5-120
8F	Logical Set	5-121
90	Reset TVD Destructive	5-73
91	Compare Equal Non Destructive	5-59
92	Compare Less Than Destructive	5-58
93	Compare Less Than Or Equal Non Destructive	5-60
94	Compare Greater Than Destructive	5-61
95	Compare Greater Than Or Equal Non Destructive	5-62
96	Compare Not Equal Destructive	5-63
97	Set TVD Non Destructive	5-64
98	Reset TVD Non Destructive	5-65
99	Compare Equal Destructive	5-66
9A	Compare Less Than Non Destructive	5-67
9B	Compare Less Than Or Equal Destructive	5-68
9C	Compare Greater Than Non Destructive	5-69
9D	Compare Greater Than Or Equal Destructive	5-70
9E	Compare Not Equal Non Destructive	5-71
9F	Set TVD Destructive	5-72
A0	No Transfer	5-76
A1	Transfer On Equal To Zero	5-77
A2	Transfer On Greater Than Zero	5-78
A3	Transfer On Greater Than Or Equal to Zero	5-79
A4	Transfer On Less Than Zero	5-80
A5	Transfer On Less Than Or Equal To Zero	5-81
A6	Transfer On Not Equal To Zero	5-82
A7	Unconditional Transfer	5-83
A8	No Transfer	5-76
A9	Transfer On Equal To Zero	5-77
AA	Transfer On Greater Than Zero	5-78
AB	Transfer On Greater Than Or Equal To Zero	5-79
AC	Transfer On Less Than Zero	5-80
AD	Transfer On Less Than Or Equal To Zero	5-81
AE	Transfer On Not Equal To Zero	5-82
AF	Unconditional Transfer	5-83

B0	Signim	5-41
B2	Minimum	5-74
B4	Maximum	5-75
B8	Unpack	5-52
B9	Load Word To Accumulator	5-54
BC	Transfer On Test Valid Set	5-84
C1	Addition	5-16
C2	Subtract	5-21
C4	Reverse Subtract	5-19
C5	Load Accumulator	5-37
CA	Load Negative	5-38
CC	Negation	5-39
CF	Absolute Value	5-40
D0	Divide	5-27
D2	Divide Short	5-33
D4	Divide Residue	5-29
D8	Reverse Divide	5-25
DA	Reverse Divide Short	5-35
DC	Reverse Divide Residue	5-31
E0	Multiply	5-23
E2	Square Root	5-44
E4	Floor	5-42
E5	Ceiling	5-43
E8	Store And Halt	5-46
EA	Store And Proceed	5-47
F0	Shift Open	5-86
F1	Shift Cyclic	5-87
F4	Normalize	5-45
F8	Shift Single Open	5-88
F9	Shift Single Closed	5-89

BLANK

APPENDIX D

Alphabetical Listing of PMU Op Codes

Instruction	Op Code	Page
Add	B0	4-62
Add Full	B4	4-64
AND	A0	4-78
AND Full	A4	4-80
Binary Normalize	D7	4-173
Command Subsystem/Address Modification	47	4-268
Command Subsystem/Immediate Execution	4F	4-272
Convert 2's to SM	91	4-164
Convert 2's to SM Full	95	4-166
Convert To 2's	A2	4-168
Convert SM To 2's	A6	4-170
D And $\overline{R}$	C2	4-86
D And $\overline{R}$ Full	C6	4-88
$\overline{D}$ And R	90	4-82
$\overline{D}$ And R Full	94	4-84
D Or $\overline{R}$	A3	4-98
D Or $\overline{R}$ Full	A7	4-100
$\overline{D}$ Or R	B1	4-94
$\overline{D}$ Or R Full	B5	4-96
Divide Full By Half	D2	4-76
Divide Half By Half	D3	4-74
Escape 0	E4	4-197
Escape 1	E5	4-197
Escape 2	E6	4-197
Escape 3	E7	4-197
Escape 4	F4	4-197
Escape 5	F5	4-197
Escape 6	F6	4-197
Escape 7	F7	4-197
Execute	27	4-244
Initiate New Task	28	4-263
Interval Timer Control/Halt	32	4-245

Load Absolute Value	30	4-207
Load Absolute Full	34	4-209
Load Left Byte	20	4-204
Load Right Byte	22	4-206
Load Control Bits	84	4-216
Load Data Kernel	44	4-218
Load Multiple	86	4-217
Load Negative	31	4-211
Load Negative Full	35	4-212
Load Page	64	4-221
Load High SP	10	4-200
Load SP	12	4-198
Load SP Full	16	4-202
Mask Load	80	4-214
Move Full And Stack	1F	4-241
Move Full To Full	1D	4-239
Move Half To Half	19	4-237
Multiply Half To Full	C3	4-72
Multiply Half To Half	C0	4-70
NAND	93	4-102
NAND Full	97	4-104
NOR	C1	4-106
NOR Full	C5	4-108
OR	B2	4-90
OR Full	B6	4-92
Proceed	01	4-243
Read Array To Output	0C	4-290
Read Indirect Word To Output	0E	4-291
Read Operand To Output	04	4-287
Read Page To Otuput	06	4-289
Read Word To Output	02	4-277
Reset Bit N	81	4-252
Return Stack To P	52	4-248
Return Stack To P And Proceed	53	4-250
Round	D6	4-172
Set Bit N	82	4-254
Set System Parameters	25	4-258
Set Task Parameters	29	4-256
Shift ALFO	E3	4-124
Shift ALHO	E2	4-122

Shift ARFO	E1	4-120
Shift ARHO	E0	4-118
Shift LLFO	F3	4-132
Shift LLHO	F2	4-130
Shift LRFC	D1	4-136
Shift LRHC	D0	4-134
Shift LRFO	F1	4-128
Shift LRHO	F0	4-126
Single Word I/O Command	60	4-279
Skip If Equal To	62	4-138
Skip If Equal To Full	66	4-140
Skip If Greater Than	61	4-146
Skip If Greater Than Full	65	4-148
Skip If Less Than	70	4-154
Skip If Less Than Full	74	4-156
Skip If Not Equal To	71	4-142
Skip If Not Equal To Full	75	4-144
Skip If Not Greater	72	4-150
Skip If Not Greater Than Full	76	4-152
Skip If Not Less Than	63	4-158
Skip If Not Less Than Full	67	4-160
Skip On Bit N	83	4-162
Store Byte Left	21	4-228
Store Byte Right	23	4-230
Store Control Bits	85	4-232
Store Interval Timer	33	4-247
Store Multiple	87	4-233
Store Page	77	4-235
Store SP	13	4-223
Store SP Full	17	4-227
Store High Scratchpad	11	4-225
Subtract	B3	4-66
Subtract Full	B7	4-68
Test And Reset	36	4-266
Test And Reset To Output	00	4-276
Transfer And Stack	51	4-192
Transfer And Stack Kernel 0	54	4-196
Transfer And Stack Kernel 1	55	4-196
Transfer And Stack Kernel 2	56	4-196
Transfer And Stack Kernel 3	57	4-196
Transfer If Not Equal	43	4-184
Transfer If R Is Zero	42	4-176
Transfer If R Is Zero Full	46	4-178
Transfer If R Negative	41	4-180
Transfer If R Negative Full	45	4-182

