IBM Maintenance Library

3145 Processing Unit Theory – Maintenance

Volume 30	Volume 31	Volume 32	Volume 33	Volume 34
INDEX	MICroprogram	PoWeR	CHaNneLs	System CoNSoLe
PLAN	Console File Adapter	STORage	Integrated File Adapter	RECovery Features
INTRoduction	Console Printer Keyboard		Optional FEATures	DIAGnostic Functions
CPU Hardware				REFerence

SY24-3581-4

OUTLINE

3145 Processing Unit Theory-Maintenance

VOL 30 — INDEX, PLAN, INTR, CPU

VOL 31 — MIC, CFA, CPK

VOL 32 — PWR, STOR

VOL 33 — CHNL, IFA, FEAT

VOL 34 — CNSL, REC, DIAG, REF

PLAN
- Preface
- Documentation Plan
- Abbreviations
- Legend
- Index

INTR
- Model 145 General Description
- CPU Configuration
- Features
- 3145 Data Flow
- Microprograms
- Error Handling
- Compatability with System/360
- 3145 Channels General Description Description
- Integrated File Adapter

CPU
- SPTL SDBO Pre-Asm, Asm
- SDBO, Pre-Assembler unit Data Flow
- Local Storage
- Expanded Local Storage (EXPLS)
- External Facilities
- I-Cycles
- A-Register B-Register and ALU
- Z-Register and D-Register
- C-Register (Control Word Decode)
- CPU Clock
- M, N, and MB-Registers
- Secondary Control Assembler
- Control Registers
- Standard Features

MIC
- Microprogram Principles

CFA
- Introduction
- Data Flow
- Console File Clock
- Shift Register
- Sector Ready
- CF Commands
- CF Error Checks
- Initial Microprogram Load (IMPL)
- CF Removal and Replacement

CPK
- Introduction
- 3210 Console Printer-Keyboards
- 3210 PK-KB Integrated Attachment
- 3215 PK-KB Integrated Attachment
- Alter/Display Operations
- Microprogram Operation
- Programming Information

PWR
- General Information
- Safety
- Motor Generator
- Electrical Component Locations
- Power Distribution
- CPU Power Conversion and Distribution
- Power Frame, Power Conversion and Distribution
- Controls and Indicators
- Power Sequencing
- Protection and Checking
- Power Service Checks
- Regulator Removal and Replacement
- AC/DC Module Removal and Replacement
- Blowers and Air Filters
- Motor Generator Removal and Replacement
- Preventive Maintenance
- Motor Generator Troubleshooting
- Power Check Troubleshooting

STOR
- General Information
- Functional Areas
- Functional Operations
- Servicing Information

CHNL
- Model 146 I/O Channels
- Byte-Multiplexer Channel
- Selector Channels
- Block-Multiplexer Feature

IFA
- Abbreviation List
- Diagram List Introduction
- Functional Units
- Theory of Operation
- Maintenance Features
- IFA Power Supplies and Interface

FEAT
- IBM 1401/1440/1460 and 1410/7010 Compatibility
- Direct Control
- Channel-to-Channel Adapter

CNSL
- Fixed Indicators
- Roller Indicators
- Rotary Control Switches
- Rotary Switches A-H
- Toggle Switches
- Keys
- Use Meters

REC
- Introduction
- Recording Modes
- Error Checking and Correction
- Machine Checks and Error Retry
- Summary of Machine Check Handling Feature
- Stop After Log
- Software Recovery

DIAG
- Testing Philosophy
- Basic Diagnostic Troubleshooting
- Extended Tests Troubleshooting
- CPU Diagnostic Hardware
- Selector Channel GA Diagnostic Functions
- 3215 Diagnostic Functions
- IFA Diagnostic Functions
- Diagnostic Test Listings

REF
- Voltage Distribution
- Basic Logic Symbology
- 3145 Support Documents
- Functions of Diagnose Instruction
- CPU Maintenance Aids
- Channel Maintenance Aids
- Console-Printer-Keyboard Maintenance Aids
- IFA Maintenance Aids
- 370 Microprogram Load
- Major Functional Unit Card Assignments
- Plug Card Information
- MST Wiring Change Procedure
- 370 Microprogram Index

CONTENTS

SYSTEM/370 MODEL 145 — GENERAL DESCRIPTION

The System/370 Model 145 is a high-availability, general-purpose data processing system that provides the reliability, availability, performance, and convenience demanded by both business and scientific users. This is achieved by:

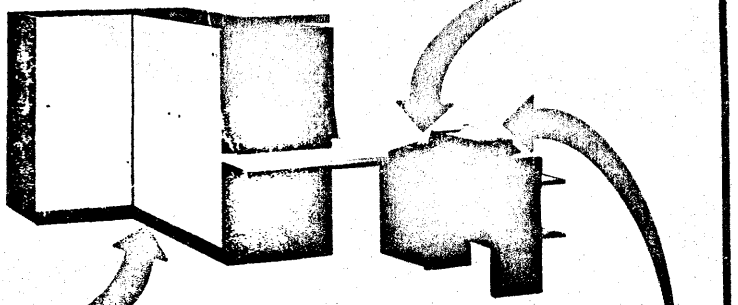
- Using monolithic system technology (MST) circuitry. All system storage–local, control, and main–is implemented using monolithic technology.
- Providing logout information. Hard copy is available under console switch control. Programming determines whether the logout information is to be written on some I/O device (disk, tape, etc).
- Providing microprogram retry. Detected CPU hardware errors can be retried automatically by CPU retry hardware. CPU retry is accomplished by additional microprogram routines and hardware.
- Providing error checking and correction (ECC). Correction circuitry for both main and control storage automatically corrects single-bit errors. Double-bit storage errors are detected, the error location is indicated in fixed storage, and a machine-check interrupt occurs.
- Providing expanded machine-check interrupt facilities to facilitate better error recording and recovery procedures.

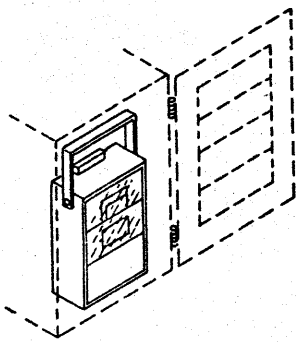
3145 PROCESSING UNIT (CPU) contains from 112k to 2048k bytes of main storage, plus 32k for control storage. Monolithic control storage is reloadable and is used to contain the microprogram necessary for system operation. Control storage is not accessible to the user.

The CPU contains all hardware controlled by the microprogram, necessary to decode and execute the System/370 instruction set and, optionally, those in the hardware compatibility features required by the 1401/1440/1460 and 1410/7010 emulator programs.

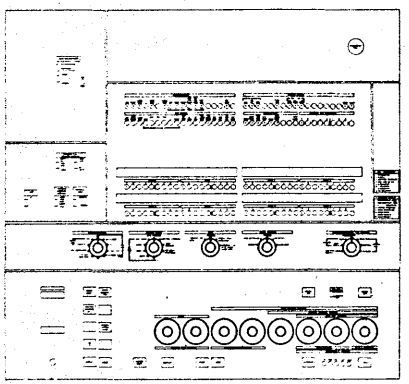
All CPU and channel operations are controlled by the microprogram contained in control storage loaded from the console file.

The CPU clock is basically one main oscillator. Its pulses are distributed to each board. Each board clock then develops the basic timing signals to time the CPU circuitry. These clocks have four basic cycle times: 202.5, 247.5, 292.5, and 315 nanoseconds.



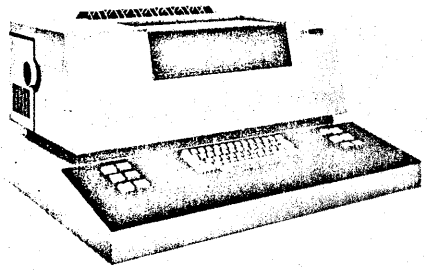


CONSOLE FILE through an integrated attachment is used to load control storage with either the System/370 microprogram for customer operations or with the microdiagnostic used to check out the CPU. For details about this integrated attachment, refer to "Console-File Adapter (CFA)."

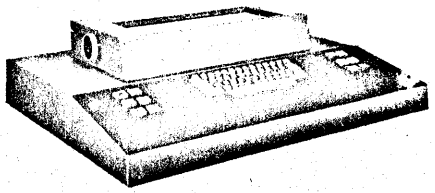


SYSTEM CONTROL PANEL contains the operator panels, the lights, and the switches used during check-out and maintenance of the system. This console contains roll charts that can be checked to determine the status of different conditions using the same lights. For a detailed description of the system control panel, refer to "System Console (CNSL)."

Either Console Printer-Keyboard can be used. For information on integrated attachment for these Console Printer-Keyboards, refer to "Console Printer-Keyboard (CPK)."



3215 CONSOLE PRINTER-KEYBOARD is a wire-matrix printer that prints 85 characters per second.



3210 CONSOLE PRINTER-KEYBOARD MODEL 1 uses the SELECTRIC ® I/O II printing unit. The print element contains 88 characters arranged in an optimized pattern to provide fast response. The printing speed is 15.5 characters per second.

FEATURES

3145 Models GE, GFD, H, HG, and I

Standard Features

3145 Processing Unit (with main power frame)
160k to 512k Bytes of Main Storage
32k Bytes of Control Storage

Model	Main Storage	Control Storage
3145FED*	114,688 bytes (112k)	
3145GE	163,840 bytes (160k)	
3145GFD	212,992 bytes (208k)	See Note 1
3145H	262,144 bytes (256k)	
3145HG**	393,216 bytes (384k)	
3145I**	524,288 bytes (512k)	

*Withdrawn from the product line.

