REFERENCE MANUAL HARRIS 800 GENERAL PURPOSE DIGITAL COMPUTER

Original Issue August, 1979



Copyright © 1978 by Harris Corporation, Computer Systems Division. All rights reserved. This publication or any part thereof is intended for use with Harris products by Harris personnel, customers, and end-users, and shall be used only for installation, operation, and maintenance of Harris products. It may not be reproduced in any form without the written permission of the publisher.

The information contained in this document is believed to be correct at the time of publication. It is subject to change without notice. Harris makes no warranties, express or implied, concerning the information contained in this document.

Printed in U.S.A.

LIST OF EFFECTIVE PAGES

TOTAL NUMBER OF PAGES IN THIS PUBLICATION IS: 178 CONSISTING OF THE FOLLOWING:

Page	Change	Page	Change	Page	Change
No.	No.	No.	No.	No.	No.
Title A i thru vi 1-1 thru 1-13 1-14 Blank 2-1 thru 2-20 3-1 thru 3-5 3-6 Blank 4-1 thru 4-21 4-22 Blank 5-1 thru 5-8 6-1 thru 6-3 6-4 Blank 7-1 thru 7-17 7-18 Blank 7-19 thru 7-87 7-88 Blank A-1 thru A-7 A-8 Blank	Original				

Insert Latest Revision Pages. Destroy Superseded Pages.

CONTENTS

Section		Page
1	INTRODUCTION	
	SCOPE OF MANUAL	. 1-1
	HARRIS 800 SYSTEMS	
	BASIC COMPUTER ORGANIZATION	. 1-2
	Basic Operation	
	Central Processing Unit (CPU)	
	Scientific Arithmetic Unit (SAU) Functions	
	Memory Units	
	Input/Output Operation	
	Priority Interrupt System	
	MAINTENANCE AID PROCESSOR	. 1-5
	SWITCH PANEL	
	DATA TERMINALS	. 1-6
	Priority Interrupts	
	120 Hertz Clock	
	Interval Timer	. 1-7
	100 kHz Real Time Clock	. 1-7
	Power Fail Alarm	
	Bootstraps	
	Bit Processor	
	Stall Alarm	
	Address Trap	
	Input/Output and Communications Processor Channels	
	Programmed Input Output Channel (PIOC)	
	Buffered Block Channel (BBC)	
	Direct Memory Access Communications Processor (DMACP-8)	
	External Block Channel (XBC)	
	Integral Block Channel (IBC)	
	Input/Output Expansion Unit	. 1-9
	Interprocessor Communication Facility	
	Multi-Channel Adapter	. 1-10
	PERIPHERAL EQUIPMENT	
	SOFTWARE	. 1-10
	Language Processors	
	Utility Programs	. 1-10
	Remote Job Entry Support Packages	
	Remote Batch Terminal Host Packages	
	Data Base Management System	. 1-10
	SUMMARY OF CHARACTERISTICS	. 1-11
	Memory System	. 1-11
	Addressing	
	Input/Output Capability	
	Priority Interrupt Structure	. 1-13
	Power Fail Protection	. 1-13
	Electrical Requirements	. 1-13
	Environmental Requirements	. 1-13

Section		Page
11	CENTRAL PROCESSING UNIT	
•	GENERAL DESCRIPTION	. 2-1
	PRINCIPAL CPU REGISTERS	
	A and B Registers	
	E Registers	
	D Registers	
	I, J, and K Registers	
	Condition Code Registers	
	C Register	
	Y Register	
	Program Address Register	
	Instruction Buffer	
	VIRTUAL MEMORY DESCRIPTION	
	Introduction	
	Virtual Memory Instruction Set	
	Principal Virtual Memory Registers	
	Virtual Address Register (VARs)	
	Virtual Base Register (VBR)	
	Virtual Limit Register (VLR)	. 2-4
	Virtual Usage Registers (VURs)	. 2-4
	Virtual Not-Modified Registers (VNRs)	. 2-4
	Virtual Usage Base Register (VUB)	. 2-4
	Virtual Source Register (VSR)	
	Virtual Destination Register (VDR)	
	Virtual Demand Page Register (VPR)	
	Demand Paging	
	Instruction Trap	
	Paging System Control	
	CPU OPERATIONAL CONTROL	
	CPU Modes of Operation	
	Compatibility Mode	
	Address Extension Mode	. 2-7
	CPU Operational States	
	ADDRESSING FUNCTIONS	
	Compatibility Mode Addressing	. 2-8
	Direct Addressing	
	Indirect Addressing	
	Indexing	
	Address Extension Mode Addressing	. 2-10
	Direct Addressing	
	Indirect Addressing	. 2-13
	Indexing	
	Address Translation	
	120 HERTZ CLOCK	
	INTERVAL TIMER	
	General Description	
	Timer Register	2-16

Section		Page
11	CENTRAL PROCESSING UNIT (CONT.)	
	Operational Description	. 2-16
	Program Control	
	REAL TIME CLOCK	. 2-16
	General Description	
	Operational Description	
	Command and Status Word Formats	
	Program Control	
	Preset Count Loading	
	Automatic Count Restart	
	Snapshot Output	
	Selection Sampling	
	FIRMWARE BOOTSTRAPS	
	BIT PROCESSOR	
	General Description	
	Bit Processor Registers	
	Operational Description	. 2-18
	Program Control	. 2-18
	Bit Processor Instruction Set	. 2-19
	STALL ALARM	
	ADDRESS TRAP	
	General Description	. 2-20
	Query Register	. 2-20
	Operational Description	. 2-20
	Program Control	. 2-20
Ш	MEMORY SYSTEM	
	GENERAL DESCRIPTION	2.1
	MEMORY MODULES	. J-1 2.1
	64K MOS Memory Module	. 3-1
	64K MOS Memory Module	. 3-1
	Read and Write Operations	. J·I
	Fast Access Operation	. 3-2
	MAIN MEMORY	. 3-2
	EXTENDED MEMORY UNIT.	. 3-2
	SHARED MEMORY UNIT	. 3-2
	General Description	. 3-2
	Programming Considerations	. 3-2
	Semaphore Operation	. 3-2
	ERROR CORRECTING AND REPORTING	. 3-3
	Error Correction	. 3-3
	Error Reporting	. 3-3
	Parity Errors and Interrupts	3-3
	Parity Error Address Register	3-3
	CACHE MEMORY	3-4
	Operational Description	3-4
	Algorithm for Filling Cache	3-4
	Programming Considerations	3-4
		J- 4

Section		Page
IV	INPUT/OUTPUT CHANNELS	
	GENERAL DESCRIPTION	. 4-1
	BASIC I/O CONCEPTS	
	Addressing	
	Disconnect/Connect Sequences	
	Block I/O Channel Priority	
	Synchronization (Handshake) Conditions	
	Output Transfer Synchronization	. 4-4
	Input Transfer Synchronization	4-4
	PIOC Synchronization	. 4-4
	XBC Synchronization	. 4-4
		. 4-5
		. 4-5
	BBC Synchronization	. 4-5
	Timing	
	Block Transfer Memory Access	. 4-5
	Block Transfer Parameters	. 4-5
	BBC Parameter Words	. 4-5
	XBC Parameter Words	. 4-7
	IBC Parameter Words	. 4-7
	DMACP-8 Parameter Words	. 4-7
	INPUT/OUTPUT INSTRUCTIONS	. 4-8
	I/O Commands	. 4-8
	I/O Status Word	. 4-9
	Programmed Data Transfers	. 4-11
	Input Data Word	. 4-11
	Output Data Word	. 4-11
	Address Transfers	. 4-11
	Output Address Word	. 4-11
	Input Address Word	. 4-13
	Input Parameter Word	. 4-13
	INTERRUPT CONTROL	
		. 4-14
	I/O CHANNEL JUMPER CONTROLS	
		4-14
	Single-Word Instruction Execution	4-14
	OCW/ODW	4-14
	IDW	
	ISW	. 4-14
	OAW	. 4-14
	IAW/IPW	. 4-15
	Block-Transfer Operations	. 4-15
	BBC Block Transfers	. 4-15
	XBC Block Transfers	. 4-15
	IBC Block Transfers	. 4-20
	DMACP-8 Channel Block Transfers	. 4-20
	Program Lists	. 4-20
	IBC Applications	. 4-20
	VPC Application	4-21

Section					Page
V	PRIORITY INTERRUPT SYSTEM				
	GENERAL DESCRIPTION				. 5-1
	INTERRUPT ORGANIZATION				
	Priority Conventions				
	Executive Traps (Group 0)				
	External Interrupts (Groups 1, 2, and 3)				
	Dedicated Memory Locations				
	OPERATION AND CONTROL				. D-I
	Executive Traps Control				. 5-3
	External Interrupts Control				
					. 5-5
	Operational State Zero Interrupt Processing				. 5-5
	Operational States One and Three Interrupt Processing			• •	. 5-6
VI	SCIENTIFIC ARITHMETIC UNIT OPERATION				
	GENERAL DESCRIPTION				. 6-1
	FLOATING-POINT DATA FORMAT	• •	• • •		. 6-1
	SAU REGISTERS				
	OPERATION AND CONTROL			• •	. 6-1
	Data Transfers			• •	. 6-1
	SAU Instructions			• •	. 0-1
	SAU INTERRUPT			• •	. 6-1
V/II		• •		• •	. 6-1
VII	INSTRUCTION SET				
	INTRODUCTION				. 7-1
	INSTRUCTION TYPES AND FORMATS				. , . . 7-1
	Introduction				
	Standard Instruction Format				. 7-1 . 7-1
	Extended Instruction Format				. /-1 . 7-1
	INSTRUCTION FORMULA				. 7-1
	INSTRUCTION DESCRIPTIONS	• •			. 7-3 . 7-4
	Arithmetic Instructions				. 7-4
					7-16
	Compare Instructions				. 7-25
					7-28
					7-30
	Transfer Instructions				7-33
	Byte Processing Instructions				7-43
	Input/Output Instructions				7-49
	Bit Processor Instructions				7-53
	Virtual Memory Instructions				7-57
	Priority Interrupt Control Instructions				7-60
	Miscellaneous Instructions				7-69
	Scientific Arithmetic Unit Instructions				7-72
	Decimal Arithmetic Instructions				7-83
	Diagnostic Instructions				
APPENDI	X A – INSTRUCTION INDEX				Α-1

ILLUSTRATIONS

igure		Page
1-1.	Major Functional Units	. 1-3
2-1. 2-2. 2-3. 2-4. 2-5. 2-6. 3-1. 4-1. 4-2. 4-3. 4-4. 4-5.	Data Formats	. 2-2 . 2-9 . 2-11 . 2-12 . 2-14 . 2-15 . 3-5 . 4-2 . 4-6 . 4-7 . 4-9 . 4-12
4-6. 4-7. 4-8. 4-9.	BBC Block Transfer Sequence; Simplified Flow Diagram	. 4-17 . 4-18
5-1. 5-2. 5-3. 5-4. 5-5. 5-6.	Functional Block Diagram, Priority Interrupt System External Interrupt Control Interrupt Subroutine Entry, Operational State Zero Interrupt Subroutine Exit, Operational State Zero Interrupt Subroutine Entry, Operational States One and Three Interrupt Subroutine Exit, Operational States One and Three	. 5-4 . 5-7 . 5-7 . 5-8
6-1. 6-2.	Floating-Point Data Formats	. 6-2 . 6-2
7-1. 7-2.	Typical Instruction Word Formats	. 7-2 . 7-17
	TABLES	
Table		Page
2-1.	VPR Status Bits Definitions and Functions	. 2-5
4-1. 4-2.	Peripheral Unit Interrupt Control	. 4-13 . 4-14
7-1.	Summary of Extended Instructions Derived from Standard Instructions	. 7-3

SECTION I

SCOPE OF MANUAL

This manual contains reference material for the Harris 800 Computer Systems designed and manufactured by Harris Corporation, Computer Systems Division. Included are descriptions of the overall computer organization, central processing unit (CPU), memory configurations, priority interrupt system, input/output (I/O) channels, and instruction set. Various hardware features and options are also described; application and programming examples are provided where appropriate.

The material in this manual is oriented toward the user/programmer with a knowledge of computer fundamentals and terminology.

HARRIS 800 SYSTEMS

This family is comprised of high-performance, disc-oriented, virtual memory computer systems for performing concurrent time-sharing, batch, remote job entry and real-time processing. The Harris 800 Computer Systems are building-block systems; each may be expanded to support a variety of applications and performance levels. Upgrades between systems are also available. Harris 800 systems provide cost-effective solutions for distributed data processing, transaction oriented processing, and communications applications. Data Base Management and Inquiry software is available for fast, efficient file maintenance and information retrieval. These multi-use systems are ideal for scientific, commercial, and real-time applications since they provide true multi-programming and multi-lingual capabilities.

The Harris 100, 500, and 800 systems comprise a series of compatible data processing systems. These systems include a family of central processors, peripheral devices and programming support systems. The processors share a common code structure and instruction formats. They differ primarily in memory capacity, computational speed, the number of instructions, and input-output throughput. Harris 800 systems are products based on the experience gained with 100 and 500 systems. Since an 800 is upward compatible from 100 or 500 systems, a move from 100 or 500 systems to 800 systems is relatively easy.

Harris 800 instructions, character codes, interrupt facilities, and programming features are functionally the same as corresponding features on 100 and 500 processors. Harris 800 systems provide the capability of running 100 and 500 operating systems, as well as operating systems designed specifically for the advanced 800 features, with a minimum impact on application programs and data. Users' programs which run on 100 and 500 systems will also run on 800 systems.

Harris 800 systems use the same standard instruction set as 100 and 500 systems. A decimal feature has been added which includes pack, unpack, decimal add, and decimal subtract instructions. The scientific instruction set executes floating-point operations on 48-bit operands which employ an 8-bit exponent and a 39-bit mantissa. Should greater precision be required, extended floating-point operations can be invoked. These operations employ a 24-bit exponent and 70-bit mantissa.

Another major aspect of the 800 is the capability to attach a wide variety of I/O devices through several types of block channels. Like the 100 and 500 systems, channels are provided for the attachment of large numbers of communication devices. Most of the 800 I/O channels and peripherals are upward compatible from 100 and 500 systems.

The increased performance of the 800 is achieved by increasing the width of data paths and incorporating a pipelining technique into the architecture. Data transfers between memory and available I/O channels proceed on a 48-bit basis. Most instruction fetches and many operand fetches are 48-bit transfers as well. Pipelining provides simultaneous instruction execution, address processing, and instruction and operand fetches. It provides prefetch of both operands and instructions. Several instructions are processed simultaneously. When conditional branch instructions are executed, instructions for both decision paths are prefetched and preprocessed so that minimal time is lost after the decision is made.

BASIC COMPUTER ORGANIZATION

Basic Operation

Figure 1-1 illustrates the functional relationship between major units of a typical system. The major functional units include the central processing unit (CPU) consisting of an Instruction Unit and an Execution Unit, main memory, cache memory, shared memory, extended memory, priority interrupt system, input/output (I/O) channels, Maintenance Aid Processor (MAP), and switch panel.

The computer has a variable word length, a multi-access bus structure, and an integral memory system. Operations are performed on, and from, 48-bit and 24-bit data and instruction words. In addition, the computer is capable of selective byte manipulation and performs Boolean functions on single, selected bits. Two's complement arithmetic is performed on parallel, binary, fixed-point or floating-point operands. Fixed-point capabilities include hardware multiply, divide, and square root functions, as and subtract 48-bit add well as Double-precision (48-bit) floating-point operations employ 8-bit exponent and 39-bit mantissa, quad-precision operations employ a 24-bit exponent and a 70-bit mantissa. Decimal addition and subtraction are additional arithmetic functions included. Decimal arithmetic is performed on data in packed format. In this format, two decimal digits are placed in one byte. Decimal pack and unpack capability is included in the instruction set.

Data or instruction words may be retrieved from or stored in memory, retained in one of the CPU registers, or received from and transmitted to peripheral devices via the I/O channels. Prior to execution, instructions must be loaded into, and subsequently retrieved from, physical memory. Main memory is accessed on a 48-bit boundary. This arrangement permits an instruction prefetch which reduces the effective access time of the memory system. In addition, the CPU employs an asynchronous cycle that automatically adjusts to the timing of the addressed memory module. If, for example, memory contention occurs, the CPU waits at a predetermined point until memory becomes available.

Memory may be accessed at the 48-bit, 24-bit, 8-bit (byte), and bit levels by the standard instruction set. Memory is divided into thirty-two, 96K byte sections (map 0 through map 31). If the system is in the Compatibility Mode, up to 96K bytes per section may be directly addressed and up to 768K bytes can be accessed

by indirect and indexed address references; executable code is restricted to 192K bytes at any given time. When in the Address Extension Mode, up to 3M bytes of memory may be accessed directly, and executable code may be located anywhere in memory and is not limited in size.

When virtual memory is enabled, two addressing modes are employed, User and Monitor. Addresses generated in the User Mode (called logical addresses) are translated into physical memory addresses by the virtual memory hardware. The logical address is translated to the physical address by selecting the appropriate 3K byte physical "page" and the offset within the page. The division of main memory into physical pages allows a program to be located in non-contiguous areas of memory, and to be transferred (in page increments) between memory and an external mass storage device under system control. When the virtual memory hardware detects a reference to a page which is not currently resident in main memory, a page fault occurs. This supports a demand-page technique which allows portions of a program to be absent from memory while the program is running. The occurrence of a page fault initiates a system process which transfers the referenced page to physical memory. The paging logic is disabled in the Monitor Mode, thus addresses generated in the Monitor Mode are used directly as physical main memory addresses.

Central Processing Unit (CPU)

Included in the CPU are several general and special-purpose registers, an arithmetic section, timing and control logic, memory interface logic, and I/O channel interface circuits. Special paging registers and control logic are provided for virtual memory operation.

Five general-purpose registers are included in a basic CPU. These registers are employed in a variety of logical, arithmetic, and manipulative operations such memory-to-register, and register-to-memory, the instructions. Three of register-to-register general-purpose registers can be used for indexing in memory addressing functions. One register serves as the register during input/output communication operations. A 48-bit register is formed by combining two general-purpose registers, and a byte register is created by using the eight least-significant bits of one general-purpose register. With the Interval Timer included in the CPU, the Timer (T) Register becomes a sixth general-purpose register in the Monitor Mode of operation. In the User Mode, the T Register can not be loaded but can be read.

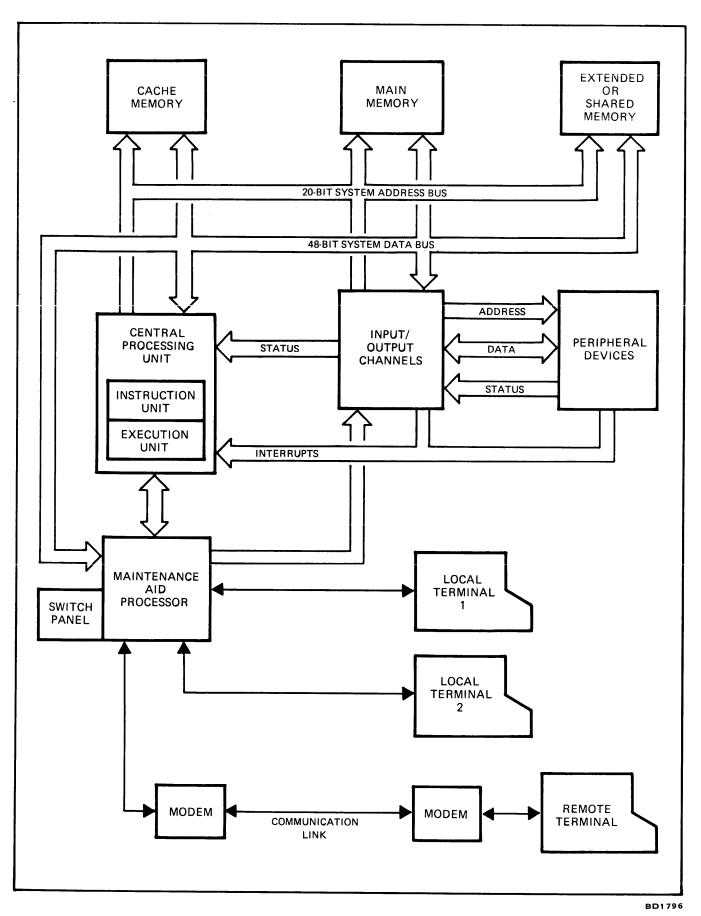


Figure 1-1. Major Functional Units

Among the special-purpose registers are those associated with integral CPU functions such as addressing, instruction decoding, and temporary storage during data manipulation. Additional special-purpose registers are those supplied with the Bit (Boolean) Processor, Interval Timer (T Register for timing applications), and the Address Trap.

The arithmetic section consists primarily of a 48-bit arithmetic logic unit (ALU) and several buses to permit data manipulation between the various registers and the ALU. Arithmetic functions performed include addition, subtraction, multiplication, division, and square root computation.

Instruction execution sequences are established and directed by the timing and control logic associated with includes This logic Instruction Unit. crystal-controlled clock generator that provides precise timing for all instruction functions. Instruction words of 24 or 48 bits are prefetched and retained in an instruction buffer. As many as four instructions may be prefetched and stored in the buffer. The control logic decodes these instruction words and provides the internal commands necessary for execution. In the User Mode of logic operates in operation, the paging control conjunction with the basic CPU timing to implement address translation and demand paging techniques.

CPU-memory interface circuits consist of address and and parity buses and registers. data-handling generation/checking or error checking and correction logic. Memory interface circuits include a 48-bit data register that retains both the read and write data, a 20-bit address register to define the location to be accessed in up to 3,072K bytes of physical memory, data multiplexing logic to control read and write data handling, and address multiplexing and control logic for selecting the proper memory segment and a location within that segment. Data to be written (stored) in memory is applied via the 48-bit system data bus. Address inputs are applied to the memory interface via the system address bus. The address source may be the CPU, one of the block transfer channels, communications processor, or, in the User Mode of operation, the paging logic addressing circuits.

Communications between the CPU and the I/O channels are conducted via the channel interface logic in the CPU. This logic makes use of the system buses and one of the general-purpose registers in order to implement data and

address flow between the CPU and I/O channels. channel-unit 1/0 channel conducts an Although and asynchronously, communications independently operations such as channel-unit input/output selection and activation, function commands, and status testing are initiated under program control.

Scientific Arithmetic Unit (SAU) Functions

The Execution Unit provides floating-point arithmetic capability. A special repertoire of instructions is provided for performing floating-point computations. The E Unit contains the X, XW and Y Registers for manipulating 48-bit quantities and for reporting arithmetic status (condition) after the operation is completed. Data and condition information may be displayed on the Map Terminal. An executive trap is provided with the E Unit for detection of overflow/underflow conditions. Refer to Section VI for a more detailed description of the E Unit execution of SAU instructions.

Memory Units

The memory system consists of main memory, extended memory, cache memory, and shared memory units. The Maintenance Aid Processor, I/O block channels, communication processors, and the CPU communicate directly with all memory modules. Each memory module contains the address decode logic necessary to determine when a particular module is selected. The CPU provides the required hand-shaking signals with the memory module to ensure proper data transfer.

Storage of information, both instruction and data words, is the function of main memory which may be located in the CPU or in the Extended Memory Unit. The basic memory module is a 192K byte MOS memory module which features single bit error correction. A system can be configured with up to 3,072K bytes of memory when the Extended Memory Unit is attached.

Cache memory provides fast access to data stored in the memory system. The 6K byte cache stores up to 1,024 memory word addresses and the information (instruction or data) contained therein. Data storage in cache is structured as two, 512 word sections, where each word is 48 bits wide. One section stores only instructions, and the other section stores only operands. When the CPU accesses the memory system, the address word is presented to the main memory and the cache. If the requested address and information is present in cache, the information is placed on the 48-bit data bus. If the cache

does not contain the requested address and information, the data is provided from main memory and the cache is updated to contain it. Cache memory effectiveness is significantly affected by program structure.

Shared memory is configured using the basic 192K byte (MOS) memory modules. Maximum memory available to a single CPU is 3,072K bytes which includes the combination of main memory and shared memory.

Refer to Section III for additional details concerning the memory system.

Input/Output Operation

Input/Output (I/O) operations consist of data, address, command, or status transfers between selected peripheral devices and the CPU or memory. Programmed and direct memory access (DMA) data transfers are supported. All such operations are initiated under program control and are conducted, asynchronously, by an I/O or communications processor channel. Various types of channel modules may be installed in a system. All channels in the system can be active simultaneously, and each channel may communicate with up to 16 controllers, however, only one device can transfer data at one time.

An I/O operation is initiated by selecting and activating a channel, and one of its assigned peripheral devices, through the execution of a computer input/output instruction. (The instruction set includes seven input/output instructions.) A specific I/O operation may involve preparing a peripheral device for a subsequent communication, determining the operational status of a device, or initiating a data transfer. Once activated, the channel provides complete functional control over the operation.

Data may be transferred on a single word basis (i.e., one data word per instruction) or automatically, in blocks of n words per operation. Block data transfers are performed by the Direct Memory Access Communication Processor (DMACP-8), External Block Channel (XBC), Integral Block Channel (IBC), or Buffered Block Channel (BBC). Each available type of I/O channel and communications processor permits data transfers to (input) and from (output) the computer. Data transfers between memory and the DMACP-8, XBC, or IBC are in a 24-bit parallel format, and between memory and the BBC in a 48-bit parallel format.

I/O operations may also be conducted on an interrupt basis through the use of interrupt logic in the channel(s). The channel interrupt system can be placed under program control and selectively enabled or disabled by an input/output instruction. Peripheral device functions may be connected directly to the computer priority interrupt system, bypassing the channel interrupt logic.

Priority Interrupt System

The interrupt system is a multi-level vectored structure that allows additional program control of input/output devices and internal CPU operations, and immediate recognition of special external conditions on the basis of priority. Receipt and recognition of an interrupt trigger permits normal program flow to be diverted to a subroutine that services the interrupt and returns the program to its normal sequence at the point where the interruption occurred.

MAINTENANCE AID PROCESSOR

The Maintenance Aid Processor (MAP), together with the terminal connected to the MAP, replace the conventional computer control panel. The MAP and terminal provide an intelligent interface between the operator and the computer.

Normal operator functions provided by the MAP Terminal include facilities for manually starting and halting operations, entering data into memory and various registers, and selecting memory and registers for display. System status and other important functions can be displayed via the MAP and its connected terminal. Master clear and initial program load (bootstrap) functions are also provided.

In addition to the normal operator functions, the MAP provides special maintenance functions. Special support hardware operates in conjunction with the MAP to provide selective monitoring and control of the computer logic. A maintenance bus provides access to all essential internal computer hardware not otherwise accessible. Maintenance functions include the capabilities to perform a limited master clear, step through instructions and microinstructions, control the CPU clock, and read/write non-programmable CPU registers. I/O channel read and write control, program and memory address compare breakpoint control, and power supply voltage monitoring are additional functions performed by the MAP.

SWITCH PANEL

Four switches are located on the Switch Panel; one keylock switch and three toggle switches. The keylock switch is used to lock and disable any of the maintenance features that may affect computer operations. It also enables or disables the Stall Alarm, and enables or disables diagnostic testing of the 192K byte MOS memory modules. Additionally, the keylock switch enables selection of a local or remote terminal for connection to the MAP.

The toggle switches provide for alternating the functions of the local terminals, connecting the MAP to a selected local terminal, and enabling the 192K byte MOS memory modules for a diagnostic mode of operation. Memory module interleaving, error correction and rewrite, and fast access functions are disabled when the diagnostic mode is selected.

DATA TERMINALS

Two local data terminals are provided with each system. A similar data terminal is available for remote diagnostic operations. All terminals are of the console type, consisting of a CRT and keyboard.

One of the local terminals is dedicated to operator communications (OPCOM). An operator using OPCOM commands at this terminal can display information about the system and exert control over the program, I/O, and user configurations. The second local terminal is used as a MAP Terminal. A switch on the Switch Panel allows for interchanging the assigned functions of the local terminals, i.e., either terminal can be designated as the OPCOM Terminal, while the second terminal is designated as the MAP Terminal. When the system is powered up, the MAP Terminal is automatically connected to the MAP. A command issued to the MAP from the MAP Terminal disconnects the MAP and connects the terminal to the CPU as a User Terminal. The MAP Terminal can be switched from the user function to the MAP function from the Switch Panel.

System maintenance is facilitated through the use of a remote terminal located at a diagnostic site. When a system error occurs, a remote MAP Terminal at the diagnostic site can be connected to the system. An auxiliary communications link is available for each system to implement this function. The remote terminal is connected to the MAP with the keylock switch. This function permits the engineer at the remote location to

examine the system for errors, and allows him to load, run, and control diagnostics from the remote terminal. The diagnostics can be run concurrently with other system activities.

STANDARD AND OPTIONAL FEATURES

Harris 800 systems contain various hardware features. Many options are also available to enhance system performance. A brief description of standard features and options are provided in the following paragraphs. Unless otherwise indicated, additional details pertaining to system features and options are contained in Section II.

A listing of the standard hardware that is provided with a typical system is as follows:

- Central Processor with hardware multiply/ divide/square root, power supplies, and two cabinets
- Scientific Arithmetic Functions
- 960K bytes of MOS Memory with error correction
- 6K byte Cache Memory
- 12,288K bytes of Virtual Memory address space
- Maintenance Aid Processor with CRT Console
- System Console CRT with Keyboard and Controller
- Switch Panel
- 16 Priority Interrupt Levels
- 120 Hertz Clock
- Power Fail Alarm
- Firmware Bootstraps
- Bit Processor
- Stall Alarm
- Executive Traps
- Interval Timer
- Address Trap
- Programmed Input Output Channel (PIOC)
- Direct Memory Access Communications Processor (DMACP-8) Channel with four asynchronous ports
- Buffered Block Channel (BBC) (2)

A summary of optional hardware items that could be added to the foregoing system follows:

- 100 kHz Real Time Clock
- Programmed Input Output Channels (PIOCs)
- External Block Channels (XBCs)
- Integral Block Channels (IBCs)
- Buffered Block Channels (BBCs)
- Direct Memory Access Communications Processor (DMACP-8) Channels
- 56 Priority Interrupt Levels
- Memory Extension Unit
- Shared Memory Unit
- I/O Expansion Unit
- Interprocessor Communication Facility
- Multi-Channel Adapter

Priority Interrupts

Four priority interrupt groups are available; groups 0, 1, 2 and 3. Group 0 is reserved for internal CPU functions and is comprised of eight executive trap interrupt levels. All executive trap levels are associated with specific functions.

Groups 1, 2, and 3 are reserved for external interrupts; each group may have up to 24 levels. A basic system is supplied with 16 external interrupt levels. Fifty-six additional external interrupt levels are available.

Complete details pertaining to the priority interrupt system are contained in Section V.

120 Hertz Clock

Continuously generated interrupt triggers are placed under software control by enabling or disabling the associated external interrupt level. By this method, the 120 Hertz Clock may be used for various timing operations. The clock continuously transmits 120 interrupt trigger pulses per second for 60 Hertz power, and 100 interrupt trigger pulses per second for 50 Hertz power.

Interval Timer

The programmable Interval Timer functions as an internal CPU timer that provides a method for regulating operating program segments and recording other intervals. Depending on the instruction used for its activation, the

Interval Timer clocks either CPU time or clock (real) time. In addition to its timing applications, the Interval Timer provides the user with an additional 24-bit general-purpose register that may be accessed through the standard instruction set when in the Monitor Mode of operation. The T Register may not be modified when in the User Mode.

100 kHz Real Time Clock

This option provides the programmer with general purpose clock pulses that are independent of the mainframe clock pulses. With an accuracy of .05%, the real time clock pulses are available whether the CPU is in standby or not. The timing pulses can be used to measure user's program running time, or to generate periodic interrupts. Programming is accomplished through normal input/output commands. One or two real time clocks may be installed on the Programmed Input Output Channel (PIOC) boards.

Power Fail Alarm

The Power Fail Alarm Module monitors the ac line voltage and generates power down signals to the CPU and memory modules in the event of an ac power failure or sustained low ac line voltage. An executive trap interrupt is triggered and, to prepare the CPU for an orderly restart, the major CPU registers and control logic are cleared. A one millisecond interval is available between the time of the interrupt and final shutdown. Once the ac line voltage is restored to a normal level, an executive trap is triggered and a restart signal is sent to the MAP. This signal causes the MAP firmware to interrogate the system status and reinstate the operation, if appropriate.

Bootstraps

Automatic program loading from a selected peripheral device is provided by the Bootstrap feature. Through the use of the MAP Terminal, the appropriate bootstrap program is loaded into memory. Once loaded, the bootstrap program will automatically load a minimum of one record from the appropriate device. Programs provide for loading from disc, magnetic tape, or punched cards.

Bit Processor

Capability is provided by the Bit Processor for selectively changing, testing, or performing logical operations on a single bit in memory. A special group of instructions enables implementation of these functions.

Stall Alarm

Certain operations in the instruction set and other internal conditions prohibit the recognition of external interrupts. A series of these instructions or conditions could, therefore, produce a situation where external interrupts are, in effect, "locked out". The Stall Alarm monitors all instructions and conditions in this interrupt-prohibiting category. If a series of these instructions or conditions have not been completed before the elapse of a predetermined time period, they are terminated and an executive trap interrupt is generated. The subsequent interrupt processing routine may then examine the situation and take any necessary corrective action. The Stall Alarm includes the appropriate control logic and is furnished with the associated executive trap interrupt.

Address Trap

This feature provides for an executive trap interrupt to occur at a specified address and under certain conditions. The Address Trap is used as an on-line debugging aid for use in applications such as breakpoint tracing. An address may be defined under program control so that when the address is referenced, an interrupt will be generated at the assigned executive level. The Address Trap may be enabled or disabled under program control. The Query Register provided with the Address Trap may not be modified when the virtual memory is in the User Mode of operation.

Input/Output and Communications Processor Channels

A variety of I/O and communications processor channel types are available with a system. Each channel is designed for a particular input/output data transfer application. Multiple channels of a given type may be used as the application demands. A brief description of each type follows. A more detailed discussion of the channels is provided in Section IV.

Programmed Input Output Channel (PIOC)

This is an I/O channel capable of implementing a single-word, eight-bit, parallel data transfer between the CPU and a suitable peripheral device. This channel has provisions for installing up to four unit interface controllers on the I/O circuit board. In addition, the PIOC can drive up to 12 additional remote device controllers. This board also contains a programmable Interrupt Generator which may be used in multi-processor installations. If required, one or two Real Time Clocks

may be installed on the board. Units which may be interfaced to the PIOC include teletypes (with or without cassette), line printers, CRT terminals, RS 232 asynchronous controllers, and communications multiplexers.

Buffered Block Channel (BBC)

The Buffered Block Channel performs and controls automatic data transfers between main memory and any of up to 16 external high-speed peripheral controllers. Data is transferred between main memory and the BBC in a 48-bit parallel word format, and between the BBC and peripheral controllers in a 24-bit parallel word format. Data transfers between the BBC and a peripheral controller may be performed simultaneously with transfers between the BBC and memory.

Data and command chaining is supported as well as programmed input/output transfers. After a block of data is transferred, the data chaining capability permits a subsequent data block to be automatically transferred without program intervention. Command chaining enables the channel to automatically access memory for a command word upon completion of the current block transfer. A new block of data is then automatically transferred under new command restraints. In addition to the standard forward motion read and write operations, the BBC can also perform a read reverse operation which alleviates the need for rewinding magnetic tapes if the controller supports this feature. For this function, the 24-bit input data words from the peripheral controller are assembled in reverse order by the BBC and are then transferred to memory in 48-bit parallel format. Addressing and transfer block sizes are established under program control. Once initiated, all BBC DMA operations proceed automatically.

A large internal buffer (48-bits wide by 16 words deep) in the BBC allows peripheral transfer rates to be maintained during periods when the CPU inhibits memory access by the channel. The internal buffer and a dual priority scheme make most BBC memory requests occur in groups. This function increases CPU performance by increasing the effective bandwidth.

A special function is provided which enables the BBC to generate odd parity on command and data transfers to designated controllers, and check for odd parity on data transfers from the units. Parity errors occurring during block or programmed word transfers are reported with a status word.

Direct Memory Access Communications Processor (DMACP-8)

The DMACP-8 is a multiport communications processor channel dedicated to serial data communications and provides direct access to main memory for up to eight devices. These communication devices can be either asynchronous or synchronous. Up to eight asynchronous interfaces can be used, or one synchronous and up to four asynchronous interfaces can be accommodated. The one synchronous interface takes the place of four synchronous interfaces. Each interface is termed a port.

Standard interfaces available with the DMACP-8 are RS-232C, 20 ma current loop, and Harris Differential. Asynchronous devices supported are Harris Standard CRT Terminals, interactive CRTs, teletypewriters, Bell Asynchronous Modems, and Bell compatible modems up to 19.2 kilobaud. Also supported are Ti 700 devices with or without cassettes. Synchronous devices supported are Bell Synchronous RS-232C Modems, and Bell compatible synchronous modems with transfer rates up to and including 50 kilobaud.

Programmed data transfers or block data transfers of 24-bits are performed between the DMACP-8 and CPU. Programmed word transfers are used for status check, initialization, and control of the DMACP-8. Block mode operations are used for data transfers between main memory and the communication devices attached to the ports. These transfers are under control of the microprocessor installed on the DMACP-8 board and require no intervention by the CPU. Transfers between the DMACP-8 and main memory are in the form of 24-bit words, while transfers between the DMACP-8 and communication devices are in the form of 8-bit bytes.

External Block Channel (XBC)

The External Block Channel provides for direct memory access operations between memory and up-to-eight user-defined external controllers. Any one of the eight controllers connected to the XBC may initiate memory transfer sequences. Since controllers may be self-starting, no CPU commands are required to perform DMA transfers. If required, however, controllers may be activated by generating commands to the XBC by means of the input/output instructions. Parameters required for block transfers, such as block length and memory transfer address, may be furnished by either the controller or the CPU. Once a controller is activated for memory transfer operations, the controller initiates word transfer sequences and controls the operational parameters. Data is

transferred between main memory and the controllers in a 24-bit parallel word format.

Integral Block Channel (IBC)

An Integral Block Channel provides automatic data transfers between main memory and one self-contained controller. The controller is dedicated to a block mode card reader. The IBC is initialized by the CPU to perform DMA transfers under self-control. Data chaining provides for the transfer of subsequent blocks of data without program intervention. Data transfers between memory and the controller are in a 24-bit parallel word format.

Input/Output Expansion Unit

Available as an option, the input/output expansion unit increases the input/output capacity of a system. The maximum number of input/output channels supported by a system is 31. I/O expansion is implemented by adding a cabinet assembly to the basic system. The cabinet contains a power distribution unit, power supplies, and a 22-slot chassis assembly. A pair of PC boards plus cables is also provided which is used to connect the expansion unit to the basic system. The I/O expansion unit may be used to expand the I/O capabilities of computer systems located in the field.

Interprocessor Communication Facility

The Interprocessor Communication Facility (ICF) provides direct communications capability between interconnected computers in a dual computer installation. Memory-to-memory transfers are made under the control of a DMA channel installed in each CPU. Either channel in the link may control the transfer. The computer link is particularly useful in real-time control applications involving dual computers.

Two Harris 800 CPUs may be linked together, or the CPU in an 800 system may be interconnected to a CPU in either a Harris 100 or 500 system. The ICF is implemented with two dedicated Buffer Block Channels, link cables, and four Priority Interrupt Generators. The interconnected systems may be separated by as much as 120 feet. Software interrupts generated by the Priority Interrupt Generators are normally used to initialize link operations.

This option is not supported by VULCAN except for the priority interrupt structure. Establishment of control is user implemented and requires user code.

Multi-Channel Adapter

Usually used in a multiple computer configuration, the Multi-Channel Adapter allows peripheral devices to be shared by two or more computers. CPUs share units via an I/O interface made common by daisy chain installations. Normally, each CPU is equipped with an interrupt generator for the purpose of generating software interrupts required for daisy chain operations. When a CPU wants to communicate with a unit on the common interface, it generates an interrupt to cause the other CPU(s) to set their daisy chained channels off-line. The CPU generating the interrupt can then exercise the shared unit without interference. This feature is not supported by VULCAN except for the priority interrupt structure. Proper establishment of control is user implemented and requires user code. If simultaneous access is made to a unit on the common interface, the results are indeterminate.

PERIPHERAL EQUIPMENT

Harris 800 systems can be expanded and enhanced by selection from a variety of peripheral equipment offered for each system, including:

- Moving Head Discs (40, 80, and 300M Bytes)
- Magnetic Tapes (45 and 75 ips)
- Card Readers (300, 600 and 1000 cpm)
- Key Punch/Card Punch (35 cpm)
- Line Printers (300, 600 and 900 lpm)
- Electrostatic printer/plotter (300, 500, 1000 and 1200 lpm)
- Paper Tape Devices
- Console Devices, Local and Remote Terminals
- Supplementary equipment to meet most custom requirements

SOFTWARE

The Virtual Memory Manager (VULCAN) operating system is a priority-structured, demand paged, multi-programming operating system. VULCAN concurrently supports multi-stream batch processing, interactive terminal time-sharing, transaction-oriented

processing, multiple remote job entry and real-time operations. Under VULCAN, the virtual memory hardware/software system is transparent to the user. Up to 3M bytes per user is available, all of which may be executable code.

Language Processors

- FORTRAN 77
- FORTRAN IV Compiler with extensions
- Extended BASIC
- COBOL Compiler
- RPG II Compiler
- SNOBOL 4 Interpreter
- FORGO (Diagnostic Load-and-Go FORTRAN Compiler)
- APL Interpreter
- Harris MACRO Assembler

Utility Programs

- Sort/Merge
- VISP (Indexed Sequential File Handler)
- ACUTIL (System Accounting)
- Cross Reference
- VBUG (Symbolic Debugger)

Remote Job Entry Support Packages

- IBM HASP II M/L
- IBM 2780
- CDC 200 UT
- UNIVAC 1004

Remote Batch Terminal Host Packages

- IBM HASP II M/L
- IBM 2780

Data Base Management System

- TOTAL*
- T-ask*

^{*} TOTAL and T-ask are registered trademarks of CINCOM Systems, Inc.

SUMMARY OF CHARACTERISTICS

The major operating characteristics and pertinent technical specifications of the Harris 800 Computer Systems are summarized below.

Computer Organization Microprogrammed general-purpose digital computer.

Multi-level instruction pipelining.

Overlapped address preparation and memory access.

Multi-access central bus structure.

Buffered I/O channels.

Arithmetic Parallel, binary, two's complement fixed and floating

point; includes hardware multiply, divide, square root,

and hardware floating point processor.

Decimal addition and subtraction on packed operands.

Memory System

Main Memory

Type N-Channel MOS

Minimum Size 192K bytes

Maximum Size 3M bytes

Increment 192K bytes

Word Length 48 bits

Parity. One bit error correct

Cache Memory

Storage Configuration . . . Divided into two, 3K byte sections. One section

stores only instructions, and the other section

stores only operands.

Shared Memory

Type N-Channel MOS
Minimum Size 192K bytes
Maximum Size 3M bytes
Increment 192K bytes
Word Length 48 bits

Parity. One bit error correct

Number of Ports (maximum) . 6

Port Access Asynchronous, ring priority

Addressing

Compatibility Mode Immediate

Direct to 96K bytes

Direct to 192K bytes via long address instructions

Indirect to 768K bytes (data only)

Indexed to 192K bytes

Address Extension Mode Immediate

Direct to 3M bytes Indirect to 3M bytes Indexed to 3M bytes

Input/Output Capability

Programmed Data Transfers To/from CPU register, 8 or 24 bits

Automatic Data Transfer Direct memory access via IBC, XBC, DMACP-8,

and BBC, 24 or 48 bits

Single Channel Maximum Transfer Rates (per sec.)

	<u>Input</u>	<u>Output</u>
IBC	80K bytes	80K bytes
XBC (no mainframe contention) . (with mainframe contention)		2M bytes 1.2M bytes
BBC	3.7M bytes	3.7M bytes
DMACP-8	8.1K bytes	8.1K bytes

Input/Output Command Modes

Normal	Normal operation for each channel type.
Multiplex	Channel released to master/slave peripheral units. Not available on IBC, XBC, DMACP-8, or BBC.
Output Special Function	Enables read reverse, channel-to-unit parity checking, channel internal turnaround, and unit master clear function. Applicable only to BBC.
Offline	Channel drivers turned off allowing second CPU to share devices without need for peripheral switches. Not available on IBC.
Reset	Resets Multiplex or Offline Mode. Channel restored online and unit selected. Not available on IBC.

Priority Interrupt Structure

Internal Maximum of eight executive traps. Multi-level

vectored structure.

External Sixteen priority interrupt levels, standard. Optionally

expandable to 72 priority interrupt levels. Multi-level

vectored structure.

Control External interrupts may be individually armed, disarmed,

enabled, inhibited or triggered under program control.

Power Fail Protection Power fail alarm, standard.

Electrical Requirements

220/240, single phase, 3-wire (foreign)

Frequency 60 \pm 2 Hz (50 \pm 2 Hz, optional)

Current (maximum) 46 amps.

Environmental Requirements

Temperature

Operating 50° F to 113 $^\circ$ F (10 $^\circ$ C to 45 $^\circ$ C), ambient air

Humidity

Operating 20% to 80%, relative (non-condensing)

Storage 20% to 90%, relative (non-condensing)

Altitude

Operating -1,000 to 6,000 ft. (-305 to 1,829 m)

Cooling Forced air provided by internal fans on each chassis

SECTION II CENTRAL PROCESSING UNIT

GENERAL DESCRIPTION

The Central Processing Unit (CPU) is a single-address parallel word-oriented, stored-program processor. Operations performed by the CPU include data transfers, arithmetic, computation, and logical manipulation. These operations are defined by instructions stored in, and retrieved from, physical memory. The specified operation is performed on single-word, double-word, byte, or single bit operands stored in memory or contained in one of the CPU registers. Data word formats, as defined by both hardware and software, are illustrated in Figure 2-1.

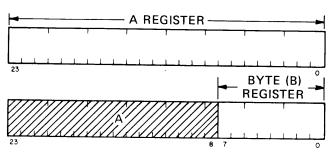
In addition to the general and special-purpose registers, the CPU contains an arithmetic section that performs the actual computation and logical manipulation of operands, and a control section that retrieves and decodes instructions from memory and directs the functional processes of the system. The control section also includes the paging logic that implements the memory address translation and demand-paging operations. The CPU contains interface elements for communications with the other computer elements; e.g., memory, the I/O channels, and the MAP Terminal.

PRINCIPAL CPU REGISTERS

The following paragraphs provide a brief description of the principal registers in a CPU. Registers associated with the priority interrupt system and SAU functions are described elsewhere, in the appropriate sections of this manual.

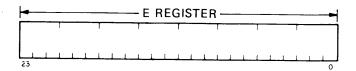
A and B Registers

Serving as the principal arithmetic accumulator, the 24-bit A Register also functions as the input/output communication register during programmed (single-word) transfers between the CPU and peripheral devices. The A Register has complete arithmetic and shift capability. Bits 7-0 of A form an 8-bit pseudo-register, termed the B (Byte) Register. Both the A and B registers are accessible to the user by means of the instruction set and the MAP Terminal.



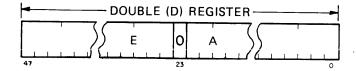
E Register

Employed as an extension of the A Register for increased arithmetic and shift capability, the 24-bit E Register also functions as a general-purpose storage element during various instructions. The E Register is accessible through both the instruction set and the MAP Terminal.



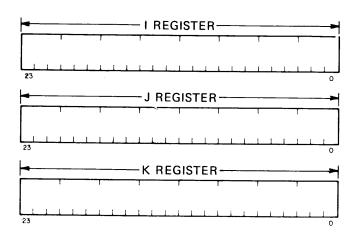
D Register

The D (Double) Register is a 48-bit pseudo register formed by combining A and E to provide double-precision arithmetic and shift capability. The A and E Registers form the least- and most-significant halves, respectively, of the 48-bit double-precision quantity. A23 is reset for most instructions. Several instructions provide direct access to the D Register; MAP Terminal entry, however, must be accomplished by accessing the E and A Registers in the proper format.



I, J, and K Registers

Each of these is an independent, 24-bit general-purpose register that can also be employed as an index register for address modification. The I, J, and K Registers are directly accessible through the instruction set and the MAP Terminal.



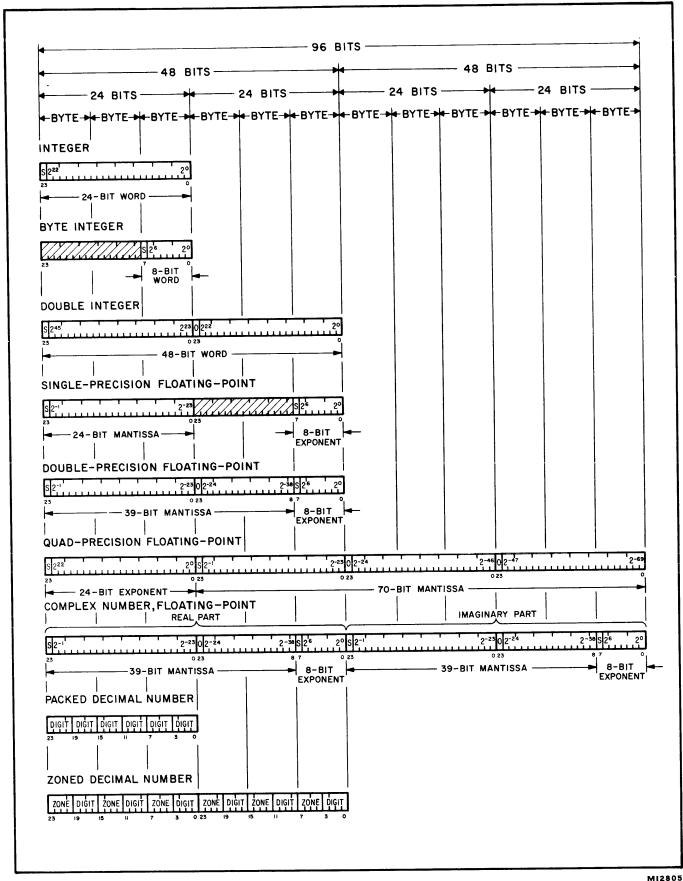


Figure 2-1. Data Formats

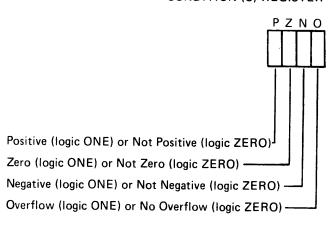
Condition Code Register

Condition codes indicate the nature of the results of an instruction. The significance of the condition code bits depends on the particular instruction just executed. Condition codes are loaded into a condition register. Two such registers are provided; a C Register and a Y Register.

C Register

A 4-bit element that stores the results of specific operations, the C (condition) Register is accessible by means of several instructions. Condition codes generated by all instructions, except SAU and decimal instructions, are loaded into the C Register. This register is termed the C or Condition Register throughout this manual and throughout related Harris 800 manuals. Display of the C Register is provided by the MAP Terminal.

CONDITION (C) REGISTER



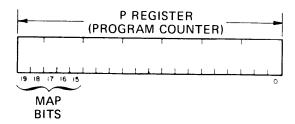
Y Register

Condition codes generated by the SAU and decimal arithmetic instructions are loaded into the Y (condition) Register. A detailed description of this register is provided in Section VI. This register is termed the Y Register throughout this, and related, Harris 800 manuals.