Transfer On Incremented SP	50	4-186
Transfer To Executive	37	4-188
Transfer Unconditional	40	4-175
Two Word I/O Command	73	4-281
Two Word I/O With Indexing	7B	4-284
Write Array From Input	0D	4-296
Write Operand From Input	05	4-292
Write Page From Input	07	4-294
Write Word From Input	03	4-278
XNOR	A1	4-114
XNOR Full	A5	4-116
XOR	92	4-110
XOR Full	96	4-112

APPENDIX E

Alphabetical Listing of AP Op Codes

Instruction	Op Code	Page
Absolute Value	CF	5-40
Addition	C1	5-16
Boolean AND	71	5-91
Boolean Equals	79	5-99
Boolean Greater Or Equal	7D	5-103
Boolean Greater Than	74	5-94
Boolean Inclusive OR	77	5-97
Boolean Less Than Or Equal	7B	5-101
Boolean Less Than	72	5-92
Boolean Load Bit	75	5-95
Boolean Load Complement	7A	5-100
Boolean NAND	7E	5-104
Boolean NOR	78	5-98
Boolean NOT	7C	5-102
Boolean Not Equal	76	5-96
Boolean Odd Even	73	5-93
Boolean One	7F	5-105
Boolean Zero	7D	5-90
Catenate Rows	64	5-143
Ceiling	E5	5-43
Compare Equal Destructive	99	5-66
Compare Equal Non Destructive	91	5-59
Compare Greater Than Destructive	94	5-61
Compare Greater Than Non Destructive	9C	5-69
Compare Greater Than Or Equal Destructive	9D	5-70
Compare Greater Than Or Equal Non Destructive	95	5-62
Compare Less Than Destructive	92	5-58
Compare Less Than Non Destructive	9A	5-67
Compare Less Than Or Equal Destructive	9B	5-68
Compare Less Than Or Equal Non Destructive	93	5-60
Compare Not Equal Destructive	96	5-63
Compare Not Equal Non Destructive	9E	5-71
Compression Along Column	6F	5-162
Dimension	51	5-137
Divide	D0	5-27
Divide Residue	D4	5-29
Divide Short	D2	5-33
Drop	6B	5-154
Execute	27	5-85
Expand Along Column	63	5-141

Floor	E4	5-42
Index Generator	60	5-138
Inner Product	6C	5-155
Laminate Rows	67	5-147
Load Accumulator	C5	5-37
Load Compliment	8A	5-116
Load Deferral	33	5-50
Load Memory Word	31	5-48
Load Negative	CA	5-38
Load Op Code	37	5-133
Load Word To Accumulator	B9	5-54
Logical AND	81	5-107
Logical Equals	89	5-114
Logical Greater Or Equals	8D	5-119
Logical Greater Than	84	5-110
Logical Inclusive OR	87	5-113
Logical Less Than	82	5-108
Logical Less Than Or Equals	8B	5-117
Logical Load	85	5-111
Logical NAND	8E	5-120
Logical NOR	88	5-115
Logical NOT	8C	5-118
Logical Not Equal	86	5-112
Logical Set	8F	5-121
Logical Zero	80	5-106
Maximum	B4	5-75
Minimum	B2	5-74
Multiply	E0	5-23
Negation	CC	5-39
Normalize	F4	5-45
No Operation	83	5-109
No Transfer	A0 or A8	5-76
Outer Product	6D	5-158
Outer Product Reduction	62	5-140
Polynominal	61	5-135
Push Data	35	5-55
Ravel	50	5-139
Reduction Along Row	6E	5-160
Reset TVD Destructive	90	5-73
Rest TVD Non Destructive	98	5-65
Reshape	69	5-151
Reversal Along Rows	66	5-146

Reverse Divide	D8	5-25
Reverse Divide Residue	DC	5-31
Reverse Divide Short	DA	5-35
Reverse Subtract	C4	5-19
Rotate Row	68	5-149
Set TVD Destructive	9F	5-72
Set TVD Non Destructive	97	5-64
Shift Cyclic	Fl	5-87
Shift Open	F0	5-86
Shift Single Closed	F9	5-89
Shift Single Open	F8	5-88
Signum	B0	5-41
Square Root	E2	5-44
Store and Halt	E8	5-46
Store and Proceed	EA	5-47
Store Deferral	34	5-51
Store Operand	36	5-56
Store Packed	32	5-49
Subtract	C2	5-23
Take	6A	5-152
Transfer On Equal To Zero	Al or A9	5-77
Transfer On Greater Than Or Equal To Zero	A3 or AB	5-79
Transfer On Greater Than Zero	A2 or AA	5-78
Transfer On Less Than Or Equal To Zero	A5 or AD	5-81
Transfer On Less Than Zero	A4 or AC	5-80
Transfer On Not Equal To Zero	A6 or AE	5-82
Transfer On Test Valid Set	BC	5-84
Transpose	65	5-145
Unconditional Transfer	A7 or AF	5-83
Unpack	B8	5-52

BLANK

APPENDIX F

CONSIDERATIONS IN PREPARING PROGRAMS FOR THE DPE

1. Indirect Dimension Words are not provided for absolute data addressing.

2. Register Replacement can only occur if the pipeline is empty or a transfer is not in the APQ. Otherwise, the present instruction will be refetched with a simultaneous pipeline clear function. This action will continue until the pipeline is clear.

3. For resident data, the packing factor on a dimension word is not interpreted.

4. For double precision word structured data, the Store and Halt instruction should be used for AP stores.

5. A page error is signalled if paged non-resident data is encountered during the executuion of AP store instructions. If the page is non-resident, the AP store should specify addressable. This will allow the fetch cycle to obtain the page. It should be noted that even if this is done, subsequently fetched instructions may, if paged data is requested, overlay the page the AP store requested. Thus, the Store Halt instruction should be used.

6. The location of an indirect dimension word should not be "write protected." When an AP store is executed which specified indirection and encountered an indirect dimension word, the accumulator dimension word replaces the dimension word in memory. Additionally, the dimension word may be stored correctly but a write protect may occur when the operand part of the array is stored.

7. If an AP store references its storage location via an indirect dimension word, the store must be a Store and Halt instruction.

8. The Deferral Overflow/Underflow Pointer maintained in SP [27] should reference page resident, or word structured data pointing to a non used RAMM. Otherwise, the system pipeline may stall under certain pipeline conditions which result in queues being full.

9. All rèferences to array or complex operands using word structured data must specify pipeline halt (bit 3 of the Kernel Word being zero).

10. All AP instructions whose op codes bits 0, 1, 2, 3 are (hex) 2, 3, 5, or 6 will cause the pipeline to halt until their execution sequences are completed.

11. Length and domain errors leave the accumulator unchanged.

12. Vectors of length 1 are not treated as scalers.

13. Monadic instructions (e.g., Transpose) should never be specified as addressables (bit 32 of the instruction being 1). Failure to adhere to this will result in unnecessary operand fetch time and ambiguity in the result accumulator when the previous accumulator is a vector and the requested operand is fetched from RAM.

14. All manipulation involving complex operands should specify the precision of the result as double precision.

15. For Blockmode Operations: SP[26;], SP[24;3] should be initialized to the next to last page entry. The word displacement field (bits 8-15) must be 1000 0001.

16. Need Appendix on Reset state of the machine.

17. When the conditional AP transfer is executed by the EXECUTE instruction, the PMU comes to a halt awaiting the instructions execution by the AP. This condition is analogous to an AP transfer being fetched when an AP transfer is presently in the APQ.

18. In Blockmode operations scratchpads 16-31 may all be used. Thus, programs written in an environment expecting blockmode operations <u>should not</u> use these scratchpads for storage of program variables.