**See Note 2.

Audible Alarm
Byte-Multiplexer Channel
Byte-Oriented Operand
Channel Indirect Data Address
Channel Retry Information
Console File
Control Registers
Dynamic Address Translation
Error Checking and Correction (ECC)
Extended Control
Interval Timer
Machine-Check Handling
Microprogram Instruction Retry
OS/DOS Compatibility
Program-Event Recording
Selector Channel 1 (without IFA) or Selector Channel 2 (with IFA)
Storage Protection (Store and Fetch)
System 370 Commercial Instruction Set
Time-of-Day Clock

Note 1: The standard control-storage is 32,768 bytes (32k). The system is equipped with a movable control-storage boundary that allows up to 64k bytes of control storage, depending upon the mix of features installed. These additional control-storage requirements are at the expense of main storage. The storage boundary is set automatically as a function of the initial microprogram load (IMPL) function. For additional information, refer to "Control Storage Requirements."

Note 2: Main storage above 256k bytes is contained in the 3345 Storage and Control Frame Model 1 or 4 (128k additional bytes for 384k bytes total) or the 3345 Storage and Control Frame Model 2 or 5 (256k additional bytes for 512k total). When any of these units is included, it contains the low-order storage addresses. The 3046 Power Unit is required for these models.

Both main and control-storage are equipped with error checking and correction (ECC).

Optional Features

Additional Byte-Multiplexer-Channel Subchannels (Note 3)
Block-Multiplexer Channels (up to four) (Note 4)
Channel-to-Channel Adapter
Clock Comparator and CPU Timer
Conditional Swapping Feature
Direct Control (with external interrupt)
System/370 Floating-Point Instruction Set and Floating-Point (includes extended precision)
Integrated File Adapter (IFA) (Note 5)
Integrated Storage Control (contained in 3345 Models 3, 4, and 5)
Selector Channels 2, 3, and 4 (without IFA) or Selector Channel 3 (with IFA) (Note 5)
Virtual Machine Assist Feature
Word Buffer (Note 6)
1401/1460, 1440 Emulator
1401/1460, 1440 and 1410/7010 Emulator
3210 Console Printer-Keyboard Model 1 (Note 7)
3210 Console Printer-Keyboard Model 2 (Note 7)
3215 Console Printer-Keyboard Model 1 (Note 7)
3345 Storage and Control Frame Model 1 or 4 (for 384k System)(Note 8)
3345 Storage and Control Frame Model 2 or 5 (for 512k System)(Note 8)

Note 3: The byte-multiplexer channel has 16 subchannels that address up to 136 I/O devices (eight shared UCWs can address up to 16 devices each; eight unshared UCWs can address one device each). Up to eight control units can be attached. Configurations with 32, 64, 128, or 256 subchannels are available.

Note 4: A block-multiplexer feature can be ordered in place of all installed selector channels. Block-multiplexer channels cannot replace selector channels 1 and 4 when integrated file adapter is installed.

Note 5: The integrated file adapter and selector channels 1 and 4 are mutually exclusive. Each selector channel addresses up to 256 I/O devices. Up to eight control units can be attached.

Note 6: The word buffer feature is installed on all selector or block-multiplexer channels or none. The word buffer feature is not available for the integrated file adapter feature.

Note 7: The 3210 Model 1 and the 3215 are mutually exclusive; one is required. The 3210 Model 2 can be used as an auxiliary printer-keyboard (except for alter/display functions).

Note 8: A 3046 Power Unit Model 1 is required for all configurations having a 3345 Storage and Control Frame Model 1, 2, 4, or 5.

3145 Models H2, HG2, I2, IH2, J2, JI2, and K2

Standard Features

3145 Processing Unit (with main power frame) (Note 1)
256k to 2048k Bytes of Main Storage
32k Bytes of Control Storage

Model	Main Storage	Control Storage
3145 H2	262,144 (256k)	
3145 HG2	393,216 (284k)	Note 2
3145 I2	524,288 (512k)	
3145 IH2	786,432 (768k)	
3145 J2	1,048,576 (1024k)	
3145 JI2	1,572,864 (1536k)	
3145 K2	2,097,152 (2048k)	

Audible Alarm
Byte-Multiplexer Channel
Byte-Oriented Operand
Channel Indirect Data Address
Channel Retry Information
Console File
Control Registers
Dynamic Address Translation
Error Checking and Correction (ECC)
Extended Control
Interval Timer
Machine-Check Handling
Microprogram Instruction Retry
OS/DOS Compatibility
Program-Event Recording
Selector Channel 1
Storage Protection (Store and Fetch)
System/370 Commercial Instruction Set
Time-of-Day Clock

Note 1: The 3145 Models H2, HG2, I2, IH2, J2, JI2, and K2 require a 3047 Power Unit Model 1.

Note 2: The standard control storage is 32,768 bytes (32k). The system is equipped with a movable control-storage boundary that allows up to 64k bytes of control storage, depending upon the mix of features installed. These additional control-storage requirements are at the expense of main storage. The storage boundary is set automatically as a function of the initial microprogram load (IMPL) function. For additional information, refer to "Control Storage Requirements."

Both main and control storage are equipped with error checking and correction (ECC).

Optional Features

Additional Byte-Multiplexer-Channel Subchannels (Note 3)
Advanced Control Program Support Feature (Note 4)
Block-Multiplexer Channels (up to four) (Note 5)
Channel-to-Channel Adapter
Clock Comparator and CPU Timer
Conditional Swapping Feature
Direct Control (with External Interrupt)
System/370 Floating-Point Instruction set and Floating-Point (includes extended precision)
Integrated Storage Control
Selector Channels 2, 3, and 4
Virtual Machine Assist Feature
Word Buffer (Note 6)
1401/1460, 1440 Emulator
1401/1460, 1440 and 1410/7010 Emulator
3210 Console Printer-Keyboard Model 1 (Note 7)
3210 Console Printer-Keyboard Model 2 (Note 7)
3215 Console Printer-Keyboard Model 1 (Note 7)

Note 3: The byte-multiplexer channel has 16 subchannels that address up to 136 I/O devices (eight shared UCWs can address up to 16 devices each; eight unshared UCWs can address one device each). Up to eight control units can be attached. Configurations with 32, 64, 128, or 256 subchannels are available.

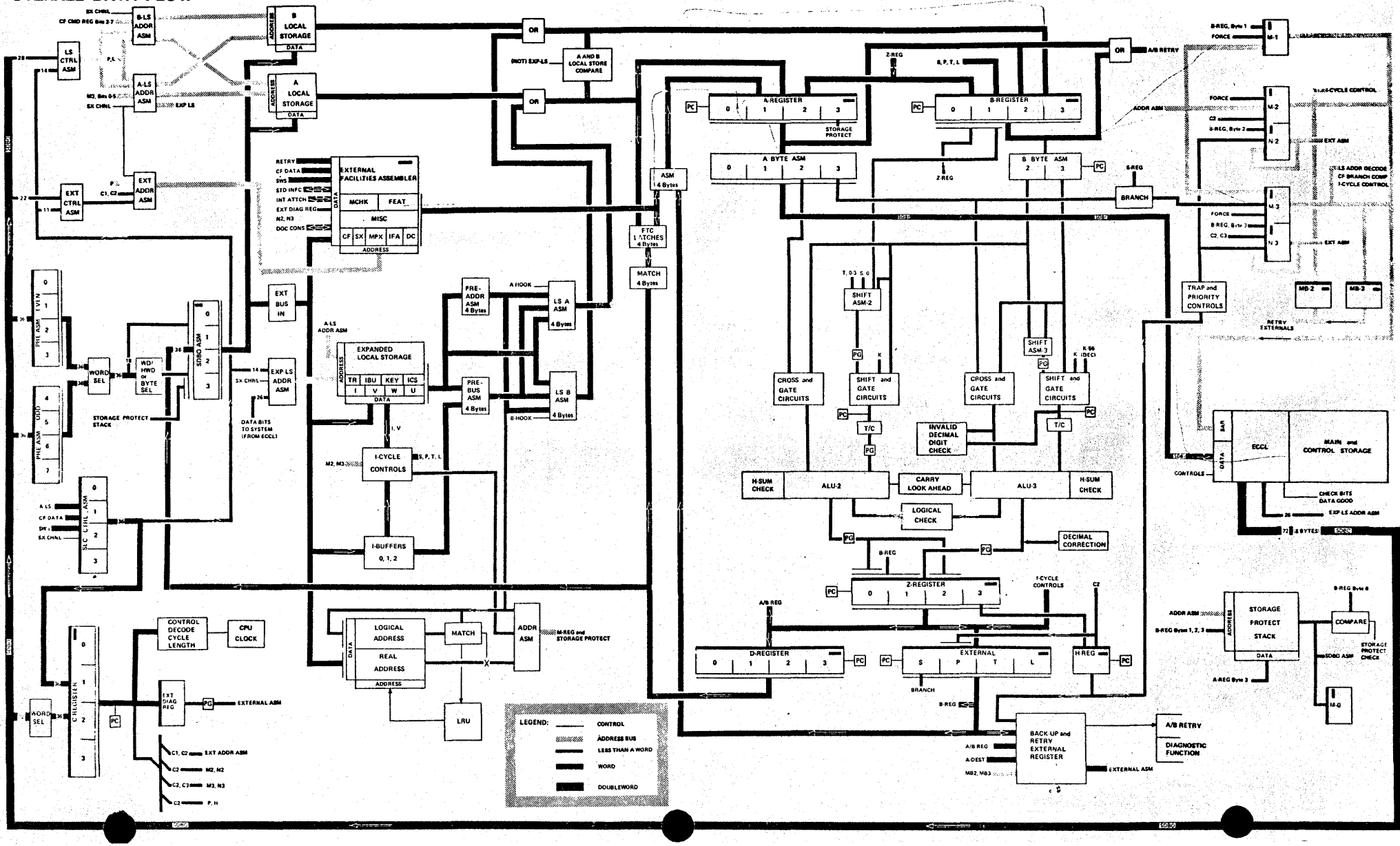
Note 4: Available only on 3145 Models IH2, J2, JI2, and K2.