Program Address Register

Also called the Program Counter, the 20-bit Program Address (P) Register retains the memory address from which the current instruction was fetched. In the Compatibility Mode of operation, bits 19 through 16 are not used and a maximum of 65,536 memory locations can be accessed via the P Register. In the Address Extension Mode, all 20 bits are used and a maximum of 1,048,576 locations can be accessed. In the Compatibility Mode, bit 15 is used as a map bit, and when in the Address Extension Mode, bits 19 through 15 serve as map

bits. The register can be loaded with a Branch and Link instruction. Contents of the register can be saved with a BSL instruction in the Compatibility Mode, or a BSL or BSX instruction in the Address Extension Mode. The contents of the P Register can be modified through the execution of any of several branch instructions. The MAP Terminal provides direct entry and display for the P Register.



Instruction Buffer

Once an instruction has been fetched from memory, it is retained in the Instruction Buffer during decoding and execution. The Instruction Buffer is not programmable. The buffer holds up to four prefetched instructions.

VIRTUAL MEMORY DESCRIPTION

Introduction

Paging is a hardware addressing scheme that allows a program's memory area to be discontiguous. Program segments may be absent from physical memory while other portions of the program are being executed. This aspect of the paging operation, termed "demand-paging", also allows the computer to execute programs larger than the available physical memory; hence, the term "virtual memory". The following paragraphs discuss the paging hardware and describe the basic functions of the VM.

Virtual Memory Instruction Set

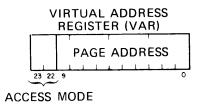
A virtual memory instruction set is provided for program control of paging functions. These instructions can only be executed in the Monitor Mode. If an attempt is made to execute any of these instructions while in the User Mode, an instruction trap interrupt is generated. A detailed description of each of these instructions is provided in Section VII of this manual.

Principal Virtual Memory Registers

Various registers are supplied with the VM paging logic. A brief description of each is provided in the following paragraphs. Entry and display of all principal VM registers is provided via the MAP Terminal.

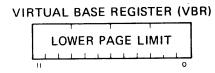
Virtual Address Register (VARs)

A total of 4,096 of these 12-bit VARs are supplied. The ten least-significant bits (9-0) retain the address of a physical memory page, while bits 23 and 22 define the manner in which the specified page may be accessed. The access modes and their corresponding bit configurations are defined in the paragraph describing demand paging operation. Specific operations within the VM instruction set provide transfers to and from the VARs.



Virtual Base Register (VBR)

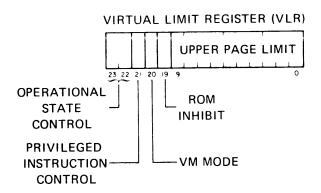
The 12-bit VBR retains the lower page limit of the user program; i.e., the address of the first assigned VAR for the currently-executing program. Special VM instructions provide for loading the VBR and retrieving its contents.



Virtual Limit Register (VLR)

Bits 9-0 of the 15-bit VLR define the upper page limit of a user program, i.e., the number of VARs minus 1 which the program may reference; bits 23 through 19 provide special controls. Bits 23 and 22 control the operational state of the CPU (see paragraph describing the CPU operational states). When bit 21 is set, any of the privileged instructions may be executed without generating an instruction trap interrupt (see paragraph describing instruction trap). Virtual memory instructions may only be executed in the Monitor Mode, regardless of the state of bit 21. When an interrupt occurs in the Address Extension Mode of operation, the virtual memory mode of operation is saved in bit position 20. The bit is set if the interrupt occurred in the User Mode, or reset if the Monitor Mode was active. When bit 19 is set, the Release Operand Mode (ROM) instruction is suppressed.

The VLR may be loaded, or its contents retrieved, by specific VM instructions.



Virtual Usage Registers (VURs)

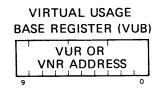
A total of 1,024 of these one-bit registers are supplied; one is associated with each physical page of memory. Each time a given memory page is accessed by a CPU instruction, a ONE is stored in the appropriate VUR. The VURs may be selectively tested and cleared under program control.

Virtual Not-Modified Registers (VNRs)

A total of 1,024 of these one-bit registers are supplied; one is associated with each physical page of memory. Each time data is written (stored) in a given memory page by an instruction reference, a ONE is stored in the appropriate VNR. The VNRs may be selectively tested and cleared under program control.

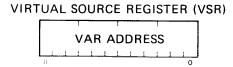
Virtual Usage Base Register (VUB)

This 10-bit register retains the address of one of the VURs or VNRs (equivalent to the associated physical page). This address is used as a pointer to access the appropriate VUR or VNR during the Query Virtual Usage Register (QUR) or Query Not-Modified Register (QNR) instruction. The VUB can be loaded or its contents retrieved by special VM instructions.



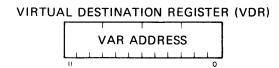
Virtual Source Register (VSR)

This 12-bit register retains the address of one of the VARs and is used as a pointer for retrieving data from the VARs during a Transfer 2 Virtual Address Registers to Double (TRD) instruction. The VSR can be loaded under program control.



Virtual Destination Register (VDR)

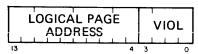
The 12-bit VDR retains the address of one of the VARs, and is used as a pointer for storing data in the VARs during Transfer A to 1 Virtual Address Register (TAR) and Transfer Double to 2 Virtual Address Registers (TDR) instructions. A special VM instruction provides program-controlled loading of the VDR.



Virtual Demand Page Register (VPR)

A special register (VPR) is used in the virtual memory system to copy the logical page address (bits 13-4) of the user program for each memory reference so that if a particular cycle causes a fault, the operating system knows which logical page is involved and the condition that caused the fault. The address of the VAR that created a demand page or limit register violation is the contents of the VBR plus the contents of the VPR. Bits 3-0 identify the type of violation. The contents of the VPR may be retrieved under program control.

VIRTUAL DEMAND PAGE REGISTER (VPR)



Demand Paging

Demand paging is the aspect of the VM hardware that permits a portion of the user's program to be absent from physical memory (and located instead on a disc mass-storage device) while the program is being executed. When the address translation logic detects a reference to a non-resident page, an executive trap interrupt (Group 0, Level 2) is triggered. Subsequent processing by the operating system may then access the desired page and load it into physical memory. If sufficient memory space is not available, the operating system may interchange inactive resident program segments with the incoming page(s) or programs (i.e., transfer the inactive segments to the disc storage device). Once the correct program sequence is loaded into physical memory, the user's program may continue its normal sequence.

A non-resident page is signified by ZEROs in bit positions 23 and 22 of the selected VAR. Each time a VAR is accessed, these bits are examined by the paging control logic to determine if a demand page is required. The last logical page presented to virtual memory is stored in bits 13-4 of the Virtual Demand Page Register (VPR).

The interrupt generated at Group 0, Level 2 may reflect a limit register or restrict mode violation as well as a demand page. Bits 3-0 of the VPR define which condition generated the interrupt; these are examined by the operating system to determine what steps are to be taken Entry in processing the interrupt. into interrupt-processing routine requires saving a return address; usually, the interrupt address plus one. Certain situations require reexecution of the instruction that created the demand page or violation, consequently, the program counter must be adjusted to fetch the instruction again. The Program Counter is automatically adjusted by hardware before the interrupt is taken; no software adjustment is made. (Note that in previous systems, software used bits VPR1 and VPR0 to adjust the Program Counter.) Table 2-1 defines the VPR status and control

Table 2-1. VPR Status Bits Definitions and Functions

Condition	VPR Bits 3 2 1 0	Type of Violation
1	0 0 0 1	Demand Page
2	0 1 0 1	Mode 3*
3	1001	Mode 2*
4	1 1 0 1	Limit Register

^{*} Page Access Mode Violation

The paging logic provides a program restrict system that permits pages of memory to be protected from unauthorized access. A user's program area is defined by the contents of the Virtual Base Register (VBR) and Virtual Limit Register (VLR). The VBR defines the lower page limit in the user's program while the VLR defines the last page, or upper limit. No user's programs can reference any memory location below the lower page limit because all addresses are biased by the VBR's contents during the address translation operation. Any attempt to reference memory above the upper limit will result in a limit register violation and trigger the Group 0, Level 2 executive trap interrupt.

Each page of memory can be further protected by placing it in one of three access modes. Bits 23 and 22 of the VARs contain the access mode bits for the associated page. Any attempt to access the selected page in any manner other than specified in the mode bits will result in triggering the Group 0, Level 2 executive trap. The access mode bits are defined below.

<u>Mode</u>	Bit 23	Bit 22	<u>Description</u>
0	0	0	Page Missing – page is not contained in physical memory (demand page).
1	0	1	Unrestricted —instructions may be executed within the page and data may be loaded from or stored within the page.
2	1	0	Execute/Read —instructions may be executed within the page or data loaded from the page; data may not be stored within the page.
3	1	1	Read —data may be loaded from the page; instructions may not be executed within the page and data may not be stored within the page.

The program restrict functions are enabled only when the VM system is in the User Mode.

Instruction Trap

An instruction trap function is included as an integral part of the paging hardware. The trap prevents the execution of certain, predetermined, instructions. When the trap is enabled, any attempt to execute one of the designated instructions will result in an executive trap interrupt at Group 0, Level 3.

The instruction trap function is automatically enabled when the paging logic is placed in the User Mode. When enabled, the trap will analyze bit 21 of the Virtual Limit Register (VLR). When VLR bit 21 is set (ONE), the following instructions may be executed without generating an instruction trap violation. If bit 21 is reset (ZERO) and the instruction trap is enabled, a violation will occur when an attempt is made to execute any of the following instructions.

Hait (HLT)
Input Address Word (IAW)
Input Data Word (IDW)
Input Status Word (ISW)
Input Parameter Word (IPW)
Output Address Word (OAW)
Output Command Word (OCW)

Output Data Word (ODW) Hold External Interrupts (HXI) Release External Interrupts (RXI) Unitarily Arm Group 1 Interrupts (UA1) Unitarily Arm Group 2 Interrupts (UA2) Unitarily Arm Group 3 Interrupts (UA3) Unitarily Disarm Group 1 Interrupts (UD1) Unitarily Disarm Group 2 Interrupts (UD2) Unitarily Disarm Group 3 Interrupts (UD3) Unitarily Enable Group 1 Interrupts (UE1) Unitarily Enable Group 2 Interrupts (UE2) Unitarily Enable Group 3 Interrupts (UE3) Unitarily Inhibit Group 1 Interrupts (UI1) Unitarily Inhibit Group 2 Interrupts (UI2) Unitarily Inhibit Group 3 Interrupts (UI3) Transfer Double to Group 1 (TD1) Transfer Double to Group 2 (TD2) Transfer Double to Group 3 (TD3) Transfer Double to Group 1 (TD4) Transfer Double to Group 2 (TD5) Transfer Double to Group 3 (TD6) Transfer Group 1 to Double (T1D) Transfer Group 2 to Double (T2D) Transfer Group 3 to Double (T3D) Transfer Group 1 to Double (T4D) Transfer Group 2 to Double (T5D) Transfer Group 3 to Double (T6D) Hold Parity Error Retry (HER) Release Parity Error Retry (RER) Load Virtual Demand Page Register (LVR) Read Parity Bits (RPB) Transfer Parity Error Address Register to A (TPA) Transfer Active Executive Traps to A (ACE) MAP Interrupt Request (MIR) Transfer CAM to Double (TCD)

If the instruction trap is enabled, the VM group of instructions will result in a violation (VLR bit 21 has no effect on this group) if the user program attempts to execute them. Any attempt to execute an Interval Timer start or stop instruction, T Register load instruction, or SAU interrupt control instruction in the User Mode when VLR bit 21 is reset causes the instruction to be treated like a NOP. No interrupt is generated. The following instructions are affected:

1) Hold Interval Timer (HIT)

Transfer CAM Hit Status to A (THA)

- 2) Release Processor Time (RPT)
- 3) Release Clock Time (RCT)
- Any register to register instruction that loads the T Register; e.g., a TAT instruction.
- 5) Hold SAU Overflow Interrupt (HSI)
- 6) Release SAU Overflow Interrupt (RSI)

Paging System Control

When a master clear is generated, the Monitor Mode is established. The paging logic then remains in the Monitor Mode until placed in the User Mode.

The User Mode is established under program control (i.e., via the RUM instruction). The RUM (Release User Mode) instruction causes the User Mode to be established at the completion of the instruction following the RUM. (This instruction should, in practice, always be an unconditional branch.) After the new program address has been calculated, the User Mode will be activated. The RUM instruction, together with the following instruction, will be handled like an EXM with respect to a demand page (VPR bits 0 and 1 will be set to ONE and ZERO, respectively). Refer to Table 2-1.

A BLU (Branch and Link-Unrestricted) instruction will automatically establish the Monitor Mode; the BLU's 5-bit effective memory address will not be mapped. Bit 20 of the J Register will be set (ONE) if the BLU was executed in the User Mode, and reset (ZERO) if the BLU was executed in the Monitor Mode.

When an interrupt occurs in the Compatibility Mode, the Monitor Mode will be established; the hardware-generated EXM (Execute Memory) instruction will not be translated. The BSL (Branch and Save Return-Long) to the dedicated interrupt location will transmit the paging mode at the time of the interrupt to the BSL's effective memory address. Bit 20 will be set (ONE) if the system was in the User Mode, and reset (ZERO) if it was in the Monitor Mode. If the interrupt occurs in the Address Extension Mode, the hardware-generated BSX will not be translated and the Monitor Mode will be established. The VM mode of operation at the time of the interrupt will be saved in bit 20 of the Virtual Limit Register (VLR). If no other interrupt is active, VLR20 will be set if the system was in the User Mode, and reset if it was in the Monitor Mode.

If a demand page interrupt occurs while executing a ROM instruction, the VM mode is recorded as Monitor. Bit 20 of the BSL save word is reset if in the Compatibility Mode, or VLR bit 20 is reset if in the Address Extension Mode. When returning from an interrupt routine via an indirect BRL instruction, bit 20 of the entry point is tested, and the User or Monitor Mode is re-established accordingly.

When in the Compatibility Mode and an indirected BRL instruction is executed in the Monitor Mode, the User Mode is established if bit 20 of the save word is set. The instruction following the indirect BRL is translated. When in the Address Extension Mode and an indirected BRL is executed in the Monitor Mode, if the currently active

interrupt is the only one active and if bit 20 is set in the VLR, the User Mode is established and the instruction following the indirect BRL is translated. If VLR bit 20 is reset, the Monitor Mode continues.

CPU OPERATIONAL CONTROL

CPU Modes of Operation

Since a Harris 800 Computer is an upward compatible extension of the SLASH 6 Computer used in Harris 100 systems, all user software which can be run on the SLASH 6 can also be run on a Harris 800 Computer. New software not previously available, can also be run on the Harris 800. Two modes of operation, termed the Compatibility Mode and the Address Extension Mode, are provided to select the particular operation required.

Compatibility Mode

In the Compatibility Mode, the Harris 800 is downward compatible with the SLASH 6. All user programs which run on the SLASH 6 can run on the Harris 800 without recompilation. Current user programs, including compilers and assemblers, may also be run in this mode. All standard instructions which can be executed on the SLASH 6 operate identically when executed on a Harris 800 in the Compatibility Mode. In addition, all extended instructions can be executed in this mode. Bits PC19-16 are kept in the cleared state when operating in this mode, therefore, the Program Counter is effectively 16 bits wide. Thus, branch addresses cannot be over 16 bits wide. Direct addressing capability is up to 64K words. In this mode of operation, indexing, indirection, and interrupt linkage function identically to the SLASH 6.

Address Extension Mode

In the Address Extension Mode, new software elements are available with the Harris 800. This mode of operation does not allow the use of earlier software such as DOS, TOS, DMS, etc. With the exception of the BSL, BRL, BLU, TLO, and GAP instructions, operation of the standard instruction set is identical to the Compatibility Mode. Operation of these five instructions is modified in the Address Extension Mode. Differences in operation are explained in Section VII. All extended instructions can be executed in the Address Extension Mode. All 20 bits of the Program Counter are functional in this mode of operation to provide the capability of direct addressing of up to 1024K words. Indexing, indirection, and interrupt linkage operations are modified versions of similar SLASH 6 operations.

CPU Operational States

Under software control, the CPU is capable of being placed in one-of-four operational states. Two software settable bits in the Virtual Limit Register (VLR), bits 23 and 22, determine state selection. The setting of these bits control the CPU operational states as follows:

VLR 23	BITS 22	STATE	OPERATION
0	0	Zero	System operates in the Compatibility Mode in both the Monitor Mode and User Mode. This state is established whenever the CPU is master cleared.
0	1	One	System operates in the Compatibility Mode when in the User Mode, and in the Address Extension Mode when in the Monitor Mode. When the CPU leaves the Monitor Mode (either following a RUM or an indirected BRL from the only active interrupt level), the CPU is placed in the User Mode. The instruction executed after a ROM instruction should be executed in the Monitor Mode. The calculation of the final EMA uses the Address Extension Mode definition of indexing; the final EMA is translated into user space. On machines with no VM hardware, State One is equivalent to State Three.
1	0	Two	Operation is not permissible and is undefined.
1	1	Three	System operates in the Address Extension Mode in both the Monitor Mode and User Mode.

ADDRESSING FUNCTIONS

Addressing is a function of the Compatibility and Address Extension Modes. Direct addressing, indirect addressing and indexing are dependent on the particular mode enabled.

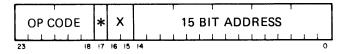
An address is calculated by the CPU without regard to virtual memory. Addresses may be indexed and/or indirected. Address generation is the same whether the virtual memory is in the Monitor or User Mode. When the CPU completes address processing and initiates a memory cycle, virtual memory translates the address if the system is in the User Mode. If in the Monitor Mode, the address is not translated, and the address generated by the CPU is the physical address. In the User Mode, all effective address references generated by the CPU are translated to physical addresses. This includes addresses defined by the Program Counter and all memory reference instructions, including indirect and indexed operations.

Compatibility Mode Addressing

Total memory available to the CPU is one megaword. When the CPU is in the Compatibility Mode, executable code (programs) is confined to 64K (0-65,536 words) of memory when executing standard instructions. However, memory above 64K may be addressed with standard instructions by means of special indirect references. Figure 2-2 illustrates the memory referencing sequence for the Compatibility Mode. Extended instructions can address up to one megaword of memory directly.

Direct Addressing

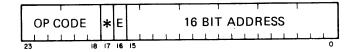
A standard memory reference instruction format is shown below. The 15-bit address field (bits 14-0) in the instruction word provides direct access to 32,768 (32K) words.



In the Compatibility Mode, the addressing logic divides the lower 64K of memory into two areas; 0 - 32K and 32K - 64K. Under this method, bit 15 (P15) of the Program Counter is used to bias all direct address references. Bits 19-16 of the Program Counter are not used in the Compatibility Mode. P15 = 0 specifies an address in the lower 32K, while P15 = 1 designates a location in the upper 32K of the 0 - 64K memory increment. By performing a logical-OR function between the immediate (direct) address reference and P15, standard instructions may directly address up to 32K words within their respective sections of memory.

Modification of a 15-bit direct address by means of the indirect bit, and with or without indexing, can permit a standard instruction to address any memory location up to 256K words.

A special group of "long branch" standard instructions permit direct addressing up to 64K words. The instruction word format for this group is shown below. Note that these instructions may be modified by indirect references (*), but have no provision for indexing. Long branch instructions are not biased by P15. Bit 16 is used to extend the op code.



An extended memory reference instruction format is shown below. The 20-bit address field in instruction word 2 permits memory access to 1,048,576 words.

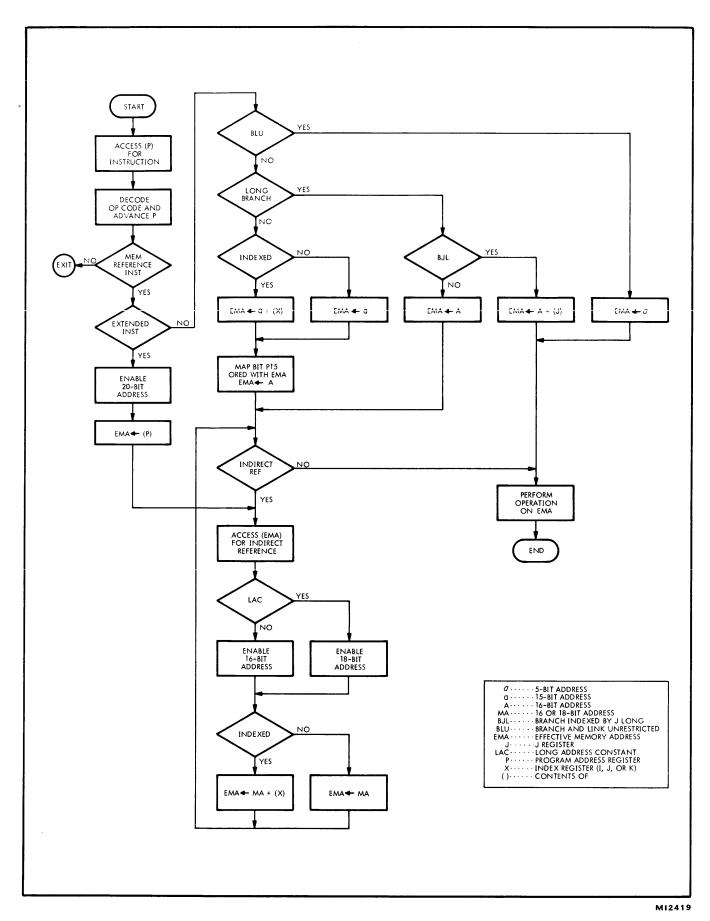
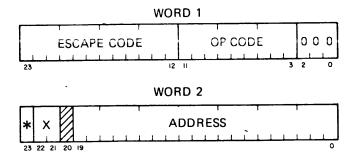


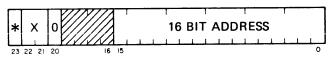
Figure 2-2. Memory Referencing Sequence, Compatibility Mode



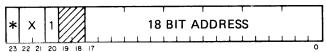
Indirect Addressing

Indirect address references permit the CPU to access up to 256K words of memory in the Compatibility Mode. When a standard memory reference instruction is decoded, bit 17 (*) of the instruction word is examined. If bit 17 is set (ONE), an indirect address reference is indicated. The same function is performed by bit 23 of word 2 in extended instructions. An indirect reference signifies that the effective address (defined by the instruction word plus any index count) contains a second address rather than an operand. The word retrieved from memory when the effective address is cycled is treated as an indirect address word. Compatibility Mode indirect address word formats are illustrated below.





LONG INDIRECT FORMAT



The standard indirect format, with its 16-bit address field, permits access of up to 64K words. Up to 256K words can be accessed by the 18-bit field in the long address word. Neither type of indirect address is affected by the P15 address bias bit.

Bit 23 (*) of either indirect format may be set to specify another level of indirect addressing. Each level of indirect reference may be individually indexed to provide further address modification.

These two indirect word formats are valid only when the CPU is in the Compatibility Mode of operation. When in the Address Extension Mode, a single indirect word is used which differs in format.

Indexing

A direct or indirect address reference may be modified by indexing. This operation adds the address in the current instruction or indirect reference to the contents of a specified index register (I, J, or K) to determine an effective address. A two-bit field (X) in the instruction or indirect reference specifies which register will be employed in each indexing operation. Figure 2-3 provides some examples of indexed addressing.

In the lower 32K memory section (P15 = 0), immediate address references may be indexed to access up to 65,536 words. However, instructions in the 32K-64K section of memory (P15 = 1) may not reference the lower section by indexing since all immediate address references will be biased by 1000008.

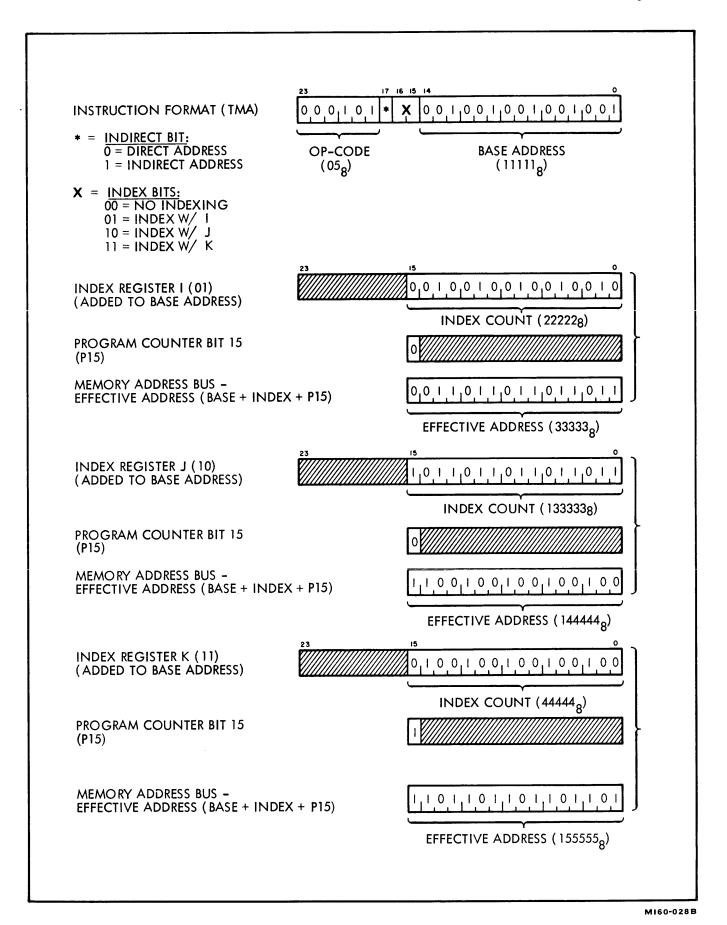
Address Extension Mode Addressing

When the CPU is operating in the Address Extension Mode, direct addressing to one megaword is enabled. Instructions are not restricted to the lower 64K of memory, but may be located anywhere in memory. All 20 bits of the Program Counter are significant so that a maximum of one megaword of memory locations can be accessed via the Program Counter. The memory referencing sequence for the Address Extention Mode is shown in Figure 2-4.

Memory is divided into thirty two, 32K maps in the Address Extension Mode. The most significant five bits of the Program Counter serve as map bits. PC19-15 specify the map in use, and PC14-0 specify the displacement within the map. When PC19, 18, 17, 16, and 15 = 00000, map 0 (0 through 32,767) is specified, and when PC19, 18, 17, 16, and 15 = 00001, map 1 (32,768 through 65,535) is specified, etc. This mapping scheme is applied to all standard memory reference instructions which contain 15-bit addresses. Standard long branch instructions (which have 16-bit addresses), and extended instructions are not mapped.

Direct Addressing

In the Address Extension Mode, the effective memory address of a non-indexed standard memory reference instruction is formed by appending bits 19-15 of the Program Counter to the most significant end of the 15-bit address contained in the instruction. The resulting 20-bit address is termed a local map reference since bits PC19-15 determine map selection. A local map is defined when bits 19-15 of the EMA are equal to bits PC19-15.



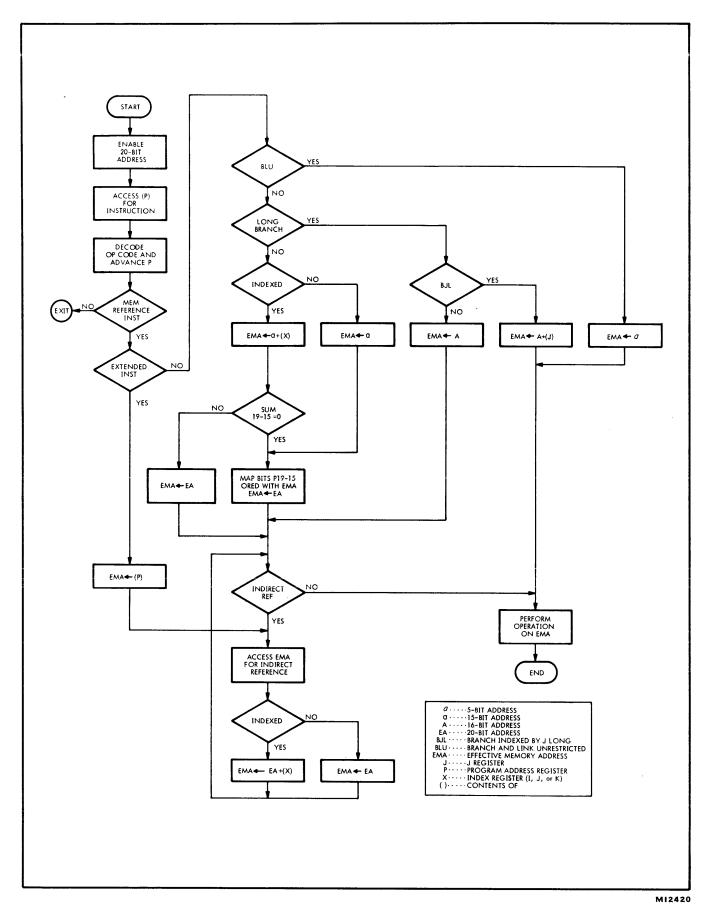


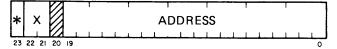
Figure 2-4. Memory Referencing Sequence, Address Extension Mode

Standard long branch and all extended instructions are not biased by the map bits. The 16-bit address in long branch instructions, and the 20-bit address in word 2 of extended instructions are used unmodified.

Indirect Addressing

Unlike the Compatibility Mode which provides for two indirect address word formats, the Address Extension Mode provides for only one indirect address word format. When the indirect bit in an instruction is set, the word retrieved from memory has the format as illustrated below.

ADDRESS EXTENSION MODE INDIRECT ADDRESS WORD



The indirect address word, with its 20-bit address field, provides for accessing up to one megaword of memory. Another level of indirect addressing may be specified by setting bit 23. Each level of indirect reference may be indexed to provide further address modification.

Indexing

Standard long branch and extended instructions are indexed in the Address Extension Mode in the manner described for indexing in the Compatibility Mode, with the exception that the EMA is 20 bits wide. Indexing of 15-bit memory reference instructions, however, differs in the Address Extension Mode.

When indexing is specified in a 15-bit address memory reference instruction, the result of the index operation may be defined to be either a local reference address (indexed address is in same map), or a global reference address (indexed address is in another map). In either case, an EMA is calculated by adding the 15-bit operand of the memory reference instruction to the 24-bit contents of the specified index register, and then examining bits 19-15 of the result to determine if the result should be qualified by map bits PC19-15.

If the sum of bits 19-15 of the result of the addition is equal to zero, the address is mapped into a 20-bit address by appending bits PC19-15 to bits 14-0 of the result. This is the local map case where the calculated address is less than 32K so that the result is a displacement within the same map.

If the sum of bits 19-15 of the result is not equal to zero, PC19-15 are not appended and the EMA is equal to the 20-bit result of the index operation. This is the global map case where bits 19-15 of the result specify another map. The map bits are not used and the EMA is the result of the index operation.

Address Translation

In a VM system, memory is divided into 1024 (1K) word "pages". A translation scheme is applied to the most-significant bits of all memory references. This scheme consists of adding a base address (VBR contents) to the 10 most significant bits of the effective memory address to select a page of memory. The remaining bits of the original memory reference are used to select a specific word within the selected page. Figure 2-5 illustrates the address translation scheme of the VM logic. Figure 2-6 provides an example of the address translation using a standard memory reference instruction.

Address translation is implemented via the Virtual Address Registers (VARs) and the Virtual Base and Virtual Limit Registers (VBR and VLR). Each VAR has a unique number, or address, from 0 through 4095. A specific VAR is selected by adding the ten most significant bits (MSB) of the 20-bit memory reference address to the contents of the VBR. The selected VAR, in turn, contains an address corresponding to 1-of-1024, 1K-word pages.

In practice, the user program is assigned (by the software operating system) a group of sequential VARs. The lower limit of the user program area, and the base for computing VAR addresses, is established by loading the VBR with the first VAR address in the group. The user program upper limit is established by the VLR contents corresponding to the number (quantity) of assigned VARs. Since the MSB value is added to the VBR to compute VAR addresses, the VLR must contain a quantity that is one less than the number of VARs assigned to the user's program. Referring to Figure 2-6, VAR address 16g is specified when the MSB value equals 0, 17g when MSB equals 1, 20g when MSB equals 2, and 21g when MSB equals 3. In this example, the VLR is preloaded with a count of 3. When the MSB value exceeds this count, a limit violation is generated. See paragraph describing demand paging operation. The VARs. VBR, and VLR are loaded under program control in the Monitor Mode.

120 HERTZ CLOCK

This clock continuously transmits 120 or 100 mainframe interrupt signals per second, depending on power line frequency. The interrupt signal is controlled completely by enabling (or disabling) the assigned CPU interrupt level. The first interrupt following an enable signal will occur in less than 1/120 (1/100) of a second because the clock never stops transmitting signals; however, all subsequent interrupts will be precisely 1/120 (1/100) seconds apart.

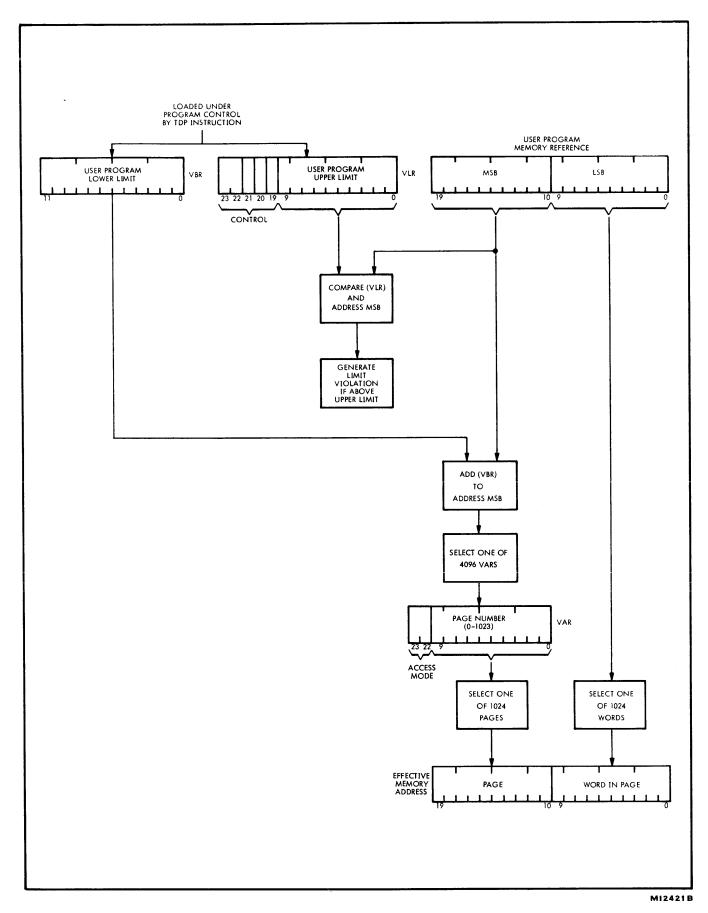


Figure 2-5. Address Translation, VM User Mode

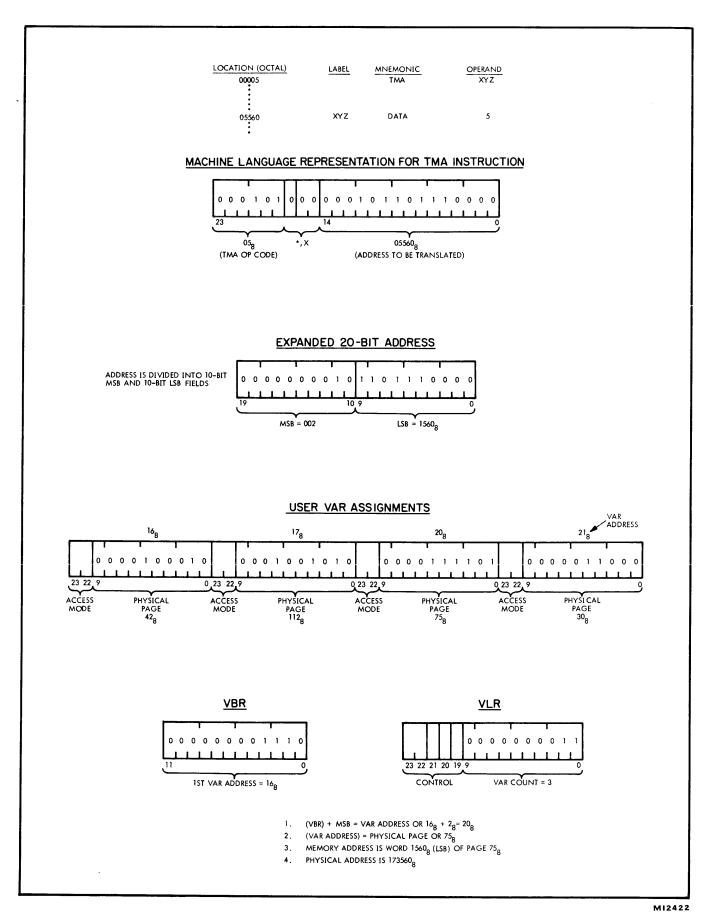


Figure 2-6. Address Translation Example, VM User Mode

The accuracy in using this clock is a function of the user interrupt routine logic. For example, if the clock is used to update a "time-in-seconds" counter by adding one count every 120 (100) interrupts, the "current time" at any given query will be accurate within 1 second. If, however, the counter-is updated each interrupt — 1/120 (1/100) — and divided by 120 (100) when "current time" is queried, the accuracy will be within 1/120 (1/100) of 1 second.

A simple example of coding, where the clock is assigned to priority interrupt Group 1, Level 22, is as follows:

*		Initialize Clock Routine
INITCT	TMA = B22	(A) = Bit 22
	TME = B22	(E) = Bit 22
	UA1	Arm G1/L22
	UE1	Enable G1/L22
	TZM CLOCK T	Zero Clock Time
	BUC 0, J	
*		Interrupt Routine
CLOCK IR	***	Enter
	AUM CLOCK T	Increment Clock Time
	BRL* CLOCK IR	Restore C register and Exit
*		Current Time Routine
CTIME	TMA CLOCK T	
	ESA	
	DVO 120	
	BUC 0, J	
		Patura: (A) - Cacanda

Return: (A) = Seconds (E) = Remainder

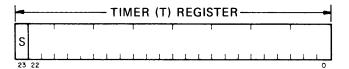
INTERVAL TIMER

General Description

The programmable interval timer consists of a 24-bit register (T Register), a clock, and associated control logic. The timer can be preset and subsequently released, under program control, to measure elapsed processor (CPU) time or clock (real) time.

Timer Register

Supplied with the interval timer, the 24-bit Timer (T) Register operates as a counter in two distinct modes of operation. When not used for timing functions, the T Register functions as an additional general-purpose register that can be accessed through the instruction set when operating in the Monitor Mode. Entry and display for the T Register is provided via the MAP Terminal.



Operational Description

A self-contained clock generates the 1 microsecond pulses used to strobe the timer. In either mode of operation, a count is loaded into the T Register and is decremented once for each elapsed period of 1 microsecond. When the count reaches zero, an executive trap interrupt is generated at Group 0, Level 5. A maximum count of 16,777,215₁₀ (77777778) may be loaded into the register. With a resolution of 1 microsecond per count, a maximum time interval of 16,777215 seconds is available.

Program Control

Interval timer operation is controlled by three instructions: Hold Interval Timer (HIT); Release Processor Time (RPT); or Release Clock Time (RCT). A HIT instruction will prohibit the start of any timing sequence or halt any in-process timing operation until the timer is released by a RPT or RCT instruction. The RPT instruction releases the timer for measuring elapsed processor (CPU) time. In this mode, counting is inhibited during block I/O channel DMA operations, whenever any interrupt is active and enabled, or the CPU is halted. Clock (real) time operation, where the timer counts continuously regardless of CPU condition, is initiated by an RCT instruction.

REAL TIME CLOCK

General Description

The Real Time Clock consists of a 100 kHz crystal-controlled clock, a counter, and associated control logic. All components are mounted on a board which is designed to plug into the internal controller locations of the Programmed Input Output Channel (PIOC) board. Each PIOC can accommodate one or two Real Time Clocks. Although the clock has no peripheral device connected to it, programming is accomplished via normal I/O instructions. More than two Real Time Clocks may be used; the limiting factor being the number of PIOCs used in the system. An external interrupt is provided which is configured in the same manner as any input/output interrupt, i.e., the interrupt can be assigned to any level in Group 1, 2, or 3. The interrupt is generated when the clock count reaches ZERO and the interrupt is enabled.

Operational Description

By means of the Real Time Clock, the programmer is provided with an interval timer which operates independent of CPU timing and provides output pulses when the CPU is either in the Run or Halt condition. Elapsed time is measured by counting down the pulses in the counter. A selected time interval is preset in the counter by loading up to three, 8-bit bytes into the counter. Clock output pulses occur at 10 microsecond intervals. A maximum time period

of 167 seconds is available when the counter is loaded with all bits set in the three bytes. Thus, the programmer can preset the clock for time intervals from 10 microseconds to 167 seconds in 10 microsecond increments. Since the Real Time Clock is asynchronous with CPU timing, the period may be off by 10 microseconds on the first count-down cycle.

Command and Status Word Formats

As a result of the CPU issuing an Output Command Word (OCW) instruction, a command word is transferred from the A Register to the Real Time Clock. The command word initiates operation of the clock, and provides the necessary set-up and control functions. A description of the function performed by each bit of the command word is given below.

7	6	5	4	3	2	1	0
Run/ Hold	Load Preset Count	Enable Snapshot	Enable Bits 0-3	Byte Count 2 ¹	Byte Count 2 ⁰	Enable Auto Restart	Enable Interrupt

Bit 0	(1) (0)	Enable count zero interrupt Disable count zero interrupt
Bit 1	(1) (0)	Enable Automatic Restart of preset count Go into hold mode at count of zero
Bits 2, 3	3	Byte count for input and output
Bit 4	(1) (0)	Sample bits 3-0 Hold bits 3-0 unchanged
Bit 5	(1) (0)	Enable count snapshot output No action
Bit 6	(1) (0)	Enable loading of preset count No action
Bit 7	(1)	Enable count down

An Input Status Word (ISW) instruction generated by the CPU results in the status word being transferred from the Real Time Clock to the A Register. The clock status word consists of bit 0 only. It is set to the ONE state whenever the clock module is plugged into the PIOC board, indicating to the CPU that it is on-line.

Hold count down

(0)

Program Control

Real Time Clock operation is controlled with four instructions: Output Command Word (OCW), Input Status Word (ISW), Output Data Word (ODW), and Input Data Word (IDW). Each Real Time Clock is addressed by a channel-unit code combination in the same manner as any I/O device. If one Real Time Clock is installed, a unit code of 00, 01, or 02 is assigned according to its plug-in location. If two Real Time Clocks are installed, unit codes of 00 and 02 are assigned. Access to the clock is via the A Register as in normal I/O operation.

Preset Count Loading

To initialize the Real Time Clock, an OCW instruction is generated by the CPU to transfer the command word with bit 6 = 1, and the desired byte count in bits 2 and 3. The CPU then provides the specified number of ODW instructions (one per byte) to transfer the bytes to the clock, with the most-significant byte transferred first. When the byte count is satisfied, an OCW instruction may be given to transfer a command word with bit 7 = 1. This enables the counter to start counting down. If bit 7 = 0 in any command word, counting is inhibited until a command word with bit 7 = 1 is received. If a byte count less than three is specified, the unused bytes in the counter are set to ZEROs.

Automatic Count Restart

If bit 1 = 0 in the command word, the automatic count restart is enabled. This causes the Real Time Clock to automatically reload the last preset count into the counter and restart the count after the interrupt is given.

Snapshot Output

During Real Time Clock operation, the current count status is made available to the CPU by means of the Snapshot mode of operation. Snapshot output is initiated with an OCW instruction and bit 5 = 1 in the command word. This loads the 24 bit current count into a register. IDW instructions, one per byte, transfer the contents of the register to the CPU, the most-significant byte being transferred first. This operation does not affect the counting as long as bit 7 = 1 in the command word. If an interrupt is generated during the Snapshot mode of operation, the mode is terminated as the count is known to be zero.

If snapshots are performed in a program with automatic count restart selected, snapshot time prior to automatic restart may be 10 microseconds different from snapshot time after automatic restart. This is because of the 10 microsecond time frame used in the Real Time Clock. Additionally, if a snapshot is performed at the trailing end

of a time out, before restarting or auto-restarting, the snapshot bytes may be all zeroes. To minimize the possibility of the foregoing occurrences, the snapshot of any time must be accomplished in the least machine time possible. An example of programming code that may be used to do a snapshot in the shortest period of machine time follows:

SNSH	DAC	•	
	TRM	SAVE	Save contents of register
	TOA	'240	Run, Snapshot command
	OCW	C/U	Output command
	IDW	C/U	Input most-significant byte
	BNZ	*-1	Possible wait
	TAI		Store most-significant byte in I register
	IDW	C/U	Input middle byte
	BNZ	*-1	Possible wait (needed if other units on channel)
	TAJ		Store middle byte in J register
	IDW	C/U	Input least-significant byte
	BNZ	*-1	Possible wait (needed if other units on channel)
	TAK		Store least-significant byte in K register
	TIA		Restore most-significant byte in A register
	LLA	8	Shift over 8 bits
	TJB		OR in middle byte into A register
	LLA	8	Shift over 8 bits
	TKB		OR in least-significant byte into A register
	TAM	TIME	Store whole word of time for later use
	TMR	SAVE	Restore registers
	BUC*	SNSH	Exit
SAVE	BLOK	5	Register save area
TIME	DATA	0	Register save area

Selection Sampling

Selection Sampling is included as a feature of the Real Time Clock for the convenience of the programmer. Since the programmer would normally want to keep command word bits 3-0 constant while he uses bits 7-5, bits 3-0 are sampled only when command word bit 4 = 1.

FIRMWARE BOOTSTRAPS

The firmware bootstrap automatically stores in memory a loader program that permits a more complex program to be stored. Any program can be loaded as long as it is in bootstrap format; however, the most common application is to load a loader program which allows other programs, operating systems, diagnostics, or other data to be stored in selected memory locations.

Sources available for transferring a program to memory via a selected peripheral device include disc, card reader (word mode), card reader (block mode), and magnetic tape. The operation of the bootstrap is implemented at the MAP Terminal.

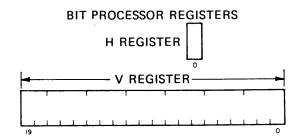
BIT PROCESSOR

General Description

The bit processor consists of the single-bit H Register, a 20-bit V Register (base register), and the associated control logic. The bit processor provides the capability to selectively change, store, or test a bit from memory.

Bit Processor Registers

Two registers are associated with the bit processor feature. A single-bit element, the H Register, retains the bit selected for use in the operation. The 20-bit V Register is employed to store a base address that is, in turn, used to define a memory location from which the designated bit will be retrieved. The V Register stores an 18-bit base address in the Compatibility Mode, or a 20-bit base address in the Address Extension Mode. Both the H and V Registers are directly programmable via the special group of bit processor instructions. Provision is made via the MAP Terminal for entry and display of the bit processor registers.



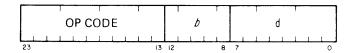
Operational Description

The V Register is loaded with a base address which specifies a memory location to be manipulated. This is accomplished by transferring an 18-bit (Compatibility Mode) or 20-bit (Address Extension Mode) memory address from the K Register. The instruction word further defines the memory location, the specific bit, and the operation to be performed.

After the operation is performed on the selected bit, the results are displayed in the Condition Register.

Program Control

Two types of instructions are associated with bit processor operations. The first (shown below) specifies a displacement (bits 7-0) to be added to the base address (V Register contents) to specify the location to be accessed. Bits 12-8 (binary coded) are used to select a specific bit to be used in the operation. The op code is defined in bits 23-13.



The second word format is used for bit movement or transfers where a specific bit from memory is not required. Bits 23-12 contain the op code; the remaining bits are undefined.



Bit Processor Instruction Set

The bit processor (Boolean function) group of instructions provides for logical manipulation and interrogation of a specified bit selected from an effective memory address or the H Register. The bit processor instructions are described in Section VII of this manual.

STALL ALARM

The stall alarm is enabled and disabled by the key switch on the switch panel. When the key switch is in the PANEL LOCK position, the stall alarm is enabled. It is disabled in the other two key switch positions. When the stall alarm is disabled, normal CPU operations take place. Once the stall alarm is enabled, a 128-cycle counter is activated whenever certain instructions are executed or certain operating conditions are encountered. Operation of the counter is program dependent. The counter is incremented once each active CPU cycle until the specified instruction or condition is removed. If the instruction or condition is still present after a minimum of 128 CPU cycles, an executive trap interrupt is generated at Level 5 of Group 0.

The following instructions and/or CPU conditions will activate the stall alarm counter.

Input Address Word (IAW)

Input Data Word (IDW)

Input Status Word (ISW)

Input Parameter Word (IPW)

Output Address Mand (O A)

Output Address Word (OAW)

Output Command Word (OCW)

Output Data Word (ODW)

Transfer Double to Source and Destination Registers (TDS)

Transfer Source and Destination Registers to Double (TSD)

Transfer A to 1 Virtual Address Register (TAR)

Transfer Double to 2 Virtual Address Registers (TDR)

Transfer 2 Virtual Address Registers to Double (TRD)

Transfer Double to Paging Limit Registers (TDP)

Transfer Paging Limit Registers to Double (TPD)

Transfer Usage Base Register and Demand Page

Register to Double (TUD)

Transfer E to Usage Base Register (TEU)

Query Virtual Usage Register (QUR)

Query Not-Modified Register (QNR)

Release Operand Mode (ROM)

Release User Mode (RUM)

Unitarily Arm Group 1 Interrupts (UA1)

Unitarily Arm Group 2 Interrupts (UA2)

Unitarily Arm Group 3 Interrupts (UA3)

Unitarily Disarm Group 1 Interrupts (UD1)

Unitarily Disarm Group 2 Interrupts (UD2)

Unitarily Disarm Group 3 Interrupts (UD3)

Unitarily Enable Group 1 Interrupts (UE1)

Unitarily Enable Group 2 Interrupts (UE2)

Unitarily Enable Group 3 Interrupts (UE3)

onitally Enable Group 5 interrupts (OES

Unitarily Inhibit Group 1 Interrupts (UI1)

Unitarily Inhibit Group 2 Interrupts (UI2)

Unitarily Inhibit Group 3 Interrupts (UI3)

Transfer Double to Group 1 (TD1)

Transfer Double to Group 2 (TD2)

Transfer Double to Group 3 (TD3)

Transfer Double to Group 1 (TD4)

Transfer Double to Group 2 (TD5)

Transfer Double to Group 3 (TD6)

Update Stack Pointer (USP)

Branch and Save Return - Long (BSL)

Branch and Save Extended (BSX)

Hold External Interrupts (HXI)

Hold Interrupts and Transfer I to Memory (HTI)

Hold Interrupts and Transfer J to Memory (HTJ)

Hold Interrupts and Transfer K to Memory (HTK)

Execute Memory (EXM)

Release External Interrupts (RXI)

Transfer Registers to Memory (TRM)

Transfer Memory to Register (TMR)

Branch and Reset Interrupt Long (BRL)

A halt condition

An indirect memory cycle

Each of the preceding instructions or conditions prohibit the recognition of external interrupts for a period of one cycle following completion of the instruction. Executing a series of these instructions sequentially will lock out external interrupts for the entire series. Multi-level indirect addressing can produce a similar effect, since the instruction must satisfy all address references before completion. (Interrupts occur only on instruction boundaries.) A halt condition — whether as a result of programmed halt or operator action — also prohibits external interrupt recognition by the CPU.

If a power failure occurs, the stall alarm becomes disabled. However, when power is restored, the stall alarm is re-enabled and operations continue in a normal routine.

With the exception of an EXM instruction or an indirect cycle, the monitored operation is terminated. An EXM chain (where an EXM instruction references another EXM which, in turn, specifies a third, etc.) has the same overall effect as an indirect chain in that all references must be completed before the sequence is complete. Therefore, if an EXM or indirect cycle is in process when the executive trap is generated, the stall alarm logic automatically terminates the sequence. If a block I/O channel is transferring data into memory when the executive trap interrupt is generated, the current cycle is completed before termination occurs and the trap takes control. If a halt condition is in effect when the executive trap interrupt is generated, the stall alarm logic automatically forces the CPU into a run mode.