APPENDIX G

PMU INSTRUCTION ATTRIBUTES

The following chart summarizes some of the more important attributes of the PMU instruction set. This chart is intended to be used only as a summary reference after gaining a thorough understanding of the instructions from their individual descriptions in Section 4. The symbolism used in the "Function" column is defined in Appendix A - Glossary. The following notes apply, where indicated in the chart.

NOTE 1 - "Inst Type" Column

This column indicates whether a given instruction is an INT (Internal) or EXT (External) type of instruction. EXT instructions are those instructions which must be received through the input queue of the channel and would cause an illegal instruction trap to occur if fetched from program counter operation. INT instructions may be fetched from program counter operation or from the channel's input queue, either directly or as the second word of a command subsystem instruction.

NOTE 2 -- "Operand Fetch" Column

This column indicates whether a given instruction is normally addressable (ADD) or normally non addressable (NON-ADD). ADD instructions may or may not have a one in bit 32 of the modified instruction word indicating a memory reference or an immediate operand, respectively. NON-ADD instructions either ignore the contents of bit 32 or, if bit 32 is one, expends a non required operand fetch from memory.

OP CODE	INSTRUCTION NAME	PAGE REF	INST TYPE (NOTE 1)	OPERAND FETCH (NOTE 2)	OPERAND SIZE	FUNCTION
PMU ARITHMETIC INSTRUCTIONS						
B0	ADD	4-62	INT	ADD	HALF	(R) + D → R
B4	ADD FULL	4-64	INT	ADD	FULL	(RS) + DD → RS
B3	SUBTRACT	4-66	INT	ADD	HALF	(R) - D → R
B7	SUBTRACT FULL	4-68	INT	ADD	FULL	(RS) - DD → RS
C0	MULTIPLY HALF TO HALF	4-70	INT	ADD	HALF	D × (R) → R
C3	MULTIPLY HALF TO FULL	4-72	INT	ADD	HALF	D × (R) → RS
D3	DIVIDE HALF BY HALF	4-74	INT	ADD	HALF	(R) ÷ D → R; REMAINDER → S
D2	DIVIDE FULL BY HALF	4-76	INT	ADD	HALF	(RS) ÷ D → R; REMAINDER → S
PMU LOGICAL INSTRUCTIONS						
A0	AND	4-78	INT	ADD	HALF	D · (R) → R
A4	AND FULL	4-80	INT	ADD	FULL	DD · (RS) → RS
90	$\overline{D}$ AND R	4-82	INT	ADD	HALF	$\overline{D} \cdot (R) \rightarrow R$
94	$\overline{D}$ AND R FULL	4-84	INT	ADD	FULL	$\overline{DD} \cdot (RS) \rightarrow RS$
C2	D AND $\overline{R}$	4-86	INT	ADD	HALF	$D \cdot (\overline{R}) \rightarrow R$
C6	D AND $\overline{R}$ FULL	4-88	INT	ADD	FULL	$DD \cdot (\overline{RS}) \rightarrow RS$
B2	OR	4-90	INT	ADD	HALF	$D \vee (R) \rightarrow R$
B6	OR FULL	4-92	INT	ADD	FULL	$DD \vee (RS) \rightarrow RS$
B1	$\overline{D}$ OR R	4-94	INT	ADD	HALF	$\overline{D} \vee (R) \rightarrow R$
B5	$\overline{D}$ OR R FULL	4-96	INT	ADD	FULL	$\overline{DD} \vee (RS) \rightarrow RS$
A3	D OR $\overline{R}$	4-98	INT	ADD	HALF	$D \vee (\overline{R}) \rightarrow R$
A7	D OR $\overline{R}$ FULL	4-100	INT	ADD	FULL	$DD \vee (\overline{RS}) \rightarrow RS$
93	NAND	4-102	INT	ADD	HALF	$\overline{D} \vee (\overline{R}) \rightarrow R$
97	NAND FULL	4-104	INT	ADD	FULL	$\overline{DD} \vee (\overline{RS}) \rightarrow RS$
C1	NOR	4-106	INT	ADD	HALF	$\overline{D} \cdot (\overline{R}) \rightarrow R$
C5	NOR FULL	4-108	INT	ADD	FULL	$\overline{DD} \cdot (\overline{RS}) \rightarrow RS$
92	XOR	4-110	INT	ADD	HALF	$D \oplus (R) \rightarrow R$
96	XOR FULL	4-112	INT	ADD	FULL	$DD \oplus (RS) \rightarrow RS$
A1	XNOR	4-114	INT	ADD	HALF	$D \overline{\oplus} (R) \rightarrow R$
A5	XNOR FULL	4-116	INT	ADD	FULL	$DD \overline{\oplus} (RS) \rightarrow RS$
PMU SHIFT INSTRUCTIONS (Note A)						
E0	SHIFT ARHO	4-118	INT	ADD	HALF	$(R) \times 2^{-N} \rightarrow R$
E1	SHIFT ARFO	4-120	INT	ADD	HALF	$(RS) \times 2^{-N} \rightarrow RS$
E2	SHIFT ALHO	4-122	INT	ADD	HALF	$(R) \times 2^{N} \rightarrow R$
E3	SHIFT ALFO	4-124	INT	ADD	HALF	$(RS) \times 2^{N} \rightarrow RS$
F0	SHIFT LRHO	4-126	INT	ADD	HALF	$(R) \times 2^{-N} \rightarrow R$
F1	SHIFT LRFO	4-128	INT	ADD	HALF	$(RS) \times 2^{-N} \rightarrow RS$
F2	SHIFT LLHO	4-130	INT	ADD	HALF	$(R) \times 2^{N} \rightarrow R$
F3	SHIFT LLFO	4-132	INT	ADD	HALF	$(RS) \times 2^{N} \rightarrow RS$
D0	SHIFT LRHC	4-134	INT	ADD	HALF	$(R) \times 2^{-N} \rightarrow R$
D1	SHIFT LRFC	4-136	INT	ADD	HALF	$(RS) \times 2^{-N} \rightarrow RS$

Note A: For OpCodes E0-E3, the shift is algebraic (the sign Bit, Bit 0, remains unchanged). For OpCodes F0-F3, D0 and D1 the shift is logical (Bit 0 is included in the shift).