Note 5: A block-multiplexer feature can be ordered in place of all installed selector channels.

Note 6: The word buffer feature is installed on all selector or block-multiplexer channels or none.

Note 7: The 3210 Model 1 and the 3215 are mutually exclusive; one is required. The 3210 Model 2 can be used as an auxiliary printer-keyboard (except for alter/display functions).

OVERALL DATA FLOW



3145 DATA FLOW

This is a high-level data flow of the 3145 CPU. The general layout is such that you can easily reference from any area of this high-level data flow to the same area of the overall data flow. The facing page illustrates the overall data flow for the 3145 CPU.

The number of bits entering or leaving a function or register is identified either by the wight (thickness) of the line (see the legend block at the lower center of the overall data flow) or by placing the number of bits in the flow line.

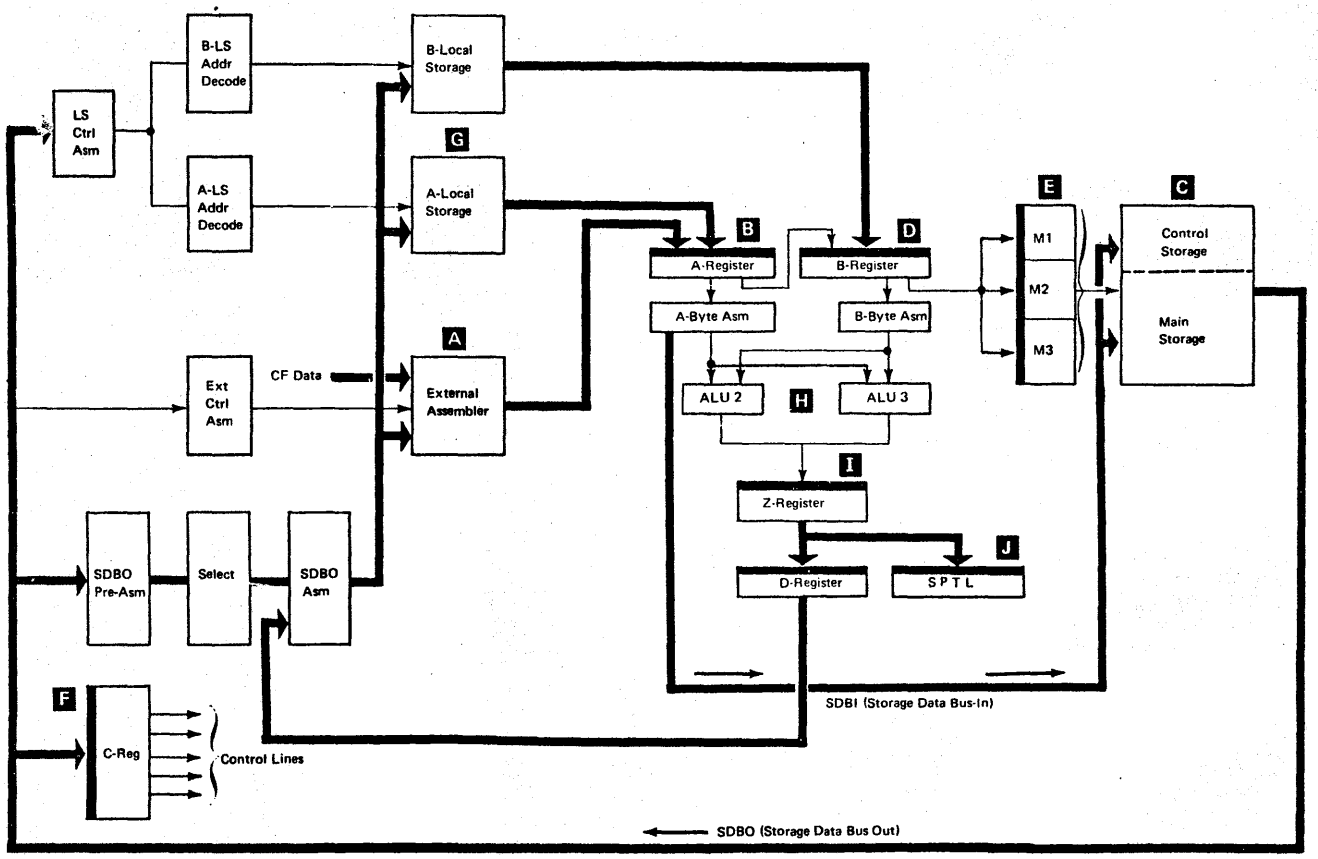
This section gives a brief description of the 3145 functional units illustrated in the high-level data flow. Some basic facts to be introduced for overall understanding of the system follow.

- When power is first applied to the system, an automatic Initial Microprogram Load (IMPL) occurs. Data is read from the console-file disk into the external assembler A. From the assembler the data moves to the A-Register B, to the A-Byte Assembler, out on Storage Data Bus-In SDBI and then the data is loaded into control storage. C
- The 3145 is a word length processor. Each time storage is accessed, a doubleword is read out. Through address bit format, the even or odd address word is gated and used for that specific operation.
- Normal addressing of storage is through the B-Register D to the M-Register. E The M-Register sets up the address of main or control storage to be accessed. Control storage is a reserved area and is unavailable to the user. All addresses are validity checked.
- The microprogram resides in control storage and is read out into the C-Register. F This control word information provides clock cycle length and sets up hardware gating controls for the handling of data. The clock is a variable cycle clock designed to accommodate operations requiring longer cycle times.
- Local Storage (LS) G consists of two monolithic stacks of 64 words each (A-LS and B-LS). Destined data is written into both A-and B-LS so that both stacks contain the same information at any corresponding address. This permits comparison checking of LS data.
- The External facilities are composed of registers, buses, status lines, and other circuitry that form the communication line between the microprogram and:
  - Channels (CHNL)
  - Console-File Adapter (CFA)
  - Console Printer-Keyboard (CPK)
  - Checking Facilities
  - Retry Circuits
  - Integrated File Adapter (IFA)
  - Features.

External facilities have restrictions associated with them because of the manner in which they are used. For example, certain externals cannot be addressed as destinations for data. Others cannot be sources for data.

- The arithmetic and logic unit (ALU) H performs the logic manipulations and adding operations in the CPU. Two ALU units are provided to allow halfword binary and word-move operations in one pass. Each ALU consists of A and B entry gates, logic and arithmetic circuits, and output gating to position the output byte in the Z-Register. I

- The SPTL J, special external registers, are used for direct and indirect addressing, byte selection, and status indications. The SPTL registers are destined in the same machine cycle.



STORAGE

For details about storage, refer to "Storage (STOR)."

Storage for the Model 145 is implemented by monolithic technology. It is based on bipolar, semiconductor storage cells with nondestructive read capabilities. Unlike magnetic core storage, the content of storage is lost when power is turned off.

The main advantages of semiconductor storage are:

- price/performance
- reliability and serviceability (a storage card can easily be replaced).

Storage Structure

All 145 storage (local, control, and main) is implemented using monolithic technology.

Main Storage

The 3145 processor has the following models and storage size options:

Model	Storage Size
FED	112k*
GE	160k
GFD	208k
H or H2	256k
HG when used with a 3345 Model 1 or 4, or HG2	384k
I when used with a 3345 Model 2 or 5, or I2	512k
IH2	768k
J2	1024k
JI2	1536k
K2	2048k

*112k withdrawn from product line.

Data-Transfer Times

Storage data path width is eight bytes. The CPU fetches a doubleword in 540 nanoseconds. It stores one word in 607.5 nanoseconds. (This involves fetching eight bytes, updating four of them, resetting the ECC, and then storing back.)

Control Storage (CS)

The amount of control storage required depends upon features in the system.