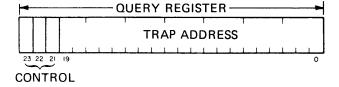
ADDRESS TRAP

General Description

The address trap queries each referenced memory address and compares it to the address preset in the Query Register. A comparison between the reference and preset address causes an executive trap interrupt to be generated. Hardware includes a register, a 20-bit comparator, an interrupt trigger circuit, and associated control logic.

Query Register

A 23-bit address Query Register is supplied with the address trap. Bit positions 19 through 0 contain the trap address. Bits 23 through 21 are the address trap control bits. When an address is reached in program that coincides with the address stored in the Query Register, an interrupt is generated. The Query Register may be loaded under program control or via the MAP Terminal.



Operational Description

The Query Register is loaded with the Transfer Memory to Query Register (TMQ) instruction. This instruction transfers the contents of the selected memory location to the Query Register. The 20 least-significant bits, representing the trap address of the memory word, are loaded into bit positions 19-0 of the Query Register. Memory word bit 20 is not used, and bits 23-21 of the

memory word are loaded into bit positions 23 through 21 of the Query Register. These three bits determine the mode of operation to be performed and have functions as follows:

Bit 23	=	ONE	Disable Address Trap
Bit 23	=	ZERO	Enable Address Trap
Bit 22	=	ONE	Trap only on write
Bit 22	=	ZERO	Trap each time selected address is referenced
Bit 21	=	ONE	Trap only during User Mode
Bit 21	=	ZERO	Trap only during Monitor Mode

The address trap is enabled or disabled with bit 23 of the Query Register. The address trap is enabled when bit 23 is reset, or ZERO. Each time a referenced memory address corresponds with the address stored in the Query Register, an executive trap interrupt at Group 0, Level 4 is generated to inform the CPU. When the trap occurs, the instruction in process is allowed to complete execution. Since up to four instructions may be prefetched from memory, an address trap interrupt caused by an instruction prefetch may occur during an instruction preceding the one causing the trap. When bit 23 is set (ONE), the address trap is disabled. Disabling the trap inhibits the executive trap interrupt.

Additional control of the address trap is provided with bits 22 and 21. With the trap enabled and bit 22 set (ONE), the executive trap interrupt is generated when a write operation is made to the referenced location. If bit 22 is reset (ZERO), the interrupt is triggered whenever the referenced location is accessed. With bit 21 set (ONE), the address trap is enabled during the User Mode of operation; if bit 21 is reset, the trap is enabled during the Monitor Mode. The memory address is taken from the CPU at a point prior to the address translation so that logical addresses are subject to the provisions of the trap.

Memory addresses that result from DMA operations by block I/O channels are not affected by the address trap.

Program Control

With the Query Register loaded and the address trap enabled, an interrupt is generated (in accordance with the control bit settings) each time a reference is made to the memory location corresponding to the address stored in the Query Register. If it is desired that a reference to the selected memory location be recognized only once, a second TMQ instruction should be executed following the first interrupt to set bit 23 of the Query Register to a ONE. This disables the address trap.

SECTION III MEMORY SYSTEM

GENERAL DESCRIPTION

Harris 800 systems are configured with main memory, cache memory and, optionally, extended memory and/or shared memory. Data and addresses gated to the system buses are available to all memory units. Maximum memory system capacity is three megabytes. Error detection and correction circuits are provided with each memory module.

Data transfers are over a 48-bit, asynchronous, bidirectional system data bus. Other buses provided include a 22-bit system address bus and a system control bus. All functional elements in the computer system communicate with each other through the system buses. The asynchronous bus system allows each system element to function at its own rate, independently of the other system elements. For example, concurrent direct memory access I/O transfer and CPU instruction execution. All buses are located on the backplane which is common to all boards in the system. This interconnection scheme eliminates the need for discrete wiring between the various boards in the system.

Transfer of data between the CPU and memory is over 24 or 48 of the data bus lines. CPU and Programmed Input Output Channel (PIOC) data transfers use 8 of the data bus lines. Data transfers between memory and the Integral Block Channel (IBC), External Block Channel (XBC), and Direct Memory Access Communication Processor (DMACP-8) is via 24 data bus lines. Buffered Block Channel (BBC) data transfers to and from memory occur on 24 or 48 lines. All block channels, once initialized, can perform blocked data transfers between memory and the peripheral device without CPU intervention.

MEMORY MODULES

Main memory, extended memory, and shared memory consist of all semiconductor memory modules. The following paragraphs describe the memory modules used, and the various modes of operation.

64K MOS Memory Module

The basic storage element of the semiconductor memory module is an N-channel metal oxide semiconductor (MOS) random access memory (RAM). A dynamic device, the RAM requires a periodic rewrite, or refresh, cycle to retain the stored data. It is also volatile — its data content

is lost when power is removed from the device. As in all semiconductor memories, the RAM has a non-destructive readout as opposed to a magnetic core memory which has a destructive readout. In addition to the RAM storage elements, each semiconductor memory module contains an address register, a memory data register, timing and control circuits, and data error correction circuits.

Memory module capacity is 192K bytes. Modules are addressable as single words of 29 bits, where five of the bits represent the error correction code. They are also addressable as double words of 58 bits to provide a double-word transfer capability. In this case, ten bits (five per word) represent the error correction bits.

The 64K MOS memory module has a cycle time of 400 nanoseconds and a normal access time of 290 nanoseconds. Fast access time is 45 nanoseconds.

Operating modes of the memory modules include the Read Mode, Write Mode, and Power Fail Refresh Mode. In either Read or Write Mode, a double word of 48 bits or a single word of 24 bits may be selected.

Read and Write Operations

A memory module always operates on two, 24-bit words at a time. These two words have the same address, except for the least significant address bit which defines the "even word" or the "odd word". If the specified word is at location 00 (even word), for example, the words at locations 00 and 01 are accessed simultaneously. If location 01 contains the specified word, the same two locations are accessed.

A read operation causes data in the location specified by the address on the system address bus to be transferred from memory to the memory data register. A single- or double-word transfer is then made from the data register to the system data bus. If a single-word read operation is specified, two words (even and odd addresses) are retrieved and loaded into the memory data register. Then, according to the address, the even or odd word is gated to the bus. If the addressed word contains an error, the parity error signal is asserted. A double-word read operation places the addressed words into the data register and onto the data bus. If an error is detected in either word, the parity error signal is asserted.

On a write to memory operation, data on the system data bus is loaded into the memory data register. Data is then transferred from the register to the location in memory specified by the address bits on the system address bus. Single or double words may be stored in memory during a write operation. For a single-word write operation, memory accepts the word from the system data bus and places it into the appropriate half of the data register according to the least significant address bit (even or odd). It calculates the error correction bit(s) and writes the word into memory. Simultaneously, the other word is read from memory and is placed into the data register. Thus, the other word is available for fast access if the next access is a read to the same address. In a double-word write operation, both the odd and even single words on the data bus are loaded into the data register. The error detection bit(s) for each word is calculated and then a write operation is performed to store the two words into the addressed location.

Fast Access Operation

Each memory module has a 48-bit memory data register, termed a Content Addressable Buffer (CAB), to improve system performance by reducing the effective cycle time of the computer. Each memory access fetches and loads the two, 24-bit words into the CAB. When the CPU requests a word from memory, a memory access is performed and the word is transferred over the system data bus to the CPU. However, if the CPU requires the next sequential word, it is transferred from the CAB to the CPU without requiring a second memory access. The CAB significantly reduces the fetch and execute time for sequential words.

MAIN MEMORY

Main memory is configured with 64K MOS semiconductor modules. Thus, minimum main memory size is 192K bytes which can be expanded in 192K byte increments up to 3M bytes.

EXTENDED MEMORY UNIT

Main memory can be expanded to the system maximum capacity of 3M bytes by including an Extended Memory Unit. This unit is housed in a separate cabinet which can be configured with up-to-16, 192K byte memory modules. CPU access is via port and interface modules supplied with the unit.

When used to expand main memory, the Extended Memory Unit is considered to be an extension of main memory so that addressing is continuous and uninterrupted. Since only one port is used in an extended memory system, no execution time is lost because of

contention between CPUs for the same memory. A loss in instruction execution time is incurred when accessing extended memory.

SHARED MEMORY UNIT

General Description

A Shared Memory Unit may be configured with up to four chassis, each of which can contain up to six memory ports. Each chassis may contain up to 3M bytes of memory. Ports may be connected to CPUs or to either types of devices. For Harris 800 CPUs, main memory plus shared memory may not exceed 3M bytes. A single Harris 800 CPU may be interfaced with up to four Shared Memory Units. Total addressable memory available to each CPU is 3M bytes.

Shared memory is designed as a six-port, asynchronous, ring priority access system. Access through the ports is nonsimultaneous. The ring priority system which services cycle requests uses a fixed rank priority for granting memory cycles to pending requests. However, no port is granted a second cycle until all previously received requests have been serviced. Priority is determined by the physical location of the port board in the shared memory chassis.

Port boards are available with a port index for use with CPUs having a cache memory system. The port index is a duplicate of the cache index, and is used to record the addresses stored in cache that correspond to the portion of shared memory associated with the port. This insures that the cache index is updated to correspond with the current data stored in shared memory.

Programming Considerations

In shared memory systems, a loss of instruction execution time is incurred when accessing the shared memory portion of the memory system. If two or more ports request entry simultaneously, access time increases for the lower priority port(s) which must wait to access memory.

Semaphore Operation

Shared memory supports read and lock, and write and unlock functions. Upon receipt of a read and lock command, the port locks the addressed memory module by initiating a read cycle, and prohibits entry to all memory modules from any other port. The shared memory remains locked until it receives a write and unlock command from the same port that initiated the lock function. The lock and unlock sequence always occurs in pairs. During the lock interval, I/O access is inhibited. Four instructions, the Transfer Flag to Memory

(TFM), Transfer Zero to Memory (TZM), Flag Bit of Memory (FBM), and Zero Bit of Memory (ZBM) instructions, implement the lock and unlock functions. Shared memory performance is directly affected by the frequency of execution of these instructions, e.g., a program containing many TFM instructions will lock up shared memory for prolonged time periods, inhibiting entry by other ports.

ERROR CORRECTING AND REPORTING

Error Correction

Single bit error correction is provided by error correction circuits contained on each memory module. All one-bit errors are corrected by this circuit. Detection of a parity error does not halt the machine.

All write operations to memory will store either 24 or 48 bits of data and 5 bits of Hamming Code parity for each 24-bit data word. These parity bits are generated by the memory module whenever data is asserted for a write operation. All memory read operations regenerate the Hamming Code from the stored data bits and compare this with the stored Hamming Code. This comparison generates an address code that points to the bit in error, and the correction circuit corrects the error. The corrected data is stored in the memory data register and is written into the appropriate memory location. This corrected data may then be obtained from the memory data register by a read operation to the same address, with no other operation intervening.

Error Reporting

Memory errors are reported to the system by means of the priority interrupt structure. A register is provided which saves the physical memory address at which a parity error occurs. In addition, if memory fails to respond within a specified time interval, a hard error is reported and the address is saved.

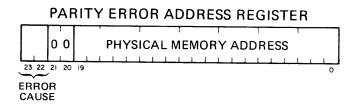
Parity Errors and Interrupts

When a parity error is generated as the result of a read operation, a retry is performed in which the location is read again. If no parity error is generated during the re-read operation, indicating that the error correction circuits corrected the error, the parity error is termed a "soft" parity error. If a parity error is asserted during a read operation and again on the re-read operation, indicating that the error was not corrected by the error correction circuits, the condition is referred to as a "hard" parity error.

Each time a hard parity error occurs, an executive trap interrupt is generated at Group 0, Level 1. This interrupt is also asserted if a memory time-out condition occurs. For each soft parity error generated, an external priority interrupt is triggered. A count of the number of hard and soft parity errors may be recorded by software. The operating system may then respond according to the type and number of errors recorded.

Parity Error Address Register

The 24-bit Parity Error Address Register (PEAR) retains the physical memory address associated with a memory parity error or memory time-out location. Two of the register bits provide for recording the type of operation causing the error. The contents of the PEAR may be retrieved with the Transfer Parity Error Address Register to A (TPA) instruction.



Bits 19 through 0 trap the address corresponding to either the parity error or memory time-out location. The time-out is defined as no response to a memory command for a period of 10 microseconds. Bits 21 and 20 are defined to be zeroes, and bits 23 and 22 record the type of operation causing the interrupt. Bits 23 and 22 are defined as follows:

23	22	Error Cause
0	0	parity error on instruction access
0	1	parity error on operand access
1	0	parity error on I/O access
1	1	time out — no memory response

In the event that a hard error is detected following a soft error before execution of the TPA instruction, the hard error address replaces the soft error address in the PEAR. If a second hard error or second soft error is detected prior to execution of the TPA instruction, no action is taken.

CACHE MEMORY

Operational Description

Cache memory enhances system performance by reducing the number of accesses made to main memory. When cache contains the data for a specified address, cache provides the data and no memory access is made. If the address and data are not stored in cache, a memory access is made to obtain the data. If the system software restricts memory accesses to small groups of locations over short periods of time, the time saving is considerable. When cache contains the requested address and provides the data, the condition is called a "hit". When the cache does not have the requested address and data in storage, and the data is provided by main memory, the condition is called a "miss". The ratio of hits to misses is dependent on the system software characteristics.

The cache data storage is configured as 1024 addresses by 48 bits and is divided into two, 512 double-word sections. One section, called the instruction section, stores only instructions. The other section, called the operand section, stores operands. A control line informs cache whether accessed data is an instruction or operand.

Addresses corresponding to the data stored in cache are stored in locations called the instruction index and the operand index. When the CPU accesses memory, the address is presented to the memory modules and the cache. The cache compares the address to the addresses stored in the instruction index and operand index. If a hit occurs, data is transferred from the cache to the data bus. When a miss occurs, data is read from main memory.

When the cache is full, a new instruction or operand is stored at a location determined by the address generated. The cache is updated whenever the CPU performs a write operation, or a read operation with a miss. I/O write operations with a hit invalidate, but do not update the cache location addressed.

Algorithm for Filling Cache

When a memory read operation is performed, the ten least significant bits of the EMA select a location in the cache. Refer to Figure 3-1. The contents of the instruction index and operand index are then compared to the ten most significant bits of the EMA. If there is a compare, the data is read from the cache. If the address does not compare, the cache location is purged. Data is then read from main memory and is loaded into the CPU and cache.

On a memory write operation, the ten least significant bits of the EMA select a location in the operand section of the cache. The contents of the selected location are purged and replaced by the CPU write data which is also loaded into main memory.

NOTES

- The cache monitors I/O memory write operations to eliminate stale data.
- 2) CPU memory write operations are monitored to eliminate stale data in the instruction storage section.
- 3) The index is updated with EMA19-10 each time the cache is loaded.

Programming Considerations

Cache is transparent to the programmer except when an address is generated outside the bounds of physical memory. If non-existing memory is addressed, the address and data are stored in cache. The cache then responds to the address, and data is read out of cache although no memory exists for the addressed location.

A search is performed by the operating system to determine the amount of memory available. An ODW instruction is executed to issue a command word to the cache to place it offline. Write and read operations are then performed to verify the amount of memory included with the system. At the conclusion of the memory search operation, the cache is placed online. All user programs are run with the cache online.

Cache stores 1024 instructions and 1024 operands. Although each section contains 512 double words, cache is updated a single word or double word at a time. Since the purpose of cache is to reduce the number of memory accesses, efficiency depends on cycling the instructions and operands stored in the cache. Better performance is achieved by looping programs within the 1K areas of cache. When running large programs straight through without looping within the 1K areas, performance may be degraded. A loss in time is associated with not obtaining a cache hit since the cache must search for requested addresses and data. In addition, a cache miss requires that a memory access be made to fetch the requested data.

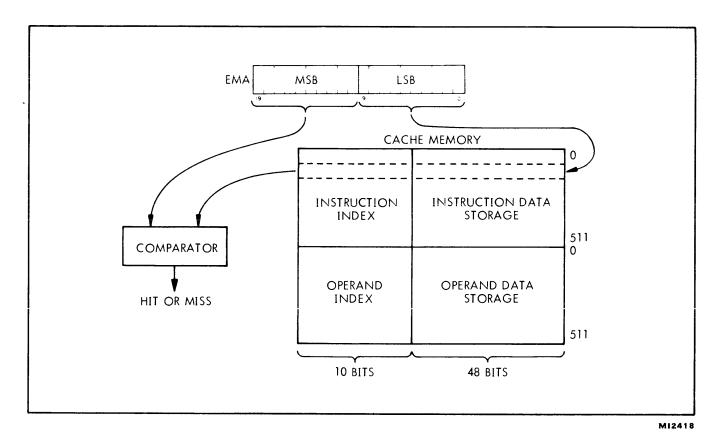


Figure 3-1. Cache Memory Operation

SECTION IV INPUT/OUTPUT CHANNELS

GENERAL DESCRIPTION

The computer system input/output (I/O) structure combines the characteristic economy of unit I/O systems with the speed of a channel I/O system. This configuration, in conjunction with the I/O instructions, permits maximum flexibility in I/O communications. The relationship between the CPU and the I/O structure is illustrated in Figure 4-1. The elements comprising the I/O structure are described in the following paragraphs.

The basic I/O structure allows single word data transfers between the Central Processing Unit (CPU) and a peripheral unit. It also allows I/O command and test operations to be program controlled. Block I/O channels may be used to control the transfer of blocks of data between the CPU and the peripheral units without program intervention.

The I/O structure involves communication (such as data transfers, addresses, and command status information) between the CPU and a peripheral unit by way of a channel. The CPU communicates with a specific channel and the channel, in turn, communicates with a peripheral unit. The I/O structure varies with CPU configurations to accommodate an applicable number of input/output channel (IOC) boards, all of which can be active concurrently. A channel can communicate with from one to sixteen peripheral units using standard I/O instructions. Only one peripheral unit per channel can be connected; however, all units can be active at any given time.

Communications between the I/O structure and the CPU may also be conducted on an interrupt basis. Logic in the channel and unit allows unit interrupts to be placed under program control and selectively enabled or disabled by executing the appropriate I/O instruction. An alternate method permits unit functions to be wired directly to the CPU priority interrupt structure and used as interrupt triggers.

The I/O interface is the link between each peripheral unit and its channel. The interface and its associated unit control facilities provide the physical means for connecting the peripheral device to the I/O structure and the logic capability that allows the unit to adapt the standard I/O controls to its specific requirements. The interface facilities and unit control logic are normally integrated with the peripheral unit. However, some controllers are available as options to the Integral Block Channel (IBC) and 8-bit Programmed Input/Output Channel (PIOC) boards.

BASIC I/O CONCEPTS

The I/O structure implements basic concepts to perform input/output operations between the CPU and a variety of channels and units. These basic concepts and their applicability are described in the following paragraphs.

Addressing

- a. Channel Addresses The I/O channels must each be addressed via a unique address contained in each I/O instruction. A channel is patched, or jumpered, to recognize its assigned address. The recognition of this code in an I/O instruction activates channel logic to execute the instruction. No other channel will respond.
- b. Unit Addresses Since a channel is capable of communicating with one or more unit controllers, any instructions involving the transfer of data, commands, or status must necessarily contain an address applicable to the unit involved. The unit address is contained in the format of the following instructions (reference Section VII for formats).

Output Command Word (OCW) - PIOC, IBC, XBC, DMACP-8, and BBC

Output Data Word (ODW) — PIOC, XBC, DMACP-8, and BBC

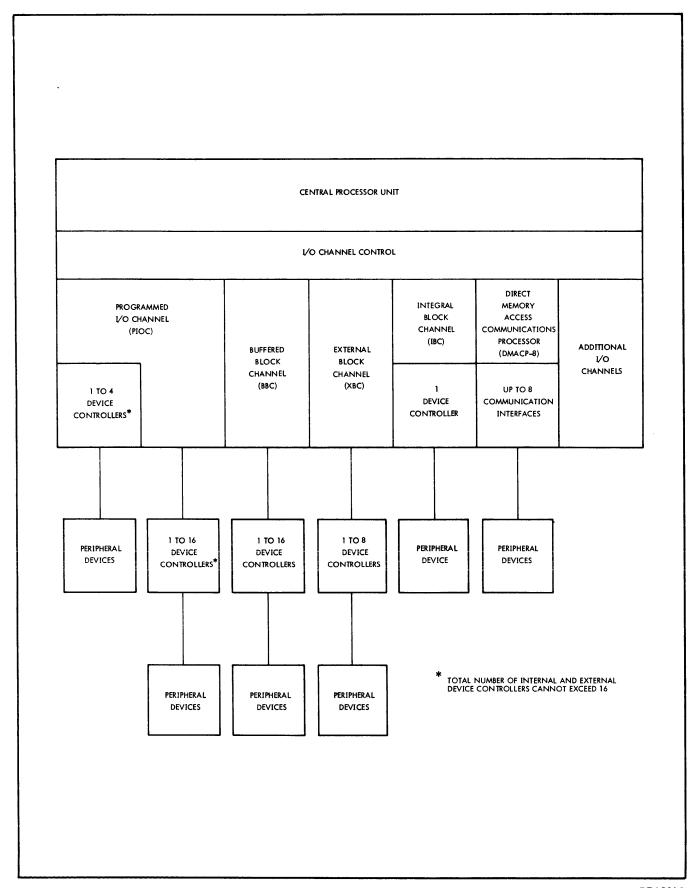
Input Data Word (IDW) — PIOC, DMACP-8, and BBC

Input Status Word (ISW) — PIOC, IBC, XBC, and DMACP-8, and BBC

Output Address Word (OAW) — IBC, XBC and DMACP-8

Input Address Word (IAW) — IBC, and DMACP-8

Input Parameter Word (IPW) — IBC, and DMACP-8



BD1641A

Figure 4-1. Computer I/O Structure Block Diagram

NOTE

The inclusion of unit addresses in the IBC, OAW, IAW, and IPW instructions has no transfer-to-unit control. The IBC contains the capability to concurrently store block transfer parameters for all unit controllers on its interface and the parameters must be addressed to reserved storage areas.

An instruction containing a unit address, sent to any channel other than the IBC is compared to the unit code of the previous instruction. If a non-compare is detected, the channel does not execute the ISW or IDW instruction. Instead, a disconnect/connect sequence is entered in order to connect the addressed unit. A non-compare detected during OCW and ODW instructions forces the disconnect/connect sequence also, but the channel loads the data/command, if not previously set busy, and holds the data/command until the addressed unit is "connected" to its interface. The transfer is then completed and the channel returns to a "not busy" condition.

Disconnect/Connect Sequences

Each IOC performs disconnect/connect sequences if the unit address contained in the instruction differs from the previously loaded address. In disconnect/connect sequences occurring during input instructions, the channel is prevented from setting the "ready" line to the CPU to verify that the instruction was executed. This requires a Branch on Not Zero (BNZ) instruction execution after each I/O instruction for a repetition of nonexecuted instructions. Timeout routines sequenced by the CPU may then detect channel/unit hangups and execute Input Status Word (ISW) instructions to pinpoint conditions.

The IBC is not equipped to sequence disconnect/connect operations; in this channel the unit is automatically connected to the channel for the purpose of instruction execution except during the time that data transfers are taking place.

Block I/O Channel Priority

Up to 31 channels may contend for memory access. The channels request memory cycles according to their assigned priorities. When a memory I/O cycle is granted, the channel with the highest active priority is enabled to access memory. All block I/O channels except the BBC are assigned a single priority. Each BBC is assigned two priorities to enable high and low priority data transfers to, and from, memory.

A programmable matrix is contained on each IOC capable of performing block transfer operations. The matrix is provided to resolve contention for simultaneous memory cycle requests.

The highest priority channel requesting a memory cycle inhibits any lower priority channel(s) from sensing a "memory cycle granted" signal from the CPU. A system should be configured to assign high speed devices a higher priority level than relatively lower speed devices. Also, no unused priority levels should appear between any two channel levels.

Two priorities are assigned to the BBC because of the 32-word buffer contained on the BBC, and because of buffers located on certain peripheral controllers which interface with the BBC. The dual priority request system prevents a high priority BBC from starving lower priority channels during buffer filling.

The BBC uses the higher of the two priorities only at the rate of I/O transfers. Additional requests to fill or empty the BBC buffer are made at the lower priority. In this way, no BBC is granted more cycles than it needs to maintain its I/O transfer rate at the expense of other channels. When a BBC is used in a system containing single priority channels, the high BBC priority should be set above the single priority channels, and the low BBC priority should be set below.

During input transfers, a BBC activates the high priority request when its buffer is full. For output transfers, the high priority request is activated when its buffer is empty. The particular BBC priority request activated is also determined by certain buffered peripheral controllers. The BBC activates the appropriate priority request based on signals received from controllers having this capability.

Synchronization (Handshake) Conditions

With few exceptions, all data and command sequences are synchronized via "handshake" operations. This convention ensures that the connected unit has received the command or data in output transfers or frees the unit to load new words in input transfers. If the unit is unable to accept the command/data, the channel sets itself busy and will honor no output transfer operation except for the OCW instruction in which "Override" is specified. The normal handshake function is modified in XBC and IBC operations and is described following the conventional handshake functions.

Output Transfer Synchronization

The output transfer handshakes are performed in OCW/ODW single-word transfer operations and in output block transfers of block I/O channels. In single-word transfers, if the channel is not busy executing a previous output instruction, the command/data is loaded into the channel's output buffer and the "Output Command Here" or "Output Data Here" line is raised to the unit. The channel sets itself busy to inhibit any new output transfer operations. When the unit gates the command/data into its own registers, it returns an "Accepted" signal. This signal resets the channel busy condition and the channel is free for a new transfer.

In block transfer sequences, the channel, having been previously initiated for output transfer operations, automatically sequences memory request operations. When the memory cycle is granted, the channel places the transfer address on line and loads the word from the specified address. The channel then raises the data transfer handshake line and, when the unit "accepts" the data, fetches another word from memory. The channel remains "busy" and the sequences continue until the transfer is completed or overriden.

Input Transfer Synchronization

A channel cannot execute an IDW instruction until it senses that the "Data Available" line from the unit has been set true. In normal operations the channel automatically transfers the input to the CPU and raises the "Data Accepted" handshake line. The unit drops "Data Available" to prepare a new word for transfer.

An input block transfer begins when a unit raises its "Data Available" line after the channel and unit have been commanded to the input mode. The channel loads the data into its input buffer and raises its "Data Accepted" line to the unit. The channel then sequences a memory cycle with the CPU to store the input word at the address specified by the Transfer Address Register (TAR). The channel will not honor any subsequent store requests until the memory cycle has been completed.

PIOC Synchronization

Programmed I/O transfers are performed by the PIOC via the OCW, ODW, IDW, and ISW instructions. The PIOC sets its "ready" line true if conditions allow it to execute an instruction from the CPU. If the unit cannot execute the I/O instruction, the channel sets itself busy. The busy condition may be removed by setting the Override bit in the OCW instruction. The PIOC performs handshake sequences with the unit controller in executing OCW, ODW, and IDW instructions. When a command is placed on line by an OCW instruction, the channel sets its "Command Data Here" line true. The controller signifies acceptance of the command by setting the "Output Data Accepted" line true. Both handshake signals are then dropped. The same sequence occurs for ODW instruction except that the channel sets its "Output Data Here" line true. The controller uses the "Output Data Accepted" signal to acknowledge receipt of the data and both the channel and controller then drop their respective handshake lines.

To perform IDW instructions, the controller signifies that it has data available by setting its "Data Available From Unit" line true. When the channel has passed the data to the CPU, the channel sets the "Data Accepted to Unit" line true. The channel and controller then drop the handshake lines.

No handshake is sequenced when the ISW instruction is executed since status information from the unit is always on line.

XBC Synchronization

The block transfer sequence control is under the control of the external units in XBC applications. The unit may be commanded to the block-transfer mode via an OCW instruction and may require parameter inputs but, once initiated, the device controls the transfers. In executing the OCW instruction the channel uses the conventional "Command Data Here" handshake signal and the unit returns "Accepted" to signal loading of the command. If required by the unit, the channel executes an OAW instruction to provide the Transfer Address (TA) to the unit. The channel raises "Address Word Here" which signals the unit to "accept" the address. This may be followed by an ODW instruction in which the word count is sent to the unit. The channel raises "Word Count Here" which the unit "accepts."

Data transfers to/from memory begin when the unit sets a priority-structured "Data Transfer Request" line to the channel. If the channel is not busy executing an instruction or servicing a higher-priority request, the channel raises its "Send" line. The unit responds via its "Ready" line. The unit then places the transfer address on line for channel storage and sets a transfer direction control line, the "In" line. If the "In" line is received in its true state, the channel loads the data from the unit, sets itself busy, and requests a memory cycle for storing the data in memory. When the "In" line is received set false, the channel requests a memory cycle for access purposes and, when the cycle is granted, the channel loads the data word from the address

furnished by the unit. The channel then pulses its "Output Data Here" line to load the data into the unit.

IBC Synchronization

The IBC is sequenced for block transfers via the units' "Data Transfer Request" lines. (See previous description for similar transfer capability.) The channel also specifies the transfer direction, but this is a reflection of the command word to the unit. In normal operation, channel parameters are loaded via the conventional block-transfer-initiate sequences into RAM locations reserved for units served by the channel. The unit, however, may be commanded to an external addressing mode in which it loads the unit's Transfer Address Register (TAR) and controls whether the TAR and/or Word Count Register (WCR) are incremented/decremented. respectively.

The IBC does not "shake hands" with the unit during command transfers; the command is automatically loaded by the unit controller since the channel "selects" the unit, bypassing the usual disconnect/connect sequence.

BBC Synchronization

For OCW/ODW instructions the handshake sequences are as previously described. For IDW/ISW instructions, the channel must have first established that a "status ready" condition exists. If this is true, the channel automatically transfers the status of the CPU during an ISW instruction; however, no handshake is sequenced with the unit. If the input data has been loaded by the channel, the data is transferred to the CPU during an IDW instruction, and the channel signals "acceptance".

The same handshake sequences occur during block transfers, but the channel is capable of 48-bit (double) word transfers to/from memory.

Timing

The PIOC and XBC depend solely on computer clock pulses for execution of single-word instructions or, where applicable, block-transfer operations. All remaining channels are synchronized to CPU timing for some sequences but may provide other sequences via independent internal timing.

Block Transfer Memory Access

Block I/O operations are controlled by the channel after it has been initiated under program control. The channel, therefore, accesses memory for read/write operations and must request memory cycles for this purpose. In memory transfers, the requested memory cycle is automatically granted unless the CPU is in an error correct cycle.

When a memory cycle is granted by memory, the control signal is permitted into the highest priority channel generating a cycle address. The "memory granted" signal activates the channel to load the word from memory (output transfer) or transfer a previously-loaded word from the unit to memory for storage (input transfer).

Block Transfer Parameters

The DMACP-8, BBC, and IBC are initiated for block-transfer operation via an OCW instruction. The command word itself must have bit 23 set to activate the block-transfer mode. These conditions activate the channel to sequence two simultaneous memory requests for parameters. The designated parameter words are illustrated in Figures 4-2 and 4-3.

BBC Parameter Words

The BBC parameter word formats are illustrated in Figure 4-2. In this channel the OAW instruction preceding the OCW used to initiate the block transfer control causes the first parameter address (PA) to be loaded into a parameter address register (PAR). This allows the parameters to be located in a separate "list" which may be located anywhere in memory. Each time the PAR is addressed for a parameter word, the channel increments the PAR for subsequent parameters.

The first parameter applicable to BBC operations contains a 16-bit word count and the most-significant 8 bits contain a "Skip Count". The skip count is significant only in block transfers designated for input and is loaded into the channel's skip count register (SCTR). This parameter controls the actual transfer operations in which data is loaded into memory. When a count is set into the SCTR, the channel provides load sequences to transfer the data from the unit to the channel but does not request memory cycles to load the data into memory. The SCTR is decremented with each word transferred and, when the counter has decremented to zero, the channel begins data transfers to memory based on the word count parameter.

The second parameter word in BBC applications contains a 20-bit TA. The two most-significant bits of PW2 are stored in the channel and specify four termination sequences that may be entered when the block transfer has been completed; these are:

a. Normal termination — the channel goes to a "not busy" state when the last data word has been transferred.

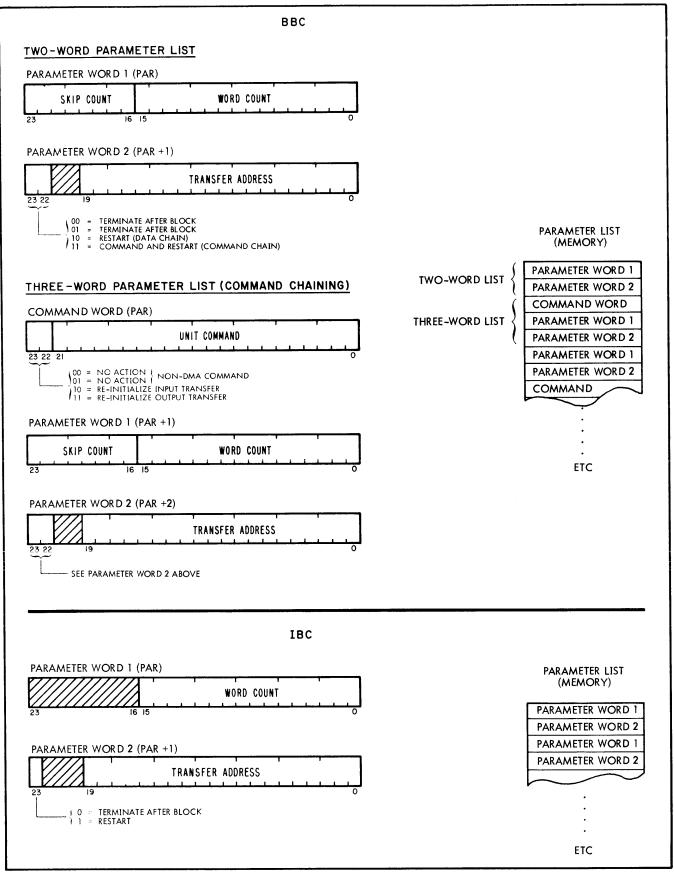


Figure 4-2. BBC and IBC Parameter Word Formats

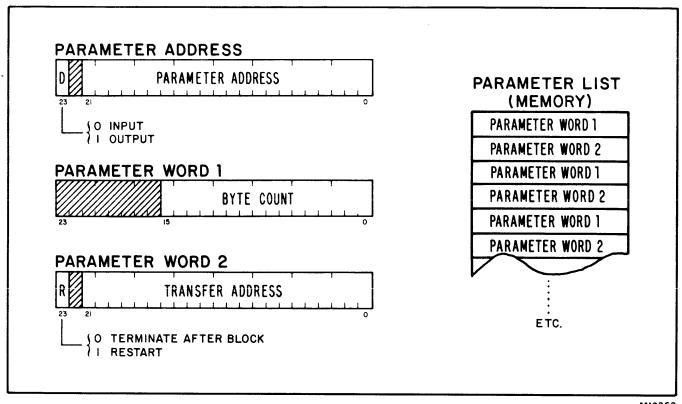


Figure 4-3. DMACP-8 Parameter Word Formats

M12362

- b. Data restart the channel goes into a re-initiate sequence to bring in two new parameters. The subsequent block transfer is as specified in the OCW initiating the previous block transfer.
- c. Chain command restart the channel goes into a re-initiate sequence in which a new command (from memory) is sent to the unit to change the transfer direction. As with the OCW initiating block mode operations, bit 23 of the command word must be set to command the initiate sequence. This is followed by bringing in PW1 and PW2 to set channel control action for the block of data to follow.
- d. Chain command terminate the channel goes into a re-initiate sequence in which a new command (from memory) is sent to the unit. If bit 23 of the command word is not set, the channel goes to a "not busy" state when the transfer sequence is completed.

XBC Parameter Words

The XBC does not contain circuits to store and control parameters. Likewise, the channel does not provide any restart actions. The parameters are controlled by the external device, but the device may require that the parameters be initially furnished from memory. In the

latter case, the channel is sequenced to execute an OAW instruction which transfers the TA to the unit. This is followed by an ODW instruction which sends the word count to the unit. After being initiated by the OCW command, each data transfer is sequenced and the unit itself furnishes the transfer address. The unit controls the word count and generates any operational interrupts.

IBC Parameter Words

The IBC is initiated to the block-transfer mode via the conventional OCW with bit 23 of the command word set. The IBC then enters the initiate sequence to load two parameter words (Figure 4-2). The first parameter word contains the word count of the subsequent block transfer. The second parameter word contains a 20-bit TA and the "restart" condition. The IBC does not provide chain command functions in a restart operation. But, since the IBC contains storage for parameter addresses, the channel may access PW1 and PW2 from a "list".

DMACP-8 Parameter Words

Each port has assigned to it a Parameter Address Register, a Byte Count Register, and a Transfer Address Register. These registers are located in the Parameter Stack located on the DMACP-8 board. Refer to Figure 4-3.

The Parameter Address Register contains the starting address in main memory of the next parameter list. The parameter list specifies the byte count to be placed in the Byte Count Register, and the transfer address to be placed in the Transfer Address Register. Along with the parameter address, a transfer direction bit (23) specifies whether the transfer is to be an output from main memory to the DMACP-8 (ONE), or an input from the DMACP-8 to main memory (ZERO).

Parameter Word 1, loaded into the Byte Count Register, contains in binary format the number of bytes of data to be transferred between main memory and the selected port. Maximum byte count per DMA sequence is 65,536.

Parameter Word 2, the transfer address, is loaded into the Transfer Address Register. The transfer address represents the location in main memory where the next data word (three bytes) is to be transferred. Each time a word is transferred, the TAR is incremented to point to the next memory address. An automatic restart function is provided to enable successive blocks of data to be transferred without CPU intervention. This is accomplished with bit 23 of the transfer address. If this bit is a ONE, the microprocessor will fetch a new byte count and transfer address from main memory as specified by the Parameter Address Register. A restart occurs when the existing count in the Byte Count Register reaches zero. When bit 23 is a ZERO, the restart function is disabled.

INPUT/OUTPUT INSTRUCTIONS

Execution of I/O instructions consists of the transfer of command (OCW), data (ODW and IDW), status (ISW), or address (OAW, IAW, IPW) words between the A Register the specified channel/unit combination. The channel/unit codes in each I/O instruction (excluding OAW, IAW, and IPW instructions in applicable block-transfer channels except the IBC) allow one channel to be selected and one of up-to-16 units to be connected to the channel. When an instruction to the same channel carries a different unit code, the previously-specified unit is disconnected and the new unit is connected automatically. During this disconnect/connect sequence, the channel is busy and does not respond to I/O instructions until the sequence is completed. If a channel is in the process of transferring commands or data to a unit, an ISW or IDW instruction addressed to a different unit on the same channel receives a busy indication.

Command and data words from the CPU are transferred to the channel output buffer and subsequently to the connected peripheral unit. Data and status words are retained in the input buffer of the selected unit and transferred to the A Register upon request (instruction) from the CPU. Address words are applicable only to those channels employing the block-control capability. (Refer to I/O instruction formats in Section VII for the following discussions.)

I/O Commands

The OCW instruction transfers a command word to the specified channel/unit combination. The command word bits specify the unit control function(s) to be performed and/or the I/O condition to be established. Following the execution of an OCW instruction, the channel remains busy until the command has been accepted by the addressed unit. Figure 4-4 shows the format for the OCW instruction.

If the channel is busy or not ready when addressed by the OCW instruction, the Condition Register is set to "Not Zero" to allow a programmed delay. The override function causes the channel to automatically perform a unit disconnect/connect sequence. This clears the channel of any other activity and allows the current instruction to assume control of the channel unconditionally upon termination of the disconnect/connect sequence.

All of the I/O channels execute the OCW instruction, but channel capabilites may require setting of the instruction contents as follows:

- a. Unit Addresses The IBC contains interface capability for up-to-two devices. The unit address must therefore be set in Unit Code bits 0 and 1. The unit addressing requirements for the XBC is contained in Unit Code bits 0-2. Unit Code 10g is the only valid code for the DMACP-8 channel. All of the remaining channels, having the capability to interface with up-to-16 units, utilize all of the Unit Code bits for addressing purposes.
- Channel Command Mode Bits 4 and 5 provide command control to set an I/O channel to one of four modes: Normal, Offline, Multiplex - Output Special Function, and Reset. The Normal mode specifies "normal" command functions. The Offline mode removes the units from the I/O channel interface, permitting a second computer and I/O channel to assume control over the units. The Multiplex-Output Special function mode serves two purposes. For the PIOC, the multiplex function allows a master unit to communicate with a slave unit without CPU intervention. The multiplex function is reset with a Master Clear, an OCW instruction with the Override bit set, or the Reset mode commanded. The Output Special Function is only applicable to the BBC. This function is used to output special channel

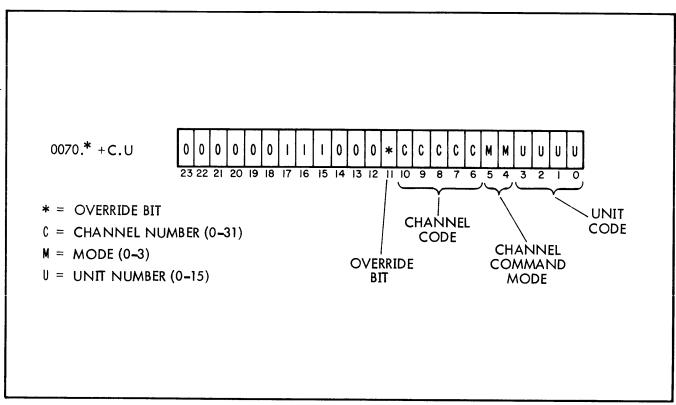


Figure 4-4. OCW Instruction Format

M12363A

functions to the BBC. These include the decrement transfer address, unit I/O parity check, internal turnaround, and unit master clear functions. The Reset mode allows a return to Normal mode operations from either the Offline or Multiplex modes.

The IBC, XBC and DMACP-8 do not respond to the mode control specifications of an OCW instruction (they thus always operate in the Normal mode).

c. Override Control — This OCW instruction control function is exercised in all I/O channels except the XBC. An OCW instruction with this bit set assumes immediate control of the channel/unit by forcing a disconnect/connect sequence.

I/O Status Word

The ISW instruction is used to test the operational status of the channel/unit. When a channel is addressed by the ISW instruction, a 24-bit status word is transferred to the A Register in the CPU. The quantity and significance of the status bits depends on the type of peripheral unit involved.

Units controlled by 8-bit interface channels (e.g., PIOC) furnish up to six unit-defined status bits which the channel sets into the least-significant bits of the input word. Channels with 24-bit unit interfaces (e.g., block controllers) may receive as many as 8 unit-defined status bits which are set into the 8 least-significant bits of the input word.

Channel status is reported in the most significant bits of the input word and reflect the current channel status. Not all of the available bits are used. Functions of the bits used to report channel status are as follows:

DMACP-8	None	
IBC	None	
PIOC	Bit 21	Multiplex
	Bit 22	Offline
XBC	Bit 22	Offline
ВВС	Bit 8	Unit Not Connected — The first ISW involving a unit code change is not accepted by the channel until the disconnect/connect sequence is completed. The disconnect/connect sequence provides a two microsecond time-out. If the unit does not acknowledge selection after this time-out, the channel accepts the ISW, regardless of unit acknowledgement, and reports the time-out in bit 8.
	Bit 9	Unit I/O Parity Error
	Bit 10	Buffer Count Not Zero — This bit is set if the channel data buffer contains data. Status reported by this bit is not valid if bit 11 of the status word is set.
	Bit 11	Transfer not complete — This bit is set when a block transfer is initialized and is reset automatically when the transfer terminates normally. Clearing the channel (e.g., OCW with Override set or Offline mode set) resets the bit.
	Bit 13	Decrement TAR
	Bit 14	Unit I/O Parity Check Output Special
	Bit 15	Internal Turnaround Function Reporting
	Bit 16	Unit Master Clear
	Bit 17	1)
	Bit 18	O Channel Identification Code = 5
	Bit 19	1 (Shanner racherheation code c
	Bit 20	0)
	Bit 22	Offline
	Bit 23	Channel Busy — This bit is set by one or more of the following conditions: Clearing the channel resets bit 23 for every condition except switched offline.
		a) Programmed offline
		b) Switched offline
		c) Transfer not complete
		d) Unit I/O parity error
		e) Executing an OCW sequence
		f) Executing an ODW sequence
		g) Executing a disconnect/connect sequence

Programmed Data Transfers

Input Data Word

The IDW instruction is a request from the CPU to a specific channel/unit combination for a data word. If data is available, the data word is transferred immediately to the A Register. If data is not available, the Condition Register is set to "Not Zero" to allow a programmed delay.

Normally, the 24-bit input data word contains a single data character. The actual number of data bits per character depends on the channel and unit involved in the transfer. For example, the console typewriter generates an 8-bit character and a card reader may generate a 12-bit character. In any case, the character is right-justified in the A Register with the unused bit positions set to ZEROS.

Assuming the data character contains no more than 12 bits, more than one character may be packed in the A Register through the use of the Merge feature. When a character Merge is employed, a logical OR is performed between the previous contents of the A Register and the new input data word. Without the Merge, the previous contents of A are destroyed upon transfer of a new character to A. An illustration of the character Merge technique, as compared to a normal IDW instruction, is shown in Figure 4-5.

The IDW instruction is executed by all I/O channels except the XBC and IBC.

Output Data Word

When an ODW instruction is executed, an 8- or 24-bit data word is transferred from the A Register to the specified channel. The data word is subsequently transferred from the channel to the unit that is currently connected. If the channel is busy or not ready to accept the data word, the Condition Register is set to "Not Zero" to allow a programmed delay. If the unit is not ready to accept the data from the channel, the data remains in the channel buffer.

As soon as the peripheral unit is able to accept the data from the channel, the channel-to-unit transfer is made, thereby freeing the channel buffer for another data (or command) word from the CPU.

The number of data bits accepted by the peripheral unit varies according to the type of unit involved. Some peripheral units are word-oriented and accept the entire 24-bit word. Others are character-oriented and accept only a specific number of bits per character.

The ODW instruction function in XBC operations serves the purpose of sending a word count parameter from the CPU

A Register to the addressed unit, if required by the unit. In subsequent block-transfer operations the unit controls the WC parameter. The IBC does not execute the ODW instruction.

Address Transfers

Three address-transfer instructions are executed by block-transfer channels for the purpose of channel or unit set-up for subsequent transfers (OAW) or for CPU checks of transfer progress (IAW and IPW). However, the PIOC board may execute the OAW instruction. The following discussions cover applicability and qualifications for the address-transfer instructions.

Output Address Word

The OAW instruction is executed by the DMACP-8, IBC, and BBC to set the starting address of parameters for block-transfer control. The XBC also executes the OAW instruction if a unit on its interface requires a TA starting address.

The DMACP-8, IBC, and BBC channels load their respective PAR during execution of the OAW instruction.

NOTE

In BBC execution of the OAW instruction the block transfer logic is cleared. Therefore, this instruction should not be programmed for execution until all block transfer operations are completed.

The XBC will not execute the OAW instruction if the channel is busy executing an output command or a data instruction. The instruction word must be addressed to the unit to which the TA parameter is intended. Therefore, a "programmed delay" should be programmed to facilitate instruction execution.

In IBC OAW execution the instruction word must be addressed to a unit controller contained on the channel board. The channel executes the instruction, writing the PA into a register reserved for the addressed unit.

Available for software interrupt purposes, an Interrupt Generator is located on the PIOC board to allow generating one-of-four possible interrupt pulses in response to an OAW instruction. The instruction is executed automatically by the addressed channel to provide one microsecond interrupt pulses which may be routed for use as interrupts in another CPU or in any peripheral unit.

EXAMPLE: THREE 8-BIT DATA CHARACTERS ARE TO BE PACKED IN THE A REGISTER. (a) NORMAL (WITHOUT MERGE) REGISTER A CODING COMMENTS BRING IN FIRST DATA CHARACTER 00000000000000000 IDW CU IDW CU BRING IN SECOND CHARACTER 000000000000000000 BRING IN THIRD CHARACTER 00000000000000000 IDW CU (b) MERGE **REGISTER A** CODING **COMMENTS** IDW CU 00000000000000000 BRING IN FIRST DATA CHARACTER LLA 8 SHIFT LEFT 8 PLACES 00000000 00000000 IDW* CU BRING IN SECOND CHARACTER 00000000 C2 AND MERGE C2 00000000 LLA 8 SHIFT LEFT 8 PLACES 16 15 C2 IDW* CU BRING IN THIRD CHARACTER Сз AND MERGE 16 15

M160-005

Figure 4-5. IDW Instruction; Data Character Formatting

The Interrupt Generator responds to the particular OAW instruction with the proper channel code. The four least-significant bits (3-0) of the A Register, during the OAW instruction, will trigger the pulse from the generator. The pulse remains at the "true" level for the 1 microsecond cycle and then is restored to the "false" state. There is no interaction between the generation of different numbered interrupts, but the generation of the same numbered interrupt is limited to not more than one per microsecond. There is no response to the mainframe C (condition) Register during the execution of the OAW, i.e., if the C Register was tested, it would indicate "not zero".

In summary, if an interrupt pulse is to be generated, the following coding could be applied:

TOA	B0B1B2B3	(Unitary bits; one for each
		interrupt pulse line.)
OAW	CU	

Input Address Word

The IAW instruction may be addressed to any of the block channels except the XBC. For IBC purposes the instruction word must be addressed to the channel and unit; otherwise the instruction is addressed only to the desired channel. In all applicable channels except the IBC the instruction is automatically executed during the current instruction cycle. The IBC executes the instruction only if it is not busy executing another instruction or transferring data. In all cases, the channel sets its "Ready" line to the CPU to clear the C Register. The address word is sent to the A Register and may be used as a check on transfer progress. The word represents the TA of the current transfer and is always 20 bits wide.

Input Parameter Word

This instruction is very similar to the IAW. The instruction is addressed only to those block channels capable of PA storage: DMACP-8, IBC, and BBC. The execution of the IPW instruction is identical to the IAW instruction.

INTERRUPT CONTROL

The OCW instruction may be used to selectively enable and disable two peripheral unit interrupts in PIOC board operations. The two interrupts are defined as Input and Output and are controlled by bits 2-0 of the command word. Table 4-1 illustrates the various bit configurations.

Table 4-1. Peripheral Unit Interrupt Control

Wo	Command Word Bit Configuration		Action
2	1	0*	
0	0	0	No Action.
0	0	1	No Action.
0	1	0	Disable Input (or Alternate) Interrupt
0	1	1	Enable Input (or Alternate) Interrupt
1	0	0	Disable Output (or Alternate) Interrupt
1	0	1	Enable Output (or Alternate) Interrupt
1	1	0	Disable Both Interrupts
1	1	1	Enable Both Interrupts

^{*}No significance to some units, i.e., the interrupts are unconditionally enabled by CW Bits 1 and/or 2.

The terms "input interrupt" and "output interrupt" are applicable only to peripheral units that are equipped with both input and output data handling facilities. Input-only devices may make use of the input interrupt and an alternate interrupt at the normal output level. Output only devices may make use of the output interrupt plus an alternate at the normal input level.

When the unit input interrupt has been previously enabled, an input interrupt signal will be generated when the input buffer in the unit is loaded (i.e., the same time the "Data Available" signal is generated). An I/O channel has no control over an input interrupt.