OP CODE	INSTRUCTION NAME	PAGE REF	INST TYPE (NOTE 1)	OPERAND FETCH (NOTE 2)	OPERAND SIZE	FUNCTION
PMU SKIP INSTRUCTIONS						
62	SKIP IF EQUAL TO	4-138	INT	ADD	HALF	IF D = (R), (P) + 1 → P
66	SKIP IF EQUAL FULL	4-140	INT	ADD	FULL	IF DD = (RS), (P) + 1 → P
71	SKIP IF NOT EQUAL TO	4-142	INT	ADD	HALF	IF D ≠ (R), (P) + 1 → P
75	SKIP IF NOT EQUAL TO FULL	4-144	INT	ADD	FULL	IF DD ≠ (RS), (P) + 1 → P
61	SKIP IF GREATER THAN	4-146	INT	ADD	HALF	IF D > (R), (P) + 1 → P
65	SKIP IF GREATER THAN FULL	4-148	INT	ADD	FULL	IF DD > (RS), (P) + 1 → P
72	SKIP IF NOT GREATER	4-150	INT	ADD	HALF	IF D ≤ (R), (P) + 1 → P
76	SKIP IF NOT GREATER THAN FULL	4-152	INT	ADD	FULL	IF DD ≤ (RS), (P) + 1 → P
70	SKIP IF LESS THAN	4-154	INT	ADD	HALF	IF D < (R), (P) + 1 → P
74	SKIP IF LESS THAN FULL	4-156	INT	ADD	FULL	IF DD < (RS), (P) + 1 → P
63	SKIP IF NOT LESS THAN	4-158	INT	ADD	HALF	IF D ≥ (R), (P) + 1 → P
67	SKIP IF NOT LESS THAN FULL	4-160	INT	ADD	FULL	IF DD ≥ (RS), (P) + 1 → P
83	SKIP ON BIT N	4-162	INT	ADD	HALF	IF BIT$_i$ = 1 WHERE i = 0, --, 15 THEN (P) + 1 → P
PMU DATA INSTRUCTIONS						
91	CONVERT 2s TO SM	4-164	INT	ADD	HALF	$\overline{D - 1} \rightarrow R$
95	CONVERT 2s TO SM FULL	4-166	INT	ADD	FULL	$\overline{DD - 1} \rightarrow RS$
A2	CONVERT SM TO 2s	4-168	INT	ADD	HALF	$\overline{D} + 1 \rightarrow R$
A6	CONVERT SM TO 2s FULL	4-170	INT	ADD	FULL	$\overline{DD} + 1 \rightarrow RS$
D6	ROUND	4-172	INT	NONADD	FULL	$(R) \rightarrow R_{ROUND}$, $O \rightarrow S$

D7	BINARY NORMALIZE	4-173	INT	NONADD	FULL	IF (RS) POS, (RS) → RS (2^{30})
						IF (RS) NEG, (RS) → RS (-2^{30})
						SHIFT COUNT → M_D
						IF (RS) ZERO, O → M_D

PMU TRANSFER INSTRUCTIONS						
40	TRANSFER UNCONDITIONAL	4-175	INT	NONADD	HALF	D → P
42	TRANSFER IF R IS ZERO	4-176	INT	NONADD	HALF	IF (R) = O, D → P
46	TRANSFER IF R ZERO FULL	4-178	INT	NONADD	FULL	IF (RS) = O, D → P
41	TRANSFER IF R NEGATIVE	4-180	INT	NONADD	HALF	IF (R) < O, D → P
45	TRANSFER IF R NEG FULL	4-182	INT	NONADD	FULL	IF (RS) < O, D → P
43	TRANSFER IF NOT EQUAL	4-184	INT	ADD	HALF	IF D ≠ (R), (S) → P
50	TRANSFER ON INCREMENTED SP	4-186	INT	NONADD	HALF	IF (R) + 1 ≠ O, D → P
37	TRANSFER TO EXECUTIVE	4-188	INT	NONADD	FULL	CONTROL OF INT TIMER PROG CONT → EXEC
51	TRANSFER AND STACK	4-192	INT	NONADD	FULL	$\left.\begin{array}{l}\text{TRAP LEVEL IND}\\ \text{MODE IND}\\ \text{HALT IND}\\ \text{DATA AND PROC}\\ \quad\text{KERNEL REG}\\ \text{SP[31]}\end{array}\right\} \quad M_{(R) + 1}$ $\left.\begin{array}{l}\text{P-SOURCE}\\ \text{P COUNTER}\end{array}\right\} \quad M_{(R) + 2}$ (R) + 2 → SP [31] O → HALT IND D → P

OP CODE	INSTRUCTION NAME	PAGE REF	INST TYPE (NOTE 1)	OPERAND FETCH (NOTE 2)	OPERAND SIZE	FUNCTION
PMU TRANSFER INSTRUCTIONS (Cont.)						
54 (N=0) 55 (N=1) 56 (N=2) 57 (N=3)	TRANSFER AND STACK KERNEL N (N = 0, 1, 2, 3)	4-196	INT	NONADD	FULL	$\left.\begin{array}{l}\text{TRAP LEVEL IND}\\ \text{MODE IND}\\ \text{HALT IND}\\ \text{DATA KERNEL REG}\\ \text{PROC KERNEL REG}\\ \text{SP}[31]\end{array}\right\} M_{(R) + 1}$ $\left.\begin{array}{l}\text{P-SOURCE}\\ \text{P COUNTER}\end{array}\right\} M_{(R) + 2}$ (R) + 2 → SP[31] O → HALT IND N → PROC KERNEL REG D → P
E4 (N=0) E5 (N=1) E6 (N=2) E7 (N=3) F4 (N=4) F5 (N=5) F6 (N=6) F7 (N=7)	ESCAPE NUMBER N (N = 0, 1, ..., 7)	4-197	INT	ADD	FULL	(P) → 33 + 2N DD → RS $\text{LOC } (32 + 2N)_{(16-31)} \rightarrow P$
	PMU LOAD/STORE INSTRUCTIONS					
12	LOAD SP	4-198	INT	ADD	HALF	D → R
10	LOAD HIGH SP	4-200	INT	ADD	HALF	D → H
16	LOAD SP FULL	4-202	INT	ADD	FULL	DD → RS
20	LOAD LEFT BYTE	4-204	INT	ADD	HALF	$D_{(0-7)} \rightarrow R_{(0-7)}, \; O \rightarrow R_{(8-15)}$
22	LOAD RIGHT BYTE	4-206	INT	ADD	HALF	$D_{(8-15)} \rightarrow R_{(0-7)}, \; O \rightarrow R_{(8-15)}$
30	LOAD ABSOLUTE VALUE	4-207	INT	ADD	HALF	$\|D\| \rightarrow R$
34	LOAD ABSOLUTE FULL	4-209	INT	ADD	FULL	$\|DD\| \rightarrow RS$
31	LOAD NEGATIVE	4-211	INT	ADD	HALF	-D → R
35	LOAD NEGATIVE FULL	4-212	INT	ADD	FULL	-DD → RS
80	MASK LOAD	4-214	INT	ADD	HALF	$D \cdot (S) \vee (R) \cdot (\bar{S}) \rightarrow R$
84	LOAD CONTROL BITS	4-216	INT	ADD	FULL	$(M_D)_{(32-35)} \rightarrow R_{(12-15)}, \; O \rightarrow R_{(0-11)}$
86	LOAD MULTIPLE	4-217	INT	NONADD	FULL	$(M_D) \rightarrow RS$ $(M_{D + 1}) \rightarrow RS + 1$ $\vdots \qquad \vdots$ $(M_{D + 15}) \rightarrow RS + 15$
44	LOAD DATA KERNEL	4-218	INT	NONADD	FULL	DATA KERNEL PAGE → TM
64	LOAD PAGE	4-221	INT	NONADD	FULL	DATA PAGE → TM