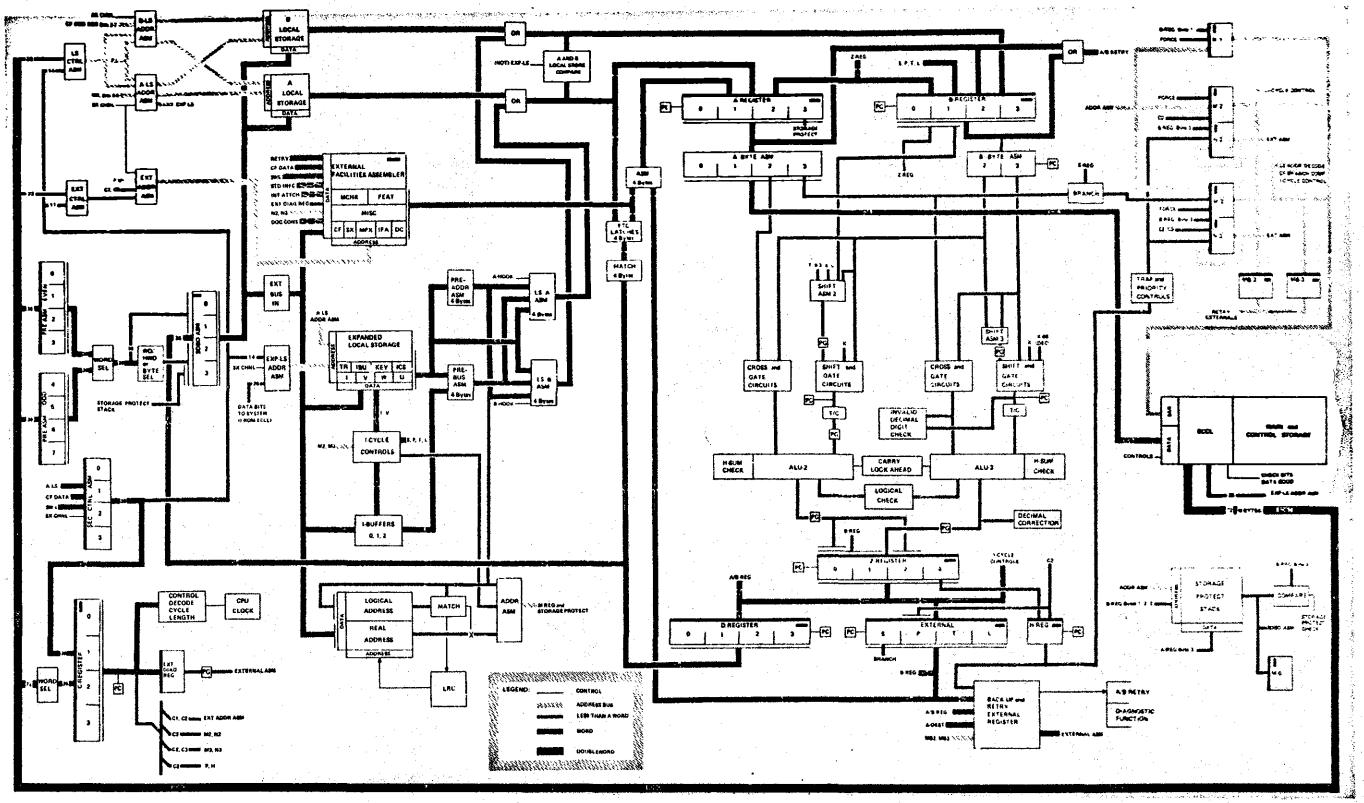
Basic system microcode contains:

Standard instructions
Standard features
Patch area and routine

and requires 24,600 bytes of CS.

Additional storage is required for:

Byte and block MPX UCWs
Block MPX feature
Console support
Integrated file adapter (IFA)
14XX/7010/DOS emulators
Floating-point arithmetic



Direct control

Clock comparator

Advanced control program

Sample requirements

CS is assigned in the high-order range of available storage.

For example: 160k Model 145

	Amount	Address Range
Main Storage	160k	0-160k
Control Storage	32k	160k-192k

Instructions accessing control storage cause address checks; attempting to access a control word from main storage causes a machine check. This is checked by comparing the address with the setting of the address check boundary register (ACB). The ACB register is part of the external facility.

Reloadable Control Storage (RCS)—Advantages

- Amount of control storage needed is minimized by recording a console-file cartridge for each customer according to the features the customer orders.
- Engineering changes to the microprogram can be easily effected.
- One storage system (single addressing design, circuits, data flow) allows greater serviceability.
- Functions implemented through RCS can be easily extended.

Speed

Fetch cycle: 540 ns, 8 bytes.

RCS SIZE

32k bytes is the minimum. RCS may be expanded in increments of 2k bytes up to 64k at the expense of main storage.

The location of the RCS/main storage boundary is established by microcode at IMPL time.

For details of items on this page, refer to "CPU Hardware (CPU)."
For storage protection, refer to "Storage (STOR)."

STORAGE DATA BUS-OUT ASSEMBLER A

- The Storage Data Bus-Out (SDBO) preassembler receives a doubleword of data from main storage.
- The output of the SDBO preassembler is gated to provide word, halfword, or byte selection.
- The SDBO assembler receives input data from the SDBO preassembler, the storage-protect stack and the D-register. It provides an output that is used as data for external bus-in (EBI) and local storage.

C-REGISTER B

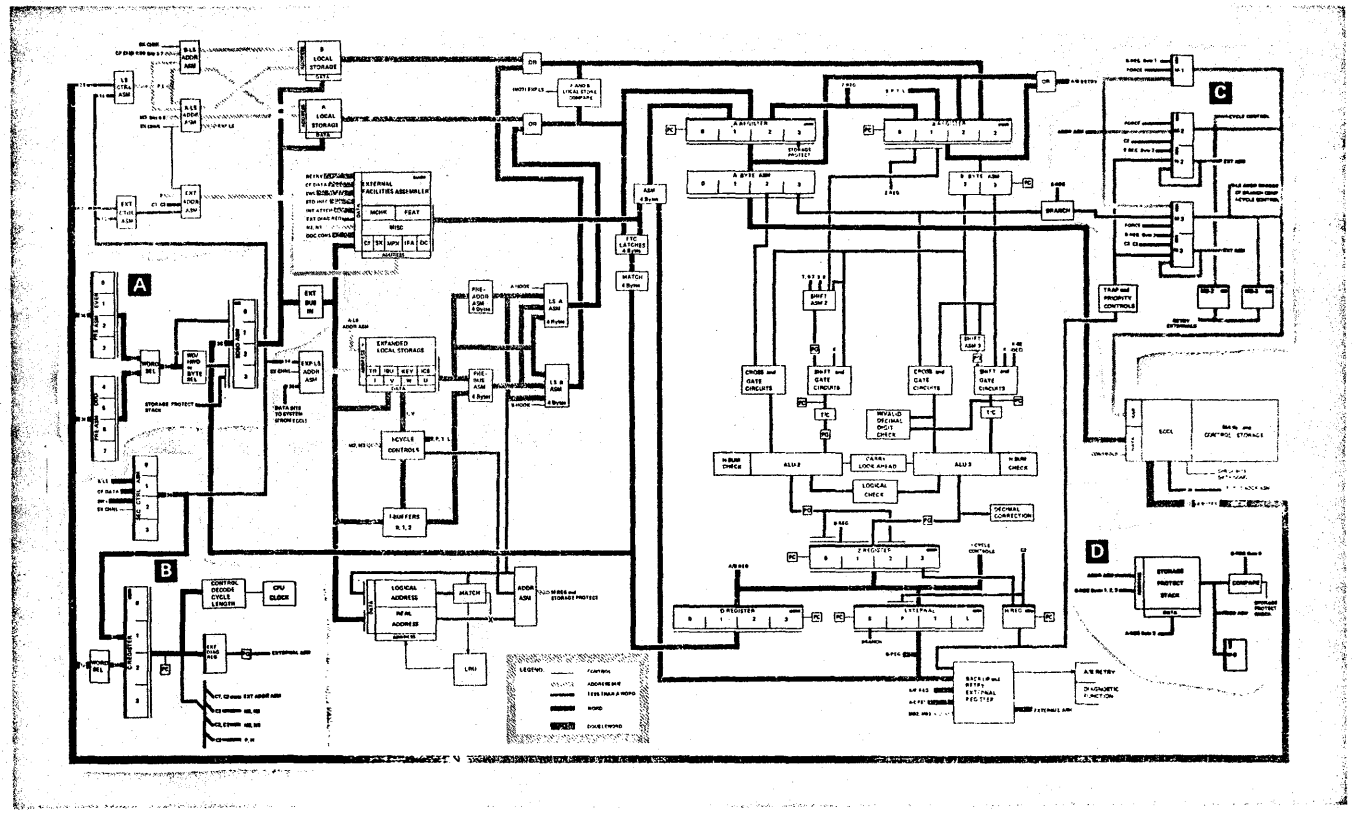
The C-register decodes the control word and provides control and gating of CPU functions. When read out of control storage and gated to the C-register, the control word is decoded to determine:

- Word type
- CPU function
- CPU clock cycle and length

The C-register is set through the secondary control assembler during certain operations; for example, manually setting a control word from the switches on the console.

M-REGISTER C

- Composed of M1, M2, and M3, which provides a 21-bit (plus three parity bits) storage address.
- Addresses main and control storage. Storage is read out on a doubleword boundary and stored on a word boundary.



N-REGISTER C

- Composed of N2 and N3.
- Backup register for control-storage addressing.
- N2 is set with the same information as M2 and is changed only when the control word being executed performs a module-switch function.
- N3 is set with the same information as M3.
- N is not changed when a trap occurs.
- When a trap occurs, the M-register is set to the trap address. The trap routine stores the contents of N (the N-Reg contains the next address that would have been used had the trap not occurred). At the end of the trap routine, M and N are restored to their original value so that the control-word sequence may continue as if there had not been any trap.

MB-REGISTER C

- Composed of MB2, MB3.
- Set with the control-word address in M2, and M3 from M2 BFR and M3 BFR.
- When the CPU clock is stopped, MB contains the address of the last word executed.
- MB output is available to the retry and backup circuits as well as the external assembler.

STORAGE PROTECTION D

The storage-protect unit has a 64 x 8 monolithic storage protection stack that applies to main storage locations zero through 131,072 (in sequential blocks of 2,048 bytes). Additional stacks are provided in the CPU when main-storage capacity extends from 131,072 bytes to 524,288 bytes. Above 524,288 bytes, storage protect is located in the Power Frame (03) and is a mix of 64 x 8 and 64 x 18 monolithic storage cards.

Storage-protect circuits prevent accessing protected areas during either store or fetch operations. To protect specific areas of storage, key bits are first stored in the array of the storage-protect circuit by a write-key operation. During a subsequent store or fetch operation, one of the prestored keys is accessed and compared with the key provided by the user. If the keys match, access to data storage is granted; if not, access is denied.

For details of items on this page, refer to "CPU Hardware (CPU)."