When the unit "output interrupt" has been previously enabled, an output interrupt signal may be generated by the channel for two sets of conditions based on a device-defined signal, "Enable Channel Buffer Empty Interrupt" (ECBEI). If the unit raises ECBEI to the channel, the output interrupt will be generated for a minimum of 325 nanoseconds if:

a. PIOC board;

- the channel has not been commanded to the Offline or Multiplex mode, and,
- the channel is not performing a disconnect/connect sequence, and.
- the channel's output buffer is not holding a command/data word for unit transfer purposes.

b. XBC and IBC boards;

These channels contain no output interrupt capability.

If the unit holds the ECBEI signal to the channel low, the output interrupt will be generated by the channel but the channel's output buffer condition (3, above) is ignored. Instead, the device-defined state of Status Bit 2 from the unit is allowed to set the output interrupt. The mode and manual conditions described for each type of channel above remain in effect.

The IBC generates a "word count complete" signal to the unit when the channel has loaded the final word, if no "restart" is specified. This signal, however, is under control of the unit for interrupt purposes.

A BBC can be patched to select either a "word count complete" interrupt or a "data chain complete" interrupt. The "word count complete" interrupt indicates that the channel has completed a single block transfer or any block within a chain. The "data chain complete" interrupt indicates that the channel has completed either a single block transfer or the last block of a data chain. For both of these interrupts, the interrupt is generated when an output transfer to the unit is completed, or an input transfer to memory is completed. The BBC also generates an "initialize" interrupt to indicate that the channel has performed an OAW instruction.

Two interrupts are generated by the DMACP-8 channel:

1) whenever a parity error is generated within the RAM located on the DMACP-8 board, and 2) whenever one of the ports requires service.

I/O CHANNEL JUMPER CONTROLS

The various I/O channels contain jumper provisions to perform a number of operational functions. The PIOC board's jumper capabilities is restricted to channel address selection. The block-transfer channels are also jumpered to encode a unique channel address, but those channels also contain a variety of other manually-activated functions. These functions are listed in Table 4-2 with I/O channel applicability specified.

Table 4-2. I/O Channels Jumper Control Capabilities

Function	PIOC	IBC	ХВС	DMACP-8	ввс
Permanent Offline/ Multiplexer mode	×				×
Channel code selection	×	x	x	X	X
Memory cycle priority		×	x	x	x
Unit selection		X			

X = function available

I/O CHANNEL OPERATIONAL SUMMARIES

The following paragraphs summarize single word and block-mode transfer capabilities of the various I/O channels interfacing with the computer. Included are program lists and suggestions. Refer to the paragraph describing input/output instructions for application to specific I/O channels.

Single-Word Instruction Execution

OCW/ODW

The channel, if not busy, loads a command or data word from the CPU A Register into its output buffer. The channel sets itself "busy" to inhibit any further instruction executions until it has completed the transfer to the addressed unit. In the event of a disconnect/connect sequence, the channel withholds the handshake until the addressed unit is "connected" to its interface. A BNZ instruction should be programmed to verify channel execution of the OCW instruction.

IDW

The channel executes this instruction if the channel is not busy executing an output transfer, is not involved in a disconnect/connect sequence, and the connected unit has signalled that data is available for transfer via its "Data Available" line. The BNZ execution performed by the CPU provides verification of transfer. The channel shakes hands with the connected unit and is ready for further instructions.

ISW

An I/O channel executes this instruction if the channel is not busy executing an output transfer and is "connected" to the addressed unit.

OAW

This instruction is addressed to block I/O channels (unit in XBC/IBC applications) for the purpose of transferring the address of the first word involved in control of a subsequent block data transfer. Channel loading of the output word from the CPU's A Register into the channel's PAR (DMACP-8, and BBC applications) is automatic. In XBC applications the instruction involves transferring a TA to the unit for subsequent control by the unit. The XBC must have gone to "not busy" prior to instruction time for execution. A programmed delay must therefore be executed by the CPU for verification of transfer. A handshake with the unit is performed in this instruction and the channel sets itself busy until the transfer-to-unit is completed. In IBC applications the addressed channel executes the instruction unless previously set busy via an instruction or data transfer sequence. This instruction causes an "initialize" interrupt to be generated by the BBC.

IAW/IPW

The IAW/IPW instructions are executed to transfer the contents of a block I/O channel's TAR/PAR to the CPU's A Register. The IPW and IAW instructions are not executed by XBC. The instructions, when applicable, are executed automatically by the BBC. The IBC is inhibited from executing the instruction if currently busy in an instruction or data transfer operation.

Block-Transfer Operations

All block I/O channels are initialized by computer control for block-transfer operations and proceed under self control or unit control to perform the transfer operations. The following paragraphs describe general performance of block transfers applicable to each channel. Refer to Figures 4-6 through 4-9 for simplified flow diagram of block-transfer operations.

BBC Block Transfers

The BBC is "set-up" via an OAW instruction and initiated via an OCW instruction with bit 23 of the unit command specifying the block transfer and bit 22 specifying the direction of transfer. During the OCW sequence the channel sets itself "busy" to all ODW, IDW, and OCW instructions (except an OCW specifying "Override"). The channel remains busy for the duration of transfers initiated by the OCW instruction. The channel automatically loads two parameter words (see Figure 4-2). If an output transfer has been specified, the channel sequences a memory request and specifes the location via its TAR. The channel increments the TAR, decrements its WCR, and loads the data word in its output buffer when the memory cycle is granted. The channel then "shakes hands" with the unit to complete the transfer. The channel then fetches another word for transfer. When the WCR has decremented to ZERO, the channel examines its "Restart" parameter (bit 23 of PW2) and either re-initiates itself for another block transfer or returns to an "idle" state, resetting its "busy" condition.

If an input transfer was specified via the OCW, the channel waits for the unit to signal data availability. The channel then loads the input data into it's input buffer and signals "accepted" to the unit to free it for the next word. The channel then requests a memory cycle, and when granted, places the TA and data on line to memory. The channel increments the TAR, decrements the WCR, and returns to sense the unit's "Data Available" line. This sequence continues until the WCR forces the restart sequence as described above. The BBC has the capability to specify an Output Special Function to decrement the TAR. This operation causes the input words from the unit to be written into memory in reverse order.

The BBC contains a Skip Count Register for added parameter control in input transfer operations and may enter an alternate "Restart" after a block of data has been transferred.

The output data transfers are sequenced in an identical fashion. The channel's capability to "Restart and Chain Command" allows the re-initiate sequence to access an additional parameter (in this application, a new command to the unit) to change transfer direction without program intervention. In this situation, the new command word initiates the channel in the same manner as did the original OCW instruction.

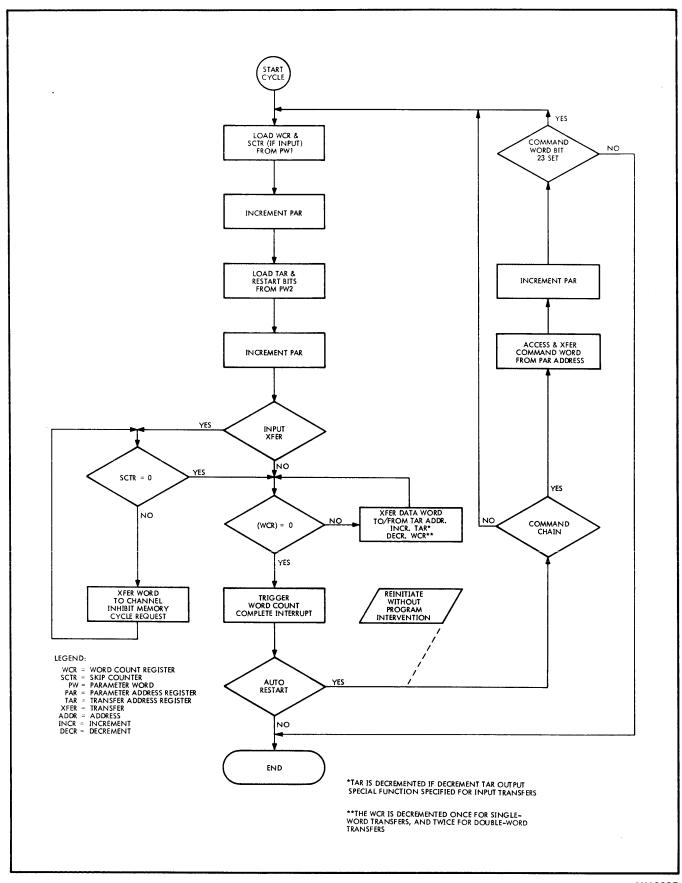
The Skip Counter affects only those transfers slated for memory. The skip count allows the channel to pass over unwanted data (sync codes, etc.) before actual data loading is sequenced. When the skip count parameter specifies a count, the channel sequences handshakes with the unit to unload the unit, but the channel does not request the memory cycles from the CPU to load the data into memory. The SCTR is decremented with each transfer, but the TAR and WCR remain unchanged. When the SCTR has decremented to ZERO, the channel begins loading data words into memory.

XBC Block Transfers

The XBC is normally initiated to block-transfer operations via an OCW instruction in which a command is transferred to the unit. If required, the OCW may have been preceded by OAW and/or ODW instructions to transfer TA and WC parameters to the unit. Once initiated, the channel is under the control of the unit for transfer purposes. When the unit signals a "Data Transfer Request" (DTR), the channel, if not previously set busy, sets itself busy and stores the TA from the unit. The unit specifies the transfer direction and, if an input transfer is specified, the channel "accepts" the data from the unit. The channel then requests a memory cycle and, when granted, transfers the data to memory, based on the TA furnished by the unit.

If the unit specifies an output transfer, the channel requests a memory cycle. When the cycle is granted, the channel places the TA on line to memory, loads the data from memory and performs a "Data Here"/"Accepted" handshake with the unit in which the data is transferred to memory.

The XBC "busy" condition is reset after each instruction or data transfer is accomplished. The unit controls the TA and WC parameters and generates any required interrupts.



M11822B

Figure 4-6. BBC Block Transfer Sequence; Simplified Flow Diagram

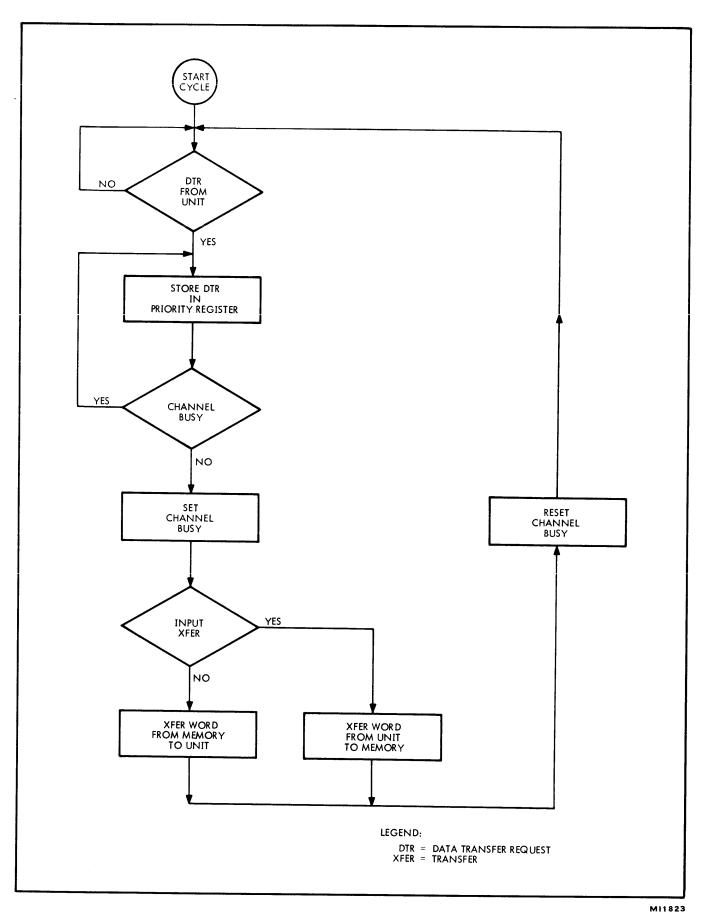


Figure 4-7. XBC Block Transfer Sequence; Simplified Flow Diagram

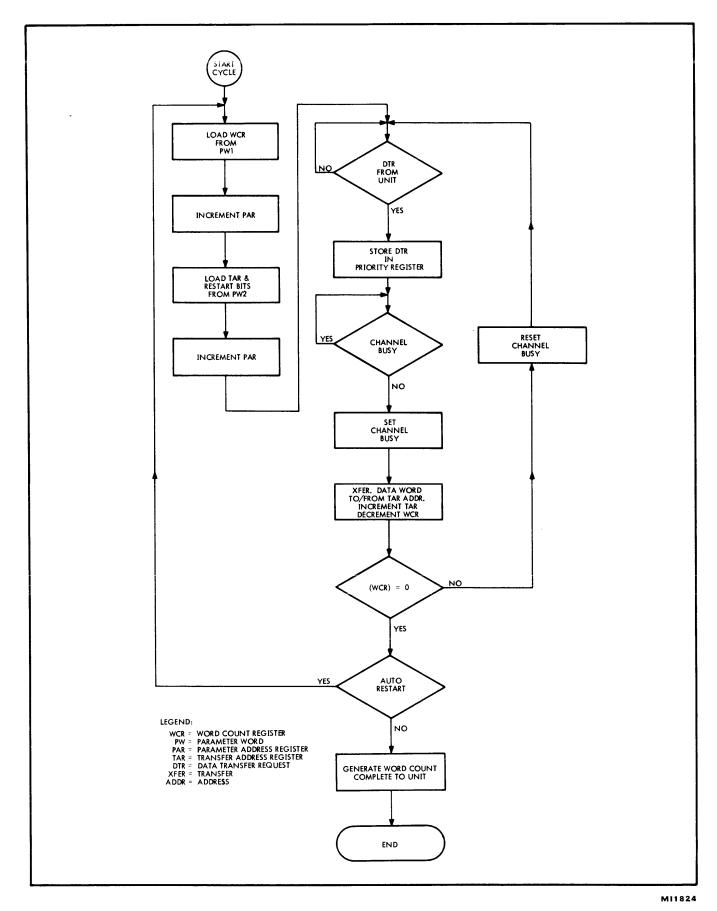


Figure 4-8. IBC Block Transfer Sequence; Simplified Flow Diagram

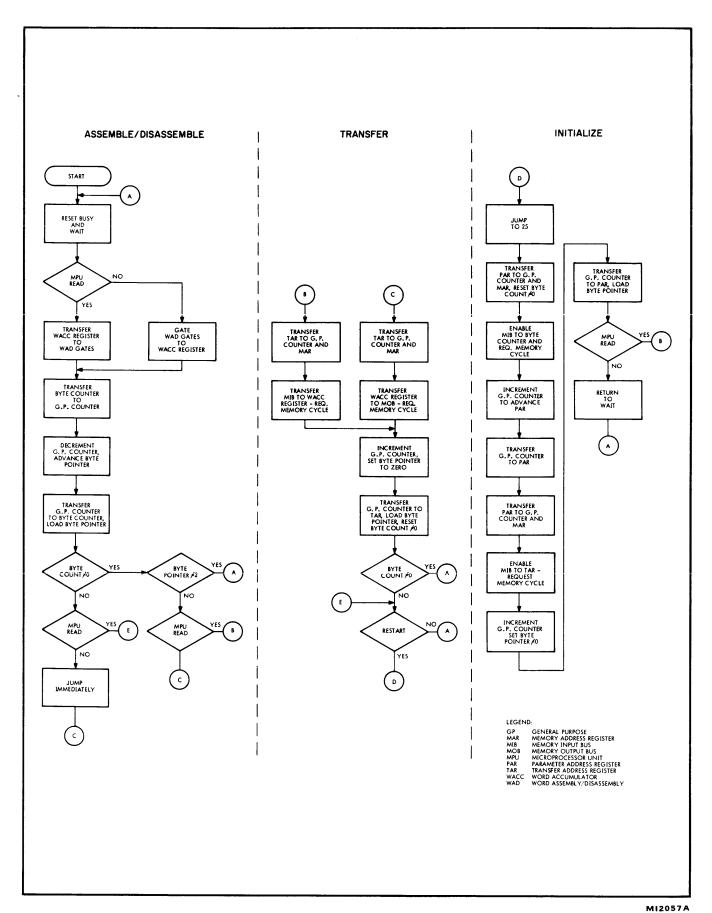


Figure 4-9. DMACP-8 Channel Block Transfer Sequence; Simplified Block Diagram

IBC Block Transfers

The IBC is set-up and initiated for block transfers via the OAW and OCW instructions, but the channel sets itself "not busy" after each instruction or data transfer. The channel may thus store the two parameter words for the self-contained unit controller (Figure 4-2).

Data transfers are sequenced by the channel based on the "Data Transfer Request" signal from the unit and the unit indicates the direction of transfer. Data transfers then proceed as described for XBC operations except as follows:

- a. the unit allows/inhibits TA and WC incrementing and decrementing by setting its "Block Mode" control line true/false (see External Addressing mode below).
- b. the unit does not furnish the TA parameter (except in the External Addressing mode).
- the channel/unit does not "shake hands" in output transfers.
- d. the channel generates "Word Count Complete" to the unit only, which then controls the interrupt to the CPU.

The IBC may enter an External Addressing mode by the unit presenting its DTR, "Address Here," and "Input" lines set to the channel. The address is then loaded into the TAR for the specified unit. The data presented with the next DTR is transferred into or out, as set, of the memory address of the unit's TAR. If the "Address Here" signal is not presented again to change the TAR, any further data transfers will use the same TAR address. This allows the use of a specified memory address as a register.

The maximum transfer rates for the IBC block transfer operations are determined by the card reader connected to the channel.

DMACP-8 Channel Block Transfers

DMA transfers between the DMACP-8 and memory are under control of the microprocessor and associated logic located on the DMACP-8 board. After a parameter address is sent with an OAW instruction, the CPU can command the microprocessor to perform a block transfer with an OCW instruction.

Data transfers are controlled by a sequencer and transfer control logic contained on the DMACP-8 board. Three main functions are performed by the transfer control logic; initialization for a DMA transfer, word assembly/disassembly, and the actual transfer. These functions are performed by three subroutines comprising a program which is stored in the sequencer PROM located on the DMACP-8.

The microprocessor starts a DMA initialize operation by accessing a special location in a RAM contained on the DMACP-8 board. Eight special locations in RAM are provided, one for each port. The DMA logic in the DMACP-8 fetches the byte count and transfer address from the RAM locations specified by the parameter address. If an output operation is specified, the first 24-bit data word is transferred to a Word Accumulator Register. The microprocessor then transfers data bytes between the word accumulator and a communications port until the byte count equals zero. A terminate interrupt is sent to the microprocessor at the completion of the operation. The microprocessor then generates an interrupt to the CPU to indicate that service is required.

Program Lists

The following program lists specify various software control functions for block-transfer I/O channels. Note the functional identity of the applicable channels.

IBC Applications

The following examples illustrate two different IBC applications.

Example 1: Simple, single buffer input.

	TOA	PA	Parameter Address
	OAW	С	Initialize TAR
	TMA	CW	Command Word
	OCW	CU	Initiate transfer
	BNZ	*-1	Delay if channel is busy
CW	DATA		Bit 23 and others as required by the I/O device
PA	DAC	n	Absolute Word Count
	DAC	BUFF	Address of Input Buffer
BUFF	BLOK	n	Reserve n words. Word n+1 is of no significance since the AR bit is not set.

Example 2: Multi-buffered output with automatic restart and buffer switching.

	TOA	PA1	Parameter Address 1
	OAW	С	Initialize TAR
	TMA	CW	Command Word
	OCW	CU	Initiate first transfer
	BNZ	*-1	Delay if channel is busy
	• • • •		
CW	DATA		Bits 23, 22, and others as required by the I/O device.
PA1	DAC	n	Word Count
	DAC*	BUF1	Address of buffer 1 and the ARF (*)
PA2	DAC	n	Word Count
	DAC	BUF2	Address of buffer 2
BUF1	BLOK	n	Reserve n words
	DAC	PA2	Automatic Reinitialization address for TAR, to switch buffers
BUF2	BLOK	n	Reserve n words
	DAC	PA1	Automatic reinitialization address for TAR, to switch buffers

NOTE Once this cycle is initiated it will continue, without program intervention, until a new command is received.

XBC Application

The following example illustrates an XBC application.

Only if Required	TOA OAW BNZ TOA OAW BNZ TMA ODW BNZ	INPAD CU *-1 OUTAD CU *-1 WC CU *-1	Set-up Input Buffer Address Start Output the Address to Channel/Unit Delay if Channel busy Set-up Output Buffer Address Start Output the Address to Channel/Unit Delay if Channel busy Set-up the required Word Count Output the WC to Channel/Unit Delay if Channel busy
INPAD	BLOK	100	Is the Starting Address of the Input Buffer that device may load data into.*
OUTAD WC	BLOK DATA	100 100	Is the Output Buffer that the device may read data from.* Number of Words to Transfer

^{*}The external device controls the addressing and interrupt requests to the XBC. The external device also controls the word count.

SECTION V PRIORITY INTERRUPT SYSTEM

GENERAL DESCRIPTION

The priority interrupt system provides added control over internal CPU operations and I/O functions, and immediate recognition of special external conditions on the basis of predetermined priority. Receipt and recognition of internal or external triggers allows the normal program flow to be diverted to interrupt service subroutines.

Four separate interrupt groups (0, 1, 2, and 3) are provided. Group 0 is reserved for internal CPU functions and is composed of eight executive trap levels. Groups 1, 2, and 3 are reserved for external interrupts. A maximum of 72 external interrupts are available.

INTERRUPT ORGANIZATION

Priority Conventions

All interrupt levels (both executive traps and external interrupts) are assigned a unique priority number. This assigned priority determines the order in which interrupts will be recognized and serviced. Interrupt levels descend in order of priority from Group 0, Level 0, to Group 3, Level 23. Group 0 has priority over Group 1; Level 0 has priority over Level 23.

Executive Traps (Group 0)

Each executive trap level is associated with a specific computer feature and is, therefore, permanently assigned. Each executive trap includes the associated internal interrupt level. Interrupt level assignments for the executive traps (Group 0) are listed below.

Level	Function
0	Power Down and Power Up
1	Hard Parity Error
2	Virtual Memory Page Fault
3	Instruction Trap
4	Address Trap
5	Stall Alarm
6	Interval Timer
7	SAU Overflow/Underflow

External Interrupts (Groups 1, 2, and 3)

A computer system includes interrupt logic and up to 72 individual external interrupt levels. Sixteen of these levels are located on the option board and represent Group 1, Levels 0 through 15. Thirty-two additional external levels are located on a priority interrupt expansion board. These represent Group 1, Levels 16 through 23, and Group 2, Levels 0 through 23. A second priority interrupt expansion board contains the Group 3, Levels 0 through 23 external interrupts. Priority assignments of the interrupt levels are determined by system requirements and are made to meet user's requirements.

Dedicated Memory Locations

Each interrupt level has a memory location dedicated for its exclusive use. This applies to both the executive traps (Group 0) and external interrupts (Groups 1, 2, and 3). Dedicated memory locations for the interrupt system are as follows:

Addresses (Octal)	Assignments (Respective)	
60-67	Executive Traps, Levels 0-7	
70-117	Group 1 Interrupts, Levels 0-23	
120-147	Group 2 Interrupts, Levels 0-23	
150-177	Group 3 Interrupts, Levels 0-23	

OPERATION AND CONTROL

Basic Operation

Figure 5-1 is a functional block diagram of the priority interrupt system. Both the executive traps and external interrupts are initiated by a trigger from their assigned functions. The primary operational difference between the two interrupt types is the method of control; executive traps are hardwired in an armed and enabled state, while external interrupts must be previously armed and enabled under program control before an interrupt trigger can be recognized and processed.

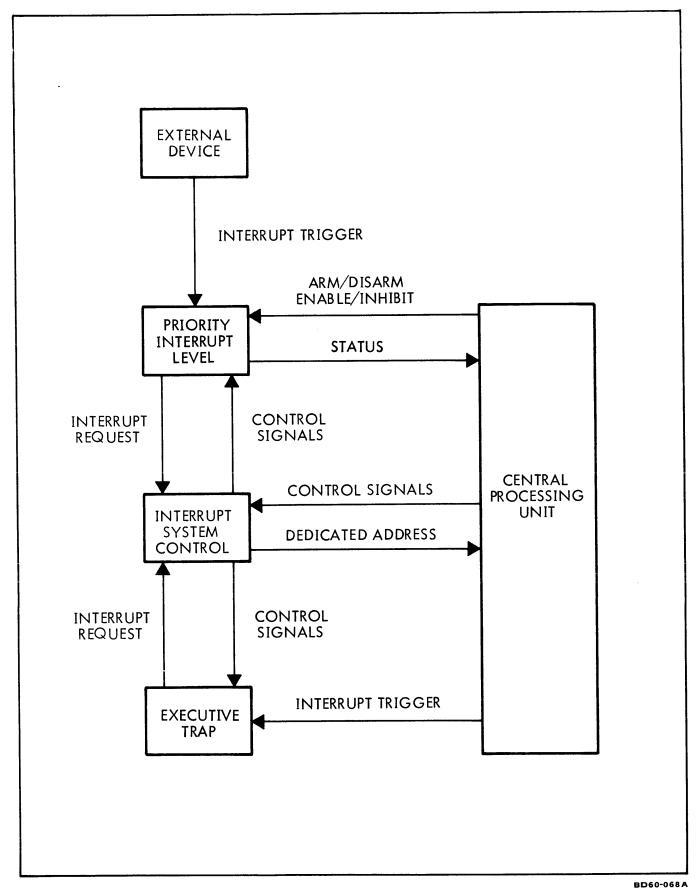


Figure 5-1. Functional Block Diagram, Priority Interrupt System

Executive Traps Control

Each executive trap interrupt is designed so as to become active immediately upon receipt of its associated internal trigger, provided no higher-priority level is active. Executive trap interrupt levels are permanently assigned to their associated function. Five executive traps are constantly armed and enabled, and no program control over the activation of these interrupts is provided. Three executive traps can be controlled by the Hold External Interrupts (HXI) and Release External Interrupts (RXI) instructions. These three include the Address Trap, Interval Timer, and SAU Overflow/Underflow executive traps.

External Interrupts Control

External interrupts are program-controlled and are not permanently assigned. Program control is afforded by several instructions. Individual levels can be selectively (unitarily) armed, disarmed, enabled, or inhibited under program control, or an entire group of interrupts can be simultaneously controlled. For a detailed description of all priority interrupt instructions, refer to the appropriate portion of Section VII in this manual.

Four 24-bit registers are associated with each external interrupt group. These registers may each be 8, 16, or 24 bits wide, depending on the number of interrupt levels within the group. As interrupt levels are added to the system, bits are added to each of the four registers in the group. The register bit positions correspond to the priority level assignments, i.e., bit 0 represents Level 0, bit 1 represents Level 1, etc. Control of the interrupt registers is accomplished by the following group of instructions.

Transfer Double to Group 1 (TD1)

Transfer Double to Group 2 (TD2)

Transfer Double to Group 3 (TD3)

Transfer Group 1 to Double (T1D)

Transfer Group 2 to Double (T2D)

Transfer Group 3 to Double (T3D)

Transfer Double to Group 1 (TD4) (software-triggered interrupt)

Transfer Double to Group 2 (TD5) (software-triggered interrupt)

Transfer Double to Group 3 (TD6) (software-triggered interrupt)

Transfer Group 1 to Double (T4D) (software interrupt status)

Transfer Group 2 to Double (T5D) (software interrupt status)

Transfer Group 3 to Double (T6D) (software interrupt status)

The armed/disarmed and enabled/inhibited states of each interrupt level are retained in the Arm/Disarm (A/D) and Enable/Inhibit (E/I) Registers, respectively. A TD1, TD2, or TD3 (Group 1, 2, or 3) instruction is used to selectively arm, disarm, enable, or inhibit individual interrupt levels within the group. Upon execution of a TD1, (TD2, or TD3) instruction, the contents of the E and A Registers are transferred, respectively, to the A/D and E/I Registers in Group 1 (Group 2, or 3). Transfers are performed in a bit-for-bit pattern. A ONE in a given bit position of the A/D Register will cause the corresponding interrupt level to be armed; a ZERO will disarm the level. An interrupt will be enabled or inhibited by a ONE or ZERO, respectively, in the corresponding bit position of the E/I Register.

An interrupt group's armed/disarmed and enabled/inhibited status may be determined under program control by the execution of a T1D, T2D or T3D (Group 1, 2, or 3) instruction. The contents of the A/D and E/I Registers are transferred to the E and A Registers, respectively. A/D and E/I Register contents are not affected by the transfer.

External interrupt triggers normally occur asynchronously with respect to CPU operation. However, interrupt triggers can be generated under program control by a TD4, TD5, or TD6 (Group 1, 2, or 3) instruction. The TD4, TD5, or TD6 instruction performs a logical OR between the contents of the E and A Registers and the Interrupt Request and Active Registers, respectively. Loading the Request Register with a ONE has the same effect as an external trigger at the corresponding interrupt level. When the Active Register is loaded with a ONE, the corresponding level will become active as long as no higher-level interrupt is active. The T4D, T5D, or T6D (Group 1, 2, or 3) instruction transfers the contents of the Request and Active Registers to the E and A Registers, respectively. The Request and Active Registers are not affected.

Figure 5-2 illustrates the control system for external interrupts. Each external interrupt operates in three distinct states: inactive, waiting, and active. In the inactive state, the level has not received an interrupt trigger. When a trigger is received, the armed/disarmed status determines whether the triggered interrupt will be placed in a waiting state or ignored. If the triggered interrupt is armed, it will be placed in the waiting state; if disarmed, it will be ignored.

If an interrupt is armed but inhibited (i.e., not enabled), it is held in the waiting state until such time as it is enabled under program control. Once enabled, the interrupt will become active as soon as the current instruction is completed, assuming that no higher level is active and that external interrupts are not being held (HXI instruction).

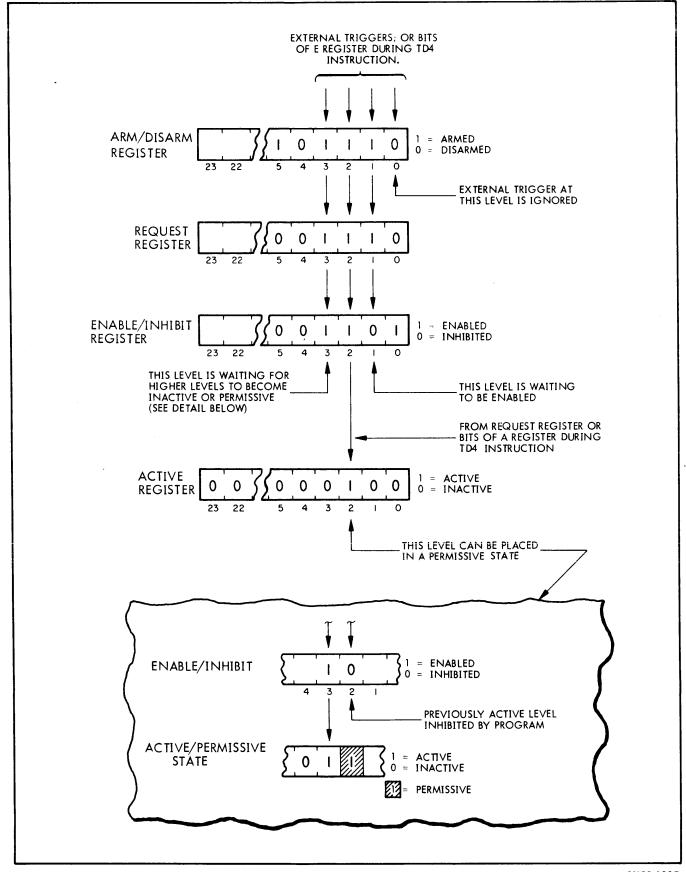


Figure 5-2. External Interrupt Control

Once an interrupt becomes active, it can be inhibited under program control (TD1, TD2, or TD3 instruction). This places the active level in an off-line mode or permissive state. The permissive state does not affect execution of the interrupt subroutine but enables lower priority armed and enabled interrupts to become active when triggered. For example, if active level two is inhibited by the program, waiting level three becomes active immediately. After level three is serviced, the processing of the level two subroutine is resumed until it is completed or another interrupt becomes active. Should another interrupt trigger be received by an interrupt that is in the permissive state, it will be saved and recognized when that level is returned to the on-line mode.

Hold and Release External Interrupts (HXI and RXI) instructions are employed to prohibit and restore the activation of any external interrupt (other than currently-active levels) regardless of that interrupt's armed/disarmed and enabled/inhibited states. Such a prohibition would ensure that another, lower-level, interrupt could complete its processing routine without interruption. This hold condition can only be released by an RXI instruction.

Should an interrupt occur during the execution of certain specified instructions, it will not be allowed to become active until the completion of the instruction following the specified instruction. The following instructions are included in this group.

Branch and Save Return Long (BSL)

Hold Interrupts and Transfer I to Memory (HTI)

Hold Interrupts and Transfer J to Memory (HTJ)

Hold Interrupts and Transfer K to Memory (HTK)

Release External Interrupts (RXI)

Execute Memory (EXM)

Transfer Memory to Registers (TMR)

Transfer Registers to Memory (TRM)

Update Stack Pointer (USP)

Transfer Double to Group 1 (TD1)

Transfer Double to Group 2 (TD2)

Transfer Double to Group 3 (TD3)

Transfer Double to Group 1 (TD4)

Transfer Double to Group 2 (TD5)

Transfer Double to Group 3 (TD6)

Unitarily Arm Group 1 Interrupts (UA1)

Unitarily Arm Group 2 Interrupts (UA2)

Unitarily Arm Group 3 Interrupts (UA3)

Unitarily Disarm Group 1 Interrupts (UD1)

Unitarily Disarm Group 2 Interrupts (UD2)

Unitarily Disarm Group 3 Interrupts (UD3)

Unitarily Enable Group 1 Interrupts (UE1)

Unitarily Enable Group 2 Interrupt (UE2)

Unitarily Enable Group 3 Interrupts (UE3)

Unitarily Inhibit Group 1 Interrupts (UI1)

Unitarily Inhibit Group 2 Interrupts (UI2)

Unitarily Inhibit Group 3 Interrupts (UI3)

Transfer Double to Source and Destination Registers (TDS)

Transfer Source and Destination Registers to Double (TSD)

Transfer A to 1 Virtual Address Register (TAR)

Transfer Double to 2 Virtual Address Registers (TDR)

Transfer 2 Virtual Address Registers to Double (TRD)

Transfer Double to Paging Limit Registers (TDP)

Transfer Paging Limit Registers to Double (TPD)

Transfer Usage Base Register and Demand Page Register to

Double (TUD)

Transfer E to Usage Base Register (TEU)

Query Virtual Usage Register (QUR)

Query Not-Modified Register (QNR)

Release Operand Mode (ROM)

Release User Mode (RUM)

INTERRUPT PROCESSING

Each external interrupt and executive trap level is assigned a unique memory location, as previously described. This location specifies an address at which to store certain system parameters. When an interrupt becomes active the contents of the C Register, program return address, and virtual memory mode of operation are saved. A branch is then made to the interrupt subroutine. At the conclusion of the subroutine, the C Register contents and virtual memory mode of operation are restored and a branch is made to the main program.

Interrupt processing is dependent upon the operational state of the CPU when the interrupt occurs. One procedure is used for Operational State Zero, while a second procedure applies to Operational States One and Three.

Operational State Zero Interrupt Processing

An interrupt, which is activated when the CPU is in Operational State Zero, generates an address and an instruction operation code. The address specifies the dedicated location and the operation code defines a pseudo (hardwired) Execute Memory (EXM) instruction. The address and EXM instruction are decoded and executed as a normal operation. This causes the instruction in the dedicated location to be executed as if it were the next instruction in the main program.

Although any instruction may be stored in an interrupt's dedicated memory location, the operation designed for subroutine entry is the Branch and Save Return Long (BSL) instruction. The BSL instruction is used to enter an interrupt subroutine because it provides a means of saving machine status and returning to the program location following that being executed at the time of the interrupt. When an interrupt is generated, the current instruction is

allowed to continue so the Program Counter can be advanced before interrupt processing begins. Figure 5-3 illustrates the sequence of events.

The BSL instruction records the paging mode (User or Monitor) in bit 20 of the effective memory address. Bit 20 is set to ONE if the CPU was in the User Mode when the interrupt occurred, or reset to ZERO if the Monitor Mode was active.

A means of exit from the interrupt routine is the Branch and Reset Interrupt Long (BRL) instruction. Normally, the BRL instruction would make use of an indirect reference (*) to the address previously referenced by the BSL instruction upon entering the routine. If this is done, the Condition Register is restored to its original contents (at the time the interrupt occurred). The state of bit 20 (in the return address) is tested by the BRL and the appropriate virtual memory mode is reestablished when the subsequent instruction is fetched. Figure 5-4 illustrates the subroutine exit sequence.

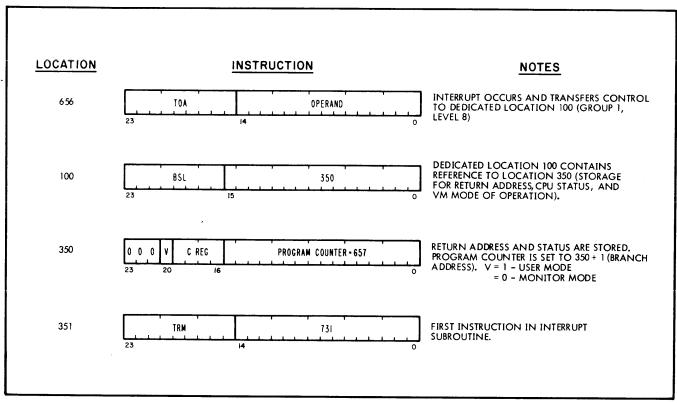
The BRL instruction resets the highest active (not in permissive state) trap or external interrupt level provided that external interrupts are not being "held" (HXI instruction). Active traps can only be reset by the BRL instruction. Active interrupts can only be reset by the BRL instruction, a TD1, TD2, or TD3 instruction, or by master clearing the CPU. A BRL instruction will not reset an interrupt that is in the permissive state.

Operational States One and Three Interrupt Processing

When an interrupt is activated in Operational States One or Three, a pseudo Branch and Save Extended (BSX) instruction is executed by the microcode. An address specifying the dedicated location is loaded into the Instruction Register; no op code is loaded into the register. Unlike Operational State Zero which stores an instruction at the dedicated location, the address word (word 2) of the BSX is stored when the CPU is in Operational States One or Three. Refer to Figure 5-5. An EMA which specifies the storage location of the BSX save word is calculated from the address stored in the dedicated location. Usually, this is a direct address, but indirection or indexing may be specified. In addition to storing the save word, which contains the Condition Register contents and program address of the next sequential instruction, the virtual memory mode of operation is recorded in bit 20 of the Virtual Limit Register (VLR). VLR20 is set if the CPU was in the User Mode when the interrupt occurred, or reset if the Monitor Mode was active.

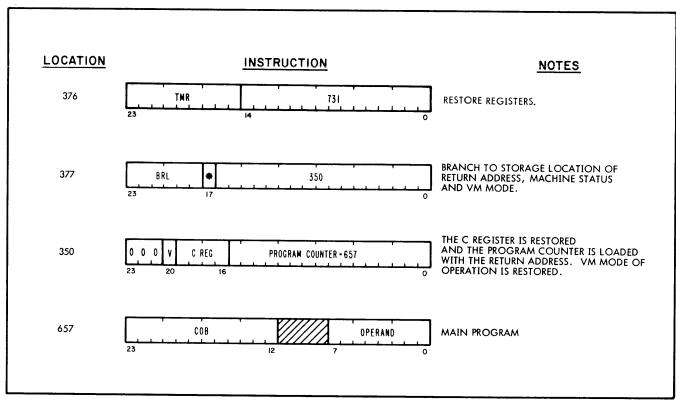
Exit from the interrupt subroutine is by means of an indirected Branch and Reset Interrupt Long (BRL) instruction (no indirect chaining allowed). See Figure 5-6. The Program Counter and Condition Register are restored from the BSX save word. VLR bit 20 remains unchanged if another interrupt is active and enabled. If no other interrupt is active and enabled, VLR20 is reset.

Control of active interrupts by execution of the BRL instruction, and certain other specified conditions is as described for Operational State Zero interrupt processing.



M160-060D

Figure 5-3. Interrupt Subroutine Entry, Operational State Zero



M160-157D

Figure 5-4. Interrupt Subroutine Exit, Operational State Zero

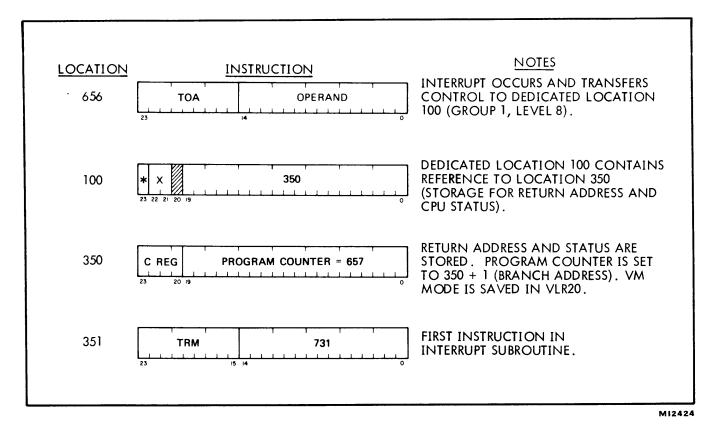


Figure 5-5. Interrupt Subroutine Entry, Operational States One and Three

LOCATION INSTRUCTION **NOTES** RESTORE REGISTERS. 376 731 **TMR** BRANCH TO STORAGE LOCATION OF 377 BRL 350 RETURN ADDRESS AND MACHINE STATUS. THE C REGISTER IS RESTORED AND THE PROGRAM COUNTER IS LOADED WITH 350 C REG PROGRAM COUNTER = 657 THE RETURN ADDRESS. VM MODE OF OPERATION IS RESTORED. MAIN PROGRAM. 657 **OPERAND** СОВ

M12425

Figure 5-6. Interrupt Subroutine Exit, Operational States One and Three

SECTION VI SCIENTIFIC ARITHMETIC UNIT OPERATION

GENERAL DESCRIPTION

The Scientific Arithmetic Unit (SAU) instructions provide double-precision, floating-point capability for the computer. The E Unit implements the execution of 47 SAU instructions, or operation codes. SAU data and condition information may be displayed via the MAP Terminal.

FLOATING-POINT DATA FORMAT

All arithmetic operations are carried out in double-precision format to yield a 39-bit mantissa and an 8-bit exponent. Figure 6-1 illustrates the floating-point data formats employed by the CPU's Double (D) Register, memory, and the SAU's X and XW Register.

Data transfers to the floating-point registers are either single-precision integers or double-precision, floating-point, normalized numbers. All arithmetic operations are executed in the double-precision, floating-point format as illustrated in Figure 6-1. Therefore, any integer number transferred to the floating-point registers for arithmetic operations is first normalized and converted to floating-point format. All double-precision transfers to the floating-point registers, whether from the D Register or memory, are assumed to be normalized, floating-point quantities. Bit 23 of the least-significant half (LSH) of the double word is truncated.

SAU REGISTERS

Three SAU registers are available to the programmer. These are:

- a. X Register (signed mantissa Figure 6-1);
- b. XW Register (signed exponent Figure 6-1); and
- c. Y Register (condition code Figure 6-2).

The XW Register can be independently modified via the SAU instruction set. Figure 6-2 illustrates the Y (Condition) Register bit configuration and their significance in reflecting the results of SAU operations.

Note that condition codes generated by the decimal arithmetic instructions are also loaded into bit positions 3-0 of the Y Register. SAU and decimal instructions should not be intermixed. If SAU instructions follow a string of decimal arithmetic instructions, the Y Register must be

initialized prior to executing the SAU instructions. Similarly, the Y Register must be initialized after executing a string of decimal arithmetic instructions if the decimal instructions are followed by SAU instructions.

After an instruction modifies the XW Register, the value in Y reflects the result of that operation and not the condition of X.

OPERATION AND CONTROL

Data Transfers

All floating-point data transfers are, effectively, confined to the X, XW, and Y Registers. Data transfers may involve the E and A Registers or memory. The transfer source and destination are selected as a function of the instruction being executed.

SAU Instructions

For a detailed description of SAU instructions, refer to Section VII of this manual.

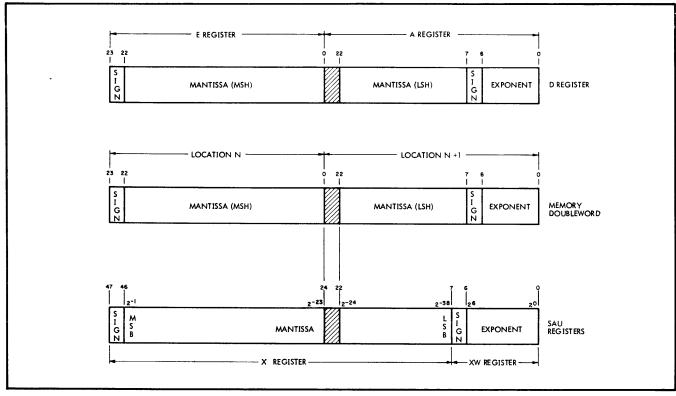
SAU INTERRUPT

The executive trap (Group 0, Level 7) provided for the SAU instructions is used to detect overflow/underflow conditions resulting from the execution of SAU instructions. The trap is controlled by two SAU instructions and the hold/release external interrupt instructions of the CPU.

The SAU instructions which control the trap are the Hold SAU Overflow Interrupt (HSI) and the Release SAU Overflow Interrupt (RSI). The trap, when enabled, is triggered by the overflow bit (bit 0) of the Y Register. In order to start SAU operation and enable the trap the following sequence may be used.

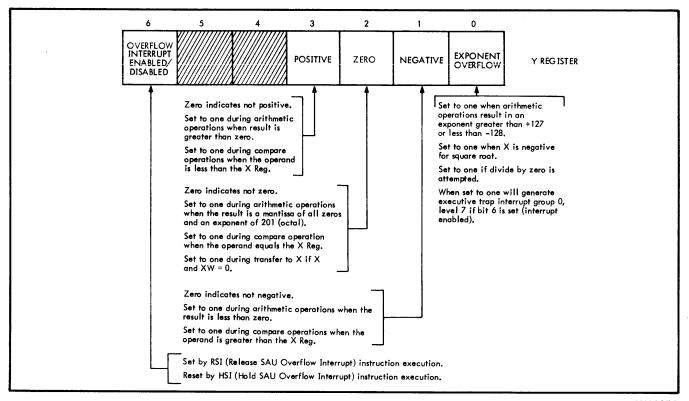
TOY 0 or TMX OPERAND RSI RSI

Either sequence clears the overflow bit and prevents an extraneous interrupt.



M11226F

Figure 6-1. Floating-Point Data Formats



M11227A

Figure 6-2. Y (Condition) Register

When the SAU trap is enabled and an overflow occurs, the SAU is set to a busy condition, preventing the execution of any other SAU instruction except an HSI. Execution of any of the decimal instructions is also inhibited. This allows the program to determine the location of the SAU instruction which caused the overflow. The SAU interrupt processing routine must execute an HSI as its first SAU instruction. Prior to exiting the service routine, bit 0 of the Y Register must be cleared and an RSI instruction performed to rearm the SAU trap. A typical entry/exit sequence is:

SAUPI ***
HSI
.
.
.
.
TOY 0
RSI
BRL* SAUPI

Note that an overflow can be caused by program control with the sequence:

HSI RSI TOY 1

It should be noted that the contents of the Program Counter at the time of the interrupt does not necessarily have a direct relation to the location of the SAU instruction which caused the overflow. This is due to the occurrence of other interrupts, the execution of the HXI/RXI instructions and the way in which the SAU and CPU instructions are intermixed.

When it is a requirement to know exactly where the instruction causing the overflow is located, careful coding is mandatory if the concurrent operation capability is to be used. It is recommended that in cases where overflow is likely, the SAU instructions be written consecutively to simplify the procedure for finding which SAU instruction caused the overflow.

SECTION VII INSTRUCTION SET

INTRODUCTION

The instruction set consists of several functional groups or families of instructions. Among these are: arithmetic; branch; compare; input/output; logical; shift; transfer; etc. Each group, in turn, is composed of individual instructions that perform specific functions.

Through the application of the instruction set, the programmer has access to each memory location and major register in the CPU. In addition, the instruction set provides for the alteration and control of program flow, manipulation and modification (arithmetic and logical) of data, servicing of priority interrupts and control of I/O operations.

INSTRUCTION TYPES AND FORMATS

Introduction

The instruction word defines the operation to be performed and the manner in which it is to be performed. All instruction formats contain an operation code (op code) that defines the general process that is to be undertaken such as add, transfer, branch, and so forth. The op code usually contains six or 12 bits, however, some instructions require expansion of the op code beyond 12 bits. Additional bits in the instruction word specify how the general operation is to be performed. For example, when adding the contents of one register to the contents of another, the additional bits indicate which registers are involved. The appropriate formats are provided with the individual instruction descriptions.

The instruction set may be divided into three general types of instructions which are designated memory reference, immediate operand, and augmented. See Figure 7-1. Memory reference instructions access memory and use formats that specify an address. The address bits are sometimes supplemented by special bits (indirect, index) in the instruction word. In other cases, the additional bits are not used for address modification, but are used to define a condition under which the specified memory location will be accessed. Instead of an address field, the immediate operand type of instruction specifies an operand in the instruction word. Instructions that are not of the memory reference or operand type are included in the augmented group. This type of instruction specifies data sources and destinations or other parameters such as shift count, I/O channel and unit numbers, and additional functions or conditions.

Two basic types of instruction word formats are used in the computer. The first of these, termed standard, is a single-word instruction. The second type of instruction word format, termed extended, is a double-word instruction.

Standard Instruction Format

Each standard instruction, with the exception of the USP and AOM instructions, is decoded from a 24-bit memory word. The USP and AOM instructions are double-word instructions which are included in this group because they are not in the extended instruction format.

The functions of several of the standard instructions are dependent upon whether the CPU is in the Compatibility or Address Extension Mode of operation. The instructions affected are the BSL, TLO, BRL, Branch and Link, and GAP instructions. The differences in operation of these instructions are provided with the individual instruction descriptions.

Extended Instruction Format

Direct memory addressing to one megaword is accomplished with instructions in the extended instruction format. These are double-word instructions that are identified by an octal 7740 (escape code) contained in bits 23-12 of word 1. The majority of the extended instructions are extensions of the standard instructions. These are identified in the individual instruction descriptions by adding a percent sign to the instruction mnemonic. As an example, TMA % indicates an instruction that can be executed in both the standard instruction format and the extended instruction format. When an instruction can be executed in both the standard and extended formats, only the standard instruction format is illustrated with the instruction descriptions provided in this section. Unless otherwise noted with the individual instruction description, the extended version of this group of instructions uses the format illustrated in figure 7-1.

Bits 11 through 3 of the first word of the extended instruction contain the op code and appear as they would in bit positions 23-15 of the standard instruction. Bits 2-0 of word 1 are not used and are defined to be zeroes. Word 2 of the extended instruction is an address word which is read from memory as an indirect operand access. Bit 23 is the indirect bit, and bits 22 and 21 are the indexing field. Bit 20 is, by definition, not used. The remaining bits, 19 through 0, comprise the 20-bit address field.