OP CODE	INSTRUCTION NAME	PAGE REF	INST TYPE (NOTE 1)	OPERAND FETCH (NOTE 2)	OPERAND SIZE	FUNCTION
	PMU LOAD/STORE INSTRUCTIONS (Cont.)					
13	STORE SP	4-223	INT	NONADD	HALF	$(R) \rightarrow M_D$
11	STORE HIGH SP	4-225	INT	NONADD	HALF	$(H) \rightarrow M_D$
17	STORE SP FULL	4-227	INT	NONADD	FULL	$(RS) \rightarrow M_D$
21	STORE BYTE LEFT	4-228	INT	NONADD	HALF	$(R)_{(0-7)} \rightarrow M_{D(0-7)}$ OR $M_{D(16-23)}$
23	STORE BYTE RIGHT	4-230	INT	NONADD	HALF	$(R)_{(0-7)} \rightarrow M_{D(8-15)}$ OR $M_{D(24-31)}$
85	STORE CONTROL BITS	4-232	INT	NONADD	FULL	$(R)_{(12-14)} \rightarrow M_{D(32-34)}$
87	STORE MULTIPLE	4-233	INT	NONADD	FULL	$(RS) \rightarrow M_D$ $(RS + 1) \rightarrow M_{D + 1}$ . . . . $(RS + 15) \rightarrow M_{D + 15}$
77	STORE PAGE	4-235	INT	NONADD	FULL	DATA PAGE → E
19	MOVE HALF TO HALF	4-237	INT	ADD	HALF	$(M_D) \rightarrow M_{(R)}$
1D	MOVE FULL TO FULL	4-239	INT	ADD	FULL	$(M_D) \rightarrow M_{(R)}$
1F	MOVE FULL AND STACK	4-241	INT	ADD	FULL	$(M_D) \rightarrow M_{(R + 1)}$, $(R) + 1 \rightarrow R$
	PMU CONTROL INSTRUCTIONS					
01	PROCEED	4-243	INT	NONADD	HALF	O → HALT IND
27	EXECUTE	4-244	INT	NONADD	FULL	(M_D) USED AS INSTRUCTION
32	INTERVAL TIMER CONTROL/ HALT	4-245	INT	ADD	HALF	LOAD AND CONTROL INT TIMER
33	STORE INTERVAL TIMER	4-247	INT	NONADD	HALF	(INT TIMER) → M_D
52	RETURN STACK TO P	4-248	INT	NONADD	HALF	$M_{(SP[31])} \rightarrow$ { P-SOURCE P COUNTER $M_{(SP[31])-1}$ { TRAP IND MODE IND HALT IND DATA KERNEL REG PROC KERNEL REG SP [31] & SPA (Note A)
53	RETURN STACK TO P AND PROCEED	4-250	INT	NONADD	HALF	$M_{(SP[31])} \rightarrow$ { P-SOURCE P COUNTER $M_{(SP[31])-1}$ (Note A) { TRAP IND MODE IND DATA KERNEL REG PROC KERNEL REG SP [31] & SPA O → HALT INDICATOR
81	RESET BIT N	4-252	INT	NONADD	HALF	$O \rightarrow M_D$ (BIT N) N = 0, 1, ..., 15
82	SET BIT N	4-254	INT	NONADD	HALF	$1 \rightarrow M_D$ (BIT N) N = 0, 1, ..., 15

Note A: If mode indicator presently O, it is left as O.

OP CODE	INSTRUCTION NAME	PAGE REF	INST TYPE (NOTE 1)	OPERAND FETCH (NOTE 2)	OPERAND SIZE	FUNCTION
PMU CONTROL INSTRUCTIONS (Cont.)						
29	SET TASK PARAMETERS	4-256	EXT	NONADD		SETS: UPPER BOUND REG LOWER BOUND REG REP ALGO REG DATA ADDRESS MODE
25	SET SYSTEM PARAMETERS	4-258	EXT	NONADD		SETS: MEMORY PARITY INHIBIT CHANNEL PARITY INHIBIT INITIALIZES CHANNEL ASSIGNS RESOURCE NAMES TO CHANNEL
28	INITIATE NEW TASK	4-263	INT	NONADD		INITIALIZE PMU FOR NEW TASK
36	TEST AND RESET	4-266	INT	NONADD	HALF	$(M_D)_{16-31} \rightarrow R$, $O \rightarrow M_D$
47	COMMAND SUBSYSTEM/ ADDRESS MODIFICATION	4-268	INT	NONADD	FULL	INSTRUCTION → E
4F	COMMAND SUBSYSTEM/ IMMEDIATE EXECUTION	4-272	INT	NONADD	FULL	INSTRUCTION → E
PMU INPUT/OUTPUT INSTRUCTIONS						
00	TEST AND RESET TO OUTPUT	4-276	EXT	NONADD	FULL	$(M_D) \rightarrow E$, $O \rightarrow M_D$
02	READ WORD TO OUTPUT	4-277	EXT	NONADD	FULL	$(M_D) \rightarrow E$
03	WRITE WORD FROM INPUT	4-278	INT	NONADD	FULL	$E \rightarrow (M_D)$
60	SINGLE WORD I/O COMMAND	4-279	INT	NONADD	FULL	$(M_D) \rightarrow E$ (M_D) IS SINGLE WORD INST
73	TWO WORD I/O COMMAND	4-281	INT	NONADD	FULL	$(M_D)_1 \rightarrow E$ $(M_D)_1$ IS INSTRUCTION $(M_D)_2 \rightarrow E$ $(M_D)_2$ IS DATA
78	TWO WORD I/O WITH INDEXING	4-284	INT	NONADD	FULL	$(M_D)_i \rightarrow E$ $(M_D)_1$ IS INSTRUCTION $(M_D)_2 \rightarrow E$ $(M_D)_2$ IS DATA INDEXING ALLOWED IN FETCHING $(M_D)_2$
04	READ OPERAND TO OUTPUT	4-287	EXT	NONADD	FULL	$(M_D) \rightarrow E$
06	READ PAGE TO OUTPUT	4-289	EXT	NONADD	FULL	$(M_D)_{\text{PAGE}} \rightarrow E$
0C	READ ARRAY TO OUTPUT	4-290	EXT	NONADD	FULL	$(M_D)_{\text{ARRAY}} \rightarrow E$
0E	READ INDIRECT WORD TO OUTPUT	4-291	EXT	NONADD	FULL	$(M_D) \rightarrow E$ $(M_{D+1}) \rightarrow E$
05	WRITE OPERAND FROM INPUT	4-292	EXT	NONADD	FULL	$E \rightarrow M_D$
07	WRITE PAGE FROM INPUT	4-294	EXT	NONADD	FULL	$E_{\text{PAGE}} \rightarrow M_D$
0D	WRITE ARRAY FROM INPUT	4-296	INT	NONADD	FULL	$E_{\text{ARRAY}} \rightarrow M_D$

APPENDIX H

AP INSTRUCTION ATTRIBUTES

The following chart summarizes some of the attributes of the AP instruction set. This chart is intended to be used only as a summary reference after gaining a thorough understanding of the instructions from their individual descriptions in Section 5. The symbolism used in the "Function" column is defined in Appendix A - Glossary. The following notes apply, where indicated in the chart.

NOTE 1 - "Operand Fetch" Column

This column indicates whether a given instruction is normally addressable (ADD) or normally non addressable (NON-ADD). ADD instructions may or may not have a one in bit 32 of the modified instruction word (PMU format) indicating a memory reference or an immediate operand, respectively. NON-ADD instructions either ignore the contents of bit 32 or, if bit 32 is one, expends a non required operand fetch from memory.