A- AND B-REGISTERS A

The A- and B-registers, each with a fullword capacity, provide the primary data inputs to the ALUs.

The B-register also feeds the M-register during address setup:

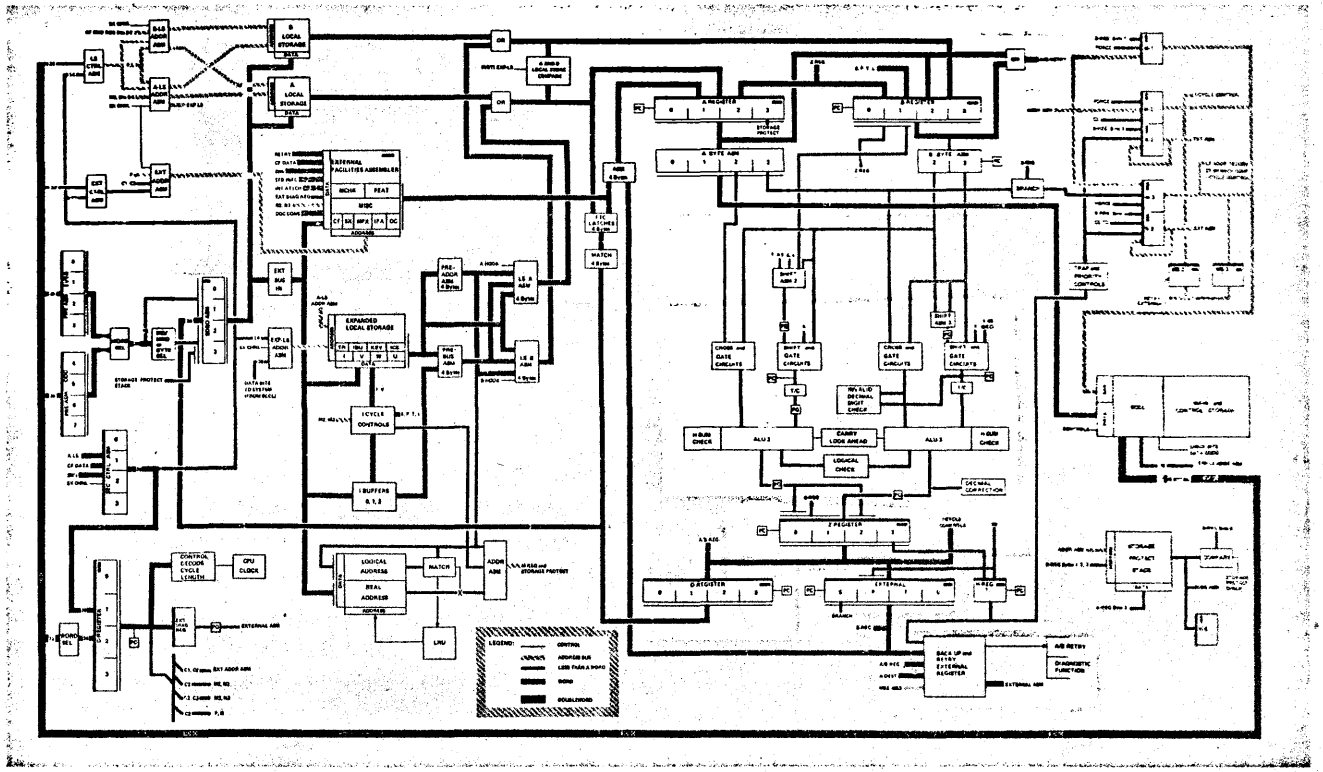
- In the first cycle of a storage word, or
- During a return function in which the return address is taken from local storage or an external facility.

ARITHMETIC AND LOGIC UNITS (ALUs) B

Two one-byte ALUs (ALU2 and ALU3) perform the following operations in one CPU cycle:

- Binary addition, true or complement, of up to two fullword operands. Two halfword additions are performed to achieve the fullword add. Binary halfword addition is achieved by gating the two low-order operand bytes of each halfword into ALU3, and two high-order operand bytes of each halfword into ALU2.
- Logical operation on two 1-byte operands. The operation can be AND, OR, or Exclusive OR.
- One-byte, packed-decimal addition (true or complement).
- Operations and microprogram symbols are:

Symbol	Operation
,A,	AND
,OR,	OR
,OE,	Exclusive OR
+	True ADD
-	Complement ADD
,D+-,	Decimal ADD
+-	Binary ADD
,A-,	Complement AND



Z-REGISTER C

ALU results are set into the four-byte Z-register. The ALU result data can then be routed from Z to:

- The D-register (normal gating)
- The S, P, T, or L-registers
- The A- or B-registers.

Also, the Z-register data (that is, ALU result) is tested, if so specified in the control word being executed, to set/reset S-register bits.

D-REGISTER D

The D-register is used as an interim destination for data to be routed to external facilities or local storage. The data leaves the D-register on the following control-word cycle.

SPTL--SPECIAL EXTERNAL WORD E

- Addressed directly by control-word bits.
- Has special data path to A- and B-register inputs.
- Only external that can be used as a B-source.
- Only facility that is destined in the same cycle (except H-Reg).
- Composed of four one-byte registers: S, P, T, and L.

S-Register: holds the status of arithmetic and logical results; controls some arithmetic functions.

P-Register: base address register for local storage and external addressing.

T-Register: used in conjunction with special branch functions, shifting, storing, and indirect-byte addressing.

L-Register: used in conjunction with P-high bits to form indirect local-storage addresses.

H-REGISTER F

The setting of the latches in the H-register is used to determine the priority of traps. The bits are set during trap-2 cycle and prevent traps of lower priority from occurring.

BACKUP AND RETRY EXTERNAL REGISTER G

To allow recovery from certain kinds of errors, hardware registers (externals) are provided. The backup registers are set to the current cycle setting of the prime register; the retry registers are set to the cycle prior to the one currently being executed.

FLUSH-THROUGH CHECK H

Data routed to local storage, as the result of some control-word operation, other than a storage word read, is gated from the D-register through the SDBO assembler I to local storage. The data that is stored in A-local storage is set into the Flush-Through Check (FTC) latches and is matched to the data routed from the D-register. If the match is unequal, an error condition is set. The same check is made on information routed to the external facilities from the D-register.

For details about items on th[redacted] refer to "CPU Hardware (CPU)."

LOCAL STORAGE A

- A and B local storage are identical monolithic stacks of 64 words each.
- Both stacks contain the same information at the corresponding address. This enables checking to ensure that data being operated on is correct.
- The microprogram uses the local-storage area as a buffer between main storage and the CPU hardware.
- Addresses are formed with combinations of bits from the control word, P-register, L-register, T-register, console-file command register, and forced by the selector channel.
- Access time is 24 nanoseconds.

EXTERNAL FACILITIES B

- External facilities are composed of registers, buses, status lines, and other circuitry that form the communications line between the microprogram and:
  - Channels
  - Console file
  - Documentary console
  - Checking facilities
  - Retry circuits
  - Integrated file adapter
  - Features
- Addresses are formed from control words, console-file data, selector-channel circuits, console switches, retry information, and local-storage address data.
- Data from the externals enters the data flow through the external assembler to the A-Reg.
- Data to the externals is gated through the SDBO assembler on the external bus-in (EBI).

EXPANDED LOCAL STORAGE C

The expanded local storage (EXPLS) registers are hardware registers addressed as though they are local-storage (LS) registers. EXPLS operates similar to the external facilities; however, the output is routed to the A- and B-registers similar to LS.

I-CYCLE CONTROLS D

This hardware improves the CPU performance for System/360 and System/370 instructions by reducing the time during I-Phase of instruction processing.

I-BUFFERS E

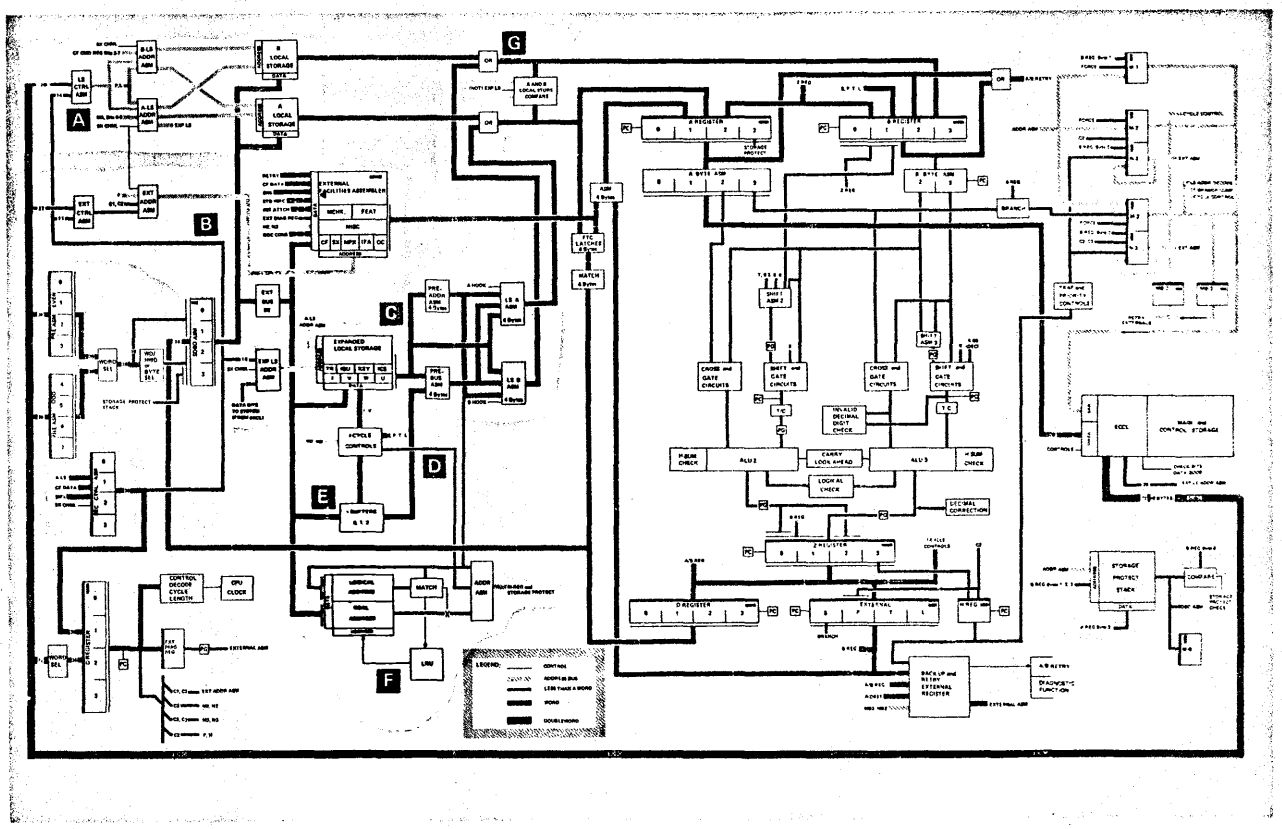
The I-buffers consist of three 1-word registers and are used to hold the present instruction plus, in most cases, the next doubleword of the instruction stream.