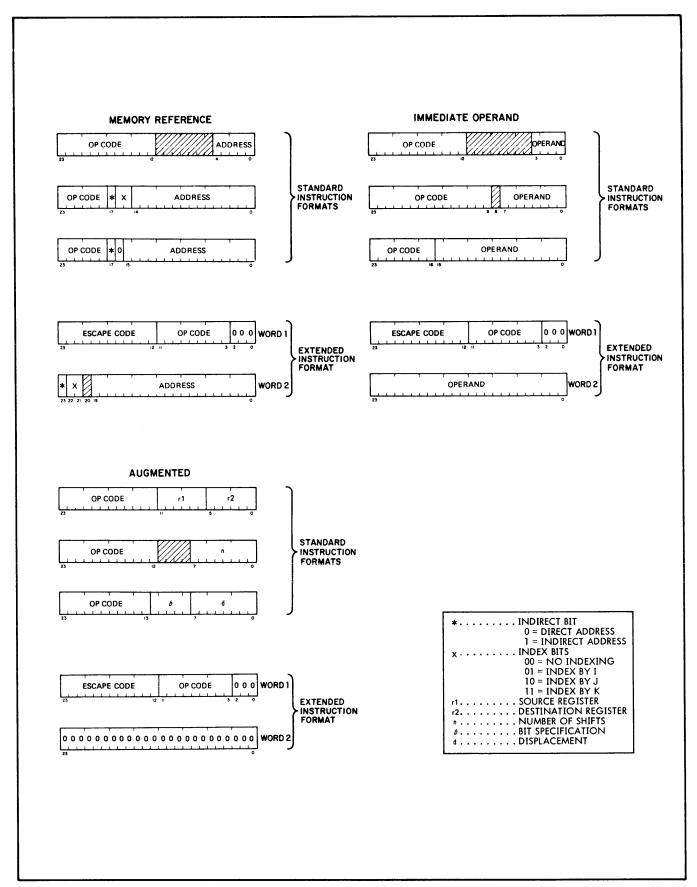


Figure 7-1. Typical Instruction Word Formats

Two instructions, the BSX and RPB, are extended instructions which are in the format shown in Figure 7-1, but which cannot be executed in the standard instruction format. Seven additional extended instructions that cannot be executed in the standard instruction format include the TLK, TPA, HER, RER, LVR, TCD, and THA instructions. The word 1 format of these instructions is as shown in Figure 7-1, but the word 2 contents differ. The word 2 format of the TLK instruction contains a 24-bit operand, while word 2 of the remaining six instructions contain all zeroes.

Table 7-1 is a list of standard instructions which can also be executed in the extended format. Included with each instruction mnemonic is the op code which is contained in bit positions 11-3 of word 1 of the extended instruction format.

Table 7-1. Summary of Extended Instructions Derived From Standard Instructions

İ	OP		OP		OP
INST	CODE	INST	CODE	INST	CODE
AAM	500	BPR	650	ОМА	350
AEM	470	BPS	657	RBM	270
AIM	461	BRL	252	SMA	530
AJM	462	BSL	250	SMB	550
AKM	463	BUC	210	SMD	540
AMA	430	BUL	260	SME	520
AMB	450	BWI	231	SMI	511
AMD	440	BWJ	232	SMJ	512
AME	420	BWK	233	SMK	513
AMI	411	BZR	640	SMX	740
AMJ	412	BZS	647	TAM	150
AMK	413	CMA	330	TBM	170
AMX	730	CMB	340	TDM	160
AUM	300	CME	320	TEM	140
BBI	607	CMI	311	TFM	460
BBJ	617	CMJ	312	TIM	110
BLI	241	CMK	313	TJM	120
BLJ	242	CZM	410	TKM	130
BLK	243	DMA	360	TMA	050
BLL	262	DMX	760	TMB	070
BNN	225	DVM	570	TMD	060
BNO	224	EMB	310	TME	040
BNP	227	EXM	400	TMI	010
BNR	630	HTI	271	TMJ	020
BNS	637	HTJ	272	TMK	030
BNZ	226	HTK	273	TMQ	510
BON	221	IMA	700	TMR	100
воо	220	IME	670	TMX	710
BOP	223	IMI	661	TRM	200
BOR	772	IMJ	662	TXM	720
BOS	773	IMK	663	TZM	660
BOX	627	MMX	750	XMA	370
BOZ	222	MYM	560		

INSTRUCTION FORMULA

The instruction formula, presented with each instruction description, provides a graphic representation of a 24-bit instruction word. The formula expresses an instruction word as a concatenation of its various fields where each field is represented by one or more octal digits. For example, the formula 21.*+X:a expresses a memory reference branch where "21" represents a 6-bit (2 octal digits) Op Code, * and X are additive quantities defining the indirect (*) and index (X) field, and "a" is a memory reference in a 15-bit address field.

The period (.) and colon (:) provide field separation in the formula, with the colon indicating right/left justification. All digits or references to the left of the colon are left-justified, and those to the right are right-justified in their respective fields. The absence of a colon indicates that all digits or references are left-justified in their fields. Examples of instruction formulas are as follows:

INSTRUCTION DESCRIPTIONS

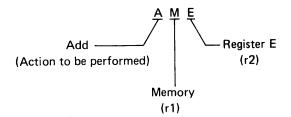
The following paragraphs describe, in detail, the various instructions. The instructions are arranged by functional groups (arithmetic, branch, compare, etc.). General information pertaining to each group is presented in the introductory paragraphs.

Each instruction description includes the three-letter mnemonic identifier, instruction name, instruction formula, and lists the registers affected. Bit assignments for each instruction are shown by means of the binary word format illustration, and a brief explanation of the instruction operation is provided. Special notes are given, where required, to complete the instruction description.

Arithmetic Instructions

The arithmetic instruction group includes the standard arithmetic operations — addition, subtraction, multiplication and division — as well as square root, normalization and sign extension instructions. Also included are several register-to-register operations which compute the absolute value, negate or round off the contents, or negate the sign of one register and subsequently transfer its contents to a second register.

The arithmetic instruction mnemonics provide a brief definition of specific operations to be performed. The first letter of the mnemonic signifies the action or type of operation to be performed, the second letter identifies the first quantity or reference (r1) to be used in the operation, and the third letter identifies the second reference (r2). For example:



In the majority of arithmetic instructions, the result of the operation remains in r2 leaving r1 unchanged (except where r1 and r2 are the same). Certain instructions — notably, those performing multiplication, division, sign extension and square root computation — do not comply with the r1 and r2 conventions stated above. These instructions are described thoroughly in the individual instruction descriptions.

Unless noted otherwise, each arithmetic operation causes the Condition (C) Register to be set reflecting the status of

the result. The various arithmetic conditions are defined as follows:

- Positive Result is arithmetically greater than zero, indicated by a ONE in bit position 3 of the C Register. A ZERO in bit position 3 indicates "Not Positive".
- b. Zero All bits of the quantity under consideration are ZEROs, indicated by a ONE in bit position 2 of the C Register. A ZERO in bit position 2 indicates "Not Zero".
- c. Negative Result is arithmetically less than zero, indicated by a ONE in bit position 1 of the C Register. A ZERO in bit position 1 indicates "Not Negative".
- d. Overflow An Overflow results from an operation instead of displaying the status of an operand. As a general rule, an arithmetic Overflow will occur when a bit is carried into the designated sign bit position and not carried out or vice versa. An Overflow condition is indicated by a ONE in bit position 0 of the C Register. A ZERO in bit position 0 indicates "No Overflow".

The following instructions are included in the arithmetic group.

AAM	Add A to Memory	7-6
AEM	Add E to Memory	7-7
AMA	Add Memory to A	7-5
AMB	Add Memory to Byte	7-6
AMD	Add Memory to Double	7-6
AME	Add Memory to E	7-5
AMx	Add Memory to Register	7-5
AOB	Add Operand to Byte	7-7
AOM	Add Operand to Memory	7-7
AOr	Add Operand to Register	7-7
Arr	Add Register to Register	7-8
AUM	Add Unity to Memory	7-5
AxM	Add Register to Memory	7-6
DVM	Divide by Memory	7-8
DVO	Divide by Operand	7-8
DVT	Divide by T	7-9
DVx	Divide by Register	7-9
DV2	Divide by 2	7-9
ESA	Extend Sign of A	7-10
ESB	Extend Sign of Byte	7-10
FNO	Floating Normalize	7-10
MYM	Multiply by Memory	7-10
MYO	Multiply by Operand	7-10
MYr	Multiply by Register	7-11

NBB	Negate of Byte to Byte	7-11
NDD	Negate of Double to Double	7-12
Nrr	Negate of Register to Register	7-11
NSr	Negate Sign of Register	7-12
PBB	Positive of Byte to Byte	7-12
PDD	Positive of Double to Double	7-12
Prr	Positive of Register to Register	7-13
Rrr	Round of Register to Register	7-13
SMA	Subtract Memory from A	7-14
SMB	Subtract Memory from Byte	7-14
SMD	Subtract Memory from Double	7-14
SME	Subtract Memory from E	7-14
SMx	Subtract Memory from Register	7-13
SOB	Subtract Operand from Byte	7-15
SOr	Subtract Operand from Register	7-15
SRE	Square Root Extended	7-16
SRT	Square Root	7-15
Srr	Subtract Register from Register	7-15

AUM % Add Unity to Memory

30.*+X:a **Affected** M,C OP CODE **ADDRESS**

Operation

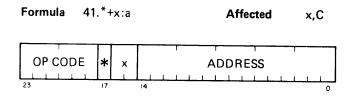
Formula

The contents of the effective memory address are incremented by one.

Note

The Condition Register is set to Positive, Negative or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out of the sign bit (23) without a carry in.

AMx % Add Memory to Register



Operation

The contents of the effective memory address are algebraically added to the contents of register I, J or K.

Notes

AMx is not a computer instruction mnemonic but represents a family of instruction mnemonics, x is coded as follows to select one of the index registers.

$$x = 1 (I)$$

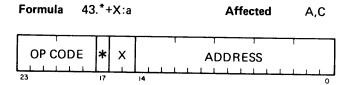
2 (J)
3 (K)

A code of 41.*+1:a, for example, implements the Add Memory to I (AMI) instruction.

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the Operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out of the sign bit (23) without a carry in.

AMA % Add Memory to A



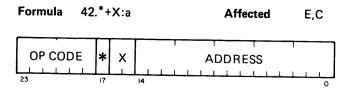
Operation

The contents of the effective memory address are algebraically added to the contents of the A Register.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out of the sign bit (23) without a carry in.

AME % Add Memory to E



Operation

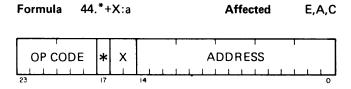
The contents of the effective memory address are algebraically added to the contents of the E Register.

0830007-000 Original 8/79

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out of the sign bit (23) without a carry in.

AMD % Add Memory to Double



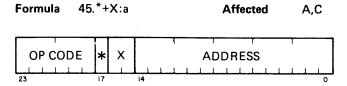
Operation

The contents of the effective memory address (EMA) and the next sequential memory address (EMA+1) are algebraically added to the contents of the D Register according to the double integer format defined in Section II.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in the D Register after the operation. Overflow is set if one occurs during the addition.

AMB % Add Memory to Byte



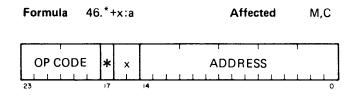
Operation

Bits 7-0 of the contents of the effective memory address are algebraically added to the contents of the B Register (A7-A0). Bits 23-8 of the A Register are unchanged.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

AxM % Add Register to Memory



Operation

The 24-bit contents of the I, J or K Register are algebraically added to the contents of the effective memory address.

Notes

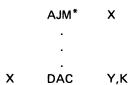
AxM is not a computer instruction mnemonic but represents a family of instruction mnemonics, x is coded as follows to select one of the index registers.

$$x = 1 (I)$$

2 (J)
3 (K)

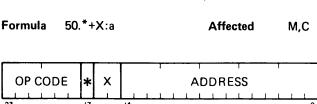
A code of 46.*+2:a, for example, implements the add J to Memory (AJM) instruction.

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,



The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

AAM % Add A to Memory



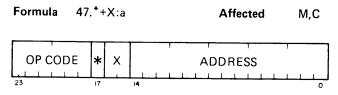
Operation

The contents of the A Register are algebraically added to the contents of the effective memory address.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

AEM % Add E to Memory



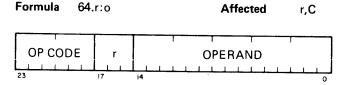
Operation

The contents of the E Register are algebraically added to the contents of the effective memory address.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

AOr Add Operand to Register



Operation

The 15-bit unsigned operand is algebraically added to the contents of the specified register.

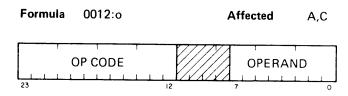
Notes

AOr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r is coded as follows to select any of the general purpose registers.

A code of 64.3:o, for example, implements the Add Operand to K (AOK) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

AOB Add Operand to Byte



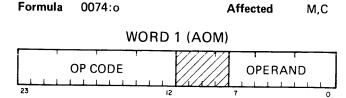
Operation

The 8-bit signed operand is algebraically added to the contents of the B Register (A7-A0). Bits 23-8 of the A Register are unchanged.

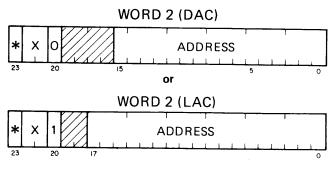
Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

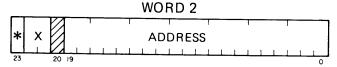
AOM (n) Add Operand to Memory Word 2(m)



COMPATABILITY MODE:







Operation

The 8-bit signed operand (n) is algebraically added to the contents of the effective memory address (m).

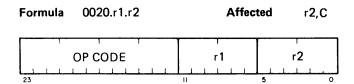
Notes

If a demand page, restrict mode violation, or limit violation occurs when attempting to access the effective memory address while in the virtual memory User mode, the Program Counter will be decremented by one. If the violation occurs during the fetch of the second word, the Program Counter will be decremented by one.

An AOM instruction may not be used after a ROM instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

Arr Add Register to Register



Operation

The contents of r1 are algebraically added to the contents of r2.

Notes

Arr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r1 and r2 are coded as follows to select one of the general purpose registers.

A code of 0020.10.40, for example, implements the Add E to T (AET) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

r1 and r2 are selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r1 or r2, they are logically ORed prior to the specified operation. The result is copied into all of the selected r2 registers. Affected registers are only those selected in group r2.

DVM % Divide by Memory

Formula 57.*+X:a Affected E,A,C

OP CODE * X ADDRESS

Operation

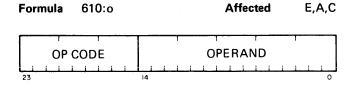
A23 is cleared and the double-precision contents of the D Register (E and A) are algebraically divided by the single-precision contents of the effective memory address. The signed, single-precision, quotient is left in A and the remainder is left in E. The remainder will have the same sign as the original dividend and the Condition Register will be set according to the status of the quotient.

Notes

If it is desired to divide a single-precision number in A by memory, an Extend Sign of A (ESA) instruction should be executed prior to the DVM. This will establish the proper format for the dividend.

If the contents of E are equal to, or greater than, the contents of memory, an Overflow condition will result and the Condition Register will be set accordingly.

DVO Divide by Operand



Operation

A23 is cleared and the double-precision contents of the D Register (E and A) are algebraically divided by the 15-bit unsigned operand. The signed, single-precision, quotient is left in A and the remainder is left in E. The remainder will have the same sign as the original dividend and the Condition Register will be set according to the status of the quotient.

Notes

If it is desired to divide a single-precision number in A by the operand, an Extend Sign of A (ESA) instruction should be executed prior to the DVO. This will establish the proper format for the dividend.

If the contents of E are equal to, or greater than, the operand, an Overflow condition will result and the Condition Register will be set accordingly.

DVx Divide by Register

Formula 61.x Affected E,A,C



Operation

A23 is cleared and the double-precision contents of the D Register (E and A) are algebraically divided by the specified register. The signed, single-precision, quotient is left in A and the remainder is left in E. The remainder will have the same sign as the original dividend and the Condition Register will be set according to the status of the quotient.

Notes

DVx is not a computer instruction mnemonic but represents a family of instruction mnemonics. x is coded as follows to select one of the index registers.

$$x = 1 (I)$$

2 (J)
3 (K)

A code of 61.1, for example, implements the Divide by I (DVI) instruction.

If it is desired to divide a single-precision number in A by the contents of the specified register, and Extend Sign of A (ESA) instruction should be executed prior to the divide instruction. This will establish the proper format for the dividend.

If the contents of E are equal to, or greater than, the contents of the specified register, an Overflow condition will result and the Condition Register will be set accordingly.

DVT Divide by T

Formula 616. Affected E,A,C

OP CODE

15

Operation

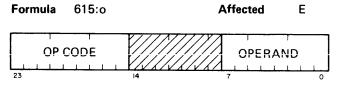
A23 is cleared and the double-precision contents of the D Register (E and A) are algebraically divided by the T Register. The signed, single-precision, quotient is left in A and the remainder is left in E. The remainder will have the same sign as the original dividend and the Condition Register will be set according to the status of the quotient.

Notes

If it is desired to divide a single-precision number in A by the contents of the T Register, an Extend Sign of A (ESA) instruction should be executed prior to the divide instruction. This will establish the proper format for the dividend.

If the contents of E are equal to, or greater, than, the contents of the specified register, an Overflow condition will result and the Condition Register will be set accordingly.

DV2 Divide by 2



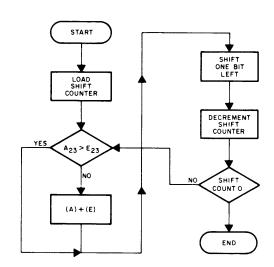
Operation

The DV2 instruction divides the contents of the E Register by the contents of the A Register, except that the arithmetic operation will be Modulo 2 (exclusive OR) instead of 2's complement arithmetic. The 8-bit operand contained in the instruction specifies the number of shifts.

Notes

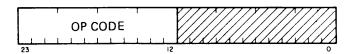
The specified number of shifts must be an even number and cannot be zero. If zero shifts are specified, the operation is the same as when a shift of one (1) is specified.

This instruction is used for generating and checking error codes based on polynomial coding techniques. The polynomial and the operand to be implemented must be left-justified in the A and E Registers. The result will be placed in the E Register while the polynomial will remain in the A Register.



ESA Extend Sign of A

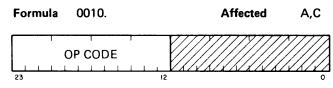
Formula 0037. Affected E,C,A



Operation

The state of the sign bit (A23) of the A Register is copied into all 24 positions of the E Register and bit A23 is then set to zero. This forms a double-precision number in E and A.

ESB Extend Sign of Byte



Operation

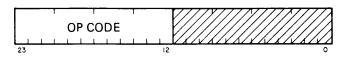
The state of the register B sign bit (A7) is copied into bit positions A8-A23, forming a sign extension of the byte.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in A at the completion of the operation.

FNO Floating Normalize

Formula 0054. Affected E,A,i,C



Operation

The contents of the D Register (E and A) are shifted left arithmetically until bit E22 differs from E23. The negative shift count (i.e., the number of shifts performed) replaces the contents of the I Register.

Notes

Example: Convert a double-precision integer in D to double-precision floating point format.

TOC 0 Clear Overflow FNO Normalize

TIB Position exponent in byte (A7-A0).

BOZ *+2 If result is zero, no exponent adjustment is necessary.

AOB 46 Adjust shift count

There are four special cases where the shifting process differs from that described above.

If the binary pattern 11000...0 is detected in register D, normalization is terminated to avoid creating the invalid pattern 10000...0.

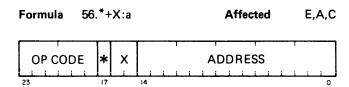
If the invalid binary pattern 10000...0 is detected, it is shifted right one position producing the pattern 11000...0. The shift count is adjusted accordingly.

If the pattern 00000...0 is detected, the shift count is set to -1778, making a zero less significant than any other value.

If an Overflow condition is present when beginning the operation, the contents of the D Register are arithmetically shifted right one position. The shift count is set to ONE and the sign of D is complemented.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

MYM % Multiply by Memory



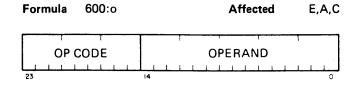
Operation

The contents of the A Register are algebraically multiplied by the contents of the effective memory address. The double-precision product replaces the previous contents of the D Register (E and A).

Note

An Overflow will result if the full-scale negative number (1000....00) is used as both the multiplier and multiplicand, and the result is full-scale negative (1000....0).

MYO Multiply by Operand



Operation

The contents of the A Register are algebraically multiplied by the 15-bit unsigned operand in the instruction word. The double-precision product replaces the previous contents of the D Register (E and A).

MYr Multiply by Register

Formula 60.r Affected E,A,C



Operation

The contents of the A Register are algebraically multiplied by the contents of the specified register. The double-precision product replaces the previous contents of the D Register (E and A).

Notes

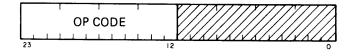
MYr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r is coded as follows to select one of the general purpose registers.

A code of 60.4, for example, implements the Multiply by E (MYE) instruction.

An Overflow will result if the full-scale negative number (1000....00) is used as both the multiplier and multiplicand, and the result is full-scale negative (1000....0).

NBB Negate of Byte to Byte

Formula 0005. Affected A,C



Operation

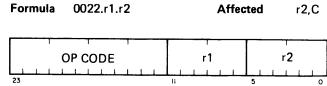
The contents of the B Register (A7-A0) are two's complemented. Bit positions A23-A8 are unchanged.

Notes

An Overflow will result when negating 27 (full-scale negative byte).

The Condition Register is set to Positive, Negative, or Zero, based on the result in the Byte Register at the completion of the operation.

Nrr Negate of Register to Register



Operation

The two's complement of the contents of r1 replace the previous contents of r2.

Notes

Nrr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r1 and r2 are coded as follows to select any of the general purpose registers.

A code of 0022.40.01, for example, implements the Negate of T to I (NTI) instruction.

An Overflow will result when negating 223 (full-scale negative number).

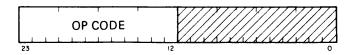
The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

r1 and r2 are selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r1 or r2, they are logically ORed prior to the specified operation. The result is copied into all of the selected r2 registers. Affected registers are only those selected in group r2.

If the Timer (T) Register is selected as source or destination, the instruction is treated as a multiple register instruction for timing.

NDD Negate of Double to Double

Formula 0033. Affected E,A,C



Operation

The contents of the D Register (E and A), in double-precision format, are two's complemented.

Notes

An Overflow will result when negating 2⁴⁶ (full-scale negative double integer).

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

NSr Negate Sign of Register

Formula 0032.r1.r2 Affected r2,C



Operation

The sign bit of the specified register is complemented.

Notes

NSr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r1 and r2 are coded as follows to select one of the general purpose registers.

A code of 0032.01.01, for example, implements the Negate Sign of I (NSI) instruction.

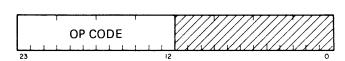
An Overflow will result when negating zero to create a full-scale negative.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

r1 and r2 are selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r1 or r2, they are logically ORed prior to the specified operation. The result is copied into all of the selected r2 registers. Affected registers are those selected in group r2 and the Condition Register.

PBB Positive of Byte to Byte

Formula 0006. Affected A,C



Operation

The absolute value of the contents of the B Register (A7-A0) is placed in the B Register.

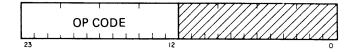
Notes

An Overflow will result when negating a full scale negative byte.

The Condition Register is set to Positive, Negative, or Zero, based on the result in the Byte Register at the completion of the operation.

PDD Positive of Double to Double

Formula 0034. Affected E,A,C



Operation

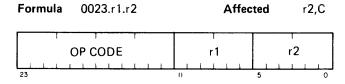
The absolute value of the contents of the D Register is placed in the D Register according to the double integer format defined in Section II.

Notes

An Overflow will result when negating a full scale negative number.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

Prr Positive of Register to Register



Operation

The absolute value of the contents of r1 replaces the previous contents of r2.

Notes

Prr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r1 and r2 are coded as follows to select any of the general purpose registers.

A code of 0023.01.02, for example, implements the Positive of I to J (PIJ) instruction.

An Overflow will result when negating a full-scale negative number.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

r1 and r2 are selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r1 or r2, they are logically ORed prior to the specified operation. The result is copied into all of the selected r2 registers. Affected registers are only those selected in group r2 and the Condition Register.

Rrr Round of Register to Register



Operation

Round the contents of r1 as a function of A and place the result in r2.

Notes

Rrr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r1 is coded to select one-of-five general purpose registers, and r2 is coded to select any of the general purpose registers.

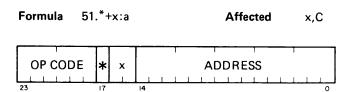
A code of 0075.04.20, for example, implements the Round of K to A (RKA) instruction.

If bit A22 is a ONE, the contents of r1+1 are transferred to r2. If A22 is ZERO, the contents of r1 replace the previous contents of r2. In either case, r1 is unchanged except when the same as r2.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

r1 and r2 are selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r1 or r2, they are logically ORed prior to the specified operation. The result is copied into all of the selected r2 registers.

SMx % Subtract Memory from Register



Operation

The contents of the effective memory address are algebraically subtracted from the contents of the I, J or K Register.

Notes

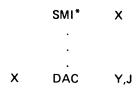
SMx is not a computer instruction mnemonic but represents a family of instruction mnemonics. x is coded as follows to select one of the index registers.

$$x = 1 (I)$$

2 (J)
3 (K)

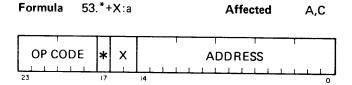
A code of 51.*+1:a, for example, implements the Subtract Memory from I (SMI) instruction.

The immediate memory reference cannot be indexed; however, indexing of indirect reference is permitted, e.g.,



The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

SMA % Subtract Memory from A



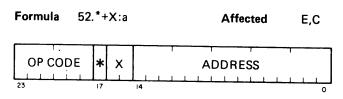
Operation

The contents of the effective memory address are algebraically subtracted from the contents of the A Register.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

SME % Subtract Memory from E



Operation

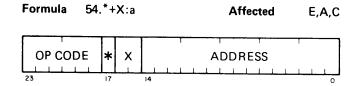
The contents of the effective memory address are algebraically subtracted from the contents of the E Register.

Note

The Condition Register is set to Positive, Negative, or Zero,

based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

SMD % Subtract Memory from Double



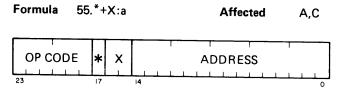
Operation

The contents of the effective memory address (EMA) and the next sequential address (EMA+1) are algebraically subtracted from the contents of the D Register (E and A), according to the double integer format defined in Section II.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

SMB % Subtract Memory from Byte



Operation

The contents of bits 7-0 of the effective memory address are algebraically subtracted from the B Register (A7-A0). Bits A23-A8 are unaffected.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

SOr Subtract Operand from Register

Formula 65.r:o Affected r,C

OP CODE r OPERAND

Operation

The 15-bit unsigned operand is algebraically subtracted from the contents of the specified register.

Notes

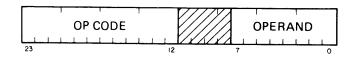
SOr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r is coded as follows to select one of the general purpose registers.

A code of 65.1:o, for example, implements the Subtract Operand from I (SOI) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the result of the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

SOB Subtract Operand from Byte

Formula 0013:0 Affected A,C



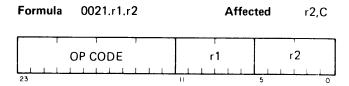
Operation

The 8-bit signed operand is algebraically subtracted from the contents of the B Register (A7-A0). Bits A23-A8 are unaffected.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

Srr Subtract Register from Register



Operation

The contents of r1 are algebraically subtracted from r2.

Notes

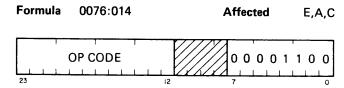
Srr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r1 and r2 are coded as follows to select any of the general purpose registers.

A code of 0020.01.02, for example, implements the Subtract I from J (SIJ) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

r1 and r2 are selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r1 or r2, they are logically ORed prior to the specified operation. The result is copied into all of the selected r2 registers.

SRT Square Root



Operation

The contents of the A Register are treated as a 23-bit positive integer. The square root of this quantity is placed in the A Register, right justified, and the remainder is placed in the E Register so that:

root² + remainder = original integer.

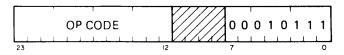
Notes

If the sign bit (23) of the A Register is set, the Condition Register will be set to Overflow.

SRT generates a root of 12 significant bits; i.e., the true integer root of any positive integer in the A Register.

Consider the following examples where An implies a binary point to the right of bit n.

	Positive Integer	Root (Octal)	
	2 at A0	1 at A0	
	2 at A20	1.3240 at A10	
SRE	Square Root E	xtended	
Formula	0076:027	Affected	E,A,C



Operation

The contents of the A Register are treated as a 23-bit positive integer. The square root of this quantity is placed in the A Register, right justified, and the remainder is placed in the E Register so that:

root² + remainder = original integer.

Notes

If the sign bit (23) of the A Register is set, the Condition Register will be set to Overflow.

SRE generates a root of 23 significant bits. This extended significance is obtained by assuming 22 zeros to the right of bit A0; effectively multiplying the contents of A by 2²² and, consequently, the root by 2¹¹.

Consider the following examples where An implies a binary point in the right of bit n.

Positive Integer	Root (Octal)
2 at A0	1.3240 at A11
2 at A20	1.3240474 at A21

Branch Instructions

The branch group of instructions can be divided into two basic types; conditional and unconditional branches. Conditional branches cause control to be transferred to a specified address upon detection of a certain machine condition as indicated by the contents of the Condition Register. Unconditional branches cause control to be transferred unconditionally to a specified address.

Branch instructions follow the mapping rules described in the addressing functions paragraph contained in Section II.

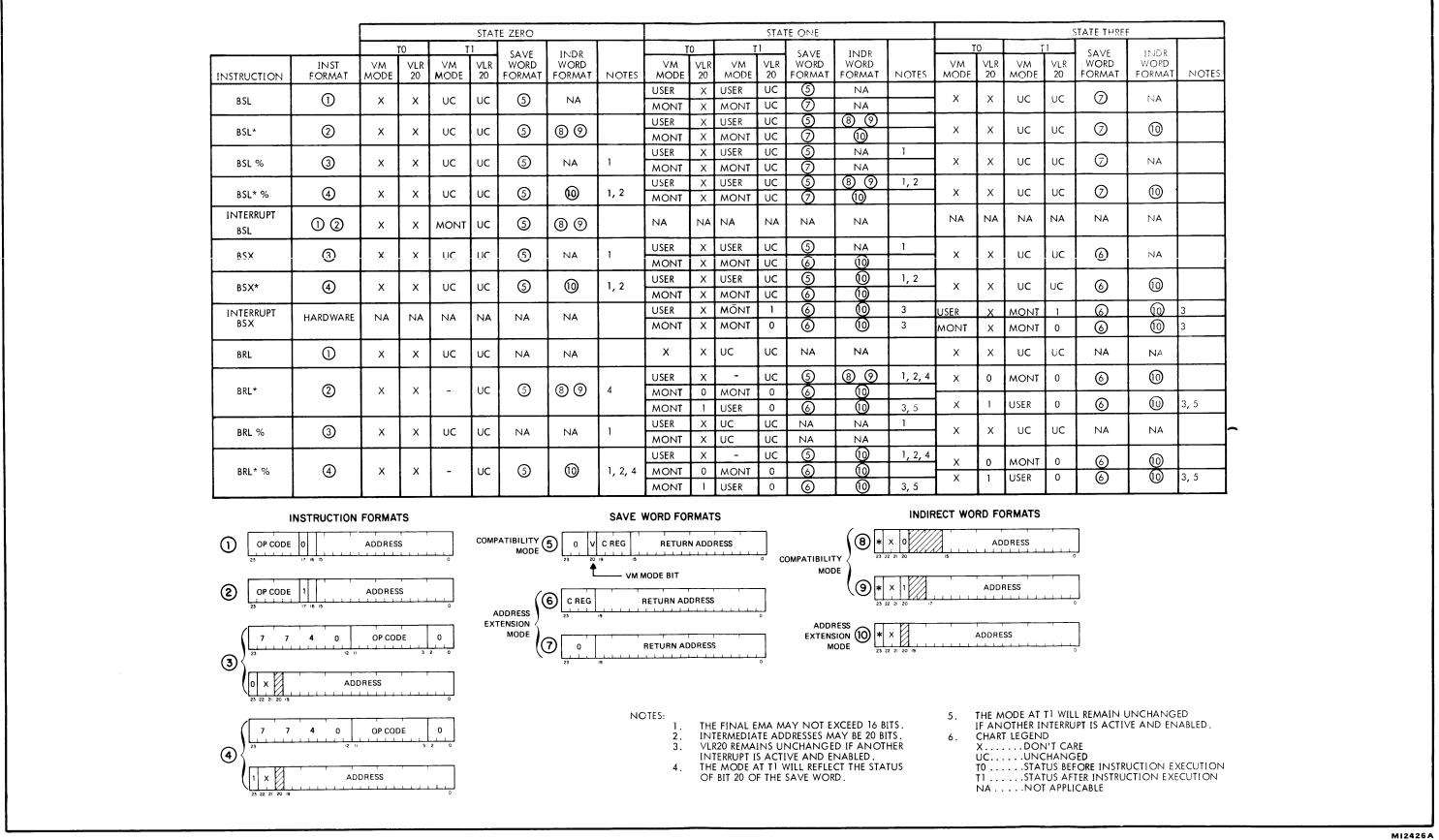
Caution should be observed when employing branch instructions in conjunction with the virtual memory system. When a Release User Mode (RUM) instruction is executed, any branch instruction following the RUM will cause the User Mode to be established. If the instruction is a conditional branch, the User Mode will be established regardless of the outcome of the conditional test. A BLU instruction automatically establishes the Monitor Mode.

Three branch instructions modify machine operation when executed. The BSL, BSX, and BRL instructions are affected by the operational state of the CPU, and by the virtual memory mode of operation. A summary of the functions of these instructions is provided in Figure 7-2.

CPU operational states are shown along the top of the chart. Under each operational state, the virtual memory mode and state of VLR20 before and after instruction execution is listed. Save word and indirect word formats are also indicated. The first instruction listed is a BSL which contains an op code and a 16-bit address; the indirect bit is reset. In Operational State Zero, the virtual memory mode and state of VLR20 are don't cares prior to instruction execution. These two functions remain unchanged after instruction execution. The Compatibility Mode save word format is used. In this format the return address is located in bits 15-0, the condition code is contained in bits 19-16, and the virtual memory mode of operation is saved in bit 20. Since the BSL is not indirected, the indirect word format does not apply.

Operation of the BSL instruction when the CPU is in state one depends on the virtual memory mode of operation. If in the User Mode, the CPU is placed in the Compatibility Mode, and when in the Monitor Mode, operation is in the Address Extension Mode. When the machine is in state one, the virtual memory mode and state of VLR20 remain unchanged after execution of the BSL. The Compatibility Mode save word format is used in the User Mode, and one of the Address Extension Mode save word formats is used in the Monitor Mode. The latter save word contains the return address in bits 19-0, and zeroes in bit positions 23-20; the condition code and virtual memory status are not saved.

Operation of the BSL in state three is similar to state zero operation except for the save word format. Since the Address Extension Mode is established in state three, one of the Address Extension Mode save word formats is specified.



An indirected BSL functions the same as a non-indirected BSL with certain exceptions. The indirect bit is set in the instruction format and an indirect word format is specified. One of the Compatibility Mode indirect word formats is used in state zero and, if in the User Mode, in state one. If in the Monitor Mode in state one, or if in either of the virtual memory modes in state three, the Address Extension Mode indirect word format is specified.

When the BSL is in the extended instruction format, operation is similar to the standard format BSL. Final EMAs may not exceed 16 bits since in the Compatibility Mode the program counter is only 16-bits wide. If the extended BSL is indirected the final EMA cannot exceed 16 bits, but intermediate addresses may be 20 bits.

The interrupt BSL is defined only for state zero. An interrupt generates a hardware Execute Memory (EXM) instruction which accesses the interrupt BSL. No hardware EXM is executed in operational states one or three.

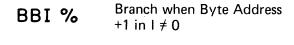
The interrupt BSX is not defined for state zero but is defined for states one and three. When an interrupt is generated, a pseudo (hardware) BSX is executed to force a branch to a dedicated location where an address is accessed as the second word. Since 20-bit addresses are used, direct accesses can be made to up to one megaword of memory. Address Extension Mode save and indirect words are specified. If the virtual memory is in the User Mode when the interrupt BSX is generated, the Monitor Mode is established after execution of the BSX. All valid interrupts reset the User Mode and place the system in the Monitor Mode. If the Monitor Mode is set when the interrupt occurs, the system remains in the Monitor Mode. VLR20 records the virtual memory mode of operation at the time of the first interrupt. This bit remains unchanged if another interrupt is active and enabled.

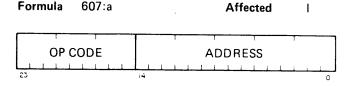
An indirect BRL instruction is usually used to exit an interrupt subroutine. Indirect chaining is allowed in the Compatibility Mode but not in the Address Extension Mode. The Condition Register and program counter are restored according to the contents of the save word stored at the indirect location. Note that the Compatibility Mode and Address Extension Mode save word formats differ.

The following instructions are included in the branch group.

BBI	Branch When Byte Address +1 in $I \neq 0$	7-19
BBJ	Branch When Byte Address +1 in $J \neq 0$	7-20
BJL	Branch Indexed by J Long	7-21
BLL	Branch and Link (J) Long	7-22
BLU	Branch and Link Unrestricted	7-24
BLx	Branch and Link Register	7-22

BNc	Branch on Condition Code	7-21
BOc	Branch on Condition Code	7-21
BRL	Branch and Reset Interrupt Long	7-23
BSL	Branch and Save Return Long	7-22
BSX	Branch and Save Extended	7-23
BUC	Branch Unconditionally	7-20
BUL	Branch Unconditionally Long	7-20
BWx	Branch When Register $+1 \neq 0$	7-21
	3	7-21





Operation

The contents of bits 22 and 23 of the I Register are incremented by one. If the result of this addition (in bits 22 and 23) is not 002, then the contents of the P Register (current program address) are replaced by the effective memory address. If the result of the addition to bits 22 and 23 is 002, then bits 22 and 23 are set to 012 and bits 21-0 are incremented by one. If the resultant sum in bits 21-0 is zero, then the P Register advances to the next sequential program location and the index register is set to 200000008. Otherwise, the contents of the P Register are replaced by the effective memory address.

Notes

In general, the BBI and BBJ instructions are used as special index register increments in order to sequentially reference consecutive bytes in memory via the EMB and RBM instructions. Consider the following example which will move 11 consecutive bytes starting from the third byte at location '200 to the first byte at location '300.

TMJ	= '60000200
TMI	= '20000300
TNK	11
EMB	0
RBM	0
BBI	*+1
BBJ	*+1
BWK	*-4

Occasionally, it is possible to use the address of a portion of the I Register as a byte counter as well as a word pointer. This may be illustrated by the following example which will set the buffer to blanks, starting at byte 3 of location '100 through byte 3 of location '102.

0830007-000 Original 8/79

However, it should be noted that this technique of using the index register as both a byte counter and word pointer may be used only in certain instances. Specifically, when the following relationship is true.

$$R\left(\frac{4-B.n.}{3}\right) = R\left(\frac{CT}{3}\right)$$

Where:

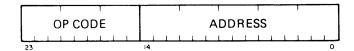
R () = remainder

B.n. = the starting byte number (1,2, or 3) CT = the number of bytes to be referenced

BBJ % Branch when Byte Address +1 in $J \neq 0$

Formula 617:a

Affected



Operation

The contents of bits 22 and 23 of the J Register are incremented by one. If the result of this addition (in bits 22 and 23) is not 002, then the contents of the P Register (current program address) are replaced by the effective memory address. If the result of the addition to bits 22 and 23 is 002, then bits 22 and 23 are set to 012 and bits 21-0 are incremented by one. If the resultant sum in bits 21-0 is zero, then the P Register advances to the next sequential program location and the index register is set to 200000008. Otherwise, the contents of the P Register are replaced by the effective memory address.

Notes

In general, the BBI and BBJ instructions are used as special index register increments in order to sequentially reference consecutive bytes in memory via the EMB and RBM instructions. Consider the following example which will move 11 consecutive bytes starting from the third byte at location '200 to the first byte at location '300.

Occasionally, it is possible to use the address of a portion of the J Register as a byte counter as well as a word pointer. This may be illustrated by the following example which will set the buffer to blanks, starting at byte 3 of location '100 through byte 3 of location '102.

However, it should be noted that this technique of using the index register as both a byte counter and word pointer may be used only in certain instances. Specifically, when the following relationship is true.

$$R\left(\frac{4\text{-B.n.}}{3}\right) = R\left(\frac{CT}{3}\right)$$

Where:

J

R() = remainder

B.n. = the starting byte number (1,2 or 3)

CT = the number of bytes to be referenced

BUC % Branch Unconditionally

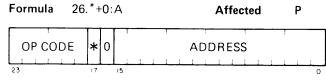
Formula 21.*+X:a Affected P

OP CODE * X ADDRESS

Operation

The contents of the P Register (current program address) are replaced by the effective memory address.

BUL % Branch Unconditionally Long

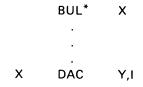


Operation

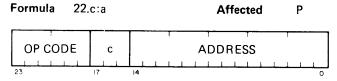
The contents of the P Register (current program address) are replaced by the effective memory address.

Note

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,



BNc % BOc % Branch on Condition Code



Operation

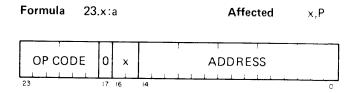
The contents of the Condition Register are tested for the specified condition. If the condition is present, the contents of the P Register (current program address) are replaced by the effective memory address. If the specified condition is not present, the program advances to the next sequential instruction.

Note

BOc and BNo are not computer instruction mnemonics but represents families of instruction mnemonics. c is coded as follows to select the branch on condition.

A code of 22.1:a, for example, implements the Branch on Negative (BON) instruction.

BWx % Branch When Register $+1 \neq 0$



Operation

The contents of the specified register are incremented by one and then tested for zero. If the contents are not zero, the contents of the P Register (current program address) are replaced by the effective memory address. If the contents are zero, the program advances to the next instruction.

Note

BWx is not a computer instruction mnemonic but represents a family of instruction mnemonics. x is coded as follows to select one of the index registers.

$$x = 1 (I)$$

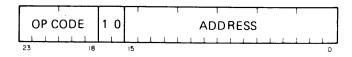
2 (J)
3 (K)

A code of 23.1:a, for example, implements the Branch When $I+1\neq 0$ (BWI) instruction.

Indexing, if specified in word 2 of the extended instruction, occurs before the register is modified.

BJL Branch Indexed by J Long

Formula 23.4:A Affected



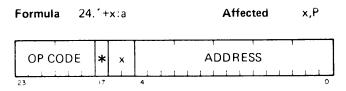
Operation

The contents of the P Register (current program address) are replaced by the effective memory address.

Note

The immediate memory reference is automatically indexed by J.

BLx % Branch and Link Register



Operation

The contents of the I, J or K Register are replaced by the program address of the next sequential instruction, and the contents of the P Register (current program address) are replaced by the effective memory address.

Notes

BLx is not a computer instruction mnemonic but represents a family of instruction mnemonics. x is coded as follows to select one of the index registers.

$$x = 1 (I)$$

2 (J)
3 (K)

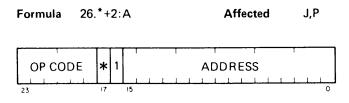
A code of 24.*+1:a, for example, implements the Branch and Link I (BLI) instruction.

If not in the extended instruction format, the immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,



On an indirect or index operation, the specified register is loaded with the contents of the P Register (address of next sequential instruction) before indexing or indirection takes place.

BLL % Branch and Link (J) Long

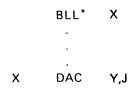


Operation

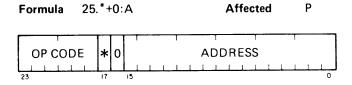
The contents of the J Register are replaced by the program address of the next sequential instruction, and the contents of the P Register (current program address) are replaced by the effective memory address.

Note

If not in the extended instruction format, the immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,



BSL % Branch and Save Return Long



Operation

In the Compatibility Mode, the program address of the next sequential instruction along with the contents of the Condition Register are stored in the effective memory address (EMA). The contents of the P Register (current program address) are then replaced by the address following the effective memory address (EMA + 1).

In the Address Extension Mode, the program address of the next sequential instruction is stored in the effective memory address (EMA). The contents of the P Register (current program address) are then replaced by the address following the effective memory address (EMA + 1).

Notes

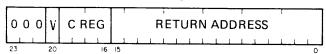
This instruction is used in the Compatibility Mode to enter an interrupt subroutine because it provides a means of returning to the main program at the point of interrupt and saves the machine status (condition) at the time of the interrupt. In the Compatibility Mode, the contents of the Condition Register are stored in bit positions 19-16 of the EMA and the return address (program address of next sequential instruction) is stored in bits 15-0. The remaining bits are set to ZEROs. When an interrupt occurs, the status of the virtual memory system is recorded. Bit 20 is set to ONE if the system is in the User Mode at the time of interrupt; bit 20 is set to ZERO if the Monitor Mode is active.

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

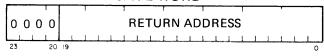
The Condition Register remains unchanged.

COMPATIBILITY MODE SAVE WORD



In the Compatibility Mode, the final EMA may not exceed 16 bits when a BSL or extended BSL is executed. Intermediate Addresses may be 20 bits when an indirect extended BSL is executed.

ADDRESS EXTENSION MODE SAVE WORD

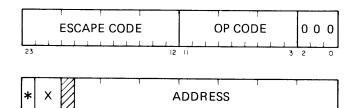


In the Address Extension Mode, the return address is stored in bit positions 19-0 of the EMA; bits 23-20 are reset to ZEROs.

BSX Branch and Save Extended

23 22 21 20 19

Formula 7740.254.0 Affected P *+X:EA



Operation

The program address of the next sequential instruction, along with the contents of the Condition Register, are stored in the 20-bit effective memory address (EMA) location. The contents of the P Register (current program address) are then replaced by the address following the effective memory address (EMA + 1).

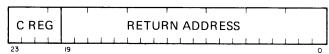
Notes

The BSX instruction is valid only in the extended instruction format. This instruction provides a means of returning to the main program and saves the machine status (condition) at the time of instruction execution.

External interrupts are prohibited for a period of one instruction following the execution of this instruction.

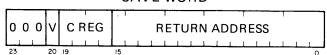
The Condition Register remains unchanged.

ADDRESS EXTENSION MODE SAVE WORD



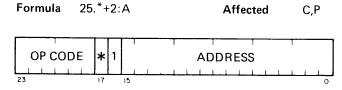
When the BSX is executed in the Address Extension Mode, the contents of the Condition Register are stored in bit positions 23-20 of the EMA location and the return address (program address of the next sequential instruction) is stored in bit positions 19-0.

COMPATIBILITY MODE SAVE WORD



In the Compatibility Mode, the final EMA may not exceed 16 bits when a BSX is executed, however, intermediate addresses may be 20 bits when the BSX is indirected.

BRL % Branch and Reset Interrupt Long



Operation

The highest-level active and enabled interrupt is reset (i.e., returned to the inactive state) and the contents of the P Register (current program address) are replaced by the effective memory address.

Notes

The BRL instruction is normally used to exit an interrupt subroutine.

In the Compatibility Mode, if the BRL contains an indirect reference, the last word in the indirect address chain contains the previous status of the virtual memory system in bit M20, the previous machine status (i.e., C Register contents at the time of the interrupt) in bit positions M19-M16, and the return address in bit positions M15-M0 as a result of the BSL instruction. The C Register is restored and the program branches to the return address (restarting the machine to the pre-interrupt status).

Example:

	•	
L	TMA AMA SMA	Interrupt occurs (EXM K).
K	BSL M	Dedicated interrupt location.
M	*** M	M becomes L+1 as a result of BSL at K. The C Register contents are stored in M19-M16.
	BRL M	Restore C Register and return to L+1.

In the Compatibility Mode, if an indirect BRL is executed in Monitor Mode, bit 20 of the effective memory address determines mode of operation to which machine returns. If bit 20 is set, User Mode is established; if reset, the Monitor Mode is established.

In the Address Extension Mode, if the BRL does not contain an indirect reference, the program branches to the return address and the states of VLR bit 20 and the C Register are unchanged. If the BRL is indirected (no indirect chaining is allowed), the destination address contains the previous machine status in bit positions M23-M20, and the return address in bit positions M19-M0 as a result of the BSX instruction. The C Register is restored and the program branches to the return address. VLR bit 20 remains unchanged if another interrupt is active and enabled. If no other interrupt is active and enabled,

VLR20 is reset. VLR bit 20 determines the mode of operation to which machine returns (if no other interrupt is active and enabled). If VLR20 is set, User Mode is established; if reset, the Monitor Mode is established.

In the Compatibility Mode, the final EMA may not exceed 16 bits when a BRL or extended BRL is executed. Intermediate address may be 20 bits when an indirect extended BRL is executed.

The immediate memory reference cannot be indexed; however, indexing indirect references is permitted, e.g.,



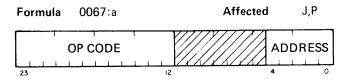
If the BRL instruction is not indirected, the Condition Register is not affected.

External interrupts are prohibited for the period of one instruction following this instruction.

The BRL will not reset the interrupt if external interrupts have been held by an HXI instruction. Control will be returned to the effective memory address.

Those executive traps, which are not affected by the HXI instruction, will be reset by the BRL.

BLU Branch and Link Unrestricted



Operation

The program address of the next sequential instruction replaces the contents of the J Register and the contents of the P Register (current program address) are replaced by the 5-bit immediate memory address.

Notes

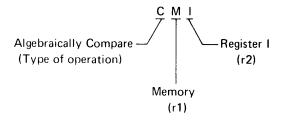
If virtual memory is enabled, execution of the BLU instruction will automatically establish the Monitor Mode. The 5-bit immediate memory address will not be mapped.

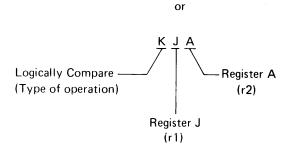
Bit 20 of the J Register will be set (ONE) if the system was in the User Mode, and reset (ZERO) if the Monitor Mode was active when the BLU was executed.

Compare Instructions

The compare group of instructions is composed of two basic types of operations; algebraic and logical comparisons. Both types of instructions compare two referenced quantities and set the Condition Register according to the result. Algebraic comparisons treat the references as signed (+ or -) quantities, while logical comparisons assume the references are unsigned quantities.

Algebraic comparisons are identified by the letter "C" as the first letter in the instruction mnemonic (e.g., CAI). Logical comparisons use a mnemonic code beginning with the letter "K" (KAI). The second letter of the mnemonic code designates the first of the compared quantities (r1) and the last letter designates the second quantity. For example:





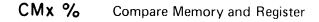
Both algebraic and logical comparisons are performed according to the formula:

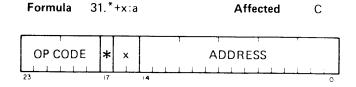
Therefore, r2 > r1, r2 < r1 and r2 = r1 will set the Condition Register (C) to positive (+), negative (-) and zero (0), respectively.

The following instructions are included in the compare group.

CMA	Compare Memory and A	7-25
CMB	Compare Memory and Byte	7-26
CME	Compare Memory and E	7-26
CMx	Compare Memory and Register	7-25
COB	Compare Operand and Byte	7-26
Crr	Compare Register and Register	7-27

CZD	Compare Zero and Double	7-27
CZM	Compare Zero and Memory	7-26
CZr	Compare Zero and Register	7-26
KOB	Kompare Operand and Byte	7-27
Krr	Kompare Register and Register	7-27





Operation

The contents of the effective memory address and the contents of the I, J, or K Register are algebraically compared.