OP CODE	INSTRUCTION NAME	PAGE REF		OPERAND FETCH (NOTE 1)	FUNCTION TYPE	FUNCTION
AP ARITHMETIC INSTRUCTIONS						
C1	ADDITION	5-16		ADD	DYADIC	$(M) + (A) \rightarrow A$
C4	REVERSE SUBTRACT	5-19		ADD	DYADIC	$(M) - (A) \rightarrow A$
C2	SUBTRACT	5-21		ADD	DYADIC	$(A) - (M) \rightarrow A$
EO	MULTIPLY	5-23		ADD	DYADIC	$(M) \times (A) \rightarrow A$
D8	REVERSE DIVIDE	5-25		ADD	DYADIC	$(M) \div (A) \rightarrow A$
DO	DIVIDE	5-27		ADD	DYADIC	$(A) \div (M) \rightarrow A$
D4	DIVIDE RESIDUE	5-29		ADD	DYADIC	$(A) \div (M);\ Q_R \rightarrow A$
DC	REVERSE DIVIDE RESIDUE	5-31		ADD	DYADIC	$(M) \div (A);\ Q_R \rightarrow A$
D2	DIVIDE SHORT	5-33		ADD	DYADIC	$(A) \div (M);\ Q_I \rightarrow A$
DA	REVERSE DIVIDE SHORT	5-35		ADD	DYADIC	$(M) \div (A);\ Q_I \rightarrow A$
C5	LOAD ACCUMULATOR	5-37		ADD	MONADIC	$(M_D) \rightarrow M$, THEN $(M) \rightarrow A$
CA	LOAD NEGATIVE	5-38		ADD	MONADIC	$(M_D) \rightarrow M$, THEN $-(M) \rightarrow A$
CC	NEGATION	5-39		NONADD	MONADIC	$-(A) \rightarrow A$
CF	ABSOLUTE VALUE	5-40		NONADD	MONADIC	$\|(A)\| \rightarrow A$

BO	SIGNUM	5-41		NONADD	MONADIC	IF $(A) < O$; $-1 \rightarrow A$
						IF $(A) = O$; $O \rightarrow A$
						IF $(A) > O$; $1 \rightarrow A$

E4	FLOOR	5-42		NONADD	MONADIC	IF $A_I - 1 < (A) < A_I$
						THEN $A_I - 1 \rightarrow A$
						IF $(A) = (A_I)$; $A_I \rightarrow A$

E5	CEILING	5-43		NONADD	MONADIC	IF $A_I + 1 > (A) > A_I$
						THEN $A_I + 1 \rightarrow A$
						IF $(A) = (A_I)$; $A_I \rightarrow A$

E2	SQUARE ROOT	5-44		NONADD	MONADIC	$\sqrt{\|(A)\|} \rightarrow A$
F4	NORMALIZE	5-45		NONADD	MONADIC	(A) NORMALIZED → A
AP LOAD/STORE INSTRUCTIONS						
E8	STORE AND HALT	5-46		NONADD	DYADIC	1 → STORE HALT $(A) \rightarrow M_D$ O → STORE HALT
EA	STORE AND PROCEED	5-47		NONADD	DYADIC	$(A) \rightarrow M_D$
31	LOAD MEMORY WORD	5-48		NONADD	MONADIC	$(M_D) \rightarrow A$
32	STORE PACKED	5-49		NONADD	DYADIC	$(A) \rightarrow M_D$
33	LOAD DEFERRAL	5-50		NONADD	MONADIC	$(M_D) \rightarrow$ DEFERRAL STACK
34	STORE DEFERRAL	5-51		NONADD	MONADIC	(DEFERRAL STACK) $\rightarrow M_D$
B8	UNPACK (NOTE A)	5-52		NONADD	MONADIC	UNPACK ACCUMULATOR
B9	LOAD WORD TO ACCUMULATOR	5-54		ADD	MONADIC	$(M_D) \rightarrow A$
35	PUSH DATA	5-55		ADD	MONADIC	$(A) \rightarrow M_D$
36	STORE OPERAND	5-56		NONADD	DYADIC	$(A) \rightarrow M_D$
Note A: Issued by Array Controller when packed data is interpreted. Should not be used in a software program.						

OP CODE	INSTRUCTION NAME	PAGE REF		OPERAND FETCH (NOTE 1)	FUNCTION TYPE	FUNCTION
		AP COMPARE AND TVD INSTRUCTIONS				

92	COMPARE LESS THAN DESTRUCTIVE	5-58		ADD	DYADIC	IF (M) < (A); 1 → A, 1 → TVD
						IF (M) ≥ (A); O → A, O → TVD

91	COMPARE EQUAL NON-DESTRUCTIVE	5-59		ADD	DYADIC	IF (M) = (A); 1 → TVD
						IF (M) ≠ (A); O → TVD

93	COMPARE LESS THAN OR EQUAL NONDESTRUCTIVE	5-60		ADD	DYADIC	IF (M) ≤ (A); 1 → TVD
						IF (M) > (A); O → TVD

94	COMPARE GREATER THAN DESTRUCTIVE	5-61		ADD	DYADIC	IF (M) > (A); 1 → A, 1 → TVD
						IF (M) ≤ (A); O → A, O → TVD

95	COMPARE GREATER THAN OR EQUAL NONDESTRUCTIVE	5-62		ADD	DYADIC	IF (M) ≥ (A); 1 → TVD
						IF (M) < (A); O → TVD

96	COMPARE NOT EQUAL DESTRUCTIVE	5-63		ADD	DYADIC	IF (M) ≠ (A); 1 → A, 1 → TVD
						IF (M) = (A); O → A, O → TVD

97	SET TVD NONDESTRUCTIVE	5-64		NONADD	MONADIC	1 → TVD
98	RESET TVD NONDESTRUCTIVE	5-65		NONADD	MONADIC	O → TVD

99	COMPARE EQUAL DESTRUCTIVE	5-66		ADD	DYADIC	IF (M) = (A); 1 → A, 1 → TVD
						IF (M) ≠ (A); O → A, O → TVD

9A	COMPARE LESS THAN NON-DESTRUCTIVE	5-67		ADD	DYADIC	IF (M) < (A); 1 → TVD
						IF (M) ≥ (A); O → TVD

9B	COMPARE LESS THAN OR EQUAL DESTRUCTIVE	5-68		ADD	DYADIC	IF (M) ≤ (A); 1 → A, 1 → TVD
						IF (M) > (A); O → A, O → TVD

9C	COMPARE GREATER THAN NONDESTRUCTIVE	5-69		ADD	DYADIC	IF (M) > (A); 1 → TVD
						IF (M) ≤ (A); O → TVD

9D	COMPARE GREATER THAN OR EQUAL DESTRUCTIVE	5-70		ADD	DYADIC	IF (M) ≥ (A); 1 → A, 1 → TVD
						IF (M) < (A); O → A, O → TVD

9E	COMPARE NOT EQUAL NON-DESTRUCTIVE	5-71		ADD	DYADIC	IF (M) ≠ (A); 1 → TVD
						IF (M) = (A); O → TVD

9F	SET TVD DESTRUCTIVE	5-72		NONADD	MONADIC	1 → TVD, 1 → A
90	RESET TVD DESTRUCTIVE	5-73		NONADD	MONADIC	O → TVD, O → A
B2	MINIMUM	5-74		ADD	DYADIC	IF (A) > (M); (M) → A
B4	MAXIMUM	5-75		ADD	DYADIC	IF (M) > (A); (M) → A

OP CODE	INSTRUCTION NAME	PAGE REF		OPERAND FETCH (NOTE 1)	FUNCTION TYPE	FUNCTION
	AP TRANSFER INSTRUCTIONS					
AO or AB	NO TRANSFER	5-76		NONADD	MONADIC	NO OPERATION