ADDRESS ADJUST AND DYNAMIC ADDRESS TRANSLATE F

This logic is used by the OS/DOS compatibility and Dynamic Address Translation features. OS/DOS compatibility feature uses the logic for executing a DOS supervisor and DOS programs under control of the OS supervisor in any main-storage location. Dynamic Address Translation uses the logic to make available by software and hardware up to 16M of virtual storage.

[redacted]ND B-LOCAL STORE COMPARE G

Data stored in local storage is located at the corresponding address in both local-storage stacks. The data is read from both stacks and compared. If the data does not compare, an error condition is set. Note that the output of expanded local storage is routed along the same path but is not compared.



MICROPROGRAMS

For details about microprograms, refer to "Microprogram (MIC)."

- All functions performed by the 3145 are controlled by a microprogram.
- Before any processing may begin, the microprogram must be loaded into the control-storage area. A
- The microprogram is loaded into control storage from a disk that is read by the console file.
- This loading process is called Initial Microprogram Program Load (IMPL).
- The microprogram is composed of microroutines of varying sizes, each having a specific task to perform.
- The microprogram handles the processing of the instructions and data that are read into the main-storage area.
- Channel operations and the operations of the integrated devices are also handled by the microprogram.
- Each microroutine is composed of bit-significant control words that handle particular functions. These functions control execution of the specified task of the microroutine.

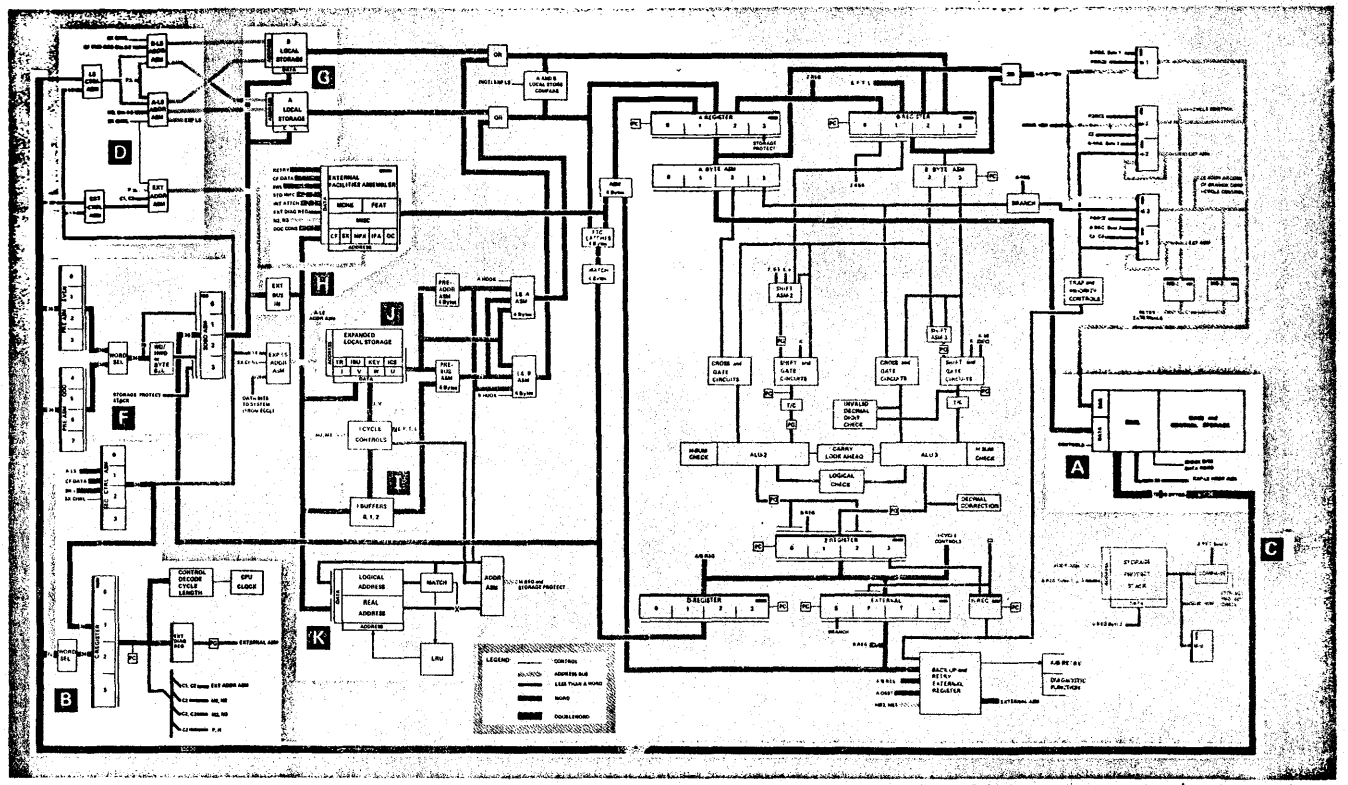
CONTROL WORD READOUT

Before a control word can perform any of its functions, it must be set into the four-byte control register (C-Reg). B The outputs of the C-Reg activate circuitry that causes the execution of specified data-flow functions.

Control words are normally read from control storage and set into the C-Reg. However, control words can be set into the C-Reg directly from the console file, and certain control-word bit combinations may be forced into the C-Reg by circuitry.

Assume that control and main storage have been loaded and that processing has begun:

1. Control words are read out of control storage and gated out on the Storage Data Bus-Out (SDBO). C
2. Portions of the control words are gated from the SDBO to either the local storage control assembler D or the external control assembler D. This is done to set up source addresses for these facilities early in the cycle.
3. The control words are gated into the control register (C-Reg) B, where they are decoded. Decoding the control words brings up control and addressing lines that access and execute the programs located in main storage.



INSTRUCTION/DATA READOUT

When a control word performs a read operation on main storage, either instructions or data is accessed. All read operations, for control or main storage, result in a double-word's being accessed from storage.

Assume that a control word is performing a read operation on main storage;

1. The doubleword from main storage is gated out on the SDBO C to the SDBO pre-assembly latches F.
2. The odd or even address word, of the doubleword, is selected and gated to the SDBO assembler.
3. If the word selected is a data word, it is gated to local storage G or some external facility H.
4. If the word is an instruction, it is gated to the I-buffers I, expanded local storage J, and in some cases to the address adjustment circuits K.

CONTROL WORD FUNCTIONS

For details about control words, refer to "Microprogram (MIC)".

The control words and their high-level functions are:

Branch and Module Switch

Functions:

- Branch
- Module switch
- Destine data to the S, T, or L-registers

Branch Word

Functions:

- Branch
- Module switch (special function)
- Set/reset bits in local storage or external registers

Branch and Link or Return

Branch and Link Functions:

- Store S, P, N2, N3 into a link register
- Set P with a value, or module switch
- Branch

Return Functions:

- Restore S, P, N2, N3
- Reset H-register bits
- Alter the link address in some cases

Word Move

- Move a fullword or selected bytes from one local storage/external location to another.
- Branch

Storage Word

Functions:

- Read data from or store data into—
  Local storage
  Main storage
  Control storage
  External registers
  Storage protect stack
- Branch