Notes

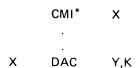
CMx is not a computer instruction mnemonic but represents a family of instruction mnemonics. x is coded as follows to select one of the index registers.

$$x = 1 (I)$$

2 (J)
3 (K)

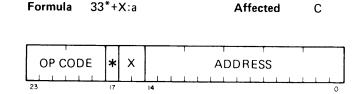
A code of 31.*+1:a, for example, implements the Compare Memory and I (CMI) instruction.

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,



The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

CMA % Compare Memory and A



0830007-000 Original 8/79

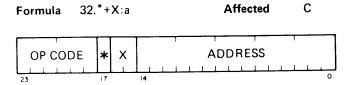
Operation

The contents of the effective memory address and the contents of the A Register are algebraically compared.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

CME % Compare Memory and E



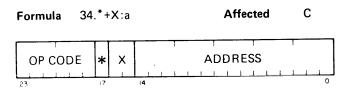
Operation

The contents of the effective memory address and the contents of the E Register are algebraically compared.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

CMB % Compare Memory and Byte



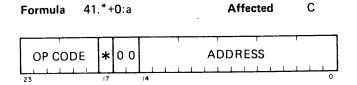
Operation

The contents of the B Register (A7-A0) and the contents of the effective memory address (M7-M0) are algebraically compared.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

CZM % Compare Zero and Memory



Operation

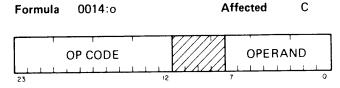
The contents of the effective memory address and zero are algebraically compared.

Notes

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

COB Compare Operand and Byte



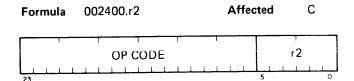
Operation

The 8-bit signed operand and the contents of the B Register (A7-A0) are algebraically compared.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

CZr Compare Zero and Register



Operation

The contents of the specified register and zero are algebraically compared.

Notes

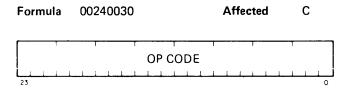
CZr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r2 is coded as follows to select any of the general purpose registers.

A code of 002400.01, for example, implements the Compare Zero and I (CZI) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

r2 is selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r2, they are logically ORed prior to the specified operation.

CZD Compare Zero and Double



Operation

The contents of the E Register are logically ORed with the contents of the A Register, and the result and zero are algebraically compared.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

Crr Compare Register and Register

Formula	0024.r1.r2	Affected	С



Operation

The contents of r1 and the contents of r2 are algebraically compared.

Notes

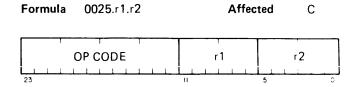
Crr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r1 and r2 are coded as follows to select any of the general purpose registers.

A code of 0024.01.02, for example, implements the Compare I and J (CIJ) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

r1 and r2 are selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r1, or r2, they are logically ORed prior to the specified operation.

Krr Kompare Register and Register



Operation

The contents of r1 and r2 are logically compared.

Notes

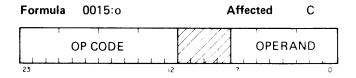
Krr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r1 and r2 are coded as follows to select any of the general purpose registers.

A code of 0025.01.02, for example, implements the Kompare I to J (KIJ) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

r1 and r2 are selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r1 and r2, they are logically ORed prior to the specified operation.

KOB Kompare Operand and Byte



Operation

The 8-bit operand and the contents of the B Register (A7-A0) are logically compared.

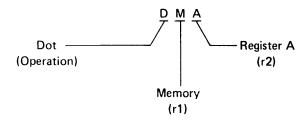
Note

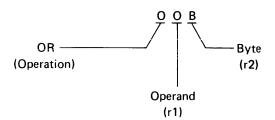
The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

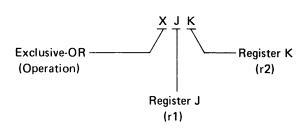
Logical Instructions

The logical group of instructions includes AND (Dot product), OR and exclusive-OR operations. All three types use two quantities to produce a logical result. The AND instructions use a mnemonic code beginning with the letter "D" for "Dot". The OR instructions use a mnemonic beginning with the letter "O", while exclusive-OR instructions are distinguished by the letter "X".

The second letter of the mnemonic code identifies the first of the two quantities (r1). The third letter signifies the second quantity (r2). Some examples are listed below.







Unless specifically noted otherwise in the individual descriptions, the result of the logical operation replaces the previous contents of r2 while r1 is unchanged. The Condition Register is set to the status of the result (Positive, Negative, or Zero) after the operation. The

various logical operations are illustrated in the following table.

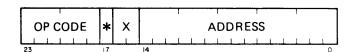
r1	r2	r1 AND r2	r1 OR r2	r1 XOR r2	
1	1	1	1	0	
0	1	0	1	1	
1	0	0	1	1	
0	0	0	0	0	

The following instructions are included in the logical group.

DMA	Dot Memory with A	7-28
DOB	Dot Operand with Byte	7-28
Drr	Dot Register with Register	7-29
OMA	OR Memory with A	7-29
OOB	OR Operand with Byte	7-29
Orr	OR Register with Register	7-29
XMA	Exclusive OR Memory with A	7-30
XOB	Exclusive OR Operand with Byte	7-30
Xrr	Exclusive OR Register with Register	7-30

DMA % Dot Memory with A

Formula 36.*+X:a Affected A,C



Operation

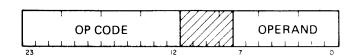
A logical AND is performed between the contents of the effective memory address and the contents of the A Register.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

DOB Dot Operand with Byte

Formula 0016:0 Affected A,C



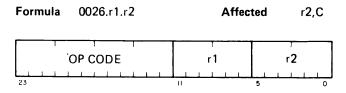
Operation

A logical AND is performed between the 8-bit operand and the contents of the B Register (A7-A0). Bits A23-A8 are unchanged.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in the Byte Register at the completion of the operation.

Drr Dot Register with Register



Operation

A logical AND is performed between the contents of r1 and r2.

Notes

Drr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r1 and r2 are coded as follows to select one of the general purpose registers.

A code of 0026.01.02, for example, implements the Dot I with J (DIJ) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

r1 and r2 are selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r1 or r2, they are logically ORed prior to the specified operation. The result is copied into all of the selected r2 registers.

OMA % OR Memory with A

Formula 3	5.*+X:a		Affected	A,C
OP CODE	* X		ADDRESS	
23	17	14		

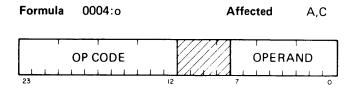
Operation

A logical OR is performed between the contents of the effective memory address and the contents of the A Register.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

OOB OR Operand with Byte



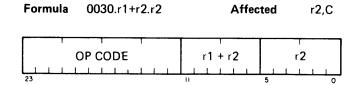
Operation

A logical OR is performed between the 8-bit operand and the contents of the B Register (A7-A0). Bits A23-A8 are unchanged.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in the Byte Register at the completion of the operation.

Orr OR Register with Register



Operation

A logical OR is performed between the contents of r1 and r2.

Notes

Orr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r1 and r2 are coded as follows to select any of the general purpose registers.

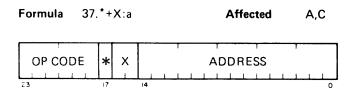
0830007-000 Original 8/79

A code of 0030.03.02, for example, implements the OR I with J (OIJ) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

r1 and r2 are selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r1, they are logically ORed prior to the specified operation. The result is copied into all of the selected r2 registers. Affected registers are the Condition Register and those selected in group r2.

XMA % Exclusive-OR Memory with A



Operation

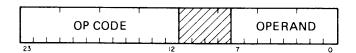
An exclusive-OR operation is performed between the contents of the effective memory address and the contents of the A Register.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

XOB Exclusive-OR Operand with Byte

Formula 0017:0 Affected A,C



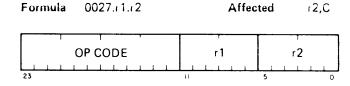
Operation

An exclusive-OR operation is performed between the 8-bit operand and the contents of the B Register (A7-A0). Bits A23-A8 are unchanged.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in the Byte Register at the completion of the operation.

Xrr Exclusive-OR Register with Register



Operation

An exclusive-OR function is performed between the contents of r1 and r2.

Notes

Xrr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r1 and r2 are coded as follows to select any of the general purpose registers.

A code of 0027.01.02, for example, implements the Exclusive-OR I with J (XIJ) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

r1 and r2 are selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r1 or r2, they are logically ORed prior to the specified operation. The result is copied into all of the selected r2 registers. Affected registers are the Condition Register and those selected in group r2.

Shift Instructions

The shift instruction group consists of arithmetic and logical shifts. The arithmetic shifts cause the contents of a register to be shifted left or right a specified number of times, while preserving the original sign. The logical shifts are similar to the arithmetic shifts, except that the sign bit is shifted along with the other bits.

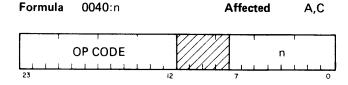
With both types of shift instructions, any number of shifts from 0 to 256 may be programmed without restriction. The number of shifts (n) are specified in bits 7-0 of the instruction word.

At the conclusion of any shift operation, the Condition Register is set to the status of the affected register contents (Positive, Negative, Zero).

The following instructions are included in the shift group.

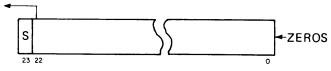
LAA	Left Shift Arithmetic A	7-31
LAD	Left Shift Arithmetic Double	7-31
LLA	Left Shift Logical A	7-31
LLD	Left Shift Logical Double	7-31
LRA	Left Rotate A	7-32
LRD	Left Rotate Double	7-32
RAA	Right Shift Arithmetic A	7-32
RAD	Right Shift Arithmetic Double	7-32
RLA	Right Shift Logical A	7-32
RLD	Right Shift Logical Double	7-33
RRA	Right Rotate A	7-33
RRD	Right Rotate Double	7-33

LAA Left Shift Arithmetic A



Operation

Bits A22-A0 are shifted left n places, with the most significant n bits being lost and n ZEROs being shifted into the least significant bit positions. The sign bit (A23) is unchanged.

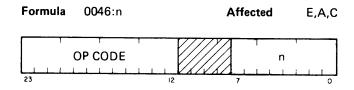


Notes

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

If a bit shifted off from A22 differs from the sign bit, the Condition Register will be set to Overflow. (This is in addition to the Positive/Negative/Zero status.)

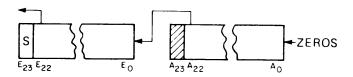
LAD Left Shift Arithmetic Double



Operation

Bits E22-E0 and A22-A0 are shifted, as one register, left n places. The most significant n bits are lost and the least

significant n bits are replaced with ZEROs. Bits E23 and A23 are bypassed. E23 is the D Register sign bit and A23 is not used in the double-precision format.

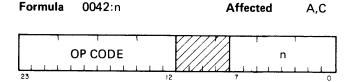


Notes

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

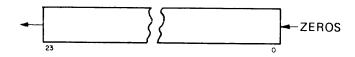
If a bit shifted off from E22 differs from the sign bit, the Condition Register will be set to Overflow. (This is in addition to the Positive/Negative/Zero status.)

LLA Left Shift Logical A



Operation

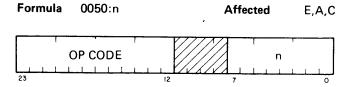
Bits A23-A0 are shifted left n places, with the most significant n bits being lost and the least significant n bits replaced by ZEROs.



Note

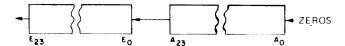
The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

LLD Left Shift Logical Double



Operation

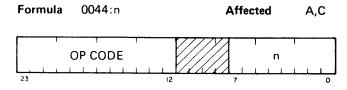
Bits E23-E0 and A23-A0 are shifted, as one register, left n places. The most significant n bits are lost and the least significant n bits are replaced with ZEROs.



Note

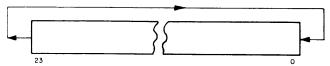
The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

LRA Left Rotate A



Operation

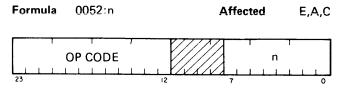
Bits A23-A0 are rotated left n places. No bits are lost.



Note

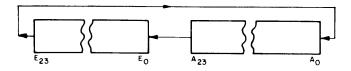
The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

LRD Left Rotate Double



Operation

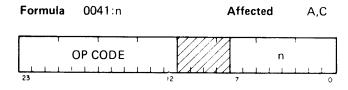
Bits E23-E0 and A23-A0 are rotated, as one register, left n places, with E23 replacing A0 and A23 replacing E0 as each shift takes place. No bits are lost.



Note

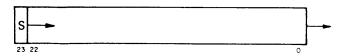
The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

RAA Right Shift Arithmetic A



Operation

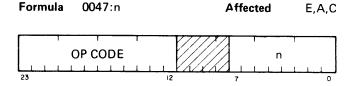
Bits A22-A0 are shifted right n places. The least significant n bits are lost and the most significant n bits are replaced by an extension of the sign bit (A23). The sign bit is not changed.



Note

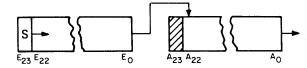
The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

RAD Right Shift Arithmetic Double



Operation

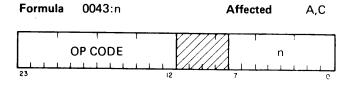
Bits E22-E0 and A22-A0 are shifted, as one register, right n places. The least significant n bits are lost and the most significant n bits are replaced by an extension of the sign bit (E23). Bit A23 is bypassed.



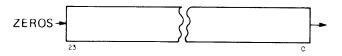
Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

RLA Right Shift Logical A



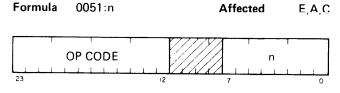
Bits A23-A0 are shifted right n places. The least significant n bits are lost and the most significant n bits are replaced by ZEROs.



Note

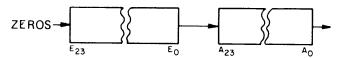
The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

RLD Right Shift Logical Double



Operation

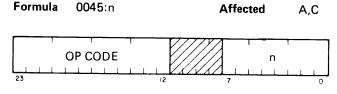
Bits E23-E0 and A23-A0 are shifted, as one register, right n places. The least significant n bits are lost and the most significant n bits are replaced by ZEROs.



Note

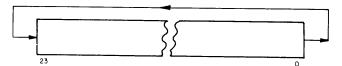
The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

RRA Right Rotate A



Operation

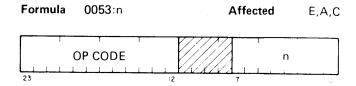
Bits A23-A0 are rotated right n places. No bits are lost.



Note

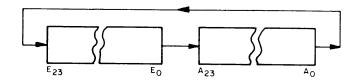
The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

RRD Right Rotate Double



Operation

Bits E23-E0 and A23-A0 are rotated, as one register, right n places, with E0 replacing A23 and A0 replacing E23 as each shift takes place. No bits are lost.



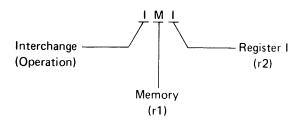
Note

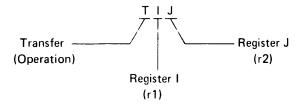
The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

Transfer Instructions

The transfer instruction group includes various types of operations. Among these are: interchanges between memory and a specified register, interchanges between registers, memory-to-register and register-to-memory transfers, and register-to-register transfers.

The mnemonic code for the transfer instruction describes the individual operation. The first letter of the mnemonic indicates what action is to be taken; "I" for interchange or "T" for transfer. The second and third letters specify the source (r1) and destination (r2), respectively. Some examples are listed below:





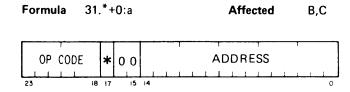
With the exception of the interchange instructions, the transfer group (r1) is not altered by the execution of any instructions in the transfer group.

The Condition Register is always set to reflect the status (Positive, Negative, or Zero) of the contents of r2, at the completion of the instruction.

The following instructions are included in the transfer group.

EMB	Extract Memory Byte	7-34
IMA	Interchange Memory and A	7-35
IME	Interchange Memory and E	7-35
IMx	Interchange Memory and Register	7-35
Irr	Interchange Register and Register	7-35
RBM	Replace Byte in Memory	7-36
TAM	Transfer A to Memory	7-41
TBM	Transfer Byte to Memory	7-41
TDM	Transfer Double to Memory	7-41
TEM	Transfer E to Memory	7-41
TFM	Transfer Flag to Memory	7-41
TIM	Transfer I to Memory	7-42
TJM	Transfer J to Memory	7-42
TKM	Transfer K to Memory	7-42
TLK	Transfer Extended Operand to K	7-39
TLO	Transfer Long Operand to K	7-39
TMA	Transfer Memory to A	7-37
TMB	Transfer Memory to Byte	7-36
TMD	Transfer Memory to Double	7-36
TME	Transfer Memory to E	7-37
TMI	Transfer Memory to I	7-37
TMJ	Transfer Memory to J	7-38
TMK	Transfer Memory to K	7-38
TMQ	Transfer Memory to Query Register	7-37
TMR	Transfer Memory to Registers	7-38
TNr	Transfer Negative Operand to Register	7-38
TOB	Transfer Operand to Byte	7-38
TOC	Transfer Operand to Condition Register	7-39
TOr	Transfer Operand to Register	7-39
TrB	Transfer Register to Byte	7-40
TRM	Transfer Registers to Memory	7-42
Trr	Transfer Register to Register	7-42
TSr	Transfer Switches to Register	7-40
TZM	Transfer Zero to Memory	7-41
TZr	Transfer Zero to Register	7-40
	=	

EMB % **Extract Memory Byte**



Operation

The effective memory address is added to the contents of the J Register, producing the word address which contains the byte to be extracted. The selected byte, as determined by the contents of bits 23 and 22 of the index J Register, is then placed in the B Register.

Notes

The following table shows the correspondence between bits 23 and 22 of J and the byte to be extracted.

Bits 23 and 22 J Register	Byte Selection
01	Leftmost Byte (bits 23-16 of EMA+J)
10	Middle byte (bits 15-8 of EMA+J)
11	Rightmost byte (bits 7-0 of EMA+J)
00	Rightmost byte (bits 7-0 of EMA+J)

The final address of any indirect/index sequence is algebraically added to the contents of the J Register.

If indirection is specified, PC mapping occurs normally for the generation of the indirect address. If indirection is not specified, the implied index register is added to the specified address with PC mapping following the rules established for Compatibility or Address Extension Modes.

Examples:

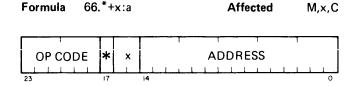
If J '40000030 and K = '00000010 when the following is executed:

	EMB*	' 40
'40	DAC*	'50,K
'42	DATA	"XYZ"
' 60	DAC	'12

then the character Y will be placed in the B Register. Note that the effective address of the indirect/index sequence is '12. However, '12 plus bits 15-0 of index J Register ('30) yields the final address of '42. Since a byte specification of 102 was made in bits 23-22 of index J Register, then the second byte (bits 15-8) of memory location '42 is placed in the B Register.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

IMx % Interchange Memory and Register



Operation

The contents of the effective memory address and the I, J, or K Register are interchanged.

Notes

IMx is not a computer instruction mnemonic but represents a family of instruction mnemonics. x is coded as follows to select one of the index registers.

$$x = 1 (I)$$

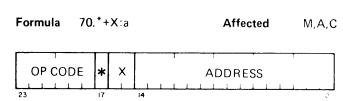
2 (J)
3 (K)

A code of 66*+1:a, for example, implements the Interchange Memory and I (IMI) instruction.

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.

The Condition Register is set to Positive, Negative, or Zero, based on the result in I, J, or K at the completion of the operation.

IMA % Interchange Memory and A



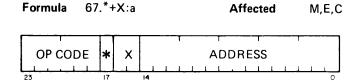
Operation

The contents of the effective memory address and the A Register are interchanged.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in A at the completion of the operation.

IME % Interchange Memory and E



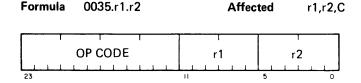
Operation

The contents of the effective memory address and the E Register are interchanged.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in E at the completion of the operation.

Irr Interchange Register and Register



Operation

The contents of r1 and r2 are interchanged.

Notes

Irr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r1 and r2 are coded as follows to select any of the general purpose registers.

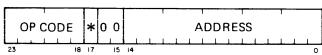
A code of 0035.01.02, for example, implements the Interchange I and J (IIJ) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result in r2 at the completion of the operation.

r1 and r2 are selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r1 or r2 they are logically ORed prior to the specified operation. The result is copied into all of the selected r2 and r1 registers. Affected registers are the Condition Register and those selected in group r1 or r2.

RBM % Replace Byte in Memory





Operation

The effective memory address is added to the contents of the I Register producing the word address which contains the byte to be replaced. The selected byte, as determined by the contents of bits 22 and 23 of the index I Register, is then replaced by the contents of the B Register.

Notes

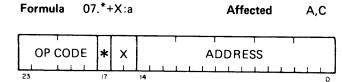
The following table shows the correspondence between bits 22 and 23 of I and the byte to be replaced.

Bits 23 and 22 I Register	Byte Selection
01	Leftmost byte (bits 23-16 of EMA+I)
10	Middle byte (bits 15-8 of EMA+I)
11	Rightmost byte (bits 7-0 of EMA+I)
00	Causes no operation

The final address of any indirect/index sequence is added algebracially to the contents of the I Register.

If indirection is specified, PC mapping occurs normally for the generation of the indirect address. If indirection is not specified, the implied index register is added to the specified address with PC mapping following the rules established for Compatibility or Address Extension Modes.

TMB % Transfer Memory to Byte



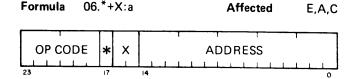
Operation

The 8 least significant bits (7-0) of the contents of the effective memory address replace the previous contents of the B Register (A7-A0). Bits A23-A8 are unaffected.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in the B Register at the completion of the operation.

TMD % Transfer Memory to Double

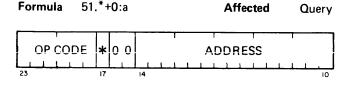


The contents of the effective memory address (EMA) and the next sequential address (EMA+1) replace the previous contents of the D Register (E and A). EMA and EMA+1 are transferred to E and A, respectively.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in D at the completion of the operation.

TMQ % Transfer Memory to Query Register



Operation

Bits 23, 22, 21 and 19-0 of the contents of the effective memory address replace the previous contents of the Query Register. These bits are loaded into the Query Register in bit positions 23, 22, 21, and 19-0, respectively.

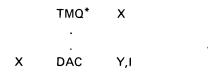
Notes

Executing this instruction will cause the Address Trap to be enabled or disabled, depending on the states of bits 23, 22, and 21 of the effective memory address.

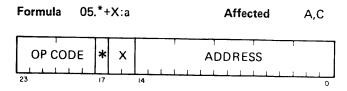
Example:

	TMQ	OA	
OA	DAC	ADDR	Enable Address Trap
OA	or DAC*	0	Disable Address Trap

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,



TMA % Transfer Memory to A



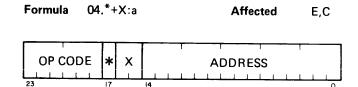
Operation

The contents of the effective memory address replace the previous contents of the specified register.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in A at the completion of the operation.

TME % Transfer Memory to E



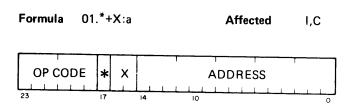
Operation

The contents of the effective memory address replace the previous contents of the specified register.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in E at the completion of the operation.

TMI % Transfer Memory to I



0830007-000 Original 8/79

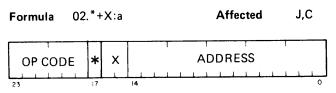
Operation

The contents of the effective memory address replace the previous contents of the specified register.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in I at the completion of the operation.

TMJ % Transfer Memory to J



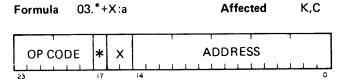
Operation

The contents of the effective memory address replace the previous contents of the specified register.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in J at the completion of the operation.

TMK % Transfer Memory to K



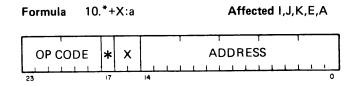
Operation

The contents of the effective memory address replace the previous contents of the specified register.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in K at the completion of the operation.

TMR % Transfer Memory to Registers



Operation

The I, J, K, E and A Registers are loaded from consecutive

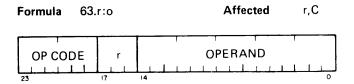
memory addresses beginning with the effective memory address.

Note

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

An indexed TMR instruction will not execute properly if a demand page occurs during the execution of the instruction.

TNr Transfer Negative Operand to Register



Operation

The two's complement of the 15-bit unsigned operand replaces the previous contents of bits 23-0 of the specified register.

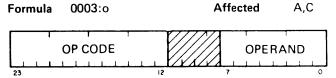
Notes

TNr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r is coded as follows to select one of the general purpose registers.

A code of 63.1:o, for example, implements the Transfer Negative Operand to I (TNI) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result in the specified register at the completion of the operation.

TOB Transfer Operand to Byte



Operation

The 8-bit signed operand replaces the previous contents of the B Register (A7-A0). Bits A23-A8 are unaffected.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in the Byte Register at the completion of the operation.

TOC Transfer Operand to Condition Register

Affected С OP CODE **DPERAND**

Operation

Formula

The 4-bit operand replaces the previous contents of the Condition Register.

Note

Operand definition is as follows:

0036:o

Bit 0 = ONE = Overflow = ZERO = No Overflow Bit 1 = ONE = Negative = ZERO = Not Negative Bit 2 = ONE = Zero = ZERO = Not Zero Bit 3 = ONE = Positive = ZERO = Not Positive

TOr Transfer Operand to Register

Formula 62.r:o **Affected** r,C **OPERAND** OP CODE

Operation

The 15-bit unsigned operand replaces the previous contents of bits 23-0 of the specified register.

Notes

TOr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r is coded as follows to select one of the general purpose registers.

> r = 1 (1)2 (J) 3 (K)

4 (E) 5 (A)

6 (T)

A code of 62.1:o, for example, implements the Transfer Operand to I (TOI) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result in the specified register at the completion of the operation.

TLO Transfer Long Operand to K

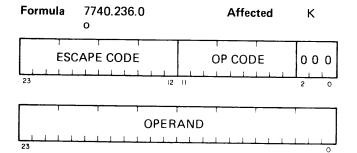
Formula 236:o **Affected** Κ OP CODE **OPERAND**

Operation

In the Compatibility Mode, the 16-bit operand replaces the previous contents of bits 15-0 of the K Register. Bits 23-16 of K are cleared (reset to ZEROs).

In the Address Extension Mode, if bit 15 is set (ONE), the operand is assumed to be a long absolute quantity which is transferred to the K Register. Bits 23-16 of K are cleared. If bit 15 is reset (ZERO), the 16-bit operand is assumed to be a local address which requires map resolution. Bits 19-15 of the Program Counter are appended to bits 14-0 of the operand and the 20-bit result is then transferred to K. Bits 23-20 of K are cleared.

TLK Transfer Extended Operand to K



Operation

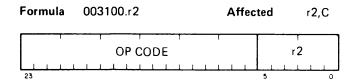
The 24-bit operand of the second word replaces the previous contents of the K Register.

Notes

The TLK instruction is valid only in the extended instruction format.

The Condition Register remains unchanged.

TSr Transfer Switches to Register



Operation

The states (set = ONE) of the switch register switches are transferred to the corresponding bit positions of the specified register.

Notes

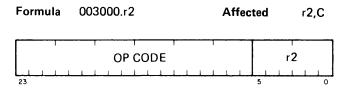
TSr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r2 is coded as follows to select any of the general purpose registers.

A code of 003100.01, for example, implements the Transfer Switches to I (TSI) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result in the specified register at the completion of the operation.

r2 is selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r2, the switches are copied into all of the selected r2 registers. Affected registers are the Condition Register and those selected in group r2.

TZr Transfer Zero to Register



Operation

The previous contents of the specified register are replaced with ZEROs.

Notes

TZr is not a computer instruction mnemonic but represents a family of instruction mnemonics. r2 is coded as follows to select any of the general purpose registers or the D register.

A code of 003000.01, for example, implements the Transfer Zero to I (TZI) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result in the specified register at the completion of the operation.

r2 is selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r2, they are logically ORed prior to the specified operation. The result is copied into all of the selected r2 registers. Affected registers are the Condition Register and those selected in group r2.

TrB Transfer Register to Byte



Operation

The least significant 8 bits (7-0) of the contents of the specified register replace the previous contents of the B Register (A7-A0). Bits A23-A8 are unchanged.

Notes

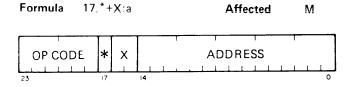
TrB is not a computer instruction mnemonic but represents a family of instruction mnemonics. r1 is coded as follows to select one-of-five general purpose registers.

A code of 0002.01, for example, implements the Transfer I to Byte (TIB) instruction.

The Condition Register is not affected.

r1 is selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r1, they are logically ORed prior to the specified operation.

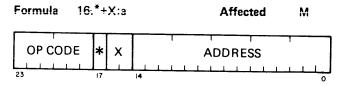
TBM % Transfer Byte to Memory



Operation

The contents of the B Register (A7-A0) replace the 8 least significant bits of the contents of the effective memory address. Bits 23-8 of the memory word are unaffected.

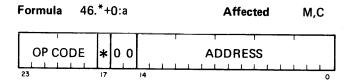
TDM % Transfer Double to Memory



Operation

The contents of the D Register (E and A) replace the previous contents of the effective memory address (EMA) and the next sequential address (EMA+1). The contents of E and A are transferred to EMA and EMA+1, respectively.

TFM % Transfer Flag to Memory



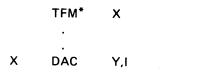
Operation

The previous contents of the effective memory address are replaced by ONEs.

Notes

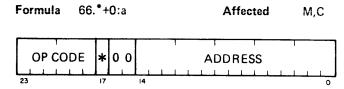
The Condition Register is set to the status of memory (Positive, Negative, or Zero) prior to the transfer.

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,



DMA transfers are inhibited and shared memory is locked up during the execution of this instruction.

TZM % Transfer Zero to Memory



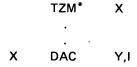
Operation

The previous contents of the effective memory address are replaced by ZEROs.

Notes

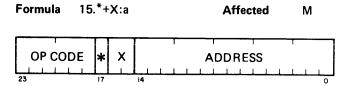
The Condition Register is set to the status of memory (Positive, Negative, or Zero) prior to the transfer.

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,



DMA transfers are inhibited and shared memory is locked up during the execution of this instruction.

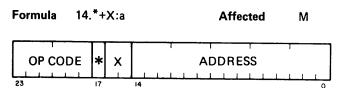
TAM % Transfer A to Memory



Operation

The contents of the A Register replace the previous contents of the effective memory address.

TEM % Transfer E to Memory

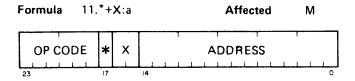


Operation

The contents of the E Register replace the previous contents of the effective memory address.

TIM %

Transfer I to Memory

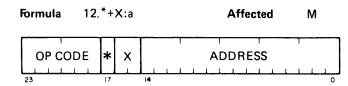


Operation

The contents of the I Register replace the previous contents of the effective memory address.

TJM %

Transfer J to Memory



Operation

The contents of the J Register replace the previous contents of the effective memory address.

TKM %

Transfer K to Memory

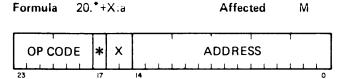


Operation

The contents of the K Register replace the previous contents of the effective memory address.

TRM %

Transfer Registers to Memory



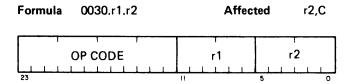
Operation

The contents of the I, J, K, E and A Registers are stored in consecutive memory locations beginning with the effective memory address.

Note

External interrupts are prohibited for the period of one instruction following the execution of this instruction.





Operation

The contents of r1 replace the previous contents of r2.

Notes

Trr is not a computer instruction mnemonic but represents a family of instruction mnemonics, r1 and r2 are coded as follows to select any of the general purpose registers.

A code of 0030.01.02, for example, implements the Transfer I to J (TIJ) instruction.

The Condition Register is set to Positive, Negative, or Zero, based on the result in r2 at the completion of the operation.

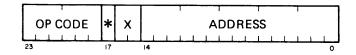
Byte Processing Instructions

The byte processing group of instructions permits program manipulation of all three bytes within the computer word (24 bits); e.g., extract, replace, etc. The following instructions are inclusive of byte processing operations.

AMB	Add Memory to Byte	7-43
AOB	Add Operand to Byte	7-43
BBI	Branch when Byte address +1 in I≠0	7-43
BBJ	Branch when Byte address +1 in $J\neq 0$	7-44
CMB	Compare Memory and Byte	7-45
COB	Compare Operand and Byte	7-45
DOB	Dot Operand with Byte	7-45
EMB	Extract Memory Byte	7-45
ESB	Extend Sign of Byte	7-46
EZB	Extend Zeros from Byte	7-46
KOB	Kompare Operand and Byte	7-46
NBB	Negate of Byte to Byte	7-46
OOB	OR Operand with Byte	7-46
PBB	Positive of Byte to Byte	7-46
RBM	Replace Byte in Memory	7-47
QBB	Query Bits of Byte	7-47
SMB	Subtract Memory from Byte	7-47
SOB	Subtract Operand from Byte	7-47
TBM	Transfer Byte to Memory	7-48
TOB	Transfer Operand to Byte	7-48
TMB	Transfer Memory to Byte	7-48
TrB	Transfer Register to Byte	7-48
XOB	Exclusive-OR Operand with Byte	7-48

AMB % Add Memory to Byte

Formula 45.*+X:a Affected A,C



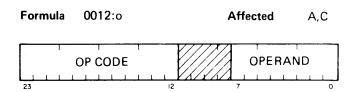
Operation

Bits 7-0 of the contents of the effective memory address are algebraically added to the contents of the B Register (A7-A0). Bits 23-8 of the A Register are unchanged.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

AOB Add Operand to Byte



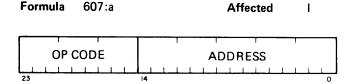
Operation

The 8-bit signed operand is algebraically added to the contents of the B Register (A7-A0). Bits 23-8 of the A Register are unchanged.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

BBI % Branch when Byte Address +1 in I ≠ 0



Operation

The contents of bits 22 and 23 of the I Register are incremented by one. If the result of this addition (in bits 22 and 23) is not 002, then the contents of the P Register (current program address) are replaced by the effective memory address. If the result of the addition to bits 22 and 23 is 002, then bits 22 and 23 are set to 012 and bits 21-0 are incremented by one. If the resultant sum in bits 21-0 is zero, then the P Register advances to the next sequential program location and the index register is set to 200000008. Otherwise, the contents of the P Register are replaced by the effective memory address.

Notes

In general, the BBI and BBJ instructions are used as special index register increments in order to sequentially reference consecutive bytes in memory via the EMB and RBM instructions. Consider the following example which will move 11 consecutive bytes starting from the third byte at location '200 to the first byte at location '300.

0830007-000 Original 8/79

TMJ = '60000200
TMI = '20000300
TNK 11
EMB 0
RBM 0
BBI *+1
BBJ *+1
BWK *-4

Occasionally, it is possible to use the address of a portion of the I Register as a byte counter as well as a word pointer. This may be illustrated by the following example which will set the buffer to blanks, starting at byte 3 of location '100 through byte 3 of location '102.

However, it should be noted that this technique of using the index register as both a byte counter and word pointer may be used only in certain instances. Specifically, when the following relationship is true.

$$R\left(\frac{4 \cdot B.n.}{3}\right) = R\left(\frac{CT}{3}\right)$$

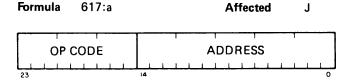
Where:

R () = remainder

B.n. = the starting byte number
(1, 2, or 3)

CT = the number of bytes to be referenced

BBJ % Branch when Byte Address +1 in $J \neq 0$



Operation

The contents of bits 22 and 23 of the J Register are incremented by one. If the result of this addition (in bits 22 and 23) is not 00₂, then the contents of the P Register (current program address) are replaced by the effective memory address. If the result of the addition to bits 22 and 23 is 00₂, then bits 22 and 23 are set to 01₂

and bits 21-0 are incremented by one. If the resultant sum in bits 21-0 is zero, then the P Register advances to the next sequential program location and the index register is set to 20000008. Otherwise, the contents of the P Register are replaced by the effective memory address.

Notes

In general, the BBI and BBJ instructions are used as special index register increments in order to sequentially reference consecutive bytes in memory via the EMB and RBM instructions. Consider the following example which will move 11 consecutive bytes starting from the third byte at location '200 to the first byte at location '300.

TMJ	= '60000200
TMI	= '20000300
TNK	11
EMB	0
RBM	0
BBI	*+1
BBJ	*+1
BWK	*-4

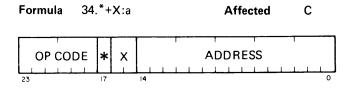
Occasionally, it is possible to use the address of a portion of the J Register as a byte counter as well as a word pointer. This may be illustrated by the following example which will set the buffer to blanks, starting at byte 3 of location '100 through byte 3 of location '102.

However, it should be noted that this technique of using the index register as both a byte counter and word pointer may be used only in certain instances. Specifically, when the following relationship is true.

$$R\left(\frac{4-B.n.}{3}\right) = R\left(\frac{CT}{3}\right)$$

Where:

CMB % Compare Memory and Byte



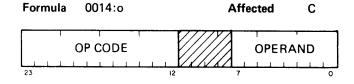
Operation

The contents of the B Register (A7-A0) and the contents of the effective memory address (M7-M0) are algebraically compared.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

COB Compare Operand and Byte



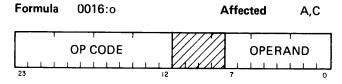
Operation

The 8-bit signed operand and the contents of the B Register (A7-A0) are algebraically compared.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

DOB Dot Operand with Byte

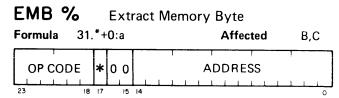


Operation

A logical AND is performed between the 8-bit operand and the contents of the B Register (A7-A0). Bits A23-A8 are unchanged.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in the Byte Register at the completion of the operation.



Operation

The effective memory address is added to the contents of the J Register, producing the word address which contains the byte to be extracted. The selected byte, as determined by the contents of bits 23 and 22 of the index J Register, is then placed in the B Register.

Notes

The following table shows the correspondence between bits 23 and 22 of J and the byte to be extracted.

Bits 23 and 22 J Register	Byte Selection
01	Leftmost byte (bits 23-16 of EMA+J)
10	Middle byte (bits 15-8 of EMA+J)
11	Rightmost byte (bits 7-0 of EMA+J)
00	Rightmost byte (bits 7-0 of EMA+J)

The final address of any indirect/index sequence is algebraically added to the contents of the J Register.

If indirection is specified, PC Mapping occurs normally for the generation of the indirect address. If indirection is not specified, the implied index register is added to the specified address with PC mapping following the rules established for Compatibility or Address Extension Modes.

Examples:

	If J =	'40000030
	and K =	'0000010 when the following
		is executed:
	EMB*	'40
40	DAC*	'50,K
42	DATA	"XYZ"
60	DAC	'12

then the character Y will be placed in the B Register. Note that the effective address of the indirect/index sequence is '12. However, '12 plus bits 15-0 of index J Register ('30) yields the final address of '42. Since a byte specification of 102 was made in bits 23-22 of index J Register, then the second byte (bits 15-8) of memory location '42 is placed in the B Register.

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

ESB Extend Sign of Byte

Formula 0010. Affected A,C

OP CODE

12

0

Operation

The state of the B Register sign bit (A7) is copied into bit positions A23-A8, forming a sign extension of the byte.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in A at the completion of the operation.

EZB Extend Zeros from Byte

Formula 0007. Affected A

OP CODE

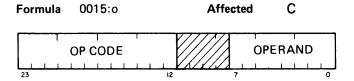
Operation

Bit positions A23-A8 are set to ZERO. The contents of the B Register (A7-A0) are not affected.

Note

The Condition Register is not affected.

KOB Kompare Operand and Byte



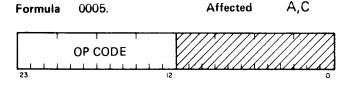
Operation

The 8-bit operand and the contents of the B Register (A7-A0) are logically compared.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation.

NBB Negate of Byte to Byte



Operation

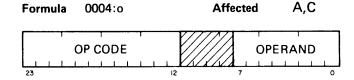
The contents of the B Register (A7-A0) are two's complemented. Bit positions A23-A8 are unchanged.

Notes

An Overflow will result when negating 27 (full-scale negative byte).

The Condition Register is set to Positive, Negative, or Zero, based on the result in the Byte Register at the completion of the operation.

OOB OR Operand with Byte



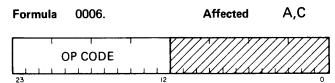
Operation

A logical OR is performed between the 8-bit operand and the contents of the B Register (A7-A0). Bits A23-A8 are unchanged.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in the Byte Register at the completion of the operation.

PBB Positive of Byte to Byte



Operation

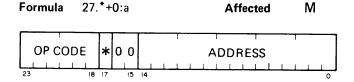
The absolute value of the contents of the B Register (A7-A0) is placed in the B Register.

Notes

An Overflow will result when negating a full scale negative byte.

The Condition Register is set to Positive, Negative, or Zero, based on the result in the Byte Register at the completion of the operation.

RBM % Replace Byte in Memory



Operation

The effective memory address is added to the contents of the I Register producing the word address which contains the byte to be replaced. The selected byte, as determined by the contents of bits 22 and 23 of the Index I Register, is then replaced by the contents of the B Register.

Notes

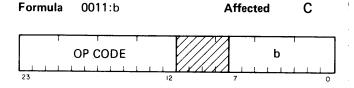
The following table shows the correspondence between bits 22 and 23 of I and the byte to be replaced.

Bits 23 and 22 I Register	Byte Selection
01	Leftmost byte (bits 23-16 of EMA+I)
10	Middle byte (bits 15-8 of EMA+I)
11	Rightmost byte (bits 7-0 of EMA+I)
00	Causes no operation

The final address of any indirect/index sequence is added algebraically to the contents of the I Register.

If indirection is specified, PC mapping occurs normally for the generation of the indirect address. If indirection is not specified, the implied index register is added to the specified address with PC mapping following the rules established for Compatibility or Address Extension Modes.

QBB Query Bits of Byte



Operation

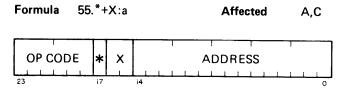
A logical AND is performed between operand bits 7-0 and the contents of the B Register. The Condition Register is set according to the status of the result; i.e., Positive, Negative, or Zero.

Note

Examples:

(1)	TOA 	В7	A = '00000200	C = Positive
	QBB	В7		C = Negative
(2)	TOA	B6	A = '00000100	C = Positive
	QBB	В6		C = Positive
(3)	TNA ···	1	A = '77777777	C = Negative
	DMA 	MASK	A = '40000000	C = Negative
MASK	DATA	4000000	00	

SMB % Subtract Memory from Byte



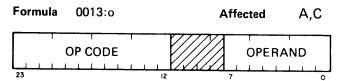
Operation

The contents of bits 7-0 of the effective memory address are algebraically subtracted from the B Register (A7-A0). Bits A23-A8 are unaffected.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

SOB Subtract Operand from Byte



The 8-bit signed operand is algebraically subtracted from the contents of the B Register (A7-A0). Bits A23-A8 are unaffected.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out without a carry in.

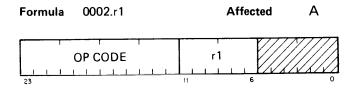
TBM % Transfer Byte to Memory



Operation

The contents of the B Register (A7-A0) replace the 8 least significant bits (7-0) of the contents of the effective memory address. Bits 23-8 of the memory word are unaffected.

TrB Transfer Register to Byte



Operation

The least significant 8 bits (7-0) of the contents of the specified register replace the previous contents of the B Register (A7-A0). Bits A23-A8 are unchanged.

Notes

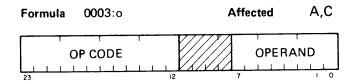
TrB is not a computer instruction mnemonic but represents a family of instruction mnemonics. r1 is coded as follows to select one-of-five general purpose registers.

A code of 0002.01, for example, implements the Transfer I to Byte (TIB) instruction.

The Condition register is not affected.

r1 is selected by unitary bits. Therefore, none, all six, or any combination of registers may be selected. If more than one register is selected in group r1, they are logically ORed prior to the specified operation.

TOB Transfer Operand to Byte



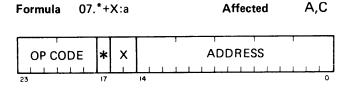
Operation

The 8-bit signed operand replaces the previous contents of the B Register (A7-A0). Bits A23-A8 are unaffected.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in the Byte Register at the completion of the operation.

TMB % Transfer Memory to Byte



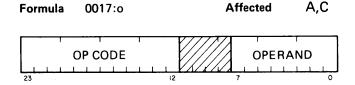
Operation

The 8 least significant bits (7-0) of the contents of the effective memory address replace the previous contents of the B Register (A7-A0). Bits A23-A8 are unaffected.

Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in the B Register at the completion of the operation.

XOB Exclusive-OR Operand with Byte



An exclusive-OR operation is performed between the 8-bit operand and the contents of the B Register (A7-A0). Bits A23-A8 are unchanged.

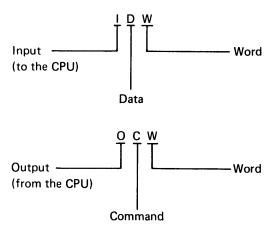
Note

The Condition Register is set to Positive, Negative, or Zero, based on the result in the Byte Register at the completion of the operation.

Input/Output Instructions

The input/output (I/O) instructions provide the required control for all communications between the CPU and the input/output structure. In addition to controlling data transfers between the CPU and peripheral units, the I/O instructions allow peripheral unit command functions and status testing to be placed under program control.

The specific I/O operation can be identified by examination of the individual instruction mnemonic. All I/O instruction mnemonics use the letter "W" to indicate that a full word is to be transferred between the CPU and the I/O structure. The first letter of the mnemonic indicates the direction of the transfer (input or output). The second letter indicates the type of word to be transferred. For example:



There is no "I/O hold", or delay, imposed by the hardware. All I/O instructions are executed unconditionally, i.e., the CPU is not forced to wait for a response from the I/O structure in order to complete the instruction execution cycle.

Although there is no built-in hold/delay provision, a programmed delay can be implemented if desired. At the beginning of each I/O instruction cycle, the Condition Register is cleared. At the end of the execution phase of each I/O instruction, bit 2 (Zero/Not Zero) is set to Zero if the selected channel was ready and accepted the command. If the selected channel was not ready, bit 2 of the

Condition Register remains set to Not Zero. The program can test the Not Zero state of bit 2 with a branch instruction following the I/O instruction. When bit 2 is set to Not Zero, a programmed delay is implemented. For example:

ODW '0103 Output word to Channel 1, Unit 3
BNZ *-1 Delay if not ready
--- Continue if ready

An example of a channel being not ready is when the peripheral unit's data transfer capability is slower than that of the program loop and therefore cannot accept data as it is available from the channel. Another example occurs in a channel/multiunit environment where the channel is connected to peripheral unit A and peripheral unit B is selected for a data transfer.

In this instance, the channel remains not ready until a disconnect/connect sequence is performed and peripheral unit B is connected to the channel. Two cycles are required for the disconnect/connect sequence.

Status returned to the Condition Register immediately after completion of an I/O instruction refers to channel status only. A ready (Zero) condition indicates the channel accepted the I/O command. This does not imply the I/O operation was completed with the selected peripheral unit.

If the program selects a non-existent channel or unit, the channel accepts the command or data and leaves bit 2 of the Condition Register set to Not Zero to indicate not ready. The channel will remain not ready for any subsequent commands.

Channel number 30g cannot be assigned to an I/O channel.

The I/O command modes are determined by the configuration of bits 5 and 4 of the OCW instruction and are as follows:

- Normal The Normal Channel Operation command is raised by bits 5 and 4 of the OCW being ZEROs (0,0).
- Multiplex-Output Special Function This command is raised by bits 5 and 4 of the OCW being a ZERO, ONE (0,1) configuration.

This is a multiplex function for the PIOC which, when executed by the CPU, releases the channel to a master/slave pair of peripheral units.

For the BBC, this is an Output Special Function which causes the channel to modify its normal operation. The CPU transfers the contents of the A Register to the Special Function Register in the BBC. The bits unitarily control the special functions. The special functions are reset unitarily by issuing an OCW with the respective A Register bit reset to zero. The bits may also be reset by a Master Clear. This instruction is not accepted while a block transfer is in process. Definition of the bits when active (set) is as follows:

Bit 0 Decrement TAR — This function provides a magnetic tape read reverse capability, alleviating the requirement to rewind the tape. This causes the channel to decrement, rather than increment, the transfer address during input block transfers. The 24-bit words from the unit are assembled into 48-bit memory words in reverse order.

Bit 1 Unit I/O Parity Check — This function enables channel-to-unit parity checking. The BBC provides an odd parity on commands and data to the unit, and it checks for odd parity on data from the unit. Since some units do not provide parity, the parity checking function has to be specifically enabled. Parity errors on block or single word (IDW) transfers are reported in bit 9 of the status word. The channel remains busy after a parity error is detected, and must be cleared to resume normal operation.

Bit 2 Internal Turnaround - This function causes the channel to transfer data in and out of the channel data buffer without accessing or signaling the unit. This function is used for block transfers only (maximum word count = 32). Incorrect results will occur if the Buffer Count Not Zero Flag is not reset prior to enabling internal turnaround. The flag can be reset by initializing an input block transfer to a non-existing unit by means of an OCW instruction, incurring a delay of at least 30 CPU cycles (e.g., executing 30 NOPs), and then aborting the transfer. Executing an OCW with the mode bits in the Offline configuration will abort the transfer. Any normal block transfer termination will also reset the flag. The state of the Buffer Count Not Zero Flag is reported in the status word.

Bit 3 Unit Master Clear — This function provides a continuous Master Clear signal to the unit interface.

Bits 4-23 Not used (reset to Zeros).

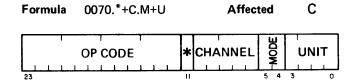
An XBC, IBC, or DM ACP-8 channel will not respond to a Multiplex-Output Special Function command.

- Offline This command is the same as the Multiplex command, except the I/O drivers in the channel are turned off, allowing the second CPU to share peripherals without need of peripheral switches. (Assumes control of I/O bus.) The command is raised by bits 5 and 4 being in a ONE, ZERO (1,0) configuration.
- Reset This command operates the same as a Normal command, but resets the channel out of either the Multiplex or Offline mode. (Channel restored on-line, unit selected.) This command is raised by bits 5 and 4 being in a ONE, ONE (1,1) configuration.

The following instructions are included in the input/output group.

IAW	Input Address Word	7-52
IDW	Input Data Word	7-52
IPW	Input Parameter Word	7-53
ISW	Input Status Word	7-51
OAW	Output Address Word	7-52
OCW	Output Command Word	7-50
ODW	Output Data Word	7-51

OCW Output Command Word



Operation

An 8-bit or a 24-bit command word is transferred from the A Register to the specified channel/unit combination.

Notes

The Condition Register is cleared, then set to Zero if the I/O channel is ready. If the selected channel is not ready, the Condition Register remains set to Not Zero which allows a programmed delay if desired.

Bits 3-0 of the OCW instruction form a 4-bit paralleled unit code that is used to select a particular peripheral unit. The configuration of bits 4 and 5 determines the Normal, Multiplex-Output Special Function, Offline, or Reset Channel mode for a particular channel. The configuration of bits 10-6 determines which channel is selected. Bit 11 is the Override Bit, and bits 23-12 define the general process that is to be performed. The only valid unit code for a DMACP-8 channel is 10g; all others are rejected.