A1
or
A9
	TRANSFER ON EQUAL TO ZERO	5-77		NONADD	MONADIC	IF $(A) = O$; $(M) \rightarrow P$
						IF $(A) \neq O$; $(P) + 1 \rightarrow P$

A2
or
AA
	TRANSFER ON GREATER THAN ZERO	5-78		NONADD	MONADIC	IF $(A) > O$; $(M) \rightarrow P$
						IF $(A) \leq O$; $(P) + 1 \rightarrow P$

A3
or
AB
	TRANSFER ON GREATER THAN OR EQUAL TO ZERO	5-79		NONADD	MONADIC	IF $(A) \geq O$; $(M) \rightarrow P$
						IF $(A) < O$; $(P) + 1 \rightarrow P$

A4
or
AC
	TRANSFER ON LESS THAN ZERO	5-80		NONADD	MONADIC	IF $(A) < O$; $(M) \rightarrow P$
						IF $(A) \geq O$; $(P) + 1 \rightarrow P$

A5
or
AD
	TRANSFER ON LESS THAN OR EQUAL TO ZERO	5-81		NONADD	MONADIC	IF $(A) \leq O$; $(M) \rightarrow P$
						IF $(A) > O$; $(P) + 1 \rightarrow P$

A6
or
AE
	TRANSFER ON NOT EQUAL TO ZERO	5-82		NONADD	MONADIC	IF $(A) \neq O$; $(M) \rightarrow P$
						IF $(A) = O$; $(P) + 1 \rightarrow P$

A7 or AF	UNCONDITIONAL TRANSFER	5-83		NONADD	MONADIC	$(M) \rightarrow P$

BC	TRANSFER ON TEST VALID SET	5-84		NONADD	MONADIC	IF $(TVD) = 1$; $(M) \rightarrow P$
						IF $(TVD) = O$; $(P) + 1 \rightarrow P$

27	EXECUTE	5-85		NONADD	MONADIC	(M_D) USED AS INSTRUCTION
	AP SHIFT INSTRUCTIONS					

FO	SHIFT OPEN	5-86		ADD	DYADIC	IF (M) +; $(A) \times 2^{4 \lvert (M) \rvert} \rightarrow A$
						IF (M) -; $(A) \times 2^{-4 \lvert (M) \rvert} \rightarrow A$

F1	SHIFT CYCLIC	5-87		ADD	DYADIC	IF (M) +; $(A) \times 2^{4 \lvert (M) \rvert} \rightarrow A$
						IF (M) -; $(A) \times 2^{-4 \lvert (M) \rvert} \rightarrow A$

F8	SHIFT SINGLE OPEN	5-88		ADD	DYADIC	$A \times 2^{M_{30-31}} \rightarrow A$
F9	SHIFT SINGLE CLOSED	5-89		ADD	DYADIC	$A \times 2^{M_{30-31}} \rightarrow A$
	AP BOOLEAN AND LOGICAL INSTRUCTIONS					
70	BOOLEAN ZERO	5-90		NONADD	MONADIC	$O \rightarrow A$; $O \rightarrow TVD$
71	BOOLEAN AND	5-91		ADD	DYADIC	$(A)_{31} \cdot (M)_{31} \rightarrow A_{31} \rightarrow TVD$

72	BOOLEAN LESS THAN	5-92		ADD	DYADIC	IF $(M)_{31} = O$ AND $(A)_{31} = 1$
						THEN $O \rightarrow A$, $1\text{-}A_{31}$, $1 \rightarrow TVD$
						IF $(M)_{31} = 1$
						THEN $O \rightarrow A$, $O \rightarrow TVD$

73	BOOLEAN ODD EVEN	5-93		NONADD	MONADIC	IF $(A)_{31} = 1$
						THEN $O \rightarrow A$, $1 \rightarrow A_{31}$, $1 \rightarrow TVD$
						IF $(A)_{31} = O$
						THEN $O \rightarrow A$, $O \rightarrow TVD$

74	BOOLEAN GREATER THAN	5-94		ADD	DYADIC	IF $(M)_{31} > (A)_{31}$
						THEN $O \rightarrow A$, $1 \rightarrow A_{31}$, $1 \rightarrow TVD$
						IF $(M)_{31} = O$
						THEN $O \rightarrow A$, $O \rightarrow TVD$

OP CODE	INSTRUCTION NAME	PAGE REF		OPERAND FETCH (NOTE 1)	FUNCTION TYPE	FUNCTION
	•AP BOOLEAN AND LOGICAL INSTRUCTIONS (Contd)					

75	BOOLEAN LOAD	5-95		ADD	MONADIC	IF $(M)_{31} = 1$
						THEN O → A, 1 → A_{31}, 1 → TVD
						IF $(M)_{31} = O$
						THEN O → A, O → TVD

76	BOOLEAN NOT EQUAL	5-96		ADD	DYADIC	IF $(M)_{31} \neq (A)_{31}$
						THEN O → A, 1 → A_{31}, 1 → TVD
						IF $(M)_{31} = (A)_{31}$
						THEN O → A, O → TVD

77	BOOLEAN INCLUSIVE OR	5-97		ADD	DYADIC	IF $(M)_{31}$ OR $(A)_{31} = 1$
						THEN O → A, 1 → A_{31}, 1 → TVD
						IF $(M)_{31}$ AND $(A)_{31} = O$
						THEN O → A, O → TVD

78	BOOLEAN NOR	5-98		ADD	DYADIC	IF $(M)_{31}$ AND $(A)_{31} = O$
						THEN O → A, 1 → A_{31}, 1 → TVD
						IF $(M)_{31}$ OR $(A)_{31} = 1$
						THEN O → A, O → TVD

79	BOOLEAN EQUALS	5-99		ADD	DYADIC	IF $(M)_{31} = (A)_{31}$
						THEN O → A, 1 → A_{31}, 1 → TVD
						IF $(M)_{31} \neq (A)_{31}$
						THEN O → A, O → TVD

7A	BOOLEAN LOAD COMPLEMENT	5-100		ADD	MONADIC	IF $(M)_{31} = O$
						THEN O → A, 1 → A_{31}, 1 → TVD
						IF $(M)_{31} = 1$
						THEN O → A, O → TVD

7B	BOOLEAN LESS THAN OR EQUAL TO	5-101		ADD	DYADIC	IF $(M)_{31} = O$ OR $(A)_{31} = 1$
						THEN O → A, 1 → A_{31}, 1 → TVD
						IF $(M)_{31} \neq O$ OR $(A)_{31} \neq 1$
						THEN O → A, O → TVD

7C	BOOLEAN NOT	5-102		NONADD	MONADIC	IF $(A)_{31} = O$
						THEN O → A, 1 → A_{31}, 1 → TVD
						IF $(A)_{31} = 1$
						THEN O → A, O → TVD

7D	BOOLEAN GREATER OR EQUAL	5-103		ADD	DYADIC	IF $(M)_{31} = 1$ OR $(A)_{31} = O$
						THEN O → A, 1 → A_{31}, 1 → TVD
						IF $(M)_{31} = O$ OR $(A)_{31} = 1$
						THEN O → A, O → TVD

7E	BOOLEAN NAND	5-104		ADD	DYADIC	IF $(M)_{31} = O$ OR $(A)_{31} = O$
						THEN O → A, 1 → A_{31}, 1 → TVD
						IF $(M)_{31} = 1$ AND $(A)_{31} = 1$
						THEN O → A, O → TVD