Arithmetic Word

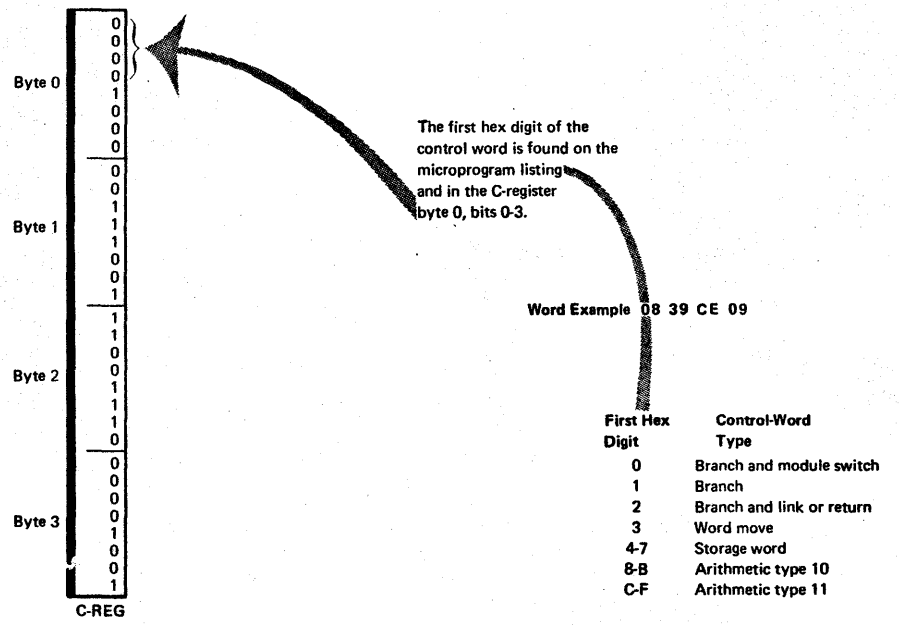
Type 10 Functions:

- Perform a variety of arithmetic and logical operations
- Operate on fullwords for arithmetic or shifting operations

Type 11 Functions:

- Operates on bytes only
- Performs OR, or true ADD only
- Provides A-register input crossing

WORD TYPE – DEFINITION



First Hex Digit	Control-Word Type
0	Branch and module switch
1	Branch
2	Branch and link or return
3	Word move
4-7	Storage word
8-B	Arithmetic type 10
C-F	Arithmetic type 11

ERROR HANDLING

If application-program errors occur (such as illogical action requests), the operating system attempts to handle the exception and provide any necessary operator messages.

If a failure occurs within the CPU or an I/O unit, provisions are made to retry the failing operation. Error-logout facilities to record any such failures are incorporated into the system. This is in addition to any provisions made by the operating system for error retry and error logging.)

Microprogram instruction retry, limited and extended channel logout, storage validation (Error Checking and Correction--ECC) the main and control storage, and other error-detection and error-handling provisions are standard.

MICROPROGRAM INSTRUCTION RETRY

The ability to recover from most intermittent failures is provided by retry techniques. CPU retry is done by microprogram routines that save the source data before it is altered by the operation. When an error is detected, a microprogram routine returns the CPU to the beginning of the operation (or to a point during the operation that was executed correctly), and the operation is repeated. For a detailed description of microprogram instruction retry, refer to the "Recovery Features (REC)."

ERROR CHECKING AND CORRECTION (ECC)

Error checking and correction circuitry for main and control storage automatically corrects single-bit errors. Automatic detection of double-bit errors is also provided. For a detailed description of ECC circuits, refer to "Storage (STOR)."
For a description of handling ECC errors, refer to "Recovery Features (REC)."

CHANNEL RETRY

This feature ensures that most failing channel operations can be retried by error-handling routines. Both a limited and an extended channel logout are implemented. When a channel error or a CPU error associated with a channel operation occurs, the channel status word (CSW) and an extended channel status word (ECSW) are stored in the fixed lower storage area during the I/O interrupt. The ECSW or limited channel log out data provides additional, more exacting status information about the channel failure. This data is formatted by the channel check handler (CCH) routine and passed to a device-dependent error recovery routine to be used in the retry of the failing I/O operation. The ECSW contains information as to:

Which unit detected the error
Which unit caused the error
Successful retries
Channel retries
Validity flags
Retry code--how far has the instruction progressed in execution

COMMAND RETRY

Command retry is a control-unit-initiated procedure between the channel and the control unit. (Not all control units have this capability.) No I/O interruption is required. The number of retries is device-dependent.

COMPATIBILITY of MODEL 145, with OTHER SYSTEM/370 MODELS and SYSTEM/360

Within the storage capacity, internal and input/output channel processing rates, and type of input/output devices that can be attached, compatibility is maintained with other System/370 and System/360 models, with the following exceptions.

1. Programs using machine-dependent data (for example, machine logouts).
2. Programs using the ASCII bit (PSW bit 12).
3. Programs that depend upon features or I/O devices that are not implemented on this system (such as special instructions for ths System/360 Model 44).
4. Programs that depend upon validity of data after the system power has been turned off and restored.

Programs written for other System/370 or System/360 models that contain the following conditions or requirements should be evaluated on an individual basis to ensure proper operation.

1. Time-dependent programs.
2. Programs written to cause deliberate program checks.
3. Programs that depend upon model-dependent features of other System/370 and System/360 models.
4. Programs that use storage locations between address 128 *(decimal)* and 704 after a diagnostic logout into program storage. *However,* such programs may be executed if:
   a. Check-control switch is set to Stop After Log position. In this case, processing stops after the diagnostic logout into program storage takes place.
   b. Program-storage locations that are overlaid by the diagnostic logout are restored with the program requirements followed by an appropriate program restart procedure.

Any attempt to continue processing after a dianostic logout to program storage *without* restoring your program information to the logout area has unpredicatable results.

The 705-byte extent (the permanently assigned program-storage locations) can be reduced to 512 bytes by moving the 192 bytes (between locations 512 and 704) into another program-storage area. The technique used to accomplish this relocation depends upon your application.

CONTROL REGISTERS

The control registers provide for loading and storing control information.

The structure provides for sixteen 32-bit registers for control purposes. These registers are not part of addressable storage.

One or more specific bit positions in control registers are assigned to each function requiring register space. Some of these functions and registers are:

Making the timer interrupt and external interrupt in control register 0.

Masking machine-check subclasses by bits set in control register 14.

Pointing to an extended CPU logout area in control register 15.

For details of the control registers, refer to "CPU Hardware (CPU)."

PROGRAM STATUS WORD CHANGES

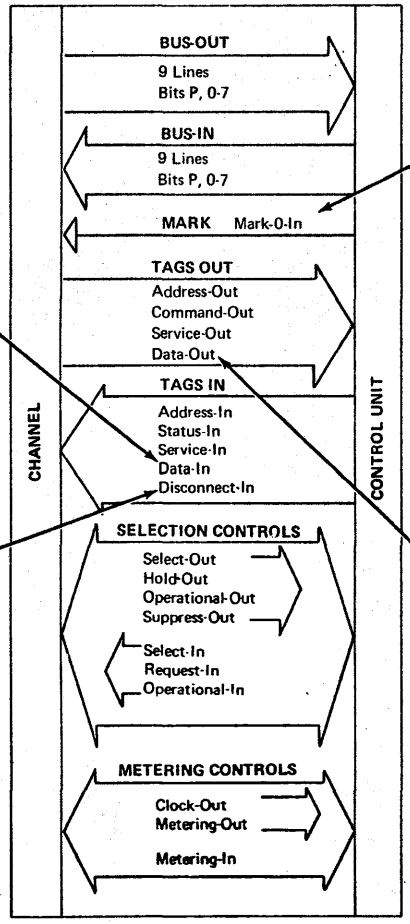
Bit 7 External Mask bit is now a summary bit with control register 1 conataining the individual mask bits.

Bit 12 is now reserved and must be zero. ASCII code is removed.

Bit 6 is the mask bit for channels 6 and over.

Bit 13 Hard Stop bit is now a summary bit. Control register 14 contains mask bits for subclasses of machine checks.

The standard interface for System/370 has all the lines used on the System/360 standard interface, plus several additional lines. The additional lines used by the 3145 are identified on this page. For details of these lines, refer to "Channel (CHNL)."



DATA-IN

During read and sense operations, Data-In rises when data is available on Bus-In. During write and control operations, Data-In indicates that the control unit is ready to receive data.

Data-In indicates to the channel that data on Bus-Out was accepted by the control unit or that the control unit provided the requested data on Bus-In.

Data-In is effective with selector/block-multiplexer channels only and is used along with the Service-In Tag line to increase data rates.

DISCONNECT-IN

Disconnect-In provides control units with the ability to alert the system of malfunctions.

The channel responds to Disconnect-In by performing a selective reset.

MARK-0-IN

Mark-0-In is used only for the Command Retry feature.

When the command being executed encounters a condition requiring retry, the control unit indicates this by raising Mark-0-In.

DATA-OUT

Data-Out is the response to Data-In.

Data-Out indicates to a control unit that data on Bus-In was accepted by the channel or that the channel provided the requested data on Bus-Out.

Data-Out is effective with selector/block-multiplexer channels only and is used along with the Service-In Tag line to increase data rates.

3145 CHANNELS--GENERAL DESCRIPTION

For details, refer to "Channel (CHNL)."

The Model 145 has two types of channels available:

- Byte Multiplexer
- Selector
  The selector channel optionally may have the block-multiplexer feature attached.

Channels on the Model 145 are integrated in the CPU and share CPU cycles for I/O operations.

STANDARD FEATURES

- Byte multiplexer channel
- Selector channel 1 (Channel 2 if the IFA is present)
- Channel retry

OPTIONAL FEATURES

- Selector channels 2-4 (only selector channel 3 if IFA is present).
- Block-multiplexer feature.
- Integrated File Adapter for 2319 DSF. (Displaces Channels 1 and 4.)
- Channel-to-Channel Adapter.

BYTE-MULTIPLEXER CHANNEL

Functionally is equivalent to the System/360 multiplexer channel.

Data transfer is on a byte basis only.