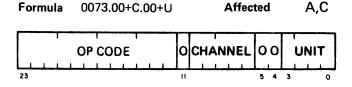
If the Override Bit (*) is set (ONE), the command word assumes immediate control over the channel. The contents of the A Register are transferred to the channel and a disconnect/connect sequence is initiated. The Condition Register is set to Zero to indicate the channel has accepted but not necessarily executed the command. Upon completion of the disconnect/connect sequence, the channel transfers the command word to the unit. In the case of a DMACP-8 channel, the Override bit clears the channel and forces the MPU to a halt; the Condition Register is not set to Zero, and no busy test is required.

If the Override Bit is not set (ZERO) and the OCW specifies a unit other than the unit connected to the channel and the channel is ready, the command word is accepted by the channel. The Condition Register is set to Not Zero to indicate the channel is not ready. A disconnect/connect sequence is performed and the command is transferred to the unit. The Condition Register is reset to Zero to indicate ready.

Following the execution of an OCW the channel remains not ready until the peripheral unit accepts the data.

This instruction is privileged.

ISW Input Status Word



Affected

Formula

A status word is transferred from the specified channel/unit combination to the A Register.

Notes

The Condition Register is cleared, then set to Zero if the I/O channel is ready. If the addressed channel/unit combination is not ready (see following notes) or status word is not available, the Condition Register is set to Not Zero to allow a programmed delay.

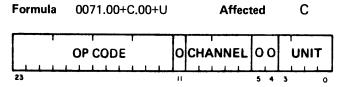
If the selected channel is in the process of executing a command (resulting from a previous OCW), the channel indicates not ready (Condition Register remains set to Not Zero) and ignores the ISW instruction until the peripheral unit accepts the OCW command. The channel indicates ready (Condition Register set to Zero) and accepts the ISW when it is executed again.

If the ISW specifies a unit other than the unit connected to the channel, the channel indicates not ready and ignores the command. A disconnect/connect is initiated.

If the selected unit is receiving data as the result of an ODW instruction, the ISW is accepted and the Condition Register is set to Zero.

This instruction is privileged.

ODW **Output Data Word**



Operation

A data word is transferred from the A Register to the specified channel/unit combination.

Notes

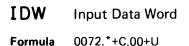
The Condition Register is cleared, then set to Zero if the I/O channel is ready. If the channel is busy and cannot accept the data word, the Condition Register is set to Not Zero to allows a programmed delay.

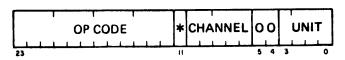
Although, a 24-bit word is transferred to the channel, the peripheral unit accepts only a predetermined number of bits (dictated by peripheral unit design).

For character-oriented units and units accepting data words of less than 24 bits, the data for transfer must be rightjustified in the A Register prior to executing the ODW instruction.

If the ODW instruction specifies a unit other than the unit connected to the channel and the channel is ready, the channel accepts the ODW, sets the Condition Register to Zero, and initiates a disconnect/connect sequence. After completion of the disconnect/connect sequence, the ODW is transferred to the unit. The channel indicates ready to subsequent I/O instructions.

This instruction is privileged.





Affected

A,C

Operation

A data word is transferred from the specified channel/unit combination to the A Register.

Notes

The Condition Register is cleared, then set to Zero if the I/O channel is ready. If the channel is not ready or data from the specified unit is not available, the Condition Register is set to Not Zero to allow a programmed delay.

If the selected unit is in the process of executing a command as the result of a previous OCW instruction, the channel indicates not ready (Condition Register remains set to Not Zero) and the IDW is ignored. At the completion of the OCW, the Condition Register is set to Zero and the IDW instruction is accepted by the channel.

If the selected unit is in the process of receiving data as a result of an ODW instruction and data is available from the unit, an ODW will be accepted and the Condition Register set to Zero.

If the IDW instruction specifies a unit other than the unit connected to the channel, the channel indicates not ready (Condition Register remains set to Not Zero), ignores the instruction, and initiates a disconnect/connect sequence.

The only valid unit code for a DMACP-8 channel is 108; all others are rejected.

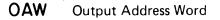
If the Merge bit (*) is ZERO the A Register is cleared prior to the data transfer. Input data is right-justified in the A Register.

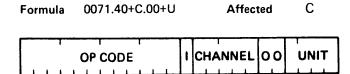
If the Merge Bit is a ONE, an OR is performed between the previous contents of the A Register and the incoming data word. This feature, in conjunction with a shift operation, allows input data characters to be packed in the A Register.

Example: Two 12-bit data characters are to be packed in the A Register.

IDW	'0102	Clear A and load first character from
		channel 01, Unit 02.
BNZ	*-1	Wait if busy
LLA	12	Shift the contents of A left 12 bits
IDW*	'0102	Merge second character
BNZ	*-1	Wait if busy
		Continue

This instruction is privileged.





Operation

The contents of the A Register are transferred to an appropriate register in the specified channel, or unit in XBC Channel executions.

Notes

The Condition Register is cleared, then set to Zero if the I/O channel is ready.

The unit is addressed only in XBC and DMACP-8 channels (bits 0-2) and IBC channels (bits 0, 1).

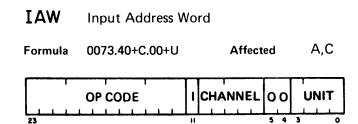
Since XBC/IBC channels involve a unit address, the unit must be "connected" before the instruction can be executed.

The OAW instruction does not activate a block-transfer channel. It transfers the starting address of the first of two parameter words from the A Register to the TAR or PAR in the selected channel. In XBC channel operations the OAW instruction transfers the contents of the A Register to the unit; the channel has no register dedicated to this function.

If an OAW instruction addresses a BBC during a block transfer sequence, the sequence will be terminated.

If the OAW instruction addresses a PIOC, the Condition Register remains set to Not Zero; the instruction is executed automatically. In this instruction the four least significant bits (3-0) of the A Register are transferred to the Interrupt Generator logic. These bits (unitarily) control the triggering of the one-to-four 1 microsecond interrupt pulses.

This instruction is privileged.



The current contents of the Transfer Address Register (TAR) in the specified channel (IBC, or DMACP-8) are transferred to the A Register.

Notes

The Condition Register is cleared, then set to zero if the I/O channel is ready. If the IAW instruction specifies an invalid channel, the Condition Register remains set to Not Zero indicating channel not ready.

The unit is addressed only in IBC and DMACP-8 channels.

Bit 5 at the ZERO level distinguishes between the IAW and IPW instructions.

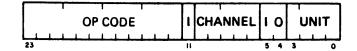
The IBC channel must go to "not busy" before executing the instruction.

This instruction is privileged.

IPW Input Parameter Word

Formula 0073.40+C.40+U

Affected A,C



Operation

The current contents of the Parameter Address Register (PAR) in the specified channel (IBC or DMACP-8) are transferred to the A Register.

Notes

The Condition Register is cleared, then set to zero if the I/O channel is ready. If the IPW instruction specifies an invalid channel, the Condition Register remains set to Not Zero, indicating channel not ready.

The unit is addressed only in IBC and DMACP-8 channels.

IPW instructions addressed to an IBC channel must specify, via the unit address, which of three possible channel PARs is read.

Bit 5 at the ONE level distinguishes between the IPW and IAW instructions.

The IBC channel must go to "not busy" before executing the instruction.

This instruction is privileged.

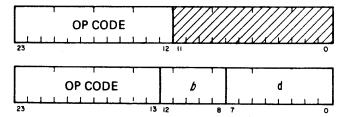
Bit Processor Instructions

The bit (Boolean function) processor group of instructions include branches, logical manipulation, and interrogation of a specified bit selected from an effective memory address or the H Register. In most instances, bit 2 (Zero/Not Zero) of the Condition Register is used to display either the result of an operation or the status of a bit before the operation is performed.

The bit processor employs two instruction word formats. The first format uses an Op Code (bits 23-12) to specify the operation to be performed. The remaining 12 bits (bits 11-0) are undefined. The second instruction format contains a displacement, bit specification, and an Op Code. Eight bits (bits 7-0) are added to the base address contained in the V Register to obtain a displacement from the base address which is an effective memory address for the word containing the bit in question. Five bits (bits 12-8) are used to select a specific bit in the effective memory address for an operation as specified in the 11-bit (bits 23-13) Op Code. Both instructions word formats are illustrated below.

NOTE

If a bit number greater than 23 is specified in bit positions 12-8, the result is unpredictable.



The following instructions are included in the bit processor group.

DMH	Dot Memory with H	7-55
DNH	Dot Not (memory) with H	7-55
FBM	Flag Bit of Memory	7-56
NHH	Negate of H to H	7-54
OMH	OR Memory with H	7-55
ONH	OR Not (memory) with H	7-55
QBH	Query bit of H	7-54
QBM	Query bit of Memory	7-56
TFH	Transfer Flag to H	7-54
THM	Transfer H to Memory	7-56
TKV	Transfer K to V	7-54
TMH	Transfer Memory to H	7-56
TVK	Transfer V to K	7-54
TZH	Transfer Zero to H	7-54
XMH	Exclusive-OR Memory with H	7-55
XNH	Exclusive-OR Not (memory) with H	7-56
ZBM	Zero Bit of Memory	7-56

TZH Transfer Zero to H

Formula 7742. Affected H,C

OP CODE

12

0

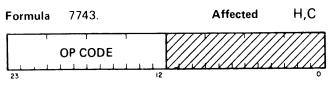
Operation

A ZERO is placed in the H Register. The Condition Register is set to reflect the original contents of H.

Note

If the original contents of the H Register were ZERO, Condition Register Bit 2 is set to 1 (Zero). If the contents were ONE, Bit 2 is set to 0 (Not Zero).

TFH Transfer Flag to H



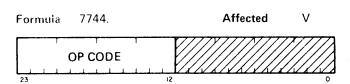
Operation

A ONE is placed in the H Register and the Condition Register is set to reflect the original contents of H.

Note

If the original contents of the H Register were ZERO, Condition Register Bit 2 is set to 1 (Zero). If the contents were ONE, Bit 2 is set to 0 (Not Zero).

TKV Transfer K to V



Operation

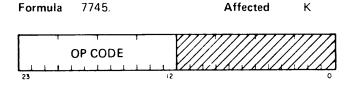
In the Compatibility Mode, the 18 least significant bits of the K Register replace the present contents of the V Register.

In the Address Extension Mode, the 20 least significant bits of the K Register replace the present contents of the V Register.

Note

The Condition Register is Unaffected.

TVK Transfer V to K



Operation

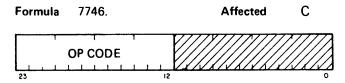
In the Compatibility Mode, the contents of the V Register are transferred to the 18 least significant bit positions of the K Register. Bits 23-18 of the K Register are reset to ZEROs.

In the Address Extension Mode, the contents of the V Register are transferred to the 20 least significant bit positions of the K Register. Bits 23-20 of the K Register are reset to ZEROs.

Note

The Condition Register is unaffected.

QBH Query Bit of H



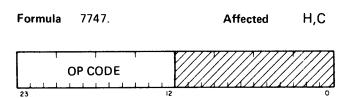
Operation

The H Register bit is tested and the Condition Register is set to display the result of the query.

Note

The Condition Register is cleared. If the resultant content of the H Register is ZERO, Condition Register Bit 2 is set to 1 (Zero). If the content is ONE, Bit 2 is set to 0 (Not Zero).

NHH Negate of H to H



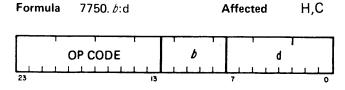
Operation

The current content of the H Register is complemented and returned to H. The Condition Register is set to display the result.

Note

The Condition Register is cleared. If the resultant content of the H Register is ZERO, Condition Register Bit 2 is set to 1 (Zero). If the content is ONE, Bit 2 is set to 0 (Not Zero).

DMH Dot Memory with H



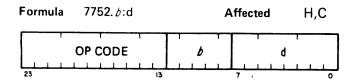
Operation

A logical AND is performed between the selected bit in the effective memory address and the contents of the H Register. The result is returned to the H Register and the Condition Register is set to display the result.

Note

The Condition Register is cleared. If the resultant content of the H Register is ZERO, Condition Register Bit 2 is set to 1 (Zero). If the content is ONE, Bit 2 is set to 0 (Not Zero).

DNH Dot Not (memory) with H



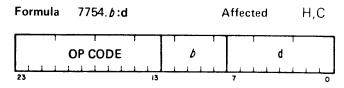
Operation

A logical AND is performed between the complement of the selected bit in the effective memory address and the content of the H Register. The result is returned to the H Register and the Condition Register is set to display the result.

Note

The Condition Register is cleared. If the resultant content of the H Register is ZERO, Condition Register Bit 2 is set to 1 (Zero). If the content is ONE, Bit 2 is set to 0 (Not Zero).

OMH OR Memory with H



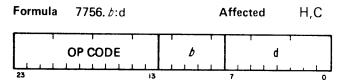
Operation

A logical OR is performed between the selected bit in the effective memory address and the content of the H Register. The Condition Register is set to display the result.

Note

The Condition Register is cleared. If the resultant content of the H Register is ZERO, Condition Register Bit 2 is set to 1 (Zero). If the content is ONE, Bit 2 is set to 0 (Not Zero).

ONH OR Not (memory) with H



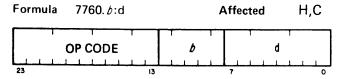
Operation

A logical OR is performed between the complement of the selected bit in the effective memory address and the content of the H Register. The Condition Register is set to display the result.

Note

The Condition Register is cleared. If the resultant content of the H Register is ZERO, Condition Register Bit 2 is set to 1 (Zero). If the content is ONE, Bit 2 is set to 0 (Not Zero).

XMH Exclusive-OR Memory with H



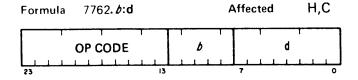
Operation

An exclusive-OR function is performed between the selected bit in the effective memory address and the content of the H Register. The Condition Register is set to display the result.

Note

The Condition Register is cleared. If the resultant content of the H Register is ZERO, Condition Register Bit 2 is set to 1 (Zero). If the content is ONE, Bit 2 is set to 0 (Not Zero).

XNH Exclusive-OR Not (memory) with H



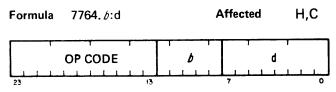
Operation

An exclusive-OR function is performed between the complement of the selected bit in the effective memory address and the content of the H Register. The Condition Register is set to display the result.

Note

The Condition Register is cleared. If the resultant content of the H Register is ZERO, Condition Register Bit 2 is set to 1 (Zero). If the content is ONE, Bit 2 is set to 0 (Not Zero).

TMH Transfer Memory to H



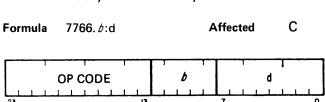
Operation

The selected bit in the effective memory address is transferred to the H Register. The Condition Register is set to display the resultant content of the H Register.

Note

The Condition Register is cleared. If the resultant content of the H Register is ZERO, Condition Register Bit 2 is set to 1 (Zero). If the resultant content is ONE, bit 2 is set to 0 (Not Zero).

QBM Query Bit of Memory



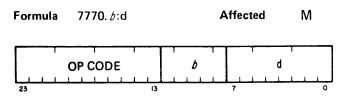
Operation

The selected bit in the effective memory address is tested and the Condition Register is set to display the result of the query.

Note

The Condition Register is cleared. If the resultant content of memory is ZERO, Condition Register Bit 2 is set to 1 (Zero). If the resultant content is ONE, Bit 2 is set to 0 (Not Zero).

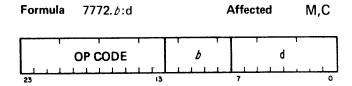
THM Transfer H to Memory



Operation

The content of the H Register is placed in the selected bit position in the effective memory address. The Condition Register is not affected.

FBM Flag Bit of Memory



Operation

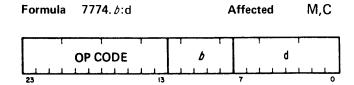
A ONE is placed in the selected bit position in the effective memory address. The Condition Register is set to display the original state of the selected bit in memory.

Notes

If the original state of the selected bit in memory was ZERO, Condition Register Bit 2 is set to 1 (Zero). If the original state was ONE, Bit 2 is set to 0 (Not Zero).

DMA transfers are inhibited and shared memory is locked up during the execution of this instruction.

ZBM Zero Bit of Memory



A ZERO is transferred to the selected bit position in the effective memory address. The Condition Register is set to display the original state of the selected bit in memory.

Notes

If the original state of the selected bit in memory was ZERO, Condition Register Bit 2 is set to 1 (Zero). If the original state was ONE, Bit 2 is set to 0 (Not Zero).

DMA transfers are inhibited and shared memory is locked up during the execution of this instruction.

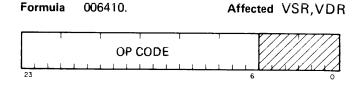
Virtual Memory Instructions

The majority of the virtual memory instructions involve transfers between the paging registers and the A, E and D Registers. The remaining instructions are special control operations for activating and testing the virtual memory logic.

The following instructions are included in the virtual memory group.

ONR	Query Not-modified Register	7-59
QUR	Query Usage Register	7-59
ROM	Release Operand Mode	7-59
RUM	Release User Mode	7-60
TAR	Transfer A to 1 Virtual Address	
	Register	7-57
TDP	Transfer Double to Paging Limit	
	Registers	7-58
TDR	Transfer Double to 2 Virtual Address	
	Registers	7-58
TDS	Transfer Double to Source and	
	Destination Registers	7-57
TEU	Transfer E to Usage Base Register	7-59
TPD	Transfer Paging Limit Registers	
	to Double	7-58
TRD	Transfer 2 Virtual Address Registers	
	to Double	7-58
TSD	Transfer Source and Destination	
	Registers to Double	7-57
TUD	Transfer Usage Base Register and	
	Demand Page Register to Double	7-58

TDS Transfer Double to Source and Destination Registers



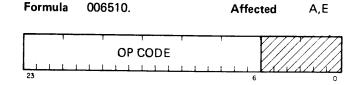
Operation

Bits 11-0 of the A Register replace the previous contents of the Virtual Destination Register (VDR) and bits 11-0 of the E Register replace the previous contents of the Virtual Source Register (VSR). The contents of A and E are not changed.

Note

This instruction is privileged.

TSD Transfer Source and Destination Registers to D



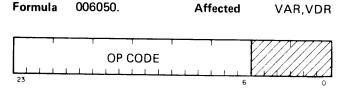
Operation

The contents of the Virtual Source Register (VSR) replace the previous contents of bits 11-0 of the E Register; the contents of the Virtual Destination Register (VDR) replace the previous contents of bits 11-0 of the A Register. Bits 23-12 of both A and E are cleared (reset to ZEROs). The contents of the VSR and VDR are not changed.

Note

This instruction is privileged.

TAR Transfer A to 1 Virtual Address Register



Operation

Bits 23, 22, and 9-0 of the A Register replace the previous contents of the Virtual Address Register (VAR) specified

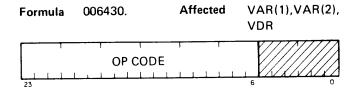
0830007-000 Original 8/79

by the Virtual Destination Register (VDR). The VDR is incremented by one. The contents of the A Register are not changed.

Note

This instruction is privileged.

TDR Transfer Double to 2 Virtual Address Registers



Operation

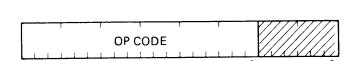
Bits 23, 22, and 9-0 of the E Register replace the previous contents of the Virtual Address Register (VAR) specified by the Virtual Destination Register (VDR); the VDR is then incremented by one to specify the second VAR. Bits 23, 22 and 9-0 of A replace the previous contents of the second VAR. The VDR is again incremented by one. The contents of the E and A Registers are not changed.

Note

This instruction is privileged.

TRD Transfer 2 Virtual Address Registers to Double

006530.



Affected

Operation

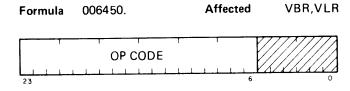
Formula

The contents of the Virtual Address Register (VAR) specified by the Virtual Source Register (VSR) replace the previous contents of bits 23, 22, and 9-0 of the E Register. The VSR is then incremented by one to specify the second VAR. The contents of the second VAR replace the previous contents of bits 23, 22, and 9-0 of the A Register. The VSR is again incremented by one. Bits 21-10 of both E and A are cleared (reset to ZERO).

Note

This instruction is privileged.

TDP Transfer Double to Paging Limit Registers



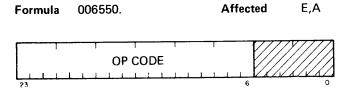
Operation

Bits 11-0 of the A Register replace the previous contents of the Virtual Base Register (VBR), and 23-19 and 9-0 of the E Register replace the previous contents of the Virtual Limit Register (VLR). The contents of A and E are not changed.

Note

This instruction is privileged.

TPD Transfer Paging Limit Registers to Double



Operation

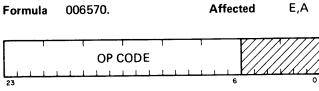
The contents of the Virtual Base Register (VBR) replace the previous contents of A Register bits 11-0, and the contents of the Virtual Limit Register (VLR) replace the previous contents of E Register bits 23-19 and 9-0. The remaining bits of both A and E are reset to ZEROs. The contents of the VBR and VLR are not changed.

Note

E,A,VSR

This instruction is privileged.

TUD Transfer Usage Base Register and Demand Page Register to Double



Operation

The contents of the Virtual Demand Page Register (VPR) replace the previous contents of A Register bits 13-0, and the contents of the Virtual Usage Base Register (VUB)

replace the previous contents of E Register bits 9-0. A Register bits 23-14 and E Register bits 23-10 are reset to ZEROs. The contents of the VPR and VUB are not changed.

Note

This instruction is privileged.

006470.

TEU Transfer E to Virtual Usage Base Register

OP CODE

Affected

VUB

Operation

Formula

The contents of E Register bits 9-0 replace the previous contents of the Virtual Usage Base Register (VUB). The E Register contents are not changed.

Note

This instruction is privileged.

QUR Query Usage Register

007030.

VUR, VUB, C OP CODE

Affected

Operation

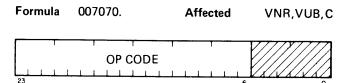
Formula

The contents of the Virtual Usage Register (VUR) specified by the Virtual Usage Base Register (VUB) - is tested. The Condition Register is set to "Not Zero" or "Zero" if the content of the VUR is ONE or ZERO, respectively. The specified VUR is cleared and the VUB is incremented by one.

Note

This instruction is privileged.

QNR Query Not-modified Register



Operation

The contents of the Virtual Not-modified Register (VNR) - specified by the Virtual Usage Base Register (VUB) - is tested. The Condition Register is set to "Not Zero" or "Zero" if the content of the VNR is ONE or ZERO, respectively. The specified VNR is cleared and the VUB is incremented by one.

Note

This instruction is privileged.

ROM Release Operand Mode

006010.

Affected None OP CODE

Operation

Formula

The operand address of the following instruction is translated.

Notes.

With the exception of the PC mapping function, the ROM instruction is nullified if bit 19 (ROM Inhibit) of the VLR is set.

No double-word instructions (AOM, USP, or extended instructions) are permitted after a ROM instruction.

No branch instructions are permitted after the ROM instruction.

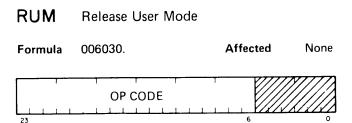
If an EXM is executed after a ROM, the ROM is treated as a NOP, no translation occurs, and the EXM executes as normal.

The ROM instruction translates only the final EMA after indexing and/or indirection.

The ROM instruction controls the map bit of the following instruction in accordance with the following example.

> TMA Inhibits mapping of operand TMA,I Inhibits mapping of operand TMA* First address access is mapped, succeeding accesses are not mapped, operand is not mapped TMA*,I Same as TMA*

This instruction is privileged.



The User Mode is established upon completion of the following instruction.

Notes

The instruction following the RUM should always be a branch instruction which may be indexed and/or indirected. No conditional branches are allowed.

After the new program address is calculated, the User Mode is activated.

The RUM instruction, together with the following instruction, are handled as an EXM instruction with respect to a demand page (bits 1 and 0 of the Virtual Demand Page Register set to ZERO and ONE, respectively).

Only the final EMA, after indexing and/or indirection, of the instruction following the RUM instruction is translated.

Execution of the RUM instruction inhibits mapping of the following branch fetch.

This instruction is privileged.

Priority Interrupt Control Instructions

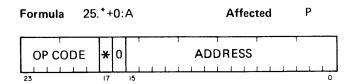
The priority interrupt instruction group provides the means for program control of external interrupts. External interrupts may be selectively armed, disarmed, enabled or inhibited under program control. Other instructions provide the means for holding and releasing external interrupts, while others are available for transferring control upon interrupt detection. For a detailed description of the priority interrupt system, refer to Section V of this manual.

The following instructions are included in the priority interrupt group.

BRL	Branch and Reset Interrupt Long	7-61
BSL	Branch and Save Return Long	7-60
BSX	Branch and Save Extended	7-61
HTx	Hold Interrupts and Transfer Register	
	to Memory	7-62
HXI	Hold External Interrupts	7-63
RXI	Release External Interrupts	7-63
T1D	Transfer Group 1 to Double	7-64

T2D	Transfer Group 2 to Double	7-64
T3D	Transfer Group 3 to Double	7-64
T4D	Transfer Group 1 to Double	7-64
T5D	Transfer Group 2 to Double	7-65
T6D	Transfer Group 3 to Double	7-65
TD1	Transfer Double to Group 1	7-63
TD2	Transfer Double to Group 2	7-63
TD3	Transfer Double to Group 3	7-63
TD4	Transfer Double to Group 1	7-65
TD5	Transfer Double to Group 2	7-65
TD6	Transfer Double to Group 3	7-65
UA1	Unitarily Arm Group 1 Interrupts	7-65
UA2	Unitarily Arm Group 2 Interrupts	7-66
UA3	Unitarily Arm Group 3 Interrupts	7-66
UD1	Unitarily Disarm Group 1 Interrupts	7-66
UD2	Unitarily Disarm Group 2 Interrupts	7-67
UD3	Unitarily Disarm Group 3 Interrupts	7-67
UE1	Unitarily Enable Group 1 Interrupts	7-67
UE2	Unitarily Enable Group 2 Interrupts	7-68
UE3	Unitarily Enable Group 3 Interrupts	7-68
UI1	Unitarily Inhibit Group 1 Interrupts	7-68
UI2	Unitarily Inhibit Group 2 Interrupts	7-69
UI3	Unitarily Inhibit Group 3 Interrupts	7-69

BSL % Branch and Save Return Long



Operation

In the Compatibility Mode, the program address of the next sequential instruction along with the contents of the Condition Register are stored in the effective memory address (EMA). The contents of the P Register (current program address) are then replaced by the address following the effective memory address (EMA + 1).

In the Address Extension Mode, the program address of the next sequential instruction is stored in the effective memory address (EMA). The contents of the P Register (current program address) are then replaced by the address following the effective memory address (EMA + 1).

Notes

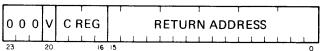
This instruction is used in the Compatibility Mode to enter an interrupt subroutine because it provides a means of returning to the main program at the point of interrupt and saves the machine status (condition) at the time of the interrupt. In the Compatibility Mode, the contents of the Condition Register are stored in bit positions 19-16 of the EMA and the return address (program address of next sequential instruction) is stored in bits 15-0. The remaining bits are set to ZEROs. When an interrupt occurs, the status of the virtual memory system is recorded. Bit 20 is set to ONE if the system is in the User Mode at the time of interrupt; bit 20 is set to ZERO if the Monitor Mode is active.

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

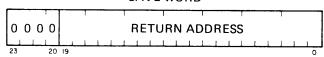
The Condition Register remains unchanged.

COMPATIBILITY MODE SAVE WORD



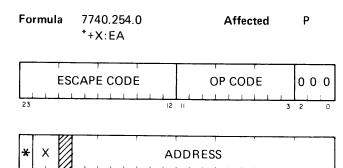
In the Compatibility Mode, the final EMA may not exceed 16 bits when a BSL or extended BSL is executed. Intermediate Addresses may be 20 bits when an indirect extended BSL is executed.

ADDRESS EXTENSION MODE SAVE WORD



In the Address Extension Mode, the return address is stored in bit positions 19-0 of the EMA; bits 23-20 are reset to ZEROs.

BSX Branch and Save Extended



Operation

The program address of the next sequential instruction, along with the contents of the Condition Register, are stored in the 20-bit effective memory address (EMA) location. The contents of the P Register (current program address) are then replaced by the address following the effective memory address (EMA + 1).

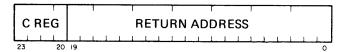
Notes

The BSX instruction is valid only in the extended instruction format. This instruction provides a means of returning to the main program and saves the machine status (condition) at the time of instruction execution.

External interrupts are prohibited for a period of one instruction following the execution of this instruction.

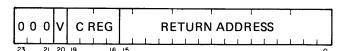
The Condition Register remains unchanged.

ADDRESS EXTENSION MODE SAVE WORD



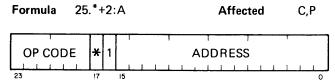
When the BSX is executed in the Address Extension Mode, the contents of the Condition Register are stored in bit positions 23-20 of the EMA location and the return address (program address of the next sequential instruction) is stored in bit positions 19-0.

COMPATIBILITY MODE SAVE WORD



In the Compatibility Mode, the final EMA may not exceed 16 bits when a BSX is executed, however, intermediate addresses may be 20 bits when the BSX is indirected.

BRL % Branch and Reset Interrupt Long



Operation

The highest-level active and enabled interrupt is reset (i.e., returned to the inactive state) and the contents of the P

0830007-000 Original 8/79

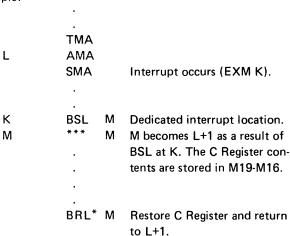
Register (current program address) are replaced by the effective memory address.

Notes

The BRL instruction is normally used to exit an interrupt subroutine.

In the Compatibility Mode, if the BRL contains an indirect reference, the last word in the indirect address chain contains the previous status of the virtual memory system in bit M20, the previous machine status (i.e., C Register contents at the time of the interrupt) in bit positions M19-M16, and the return address in bit positions M15-M0 as a result of the BSL instruction. The C Register is restored and the program branches to the return address (restarting the machine to the pre-interrupt status).

Example:



In the Compatibility Mode, if an indirect BRL is executed in Monitor Mode, bit 20 of the effective memory address determines mode of operation to which machine returns. If bit 20 is set, User Mode is established; if reset, the Monitor Mode is established.

In the Address Extension Mode, if the BRL does not contain an indirect reference, the program branches to the return address and the states of VLR bit 20 and the C Register are unchanged. If the BRL is indirected (no indirect chaining is allowed), the destination address contains the previous machine status in bit positions M23-M20, and the return address in bit positions M19-M0 as a result of the BSX instruction. The C Register is restored and the program branches to the return address. VLR bit 20 remains unchanged if another interrupt is active and enabled. If no other interrupt is active and enabled, VLR20 is reset. VLR bit 20 determines the mode of operation to which machine returns (if no other interrupt is active and enabled). If VLR20 is set, User Mode is established; if reset, the Monitor Mode is established.

In the Compatibility Mode, the final EMA may not exceed 16 bits when a BRL or extended BRL is executed. Intermediate address may be 20 bits when an indirect extended BRL is executed.

The immediate memory reference cannot be indexed; however, indexing indirect references is permitted, e.g.,



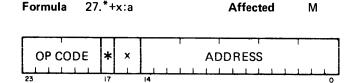
If the BRL instruction is not indirected, the Condition Register is not affected.

External interrupts are prohibited for the period of one instruction following this instruction.

The BRL will not reset the interrupt if external interrupts have been held by an HXI instruction. Control will be returned to the effective memory address.

Those executive traps, which are not affected by the HXI instruction, will be reset by the BRL.

HTx % Hold Interrupts and Transfer Register to Memory



Operation

The contents of the I, J, or K Register replace the previous contents of the effective memory address and external interrupts are prohibited for the period of one instruction following the execution of this instruction.

Notes

HTx is not a computer instruction mnemonic but represents a family of instruction mnemonics. x is coded as follows to select one of the index registers.

$$x = 1 (I)$$

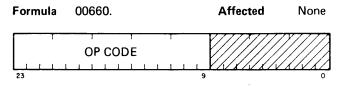
2 (J)
3 (K)

A code of 27.*+1:a, for example, implements the Hold Interrupt and Transfer I to Memory (HTI) instruction.

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,



HXI Hold External Interrupts



Operation

The activation of any external interrupt is prohibited. The prohibition is effective immediately upon execution of the instruction and lasts until the interrupts are released (see RXI instruction). Executive traps (Group 0, Levels 4, 6, and 7) are prohibited from becoming active while the HXI is in effect.

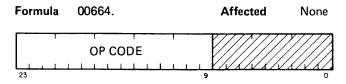
Notes

Only the three executive traps mentioned are affected by this instruction.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

RXI Release External Interrupts



Operation

The prohibition imposed by the HXI instruction is removed, allowing any external interrupt to be activated 1 cycle after this instruction. This permits the next sequential instruction to be executed without external interruption.

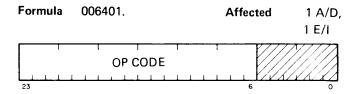
Notes

If any of the affected executive traps have been triggered while an HXI was in effect, the highest level will come in first after the RXI instruction.

External interrupts are prohibited for the period of one instruction following the execution of the instruction.

This instruction is privileged.

TD1 Transfer Double to Group 1



Operation

The contents of the D Register (E and A) replace the previous contents of the Arm/Disarm (A/D) and Enable/Inhibit (E/I) Registers of interrupt group 1. The contents of E are transferred to the A/D Register and the contents of A are transferred to the E/I Register.

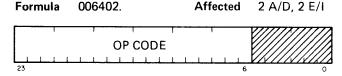
Notes

The group 1 external interrupt structure is cleared by the execution of this instruction.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

TD2 Transfer Double to Group 2



Operation

The contents of the D (E and A) Register replace the previous contents of the Arm/Disarm (A/D) and Enable/Inhibit (E/I) Registers of interrupt group 2. The contents of E are transferred to the A/D Register, and the contents of A are transferred to the E/I Register.

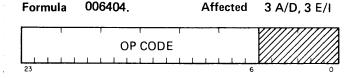
Notes

The group 2 external interrupt structure is cleared by the execution of this instruction.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

TD3 Transfer Double to Group 3



The contents of the D (E and A) Register replace the previous contents of the Arm/Disarm (A/D) and Enable/Inhibit (E/I) Registers of interrupt group 3. The contents of E are transferred to the A/D Register, and the contents of A are transferred to the E/I Register.

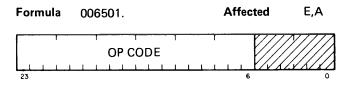
Notes

The group 3 external interrupt structure is cleared by the execution of this instruction.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

T1D Transfer Group 1 to Double



Operation

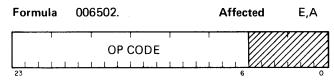
The contents of the Arm/Disarm (A/D) and Enable/Inhibit (E/I) Registers of interrupt group 1 replace the previous contents of the D Register (E and A). The contents of the A/D Register are transferred to the E Register and the contents of the E/I Register are transferred to the A Register.

Notes

The states of the external interrupts are not affected by the execution of this instruction.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

T2D Transfer Group 2 to Double



Operation

The contents of the Arm/Disarm (A/D) and Enable/Inhibit (E/I) Registers of interrupt group 2 replace the previous contents of the D (E and A) Register. The contents of the A/D Register are transferred to the E Register, and the contents of the E/I Register are transferred to the A Register.

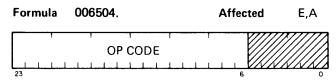
Notes

The states of the external interrupts are not affected by the execution of this instruction.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

T3D Transfer Group 3 to Double



Operation

The contents of the Arm/Disarm (A/D) and Enable/Inhibit (E/I) Registers of interrupt group 3 replace the previous contents of the D (E and A) Register. The contents of the A/D Register are transferred to the E Register, and the contents of the E/I Register are transferred to the A Register.

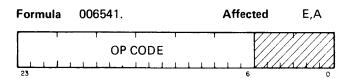
Notes

The states of the external interrupts are not affected by the execution of this instruction.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

T4D Transfer Group 1 to Double



Operation

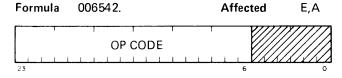
The contents of the Request and Active Registers of interrupt group 1 replace the previous contents of the D Register (E and A). The contents of the Request Register are transferred to E, and the contents of the Active Register are transferred to A.

Notes

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

T5D Transfer Group 2 to Double



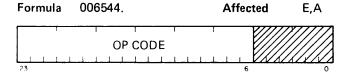
The contents of the Request and Active Registers of interrupt group 2 replace the previous contents of the D (A and E) Register. The contents of the Request Register are transferred to E, and the contents of the Active Register are transferred to A.

Note

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

T6D Transfer Group 3 to Double



Operation

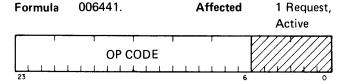
The contents of the Request and Active Registers of interrupt group 3 replace the previous contents of the D (A and E) Register. The contents of the Request Register are transferred to E, and the contents of the Active Register are transferred to A.

Note

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

TD4 Transfer Double to Group 1



Operation

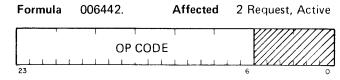
If armed, the contents of the D Register (E and A) are ORed with the current contents of the Request and Active Registers of interrupt group 1. The contents of E are ORed with the request Register and the contents of A are ORed with the Active Register.

Notes

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

TD5 Transfer Double to Group 2



Operation

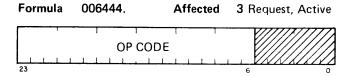
If armed, the contents of the D Register (E and A) are ORed with the current contents of the Request and Active Registers of interrupt group 2. The contents of E are ORed with the Request Register, and the contents of A are ORed with the Active Register.

Note

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

TD6 Transfer Double to Group 3



Operation

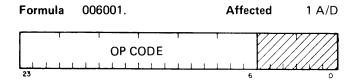
If armed, the contents of the D Register (E and A) are ORed with the current contents of the Request and Active Registers of interrupt group 3. The contents of E are ORed with the Request Register, and the contents of A are ORed with the Active Register.

Note

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

UA1 Unitarily Arm Group 1 Interrupts



Any number of the 24 interrupt levels in group 1 are selectively armed; i.e., the selected bit(s) of the Arm/Disarm (A/D) Register is (are) set to ONE.

Notes

The corresponding bit(s) of the A Register must be set to select the appropriate level(s) prior to executing this instruction.

Example: Arm levels 1 and 3, group 1

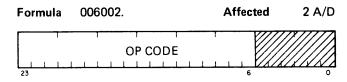
TOA B1B3 Select levels 1 and 3 (set bits 1 and 3 of A)
UA1 Arm selected levels of group 1

Execution of this instruction does not clear the interrupt structure and, therefore, does not affect any interrupt levels other than those selected. If a level selected for arming is already armed, it is not cleared by the execution of this instruction.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

UA2 Unitarily Arm Group 2 Interrupts



Operation

Any number of the 24 interrupt levels in group 2 are selectively armed, i.e., the selected bit(s) of the Arm/Disarm (A/D) Register are set to ONE.

Notes

The corresponding bit(s) of the A Register must be set to select the appropriate level(s) prior to executing this instruction.

Example: Arm levels 1 and 3, group 2

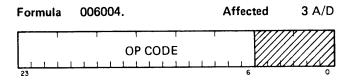
TOA B1B3 Select levels 1 and 3 (set bits 1 and 3 of A)
UA2 Arm selected levels of group 2

Execution of this instruction does not clear the interrupt structure and, therefore, does not affect any interrupt levels other than those selected. If a level selected for arming is already armed, it is not cleared by the execution of this instruction.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

UA3 Unitarily Arm Group 3 Interrupts



Operation

Any number of the 24 interrupt levels in group 3 are selectively armed, i.e., the selected bit(s) of the Arm/Disarm (A/D) Register are set to ONE.

Notes

The corresponding bit(s) of the A Register must be set to select the appropriate level(s) prior to executing this instruction.

Example: Arm levels 1 and 3, group 3

TOA B1B3 Select levels 1 and 3 (set bits 1 and 3 of A)

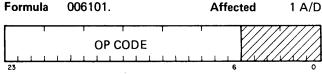
UA3 Arm selected levels of group 3

Execution of this instruction does not clear the interrupt structure and, therefore, does not affect any interrupt levels other than those selected. If a level selected for arming is already armed, it is not cleared by the execution of this instruction.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

UD1 Unitarily Disarm Group 1 Interrupts



Operation

Any number of the 24 interrupts levels in group 1 are selectively disarmed i.e., the selected bits of the Arm/Disarm (A/D) Register are reset to ZERO.

Notes

The corresponding bit(s) of the A Register must be set to select the appropriate level(s) prior to executing this instruction.

Example: Disarm level 2, group 1

TOA B2 Select level 2 (set bit 2 of A)
UD1 Disarm selected level of group 1

Execution of this instruction will clear only those levels which are selected. The remaining levels will not be affected.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

UD2 Unitarily Disarm Group 2 Interrupts

Formula	006102.	Affected	2 A/D
	OP CODE		
23		6	0

Operation

Any number of the 24 interrupt levels in group 2 are selectively disarmed, i.e., the selected bit(s) of the Arm/Disarm (A/D) Register are reset to ZERO.

Notes

The corresponding bit(s) of the A Register must be set to select the appropriate level(s) prior to executing this instruction.

Example: Disarm level 2, group 2

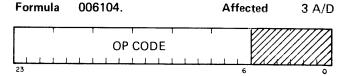
TOA B2 Select level 2 (set bit 2 of A)
UD2 Disarm selected level of group 2

Execution of this instruction will clear only those levels which are selected. The remaining levels will not be affected.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

UD3 Unitarily Disarm Group 3 Interrupts



Operation

Any number of the 24 interrupt levels in group 3 are selectively disarmed, i.e., the selected bit(s) of the Arm/Disarm (A/D) Register are reset to ZERO.

Notes

The corresponding bit(s) of the A Register must be set to select the appropriate level(s) prior to executing this instruction.

Example: Disarm level 2, group 3

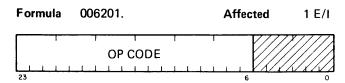
TOA B2 Select level 2 (set bit 2 of A)
UD3 Disarm selected level of group 3

Execution of this instruction will clear only those levels which are selected. The remaining levels will not be affected.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

UE1 Unitarily Enable Group 1 Interrupts



Operation

Any number of the 24 interrupt levels in group 1 are selectively enabled, i.e., the selected bits of the Enable/Inhibit (E/I) Register are set to ONE.

Notes

The corresponding bit(s) of the A Register must be set to select the appropriate level(s) prior to executing this instruction.

Enable levels 0, 2 and 5, group 1 Example:

> TOA B0B2B5 Select levels 0, 25

> > (set bits 0, 2 and 5 of A)

Enable selected levels of UE1

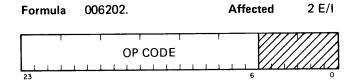
group 1

Execution of this instruction does not clear the interrupt structure and, therefore, does not affect any interrupt levels other than those selected. If a level selected for enabling is already enabled, it is not cleared by the execution of this instruction.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

UE2 Unitarily Enable Group 2 Interrupts



Operation

Any number of the 24 interrupt levels in group 2 are selectively enabled, i.e., the selected bits of the Enable/Inhibit (E/I) Register are set to ONE.

Notes

The corresponding bit(s) of the A Register must be set to select the appropriate level(s) prior to executing this instruction.

Example: Enable levels 0, 2, and 5, group 2

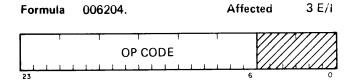
> TOA **B0B2B5** Select levels 0, 2, 5 (set bits 0, 2, and 5 of A) UE2 Enable selected levels of group 2

Execution of this instruction does not clear the interrupt structure and, therefore, does not affect any interrupt levels other than those selected. If a level selected for enabling is already enabled, it is not cleared by the execution of this instruction.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

UE3 Unitarily Enable Group 3 Interrupts



Operation

Any number of the 24 interrupt levels in group 3 are selectively enabled, i.e., the selected bits of the Enable/Inhibit (E/I) Register are set to ONE.

Notes

The corresponding bit(s) of the A Register must be set to select the appropriate level(s) prior to executing this instruction.

Enable levels 0, 2, and 5, group 3 Example:

> TOA B0B2B5 Select levels 0, 2, 5 (set bits 0, 2, and 5 of A) Enable selected levels UE3

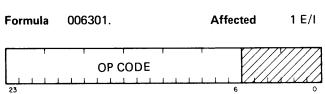
of group 3

Execution of this instruction does not clear the interrupt structure and, therefore, does not affect any interrupt levels other than those selected. If a level selected for enabling is already enabled, it is not cleared by the execution of this instruction.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

UI1 Unitarily Inhibit Group 1 Interrupts



Operation

Any number of the 24 interrupt levels in group 1 are selectively inhibited; i.e., the selected bits of the Enable/Inhibit (E/I) Register are reset to ZERO.

Notes

The corresponding bit(s) of the A Register must be set to select the appropriate level(s) prior to executing this instruction.

Example:

Inhibit levels 1, 4 and 7 of group 1

TOA E

B1B4B7 Select levels 1, 4, 7

(set bits 1, 4 and 7 of A)

UI1

Inhibit selected levels of group 1

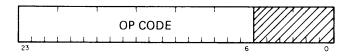
Execution of this instruction does not clear the interrupt structure and, therefore, does not affect any interrupt levels other than those selected. If one or more of the selected levels is active upon execution of this instruction, the level(s) will be placed in a "permissive" state.

External interrupts are prohibited for the period of one instruction following execution of this instruction.

This instruction is privileged.

UI2 Unitarily Inhibit Group 2 Interrupts

Formula 006302. Affected 2 E/I



Operation

Any number of the 24 interrupt levels in group 2 are selectively inhibited; i.e., the selected bits of the Enable/Inhibit (E/I) Register are reset to ZERO.

Notes

The corresponding bit(s) of the A Register must be set to select the appropriate level(s) prior to executing this instruction.

Example:

Inhibit levels 1, 4, and 7 of group 2

TOA B1B4B7 Select levels 1, 4, 7 (set bits 1, 4, and 7 of A)
UI2 Inhibit selected levels of

group 2

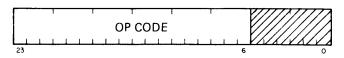
Execution of this instruction does not clear the interrupt structure and, therefore, does not affect any interrupt levels other than those selected. If one or more of the selected levels is active upon execution of this instruction, the level(s) will be placed in a "permissive" state.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

UI3 Unitarily Inhibit Group 3 Interrupts

Formula 006304. Affected 3 E/I



Operation

Any number of the 24 interrupt levels in group 3 are selectively inhibited; i.e., the selected bits of the Enable/Inhibit (E/I) Register are reset to ZERO.

Notes

The corresponding bit(s) of the A Register must be set to select the appropriate level(s) prior to executing this instruction.

Example: Inhibit levels 1, 4, and 7 of group 3

TOA B1B4B7 Select levels 1, 4, 7 (set bits 1, 4, and 7 of A)
UI3 Inhibit selected levels of

group 3

Execution of this instruction does not clear the interrupt structure and, therefore, does not affect any interrupt levels other than those selected. If one or more of the selected levels is active upon execution of this instruction, the level(s) will be placed in a "permissive" state.

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

This instruction is privileged.

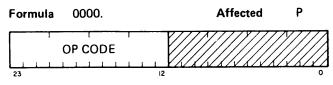
Miscellaneous Instructions

The following instructions are included in the miscellaneous group because they do not fall into any defined functional group.

EXM	Execute Memory	7-71
EZB	Extend Zeros from Byte	7-72
GAP	Generate Argument Pointer	7-70
HIT	Hold Interval Timer	7-72

HLT	Halt	7-70
NOP	No Operation	7-70
QBB	Query Bits of Byte	7-71
QSS	Query Sense Switches	7-72
RCT	Release Clock Time	7-72
RPT	Release Processor Time	7-72
USP	Update Stack Pointer	7-70

HLT Halt



Operation

The program address (i.e., the contents of the P Register) is advanced by one and program execution is terminated. When the RUN command is executed, execution will begin at the location defined by the program address.

Note

This instruction is privileged.

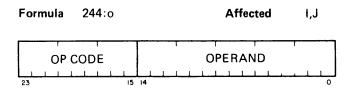
NOP No Operation

Formula 620.	Affected	Р
OP CODE		
23	15	0

Operation

The program address is advanced by one and program execution continues with the next instruction.

GAP Generate Argument Pointer



Operation

The contents of the J Register are assumed to be the first address in an indirect memory reference sequence. The effective memory address derived from this indirect sequence replaces the previous contents of the I Register. The contents of the J Register and the 15-bit operand are added, and the result is placed in the J Register.

Notes

In the Compatibility Mode, if the final EMA in the indirect sequence is a DAC format, bits 15-0 replace the contents of I. If the final EMA is a LAC, bits 20-0 replace the contents of I. A 16-bit value in the J Register is used for the address of the first indirect access.

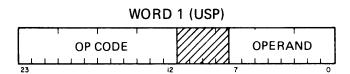
In the Address Extension Mode, a 20-bit value in the J Register is used for the address of the first indirect access. Bits 20-0 of the final EMA replace the contents of I.

The purpose of a GAP instruction is to generate an effective memory address which points to one or more data words not directly available to a subroutine. This is illustrated in the following example where subroutine B requires the data contained in location Y.

Α	BLJ	В	(J) = C, (P) = B
С	DAC*	X	
D			RETURN
	•••		
Χ	DAC	Υ	
Υ	DATA	2	
•	•••		
В	GAP	1	(I) = Y, (J) = (J) + 1
	TMA	0,1	(A) = 2
	BUC	0,J	(P) = D

USP Update Stack Pointer Word 2

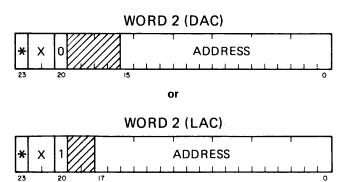
Formula 0055:o



Affected

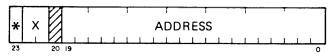
K,C

COMPATIBILITY MODE



ADDRESS EXTENSION MODE

WORD 2



Operation

The contents of the K Register are replaced by the contents of the effective memory address. The 8-bit signed operand is then added to the contents of the effective memory address.

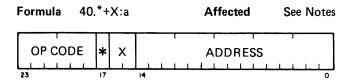
Notes

	BLJ	ENT	Call re-entrant routine
ENT	 TRM* USP	SP 5	Save registers in stack Update Stack Pointer [(K) = stack, (SP) = stack + 5]
	DAC	SP	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
	HTK	SP	Reset stack pointer
	TMR*	SP	Restore registers
	BUC	0,J	Return
SP	DAC	STACK	Stack pointer
STACK	BLOK	5N	Where N represents max-
			imum number of re-
			entrant levels

The C Register is set to Positive, Negative, or Zero, based on the result of the operation. Overflow is set if the arithmetic operation generates a carry into the sign bit without a carry out, or a carry out of the sign bit (23) without a carry in.

External interrupts are prohibited for the period of one instruction following this instruction.

EXM % Execute Memory



Operation

The instruction located in the effective memory address is executed as though it were at the address of the EXM.

Notes

In the case that the referenced instruction is a two word instruction, the second word must follow the EXM.