7F	BOOLEAN ONE	5-105		NONADD	MONADIC	O → A, 1 → A_{31}, 1 → TVD

OP CODE	INSTRUCTION NAME	PAGE REF		OPERAND FETCH (NOTE 1)	FUNCTION TYPE	FUNCTION
AP BOOLEAN AND LOGICAL INSTRUCTIONS (Contd)						
80	LOGICAL ZERO	5-106		NONADD	MONADIC	$0 \rightarrow A_{0-31}$
81	LOGICAL AND	5-107		ADD	DYADIC	$(M)_{0-31} \cdot (A)_{0-31} \rightarrow A_{0-31}$
82	LOGICAL LESS THAN	5-108		ADD	DYADIC	$(A)_{0-31} > (M)_{0-31} \rightarrow A_{0-31}$
83	NO OPERATION	5-109		NONADD	---	NO OP
84	LOGICAL GREATER THAN	5-110		ADD	DYADIC	$(M)_{0-31} > (A)_{0-31} \rightarrow A_{0-31}$
85	LOGICAL LOAD	5-111		ADD	DYADIC	$(M)_{0-31} \rightarrow (A)_{0-31}$
86	LOGICAL NOT EQUAL	5-112		ADD	DYADIC	$(M)_{0-31} \neq (A)_{0-31}; 1 \rightarrow A_{0-31}$ $(M)_{0-31} = (A)_{0-31}; 0 \rightarrow A_{0-31}$

87	LOGICAL INCLUSIVE OR	5-113		ADD	DYADIC	IF $(M)_{0-31} = 1$ OR $(A)_{0-31} = 1$
						THEN $1 \rightarrow (A)_{0-31}$
						IF $(M)_{0-31} = 0$ AND $(A)_{0-31} = 0$
						THEN $0 \rightarrow A_{0-31}$

89	LOGICAL EQUALS	5-114		ADD	DYADIC	IF $(M)_{0-31} = (A)_{0-31}$
						THEN $1 \rightarrow A_{0-31}$
						IF $(M)_{0-31} \neq (A)_{0-31}$
						THEN $0 \rightarrow A_{0-31}$

88	LOGICAL NOR	5-115		ADD	DYADIC	IF $(M)_{0-31}$ AND $(A)_{0-31} = 0$
						THEN $1 \rightarrow A_{0-31}$
						IF $(M)_{0-31}$ OR $(A)_{0-31} \neq 0$
						THEN $0 \rightarrow A_{0-31}$

8A	LOAD COMPLEMENT	5-116		ADD	DYADIC	$(\overline{M}) \rightarrow A$

8B	LOGICAL LESS THAN OR EQUAL	5-117		ADD	DYADIC	IF $(M)_{0-31} = 0$ OR $(A)_{0-31} = 1$
						THEN $1 \rightarrow A_{0-31}$
						IF $(M)_{0-31} = 1$ AND $(A)_{0-31} = 0$
						THEN $0 \rightarrow A_{0-31}$

8C	LOGICAL NOT	5-118		NONADD	MONADIC	$(\overline{A}) \rightarrow A$

8D	LOGICAL GREATER OR EQUALS	5-119		ADD	DYADIC	IF $(M)_{0-31} = 1$ OR $(A)_{0-31} = 0$
						THEN $1 \rightarrow A_{0-31}$
						IF $(M)_{0-31} = 0$ AND $(A)_{0-31} = 1$
						THEN $0 \rightarrow A_{0-31}$

8E	LOGICAL NAND	5-120		ADD	DYADIC	IF $(M)_{0-31} = 0$ OR $(A)_{0-31} = 0$
						THEN $1 \rightarrow A_{0-31}$
						IF $(M)_{0-31} = 1$ AND $(A)_{0-31} = 1$
						THEN $0 \rightarrow A_{0-31}$

8F	LOGICAL SET	5-121		NONADD	MONADIC	$1 \rightarrow A_{0-31}$

OP CODE	INSTRUCTION NAME	PAGE REF		OPERAND FETCH (NOTE 1)	FUNCTION TYPE	FUNCTION
AP ARRAY INSTRUCTIONS						
37	LOAD OP CODE	5-133		NONADD	MONADIC	OP_1 AND OP_2 LOADED TO AP
61	POLYNOMIAL	5-135		ADD	DYADIC	POWER SERIES EXPANSION OF A AND M USING OP_1 and OP_2
51	DIMENSION	5-137		NONADD	MONADIC	p(A) → A
60	INDEX GENERATOR	5-138		NONADD	MONADIC	O, 1, 2, ..., $(A)_{24-31}$ -1 (Note A)
50	RAVEL	5-139		NONADD	MONADIC	, (A) → A
62	OUTER PRODUCT REDUCTION	5-140		ADD	DYADIC	OP, /(M)∘ . OP_2, (A) → A
63	EXPAND ALONG COLUMN	5-141		ADD	DYADIC	(M)⍀(A) → A
64	CATENATE ROWS	5-143		ADD	DYADIC	(A), (M) → A
65	TRANSPOSE	5-145		NONADD	MONADIC	φ (A) → A
66	REVERSAL ALONG ROWS	5-146		NONADD	MONADIC	φ(A) → A
67	LAMINATE ROWS	5-147		ADD	DYADIC	(A), [-. 5] (M) → A (Note A)
68	ROTATE ROW	5-149		ADD	DYADIC	(M)ϕ (A) → A
69	RESHAPE	5-151		ADD	DYADIC	(M) ρ (A) → A
6A	TAKE	5-152		ADD	DYADIC	(M) ↑ (A) → A
6B	DROP	5-154		ADD	DYADIC	(M) ↓ (A) → A
6C	INNER PRODUCT	5-155		ADD	DYADIC	(M) OP_1 . OP_2 (A) → A
6D	OUTER PRODUCT	5-158		ADD	DYADIC	M∘ . OP (A) → A
6E	REDUCTION ALONG ROW	5-160		NONADD	MONADIC	OP⌿(A) → A
6F	COMPRESSION ALONG COLUMNS	5-162		ADD	DYADIC	(M)/(A) → A
Note A: Origin O						

BLANK

APPENDIX I

CHANNEL INTERRUPT CONDITIONS

The channel transmits an interrrupt to the executive if one of the following actions occur. (Reference Section 4.2.2.1.2 Channel Interrupt).

Status bus error
Send error
Receive error
PMU alert
Channel reset

The environment which causes these actions is as follows:

Status Bus Error - The detection of an illegal code on the indicated bus in the status bus lines of the primary bus.

Send Error - A second error status return is detected after the transmission of a word that has been retried. The word was retried due to the detection of an error status return when the word was initially transmitted.

Receive Error - A parity error is detected on the indicated bus on a word that is being re-transmitted on the primary bus (retry bit is on).

NOTE: The channel transmitting the word sends a send error interrupt to the executive. The channel receiving the word sends a receive error interrupt to the executive.

PMU Alert - The PMU alert signal is the computer failure signal in a DPE configuration. See Section 6.23, Computer Failure Signal, for a further description.

Channel Reset - Channel reset occurs when the timeout counter runs to completion (4096 IB clock periods). The timeout counter when enabled starts counting from 0 every time a new word is placed on the input secondary bus and is reset when the word is acknowledged by the receiving element. The enabling condition for the timeout counter is either: (1) an emergency instruction placed on the secondary bus (ref. 4.2.2.2.1); (2) a data word or non-emergency instruction placed on the secondary bus and the timeout enable allowed by the Set System Parameter instruction.