Example:

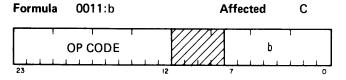
	EXM	M	
	DAC	L	Second word
М	AOM	10	Two word instruction
	AOM	20	
	AOM	30	

The registers affected will depend on the instruction in the effective memory address.

All interrupts are prohibited for the period of one instruction following the execution of this instruction.

The program address (contents of P Register) is not advanced.

QBB Query Bits of Byte



Operation

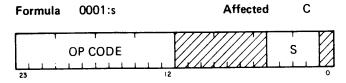
A logical AND is performed between operand bits 7-0 and the contents of the B Register. The Condition Register is set according to the status of the result; i.e., Positive, Negative, or Zero.

Note

Examples:

	(1) TOA	B7	A = '00000200	C = Positive
	QBB	В7		C = Negative
	(2) TOA	В6	A = '00000100	C = Positive
	QBB	В6		C = Positive
	(3) TNA	1	A = '77777777	C = Negative
	DMA 	MASK	A = '40000000	C = Negative
MASK	DATA	40000	000	

QSS Query Sense Switches



Operation

A logical AND is performed between operand bits 4-1 and the state(s) of the sense switches. The Condition Register is set to Positive, or Zero based on the result.

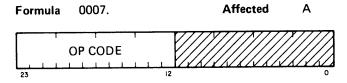
Note

Example: Test to see if either SS2 or SS3 are on, or if

both are on.

QSS B2B3

EZB Extend Zeros from Byte



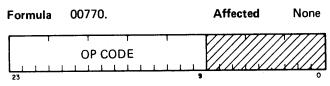
Operation

Bit positions A23-A8 are set to ZERO. The contents of the B Register (A7-A0) are not affected.

Note

The Condition Register is not affected.

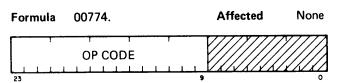
HIT Hold Interval Timer



Operation

The CPU's Interval Timer is halted and will remain so until released by an RPT or RCT instruction.

RPT Release Processor Time



Operation

The CPU's Interval Timer is started; i.e., allowed to begin counting CPU time.

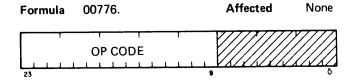
Notes

The Processor Time Mode allows the Interval Timer to count CPU time only. Counting is inhibited when an I/O block channel takes a memory cycle or when an interrupt is active

Once started, the timer counts until held by a HIT instruction or until the CPU is halted.

At each one microsecond interval, the contents of the T Register are decremented by one and tested for zero. If the contents of T are zero, an executive interrupt is triggered. The interrupt does not stop the timer.

RCT Release Clock Time



Operation

The CPU's Interval Timer is started; i.e., allowed to begin counting continuously.

Notes

The Clock Time Mode causes the Interval Timer to count continuously.

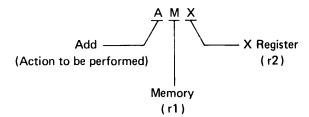
Once started, the timer will count until held by a HIT instruction.

At each one microsecond interval, the contents of the T Register are decremented by one and tested for zero. If the contents of T are zero, an executive interrupt is triggered. The interrupt does not stop the timer.

Scientific Arithmetic Unit Instructions

The Scientific Arithmetic Unit instruction set is divided into five functional groups: arithmetic, transfer, branch, compare, and interrupt control. The CPU operates on normalized floating-point numbers, and all descriptions of the arithmetic instructions are based on this fact. If an unnormalized operand is used in an arithmetic operation the results are not considered valid. Full scale maximum negative (1000....0) is an invalid input. The results of an arithmetic operation are truncated, not rounded.

Standard arithmetic instructions — add, subtract, multiply, and divide — as well as square, square root, fix and float are included in the group. The instruction mnemonics provide a brief definition of specific operations to be performed. The first letter in the mnemonic specifies the action or type of operation that is to be performed. The second letter identifies the first quantity or reference (r1) to be used in the operation, and the third letter identifies the second reference (r2). For example:



In the majority of SAU arithmetic instructions, the result of the operation remains in r2 while r1 remains unchanged (except where r1 and r2 are the same).

Unless otherwise noted, each arithmetic operation sets a bit in the Y Register to reflect the status of the result. Various conditions are described below:

- a) Positive The result is arithmetically greater than zero, indicated by a ONE in bit position 3 of the Y Register. A ZERO in bit position 3 indicates "Not Positive".
- b) Zero All of the mantissa bits comprising the quantity under consideration are ZERO and the exponent is '201, indicated by a ONE in bit position 2 of the Y Register. A ZERO in bit position 2 indicates "Not Zero".
- Negative The result is arithmetically less than zero, indicated by a ONE in bit position 1 of the Y Register.
 A ZERO in bit position ONE indicates "Not Negative".
- d) Overflow An overflow results from an arithmetic operation which causes exponent overflow, i.e., an exponent greater than $2^7 1$ (127) or less than -2^7 (-128).

NOTE

If the SAU Overflow/Underflow executive trap is enabled, any instruction causing the overflow bit of the Y Register to be set will cause an interrupt.

Bits 1, 2 and 3 (Negative, Zero, Positive) of the Y Register are normally mutually exclusive. In certain instances it is desirable to know what operation caused an Overflow, e.g., a division by zero. The following operations cause more than two bits to be set in the Y Register:

- a) Division by zero sets bits 0, 2, 3 ('15)
- b) $\sqrt{-x}$ sets bits 0, 1, 2, 3 ('17)
- c) Float to Fix, X>8388607 sets bits 0, 1, 3 ('13)

If the bit pattern of the mantissa is 100....0 (maximum negative value), invalid answers may result. The floating-point number containing the maximum negative mantissa may be corrected by adding a floating-point zero to it.

If a conditional branch SAU instruction follows an SAU instruction that generates an overflow condition, the interrupt is serviced prior to taking the branch.

Note that condition codes generated by the decimal arithmetic instructions are also loaded into bit positions 3-0 of the Y Register. SAU and decimal instructions should not be intermixed. If SAU instructions follow a string of decimal arithmetic instructions, the Y Register must be initialized prior to executing the SAU instructions. Similarly, the Y Register must be initialized after executing a string of decimal arithmetic instructions if the decimal instructions are followed by SAU instructions.

The algebraic compare instructions which are included in the SAU instruction set compare two referenced, signed (+ or -) quantities. The Y (condition) Register is set according to the result of the comparison. Algebraic comparisons are identified by the letter "C" as the first letter in the instruction mnemonic (e.g., CZX). The second letter in the mnemonic code identifies the first of the compared quantities (r1) and the remaining letter identifies the second quantity (r2). For example:

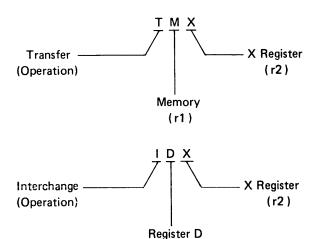
Comparisons are performed according to the following formula:

r2 - r1 = Y (Positive, Zero, or Negative)

Therefore, r2 > r1, r2 < r1, and r2 = r1, will set the condition (Y) register to Positive (+), Negative (-), and Zero (0), respectively.

Two instructions provide control of the SAU interrupt. These instructions either release or hold the interrupt.

The transfer instruction group includes various types of operations. Among these are transfers between memory and registers, registers and memory, and register-to-register. The transfer operation mnemonic code describes the individual operation. What operation is to be performed is described by the first letter in the mnemonic; "T" for transfer and "I" for interchange. The second and third letters of the mnemonic specify the source (r1) and destination (r2) of the transfer, respectively. Listed below are two examples:



With the exception of the interchange instruction, the transfer source (r1) is not altered as a result of the execution of a transfer instruction.

(r1)

The following instructions are included in the SAU group.

ARITHMETIC

Add A Register to X Register	7-75
Add D Register to X Register	7-75
Add Memory to X Register	7-75
Add Operand to W Register	7-75
Add Operand to X Register	7-75
Divide A Register into X Register	7-75
Divide D Register into X Register	7-75
Divide Memory into X Register	7-76
Divide Operand into X Register	7-76
Floating Normalize of A Register	7-77
to X Register	7-76
Fix of X Register to A Register	7-76
	Add D Register to X Register Add Memory to X Register Add Operand to W Register Add Operand to X Register Divide A Register into X Register Divide D Register into X Register Divide Memory into X Register Divide Operand into X Register Floating Normalize of A Register to X Register

INX	Inverse of X Register	7-77
MAX	Multiply A Register and X Register	7-77
MDX	Multiply D Register and X Register	7-77
MMX	Multiply Memory and X Register	7-77
MOX	Multiply Operand and X Register	7-77
NXX	Negative of X Register to X Register	7-77
PXX	Positive of X Register to X Register	7-78
SAX	Subtract A Register from X Register	7-78
SDX	Subtract D Register from X Register	7-78
SEX	Square X Register	7-78
SMX	Subtract Memory from X Register	7-78
SOX	Subtract Operand from X Register	7-78
SRX	Square Root of X Register	7-78

BRANCH

BNR	Branch on Negative Reset	7-79
BNS	Branch on Negative Set	7-79
BOR	Branch on Overflow Reset	7-80
BOS	Branch on Overflow Set	7-80
BOX	Branch on SAU Ready	7-80
BPR	Branch on Positive Reset	7-79
BPS	Branch on Positive Set	7-79
BZR	Branch on Zero Reset	7-79
BZS	Branch on Zero Set	7-79

COMPARE

CDX	Compare D Register to X Register	7-80
COW	Compare Operand to W Register	7-80
CZX	Compare Zero to X Register	7-81

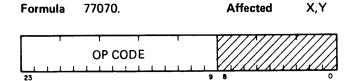
INTERRUPT

HSI	Hold SAU Overflow Interrupt	7-81
RSI	Release SAU Overflow Interrupt	7-81

TRANSFER

IDX	Interchange D Register and	
	X Register	7-81
TDX	Transfer D Register to X Register	7-81
TMX	Transfer Memory to X Register	7-81
TOW	Transfer Operand to W Register	7-81
TOY	Transfer Operand to Y Register	7-82
TXD	Transfer X Register to D Register	7-82
TXM	Transfer X Register to Memory	7-82
TYA	Transfer Y Register to A Register	7-82
TZX	Transfer Zero to X Register	7-82

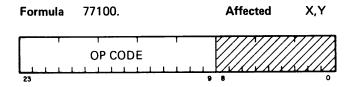
AAXAdd A Register to X Register



Operation

The signed integer in the A Register is converted to floating-point format and added to the number in the X Register. The sum replaces the previous contents of the X Register.

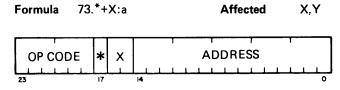
ADX Add D Register to X Register



Operation

The floating-point number in the D Register is added to the number in the X Register. The sum replaces the previous contents of the X Register.

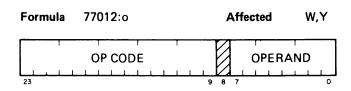
AMX % Add Memory to X Register



Operation

The contents of the effective memory address (EMA) and the next sequential address (EMA+1) are added to the contents of the X Register. The sum replaces the previous contents of the X Register.

AOW Add Operand to W Register (exponent)



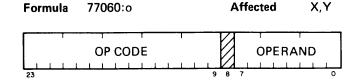
Operation

The 8-bit, signed operand is algebraically added to the contents of the W Register.

X,Y

A subtraction may be accomplished by adding a negative operand.

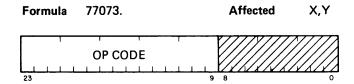
AOX Add Operand to X Register



Operation

The signed, 8-bit integer operand is converted to floating-point format and added to the contents of the X Register. The sum replaces the previous contents of the X Register.

DAX Divide A Register (integer) into X Register



Operation

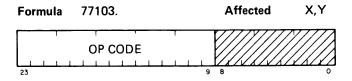
The signed integer in the A Register is converted to floating point format. The contents of the X Register are divided by the converted number. The quotient replaces the previous contents of the X Register.

Notes

If division by zero occurs, the condition register (Y) is set to Overflow, Positive, and Zero, i.e., (Y) = '15.

Division by zero results in a quotient of zero.

DDX Divide D Register (floating-point) into X Register



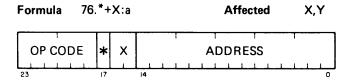
The floating-point contents of the D Register are divided into the contents of the X Register. The quotient replaces the previous contents of the X Register.

Notes

If division by zero occurs, the condition register (Y) is set to Overflow, Positive, and Zero, i.e., (Y) = '15.

Division by zero results in a quotient of zero.

DMX % Divide Memory into X Register



Operation

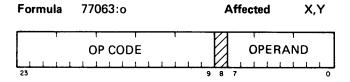
The contents of the X Register are divided by the contents of the effective memory address (EMA) and the next sequential address (EMA+1). The quotient replaces the previous contents of the X Register.

Notes

If division by zero occurs, the condition register (Y) will be set to Overflow, Positive, and Zero, i.e., (Y) = '15.

Division by zero results in a quotient of zero.

DOX Divide Operand into X Register



Operation

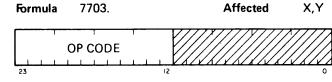
The signed, 8-bit integer operand is converted to floating-point and is divided into the contents of the X Register. The quotient replaces the previous contents of the X Register.

Notes

If division by zero occurs, the condition register (Y) will be set to Overflow, Positive, and Zero, i.e., (Y) = '15.

Division by zero results in a quotient of zero.

FAX Floating Normalize of A Register to X Register



Operation

The signed integer quantity in the A register is converted to a floating-point normalized quantity which replaces the previous quantity in the X Register.

Notes

A positive normalized number will have as the sign and most significant bit the following pattern:

01

A negative normalized number (where the value is not -1) has the configuration

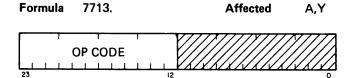
10

If the result is zero, the mantissa will be zero and the exponent will be set to a full scale negative value, i.e., (W) = '201.

The FAX instruction gives the same result as the FNO instruction for -2^N (0 \le N \le 23₁₀).

FNO of
$$-2^N = 1.10...0$$
 EXP = $(N + 1)$
FAX of $-2^N = 1.10...0$ EXP = $(N + 1)$

FXA Fix of X Register to A Register



Operation

The floating-point number in the X Register is converted to a 24-bit signed integer which replaces the previous contents of the A Register.

Notes

If the exponent is greater than 23, the result loaded into the A Register is unspecified and the condition register (Y) will be set to Overflow, Negative and Positive, i.e., (Y) = '13.

If the mantissa is negative, the result when truncated will go toward zero.

INX Inverse of X Register



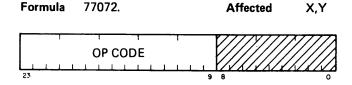
Operation

The inverse of the contents $\left[\frac{1}{(X)}\right]$ of the X Register rereplaces the contents of the X Register.

Note

If division by zero occurs, the condition register will be set to Overflow, Positive, and Zero, i.e., (Y) = '15.

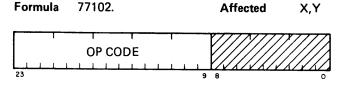
MAX Multiply A Register (integer) and X Register



Operation

The signed integer in the A Register is converted to floating-point format and multiplied by the contents of the X Register. The product replaces the previous contents of the X Register.

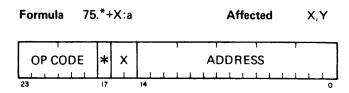
MDX Multiply D Register (floating point) and X Register



Operation

The floating-point contents of the D Register are multiplied by the contents of the X Register. The product replaces the previous contents of the X Register.

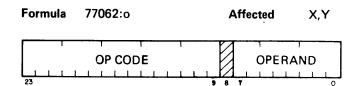
MMX % Multiply Memory and X Register



Operation

The contents of the X Register are multiplied by the contents of the effective memory address (EMA) and the next sequential address (EMA+1). The product replaces the previous contents of the X Register.

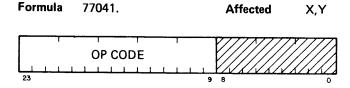
MOX Multiply Operand and X Register



Operation

The signed, 8-bit integer operand is converted to floating-point format and is multiplied by the contents of the X Register. The floating-point product replaces the previous contents of the X Register.

NXX Negative of X Register to X Register



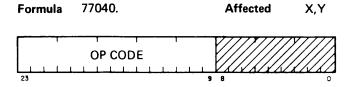
Operation

The mantissa in the X Register is two's complemented and the result is loaded into the X Register. The Y Register is changed to reflect the status of the new quantity.

Note

If the bit pattern of the mantissa is 100....0, the result will be negative.

PXX Positive of X Register to X Register



Operation

The absolute value of the contents of the X Register replaces the previous contents of the X Register.

Notes

If the bit pattern of the mantissa is 100....0, the result will be negative.

The operation noted above may cause a significant difference in a result, i.e., TNA (1), FAX, NXX, FXA generate A = 0; the result should have been 1. However, this may be alleviated by preceeding the NXX with an AOX (0) to normalize the X Register.

SAX Subtract A Register (integer) from X Register

Formula 77071. Affected X,Y

OP CODE

Operation

The signed integer in the A Register is converted to floating-point format and subtracted from the contents of the X Register. The difference replaces the previous contents of the X Register.

SDX Subtract D Register (floating point) from X Register

Formula 77101. Affected X,Y

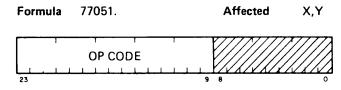
OP CODE

23

Operation

The floating-point contents of the D Register are subtracted from the X Register. The difference replaces the previous contents of the X Register.

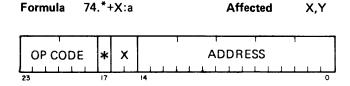
SEX Square X Register



Operation

The square of the contents of the X Register replaces the previous contents of the X Register. (i.e., the X Register is replaced by X times X.)

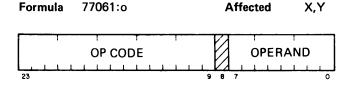
SMX % Subtract Memory from X Register



Operation

The contents of the effective memory address (EMA) and the next sequential address (EMA+1) are subtracted from the contents of the X Register. The difference replaces the contents of the X Register.

SOX Subtract Operand from X Register



Operation

The signed, 8-bit integer operand is converted to a floating-point format and subtracted from the contents of the X Register. The difference replaces the previous contents of the X Register.

SRX Square Root of X Register



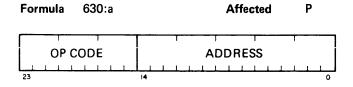
The square root of the contents of the X Register replaces the previous contents of the X Register.

Note

If the content of the X Register is negative, the condition register is set to Positive, Zero, Negative and Overflow, i.e., (Y) = '17.

The contents of X remain unchanged for negative numbers.

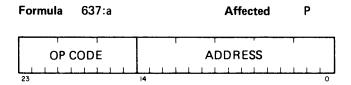
BNR % Branch on Negative Reset



Operation

The contents of the condition (Y) register are tested for the specified condition. If the condition is present, the contents of the P Register (current program address) are replaced by the effective memory address. If the specified condition is not present, the program address advances to the next sequential instruction.

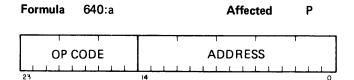
BNS % Branch on Negative Set



Operation

The contents of the condition (Y) register are tested for the specified condition. If the condition is present, the contents of the P Register (current program address) are replaced by the effective memory address. If the specified condition is not present, the program address advances to the next sequential instruction.

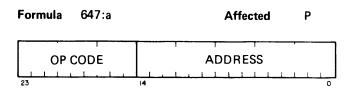
BZR % Branch on Zero Reset



Operation

The contents of the condition (Y) register are tested for the specified condition. If the condition is present, the contents of the P Register (current program address) are replaced by the effective memory address. If the specified condition is not present, the program address advances to the next sequential instruction.

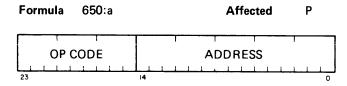
BZS % Branch on Zero Set



Operation

The contents of the condition (Y) register are tested for the specified condition. If the condition is present, the contents of the P Register (current program address) are replaced by the effective memory address. If the specified condition is not present, the program address advances to the next sequential instruction.

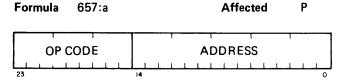
BPR % Branch on Positive Reset



Operation

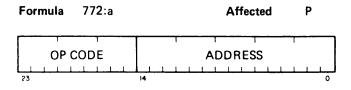
The contents of the condition (Y) register are tested for the specified condition. If the condition is present, the contents of the P Register (current program address) are replaced by the effective memory address. If the specified condition is not present, the program address advances to the next sequential instruction.

BPS % Branch on Positive Set



The contents of the condition (Y) register are tested for the specified condition. If the condition is present, the contents of the P Register (current program address) are replaced by the effective memory address. If the specified condition is not present, the program address advances to the next sequential instruction.

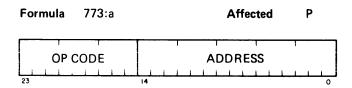
BOR % Branch on Overflow Reset



Operation

The contents of the condition (Y) register are tested for the specified condition. If the condition is present, the contents of the P Register (current program address) are replaced by the effective memory address. If the specified condition is not present, the program address advances to the next sequential instruction.

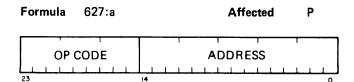
BOS % Branch on Overflow Set



Operation

The contents of the condition (Y) register are tested for the specified condition. If the condition is present, the contents of the P Register (current program address) are replaced by the effective memory address. If the specified condition is not present, the program address advances to the next sequential instruction.

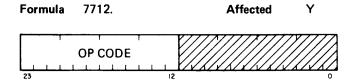
BOX % Branch on SAU Ready



Operation

A determination is made as to whether or not the SAU is processing an instruction (the SAU busy latch is tested). If the SAU is able to process another instruction (i.e., ready) then the contents of the P Register (current program address) are replaced by the effective memory address. If the SAU is currently processing an instruction (i.e., not ready) the program address advances to the next sequential instruction.

CDX Compare D Register to X Register



Operation

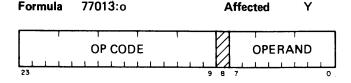
The contents of the D Register and the contents of the X Register are compared and the Y (condition) Register is set to the status of the result.

Note

Comparison results are as follows:

If X is greater than D; Y = Positive
If X is equal to D; Y = Zero
If X is less than D; Y = Negative

COW Compare Operand to W Register (exponent)



Operation

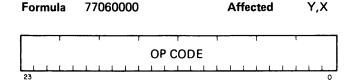
The 8-bit, signed operand and the contents of the W Register are algebraically compared and the Y (condition) Register is set to the status of the result.

Note

Comparison results are as follows:

If W is greater than the operand; Y = Positive If W is equal to the operand; Y = Zero If W is less than the operand; Y = Negative

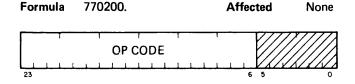
CZX Compare Zero to X Register



Operation

The contents of the X Register and floating-point zero are compared and the Y (condition) Register is set to the status of the result.

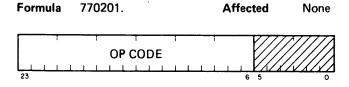
HSI Hold SAU Overflow Interrupt



Operation

This instruction disarms the overflow/underflow interrupt (Executive trap Group 0, Level 7). The trap remains disarmed until the execution of the release instruction.

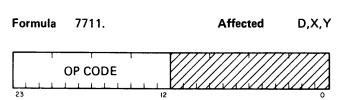
RSI Release SAU Overflow Interrupt



Operation

This instruction arms the overflow/underflow interrupt (Executive Trap Group 0, Level 7). When the trap is armed, and not inhibited by an HXI instruction, any SAU operation which causes bit 0 of the Y Register to be set (Overflow) will generate an interrupt request.

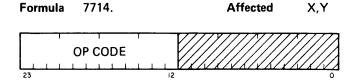
IDX Interchange D Register and X Register



Operation

The contents of the X Register and the D Register are interchanged. The Y (condition) Register is set to the status of the X Register on completion of the instruction.

TDX Transfer D Register to X Register



Operation

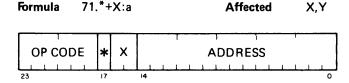
The contents of the D Register replace the previous contents of the X Register.

Notes

An unnormalized number transferred to X may not set the Y Register properly.

A binary zero transferred to X will set Zero.

TMX % Transfer Memory to X Register



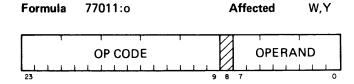
Operation

The contents of the effective memory address (EMA) and the next sequential address (EMA+1) replace the previous contents of the X Register. EMA and EMA+1 replace the most significant and least significant part of X, respectively.

Note

With the exception of a binary ZERO, the Y Register setting is unspecified based on the result of loading the X Register with an unnormalized number. When a binary ZERO is loaded into the X Register, the Y Register is set to zero.

TOW Transfer Operand to W Register (exponent)

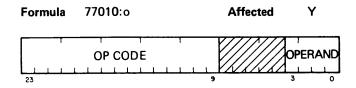


The 8-bit, signed operand replaces the previous contents of the W Register. All other bits within the X Register are unaffected.

Note

The Y (condition) Register is set to the status of the X and XW Registers upon completion of the instruction. The SAU uses the two most significant bits of the mantissa and the sign of the exponent to set the Y Register.

TOY Transfer Operand to Y Register



Operation

The four bit operand replaces the previous contents of the Y (condition) Register.

Note

Operand definition is as follows:

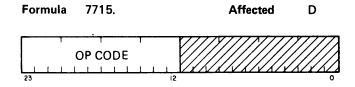
Bit 0 = ONE = Overflow
= ZERO = No Overflow

Bit 1 = ONE = Negative
= ZERO = Not Negative

Bit 2 = ONE = Zero
= ZERO = Not Zero

Bit 3 = ONE = Positive = ZERO = Not Positive

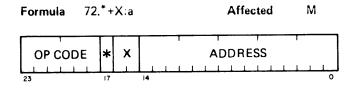
TXD Transfer X Register to D Register



Operation

The contents of the X Register replaces the previous contents of the D Register. The X Register is unchanged.

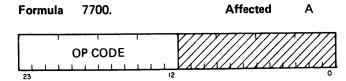
TXM % Transfer X Register to Memory



Operation

The contents of the X Register replaces the previous contents of the effective memory address (EMA) and the next sequential address (EMA+1). The most and least significant portions of X are transferred to EMA and EMA+1, respectively.

TYA Transfer Y Register to A Register



Operation

The contents of the Y Register are transferred to the A Register and the status of the SAU overflow/underflow interrupt is placed in bit position 6 in the A Register.

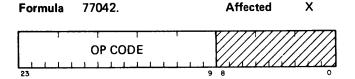
Note

The following table shows the bit placements of the various Y (condition) Register settings when transferred to the A Register.

A Register	Bit Function
Bit 0 = 1	Overflow/Underflow
Bit 1 = 1	Negative
Bit 2 = 1	Zero
Bit 3 = 1	Positive
Bit 6 = 0	SAU Interrupt Enablèd
Bit 6 = 1	SAU Interrupt Disabled

All other bits within the A Register are set to zero.

TZX Transfer Zero to X Register



The floating-point representation of zero (0000000000000001) replaces the previous contents of the X Register. The Y (condition) Register is unaffected.

Decimal Arithmetic Instructions

Decimal arithmetic is performed on data in packed format. In this format, two decimal digits are placed in one byte (four bits each). The operands may be variable in length. The format contains no sign codes; the signs of the fields are maintained in software.

All decimal instructions are executed only in the standard format. The instruction set includes addition and subtraction. Since data sent to, and from, external devices are usually in zoned format (one digit in one byte), there are also instructions for converting to, and from, packed and zoned format.

In packed format, one byte represents two decimal digits. Decimal operands are four-bit binary-coded decimal (BCD) digits packed two to a byte. The packed format is as follows:

BYTE		BYTE		BYTE	
DIGIT	DIGIT	DIGIT	DIGIT	DIGIT	DIGIT

In zoned (unpacked) format, the low-order four bits of each eight-bit byte contain the decimal digit and the high-order four bits contain the zone. The zoned format is as follows:

BYTE		BYTE		BYTE	
ZONE	DIGIT	ZONE	DIGIT	ZONE	DIGIT

BYTE		BYTE		BYTE	
ZONE	DIGIT	ZONE	DIGIT	ZONE	DIGIT

ASCII digits, ASCII space (0408), and binary zeros are the only valid characters permitted. The zone code is 00112. The digit codes are as follows:

DIGIT	CODE
0	0000
1	0001

2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

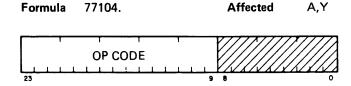
Decimal arithmetic instructions operate from right to left. Operands may be variable in length, but unused bytes must be filled with valid characters or the results are unspecified.

Condition codes generated by the decimal arithmetic instructions are loaded into bits 3-0 of the Y Register. Since condition codes generated by SAU instruction are also transferred to the Y Register, decimal and SAU instructions should not be intermixed. See also paragraph describing SAU interrupts in Section VI. If decimal instructions follow a string of SAU instructions, the Y Register must be initialized prior to executing the decimal instructions. Similarly, the Y Register must be initialized after executing a string of SAU instructions if the SAU instructions are followed by decimal instructions.

The following instructions are included in the decimal arithmetic group.

APA	Add Packed to A	7-84
PDA	Pack Double to A	7-83
SPA	Subtract Packed from A	7-84
UAD	Unpack A to Double	7-84

PDA Pack Double to A



Operation

The contents of the D Register are converted from zoned format to packed format and the result is placed in the A Register.

Notes

The contents of D must be in zoned format.

If D contains less than six valid characters, the unused bytes must be filled with ASCII zeros, ASCII spaces, or binary zeros.

All zones are checked for validity; the four-bit digit portions (stripping the four-bit zone) of each byte are then placed adjacent to each other to fill the result field in A.

The condition code settings are as follows:

If all characters are valid and the result in A is not zero, the Y Positive bit is set.

If all characters are valid and the result in A is zero, the Y Zero bit is set.

If there is at least one invalid character; i.e., if any of the characters in D are not valid ASCII digits, ASCII spaces, or binary zeros, the Y Negative bit is set.

UAD Unpack A to Double

Formula 77105. Affected E,A



Operation

The contents of the A register are converted from packed format to zoned format, and the result is placed in the D Register.

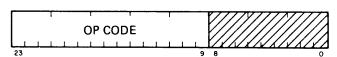
Notes

The three bytes in A are translated to equivalent ASCII characters in D. Each of the eight-bit bytes of the packed field in A represents two, four-bit digits. Each of the four-bit digits is stored in a byte in D in the low-order four-bit positions. A zone code of 0011 is inserted into the high-order four bits of each byte.

The Y Register is unaffected.

APA Add Packed to A

Formula 77106. Affected A,Y



Operation

The contents of the E Register and the carry in (Y Negative bit) are added to the contents of the A Register. The Binary-Coded Decimal (BCD) result is placed in the A

Register (A = A + E + Y Neg), and the carry out is stored in Y Neg.

Notes

The contents of either A or E, or both, may be in ten's complement form if the result is properly interpreted.

If the contents of both A and E represent positive BCD numbers, and

if $A + E + Y \text{ Neg } \neq 0$; Y Pos (not Zero) is set and Y Neg is set if carry out is true.

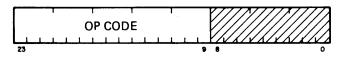
if A + E + Y Neg = 0; Y Pos (not Zero) is not changed and Y Neg is set if carry out is true.

The Y Register should be reset to zeros at the beginning of an APA. Two positive operands of 6N digits may then be added without further program manipulation or testing of the Y Register until after the Nth set of digits have been added. The Y Register may then be examined for the condition Y Pos = not Zero and Y Neg = carry out.

In all cases, the Y Zero and Overflow bits are reset.

SPA Subtract Packed from A

Formula 77107. Affected A,Y



Operation

The contents of the E Register and the not borrow (Y Negative bit) are subtracted from the contents of the A Register. The Binary-Coded Decimal (BCD) result is placed in the A Register (A = A - E - 1 + Y Neg), and the not borrow is stored in Y Neg.

Notes

The contents of either A or E, or both, may be in ten's complement form if the result is properly interpreted.

If the contents of both A and E represent positive BCD numbers, and

if A > E, the result is positive; Y Pos (not Zero) is set and Y Neg (not borrow) is set.

If A < E, the result is negative (ten's complement form); Y Pos (not Zero) is set and Y Neg (not borrow) is reset.

If A = E, the result is zero; Y Pos and Y Neg are unchanged.

The Y Positive, Y Zero, and Y Overflow bits should be reset to zeros, and the Y Negative bit set to one, at the beginning of an SPA. Two positive operands of 6N digits may then be subtracted without further program manipulation or testing of the Y Register until after the Nth set of digits have been subtracted. The Y Register may then be examined for the condition Y Pos = not Zero and Y Neg = not borrow. If the result is positive, Y Neg = 1. If the result is negative, Y Neg = 0.

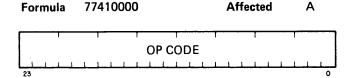
In all cases, the Y Zero and Overflow bits are reset.

Diagnostic Instructions

Diagnostic instructions are used primarily to support the diagnostic software. The following instructions are included in the diagnostic group.

ACE	Transfer Active Executive Traps to A	7-85
HER	Hold Parity Error Retry	7-85
LVR	Load Virtual Demand Page Register	7-87
MIR	MAP Interrupt Request	7-87
RER	Release Parity Error Retry	7-85
RPB	Read Parity Bits	7-86
TCD	Transfer CAM to Double	7-86
THA	Transfer CAM Hit Status to A	7-87
TPA	Transfer Parity Error Address	
	Register to A	7-86

ACE Transfer Active Executive Traps to A



Operation

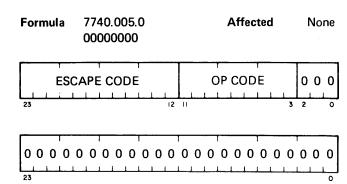
The current status of the executive trap interrupts (Group 0, Levels 7-0) is transferred to A7-A0, and the status of the HXI instruction execution is transferred to A11. The remaining bits of A are cleared.

Notes

The Condition Register remains unchanged.

This is a privileged instruction.

HER Hold Parity Error Retry



Operation

Memory read retry operations following parity errors are inhibited.

Notes

The HER instruction is valid only in the extended instruction format.

Once executed, the CPU or I/O will not retry a memory read operation if a parity error has occurred.

When this instruction is executed, all subsequent parity errors are reported as being hard errors and an executive trap (Group 0, Level 1) interrupt is generated.

The inhibit is removed by either executing the RER instruction or executing the Master Clear (MCL) command from the MAP.

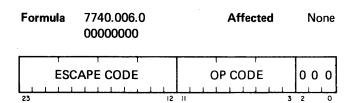
If a parity error occurs when memory read retries are inhibited, operation is unpredictable and the erroneous data is retained in cache memory.

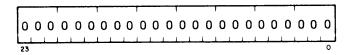
The Condition Register remains unchanged.

The second word of the extended instruction is read but not used.

This instruction is privileged.

RER Release Parity Error Retry





The memory read retry inhibit imposed by execution of the HER instruction is removed.

Notes

The RER instruction is valid only in the extended instruction format.

When this instruction is executed, subsequent parity errors are reported as being either soft or hard, providing no parity error occurs after execution of the HER instruction and before execution of the RER instruction.

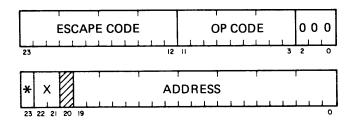
The Condition Register remains unchanged.

The second word of the extended instruction is read but not used.

This instruction is privileged.

RPB Read Parity Bits

Formula 7740.004.0 Affected A *+X:EA



Operation

The error correction bits of the 64K MOS memory module selected by the effective memory address are read and loaded into the A Register. The even word error correction bits are transferred to A9-A5, and the odd word error correction bits are transferred to A4-A0.

Notes

The RPB instruction is valid only in the extended instruction format and only for 64K MOS memory modules.

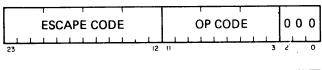
Following execution of the RPB instruction, a soft parity error is reported the next time the same memory module is accessed.

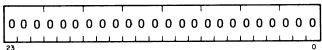
The Condition Register remains unchanged.

This instruction is privileged.

TPA Transfer Parity Error Address Register to A

Formula 7740.001.0 Affected A, PEAR 000000000





Operation

The 24-bit contents of the Parity Error Address Register (PEAR) replace the previous contents of the A Register. The PEAR is cleared and armed for the next parity error occurrence immediately after the transfer has been completed.

Notes

The TPA instruction is valid only in the extended instruction format.

The Condition Register remains unchanged.

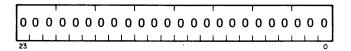
The second word of the extended instruction is read but not used.

This instruction is privileged.

TCD Transfer CAM to Double

Formula 7740.000.0 Affected E,A 00000000





The contents of the Virtual Physical Fetch (VPF) Register replace the previous contents of E Register bits 11-0, and the contents of the Virtual Physical Operand (VPO) Register replace the previous contents of A Register bits 11-0.

Notes

The TCD instruction is valid only in the extended instruction format.

The Condition Register remains unchanged.

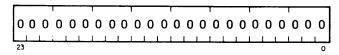
The second word of the instruction is read but not used.

This instruction is privileged.

THA Transfer CAM Hit Status to A

Formula 7740.003.0 Affected A 000000000





Operation

The contents of the Virtual Hit Status (VHS) Register replace the previous contents of A Register bits 2-0.

Notes

Bit VHS2 (fetch hit status) is transferred to A2, bit VHS1 (operand 1 hit status) is transferred to A1, and VHS0 (operand 2 hit status) is transferred to A0.

The THA instruction is valid only in the extended instruction format.

The instruction following a ROM instruction leaves the fetch hit status bit unchanged and both operand hit status bits reset.

The Condition Register remains unchanged.

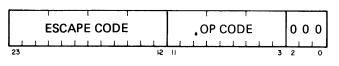
The second word of the extended instruction is read but not used.

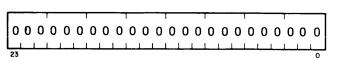
This instruction is privileged.

LVR Load Virtual Demand Page Register

Formula 7740.007.0 Affected VPR

00000000





Operation

Bits 19-10 and 3-0 of the A Register replace the previous contents of the Virtual Demand Page Register (VPR). Bits A19-10 are transferred to VPR13-4, and bits A3-0 are transferred to VPR3-0.

Notes

The LVR instruction is valid only in the extended instruction format.

The Condition Register remains unchanged.

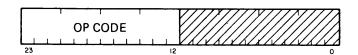
Execution of this instruction is valid only in the Monitor Mode.

The second word of the extended instruction is read but not used.

This is a privileged instruction.

MIR MAP Interrupt Request

Formula 7776. Affected



Operation

The CPU generates an interrupt request to the Maintenance Aid Processor.

Notes

The C Register remains unchanged.

This instruction is privileged.

None

APPENDIX A INSTRUCTION INDEX

Mnemonic	Instruction	Page
AAM	Add A Register to Memory	7-6
AAX	Add A Register to X Register	7-75
ACE	Transfer Active Executive Traps to A	7-85
ADX	Add D Register to X Register	7-75
AEM	Add E Register to Memory	7-7
AMA	Add Memory to A	7-5
AMB	Add Memory to Byte	7-6, 7-43
AMD	Add Memory to Double Register	7-6
AME	Add Memory to E Register	7-5
AMx	Add Memory to Register	7-5
AMX	Add Memory to X Register	7-75
AOB	Add Operand to Byte	7-7, 7-43
AOM	Add Operand to Memory	7-7
AOr	Add Operand to Register	7-7
AOW	Add Operand to W Register	7-75
AOX	Add Operand to X Register	7-75
APA	Add Packed to A	7-84
Arr	Add Register to Register	7-8
AUM	Add Unity to Memory	7-5
AxM	Add Register to Memory	7-6
BBI	Branch when Byte Address +1 in I \neq 0	7-19, 7-43
BBJ	Branch when Byte Address +1 in $J \neq 0$	7-20, 7-44
BJL	Branch Indexed by J Long	7-21
BLL	Branch and Link (J) Long	7-22
BLU	Branch and Link Unrestricted	7-24
BLx	Branch and Link Register	7-22
BNc	Branch on Condition Code	7-21
BNR	Branch on Negative Reset	7-79
BNS	Branch on Negative Set	7-79
BOc	Branch on Condition Code	7-21
BOR	Branch on Overflow Reset	7-80
BOS	Branch on Overflow Set	7-80
вох	Branch on SAU Ready	7-80
BPR	Branch on Positive Reset	7-79
BPS	Branch on Positive Set	7-79
BRL	Branch and Reset Interrupt Long	7-23, 7-61
BSL	Branch and Save Return Long	7-22, 7-60
BSX	Branch and Save Extended	7-23, 7-61

Mnemonic	Instruction	Page
BUC	Branch Unconditionally	7-20
BUL	Branch Unconditionally Long	7-21
BWx	Branch when Register +1 ≠ 0	7-21
BZR	Branch on Zero Reset	7-79
BZS	Branch on Zero Set	7-79
CDX	Compare D Register to X Register	7-80
CMA	Compare Memory and A	7-25
CMB	Compare Memory and Byte	7-26, 7-45
CME	Compare Memory and E	7-26
CMx	Compare Memory and Register	7-25
COB	Compare Operand and Byte	7-26, 7-45
COW	Compare Operand to W Register	7-80
Crr	Compare Register and Register	7-27
CZD	Compare Zero and Double	7-27
CZM	Compare Zero and Memory	7-26
CZr	Compare Zero and Register	7-26
CZX	Compare Zero to X Register	7-81
DAX	Divide A Register into X Register	7-75
DDX	Divide D Register into X Register	7-75
DMA	Dot Memory with A	7-28
DMH	Dot Memory with H	7-55
DMX	Divide Memory into X Register	7-76
DNH	Dot Not (memory) with H	7-55
DOB	Dot Operand with Byte	7-28, 7-45
DOX	Divide Operand into X Register	7-76
Drr	Dot Register with Register	7-29
DVM	Divide by Memory	7-8
DVO	Divide by Operand	7-8
DVT	Divide by T	7-9
DVx	Divide by Register	7-9
DV2	Divide by 2	7-9
ЕМВ	Extract Memory Byte	7-34, 7-45
ESA	Extend Sign of A	7-10
ESB	Extend Sign of Byte	7-10, 7-46
EXM	Execute Memory	7-71
EZB	Extend Zeros from Byte	7-46, 7-72
FAX	Floating Normalize of A Register to X Register	7-76
FBM	Flag Bit of Memory	7-56
FNO	Floating Normalize	7-10
FXA	Fix of X Register to A Register	7-76

Mnemonic	Instruction	Page
GAP	Generate Argument Pointer	7-70
HER	Hold Parity Error Retry	7-85
HIT	Hold Interval Timer	7-72
HLT	Halt	7-70
HSI	Hold SAU Overflow Interrupt	7-81
HTx	Hold Interrupts and Transfer Register to Memory	7-62
HXI	Hold External Interrupts	7-63
IAW	Input Address Word	7-52
IDW	Input Data Word	7-52
IDX	Interchange D Register and X Register	7-81
IMA	Interchange Memory and A	7-35
IME	Interchange Memory and E	7-35
IMx	Interchange Memory and Register	7-35
INX	Inverse of X Register	7-77
IPW	Input Parameter Word	7-53
Irr	Interchange Register and Register	7-35
ISW	Input Status Word	7-51
ков	Kompare Operand and Byte	7-27
Krr	Kompare Register and Register	7-27
LAA	Left Shift Arithmetic A	7-31
LAD	Left Shift Arithmetic Double	7-31
LLA	Left Shift Logical A	7-31
LLD	Left Shift Logical Double	7-31
LRA	Left Rotate A	7-32
LRD	Left Rotate Double	7-32
LVR	Load Virtual Demand Page Register	7-87
MAX	Multiply A Register and X Register	7-77
MDX	Multiply D Register and X Register	7-77
MIR	MAP Interrupt Request	7-87
MMX	Multiply Memory and X Register	7-77
MOX	Multiply Operand and X Register	7-77
MYM	Multiply by Memory	7-10
MYO	Multiply by Operand	7-10
MYr	Multiply by Register	7-11

Mnemonic	Instruction	Page
NBB	Negate of Byte to Byte	7-11, 7-46
NDD	Negate of Double to Double	7-12
NHH	Negate of H to H	7-54
NOP	No Operation	7-70
NSr	Negate Sign of Register	7-12
Nrr	Negate of Register to Register	7-11
NXX	Negative of X Register to X Register	7-77
OAW	Output Address Word	7-52
OCW	Output Command Word	7-50
ODW	Output Data Word	7-51
OMA	OR Memory with A	7-29
ОМН	OR Memory with H	7-55
ONH	OR Not (memory) with H	7-55
OOB	OR Operand with Byte	7-29, 7-46
Orr	OR Register with Register	7-29
PBB	Positive of Byte to Byte	7-12, 7-46
PDA	Pack Double to A	7-83
PDD	Positive of Double to Double	7-12
Prr	Positive of Register to Register	7-13
PXX	Positive of X Register to X Register	7-78
QBB	Query Bits of Byte	7-47, 7-71
QBH	Query Bit of H	7-54
QBM	Query Bit of Memory	7-56
QNR	Query Not-modified Register	7-59
QSS	Query Sense Switches	7-72
QUR	Query Usage Register	7-59
RAA	Right Shift Arithmetic A	7-32
RAD	Right Shift Arithmetic Double	7-32
RBM	Replace Byte in Memory	7-36, 7-47
RCT	Release Clock Time	7-72
RER	Release Parity Error Retry	7-85
RLA	Right Shift Logical A	7-32
RLD	Right Shift Logical Double	7-33
ROM	Release Operand Mode	7-59
RPB	Read Parity Bits	7-86
RPT	Release Processor Time	7-72
RRA	Right Rotate A	7-33
RRD	Right Rotate Double	7-33

Mnemonic	Instruction	Page
Rrr	Round of Register to Register	7-13
RSI	Release SAU Overflow Interrupt	7-81
RUM	Release User Mode	7-60
RXI	Release External Interrupts	7-63
SAX	Subtract A Register from X Register	7-78
SDX	Subtract D Register from X Register	7-78
SEX	Square X Register	7-78
SMA	Subtract Memory from A	7-14
SMB	Subtract Memory from Byte	7-14, 7-47
SMD	Subtract Memory from Double	7-14
SME	Subtract Memory from E	7-14
SMx	Subtract Memory from Register	7-13
SMX	Subtract Memory from X Register	7-78
SOB	Subtract Operand from Byte	7-15, 7-47
SOr	Subtract Operand from Register	7-15
SOX	Subtract Operand from X Register	7-78
SPA	Subtract Packed from A	7-84
SRE	Square Root Extended	7-16
Srr	Subtract Register from Register	7-15
SRT	Square Root	7-15
SRX	Square Root of X Register	7-78
TAM	Transfer A to Memory	7-41
TAR	Transfer A to 1 Virtual Address Register	7-57
ТВМ	Transfer Byte to Memory	7-41, 7-48
TCD	Transfer CAM to Double	7-86
TDM	Transfer Double to Memory	7-41
TDP	Transfer Double to Paging Limit Registers	7-58
TDR	Transfer Double to 2 Virtual Address Registers	7-58
TDS	Transfer Double to Source and Destination Registers	7-57
TDX	Transfer D Register to X Register	7-81
TD1	Transfer Double to Group 1	7-63
TD2	Transfer Double to Group 2	7-63
TD3	Transfer Double to Group 3	7-63
TD4	Transfer Double to Group 1	7-65
TD5	Transfer Double to Group 2	7-65
TD6	Transfer Double to Group 3	7-65
TEM	Transfer E to Memory	7-41
TEU	Transfer E to Virtual Usage Base Register	7-59
TFH	Transfer Flag to H	7-54

Mnemonic	Instruction	Page
	T. (El . Marrom	7-41
TFM	Transfer Flag to Memory	7-41 7-87
THA	Transfer CAM Hit Status to A	. •
THM	Transfer H to Memory	7-56 7-42
TIM	Transfer I to Memory	7-42 7-42
TJM	Transfer J to Memory	7-42 7-42
TKM	Transfer K to Memory Transfer K to V	7-42 7-54
TKV	Transfer K to V Transfer Extended Operand to K	7- 54 7-39
TLK TLO	Transfer Extended Operand to K Transfer Long Operand to K	7-3 9 7-39
TMA		7-3 9 7-37
TMB	Transfer Memory to A Transfer Memory to Byte	7-36, 7-48
TMD	Transfer Memory to Double	7-36, 7-48 7-36
TME	Transfer Memory to E	7-37
TMH	Transfer Memory to H	7-57 7-56
TMI	Transfer Memory to I	7-30 7-37
TMJ	Transfer Memory to J	7-37 7-38
TMK	Transfer Memory to K	7-38 7-38
TMQ	Transfer Memory to Query Register	7-30 7-37
TMR	Transfer Memory to Registers	7-37 7-38
TMX	Transfer Memory to X Register	7-81
TNr	Transfer Negative Operand to Register	7-31 7-38
ТОВ	Transfer Operand to Byte	7-38, 7-48
TOC	Transfer Operand to Condition Register	7-39
TOr	Transfer Operand to Register	7-39
TOW	Transfer Operand to W Register	7-81
TOY	Transfer Operand to Y Register	7-82
TPA	Transfer Parity Error Address Register to A	7-86
TPD	Transfer Paging Limit Registers to Double	7-58
TrB	Transfer Register to Byte	7-40, 7-48
TRD	Transfer 2 Virtual Address Registers to Double	7-58
TRM	Transfer Registers to Memory	7-42
Trr	Transfer Register to Register	7-42
TSD	Transfer Source and Destination Registers to D	7-57
TSr	Transfer Switches to Register	7-40
TUD	Transfer Usage Base Register and Demand Page Register to Double	7-58
TVK	Transfer V to K	7-54
TXD	Transfer X Register to D Register	7-82
TXM	Transfer X Register to Memory	7-82
TYA	Transfer Y Register to A Register	7-82
TZH	Transfer Zero to H	7-54
TZM	Transfer Zero to Memory	7-41
TZr	Transfer Zero to Register	7-40

Mnemonic	Instruction	Page
TZX	Transfer Zero to X Register	7-82
T1D	Transfer Group 1 to Double	7-64
T2D	Transfer Group 2 to Double	7-64
T3D	Transfer Group 3 to Double	7-64
T4D	Transfer Group 1 to Double	7-64
T5D	Transfer Group 2 to Double	7-65
T6D	Transfer Group 3 to Double	7-65
UAD	Unpack A to Double	7-84
UA1	Unitarily Arm Group 1 Interrupts	7-65
UA2	Unitarily Arm Group 2 Interrupts	7-66
UA3	Unitarily Arm Group 3 Interrupts	7-66
UD1	Unitarily Disarm Group 1 Interrupts	7-66
UD2	Unitarily Disarm Group 2 Interrupts	7-67
UD3	Unitarily Disarm Group 3 Interrupts	7-67
UE1	Unitarily Enable Group 1 Interrupts	7-67
UE2	Unitarily Enable Group 2 Interrupts	7-68
UE3	Unitarily Enable Group 3 Interrupts	7-68
UI1	Unitarily Inhibit Group 1 Interrupts	7-68
UI2	Unitarily Inhibit Group 2 Interrupts	7-69
UI3	Unitarily Inhibit Group 3 Interrupts	7-69
USP	Update Stack Pointer	7-70
XMA	Exclusive-OR Memory with A	7-30
XMH	Exclusive-OR Memory with H	7-55
XNH	Exclusive-OR Not (memory) with H	7-56
XOB	Exclusive-OR Operand with Byte	7-30, 7-48
Xrr	Exclusive-OR Register with Register	7-30
ZBM	Zero Bit of Memory	7-56