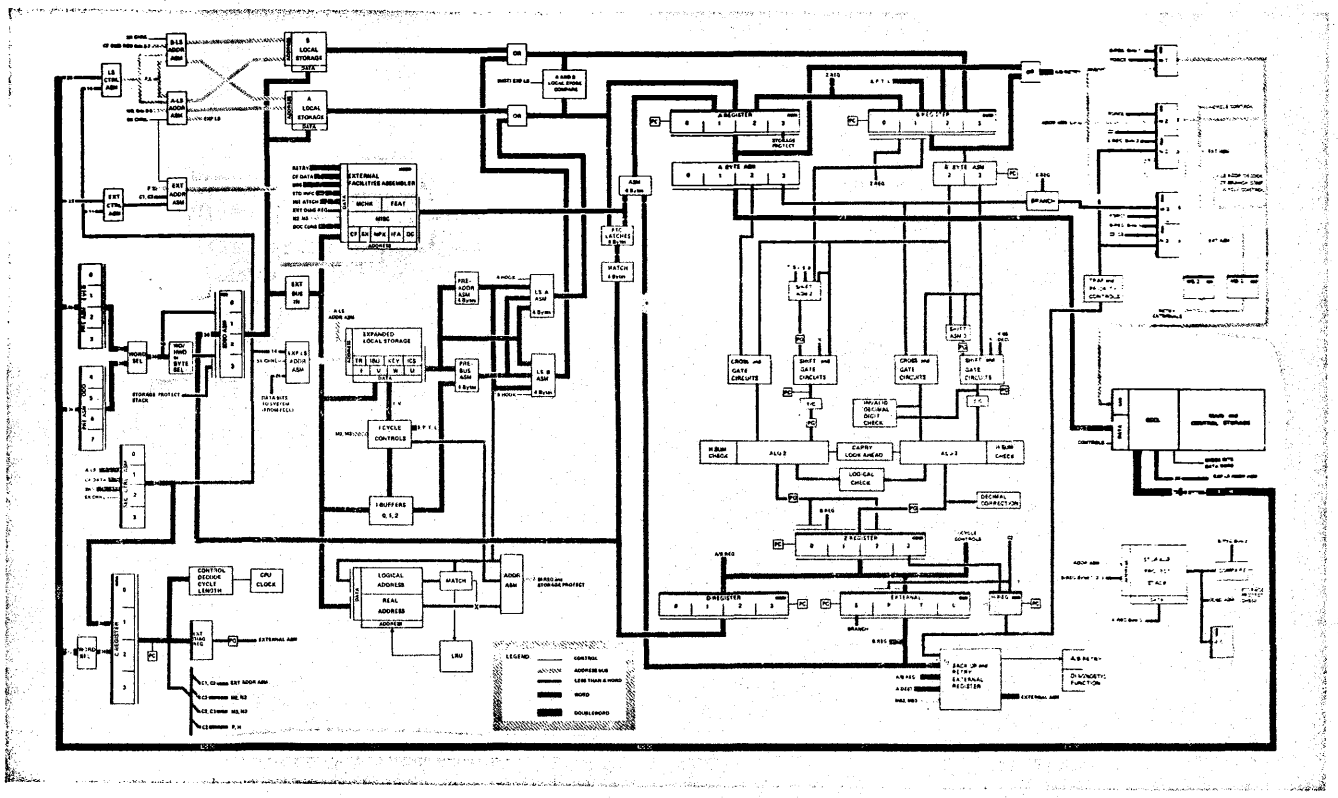
UCWs (Unit Control Words) are provided for subchannels in control storage. Each UCW is contained in four words (16 bytes).

- UCWs provide a place to store channel register data between data transfers, thus allowing multiplexing of data.
- A maximum of 256 subchannels is available on the Model 145.
- 16 UCWs are standard on the Model 145.
  A shared UCW can be shared by up to 16 devices, of which only one can operate at a time.

  A non-shared UCW can be used by one device only.

  Thus, with 16 UCWs, if 8 are shared and 8 non-shared, a total of 136 I/O devices can be attached.

  Up to eight control units can be attached per channel.
- Configurations of 32, 64, 128, or 256 subchannels are available. The number of subchannels must be specified so that the proper amount of control storage may be allocated and written on the console file.



Data Rates

- Aggregate data rate in byte mode is 50 kb. Note that the selector channels and IFA can interfere with the byte multiplexer channel.
- Burst mode data rate is 180 kb.

SELECTOR CHANNELS

The 3145 has one selector channel as standard. Up to three more may be attached as an optional feature. DASD devices without command retry feature should not be attached to channel 4.

- Functionally is equivalent to the System/360 selector channel.
- If the IFA is installed, only channels 2 and 3 can be installed.

Data Transfer

Data is transferred one byte at a time unless the optional word buffer feature is installed on the channel.

- A four-byte buffer is provided for each channel if the feature is installed.
- The word buffer feature allows fewer accesses to main storage to be made while transferring data from the selector channels and increases channel data rates.
- A one-byte operation requires 585 nanoseconds; a one-byte fetch operation requires 517.5 nanoseconds. The word buffer feature allows four bytes to be transferred rather than one.
- The word buffer is required if the 2305 is attached. This buffer is recommended if the 3330 is attached.

Data Rates

- Single Channel
  without word buffer .82 megabytes
  with word buffer 1.85 megabytes
- Aggregate data rate
  without word buffer 1.5 megabytes
  with word buffer 5.0 megabytes

BLOCK-MULTIPLEXER FEATURE

The block-multiplexer feature may be installed on the selector channels as an optional feature.

It is required if the 3330 and 2305 are attached.

The selector channel operates as a block-multiplexer channel when the mode bit in the control register is set on.

A maximum of 512 UCWs is provided when the block-multiplexer feature is installed. These UCWs provide a pool that may be assigned to devices. Each UCW is contained in two words.

- UCWs may be shared or non-shared
- UCWs are contained in control storage
- Shared UCWs must be determined and assigned device addresses.
- Non-shared UCWs are dynamically assigned in blocks of eight to devices at start I/O time. If no UCWs are available, a not-operational-condition code is returned.
- UCWs are provided in control storage in increments of 16 UCWs. Each increment contains two groups of 8 UCWs.

Block Multiplexing

Block multiplexing allows the channel to disconnect a device at channel-end time. During the interval between channel-end and device-end, another device on the channel could be started or could complete data transfer for a previously started operation. Thus, a block-multiplexer channel can multiplex blocks of data from different devices giving a much greater effective data rate than a selector channel.

Block multiplexing occurs only if a control unit presents channel-end and not device-end during command chaining, and the channel is in block MPX mode.

The block-multiplexer channel can operate as a selector channel so that existing System/360 channel programs can run unchanged.

Block-Multiplexer Operation

Because the channel is busy only during the time when data is actually being transferred, several channel programs can be executed concurrently by sharing the channel hardware. This is called "Channel Multiprogramming."

The sequence of events in channel multiprogramming is:

A channel program controlling a device is started by the channel and remains active until the device signals that it has no need for the channel path at that stage of its operation.

The channel disconnects the channel program and stores all information needed to restart the program in UCWs that are in control storage.

The channel can start another channel program at this point if one is ready.

Upon receipt of a signal from the previously disconnected device indicating that it is ready to use the channel data path again, the channel restarts the appropriate channel program.

The process is repeated for all active devices until each one is completely serviced.

If a channel is busy when a device reconnection is requested, the device must wait until the channel becomes available.

To facilitate channel scheduling, a new *channel available interrupt* has been defined for the block-multiplexer channel.

At disconnect time for a channel program, the channel is available for the resumption of an uncompleted channel program or the initiation of a new one. A channel available interrupt occurs at disconnect time if any I/O command was issued previously while the channel was busy.

INTEGRATED FILE ADAPTER

- The Integrated File Adapter (IFA) feature connects three to eight 2314-type disk drives to the System/370 Model 145.
- The IFA feature is assigned exclusive use of the channel-1 address and functions as both channel and control unit for the files.
- Data transfer takes place one byte at a time on a share-cycle basis the same selector channels.
- The initiation of operations and each step of the file sequence requires the CPU controls and microprogramming.

The primary control for the IFA is contained in the CPU, where it can make use of the CPU hardware and microprogram for operation. The 2319-A1 contains the read clocking circuits, the write oscillator, and the storage module switching for up to eight files. The disk storage drives operate the same as the 2314 system connected to a selector channel. The record format is identical, and the operation requires the same programming systems.

The IFA control-unit operation is initiated as a channel operation using the I/O instructions and channel commands. Primary control information for the file operation is stored in the CPU. Operating commands are processed by micro-programs stored in the CPU. The microprogram starts the operation by developing the appropriate information for a portion of the sequence and issues a mini-op to the control-unit hardware. While the hardware is performing the mini-op, the microprogram stores a link address and returns to CPU operation. When the hardware finishes that portion of the sequence, it requests a trap to return to the microprogram link address to continue the operation. To complete a command, an operation may require several of these transfers between the microprogram and hardware.

Data movement during the hardware control period is performed by requesting a selector-channel share cycle for each byte. The CPU or other channel operations can use the CPU hardware and microprogram for other operations when time is not required by the IFA controls either for setup or data transfer. The file operation should never overrun during normal operation because of the assigned priorities.

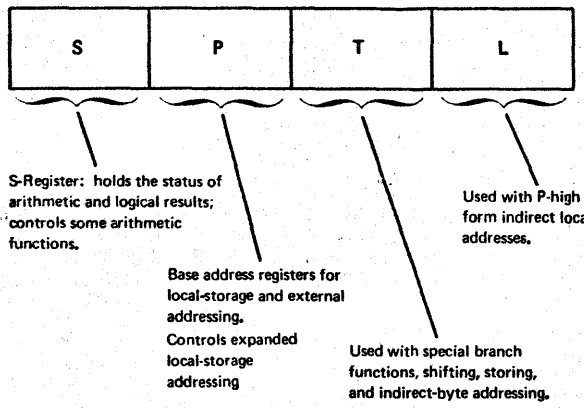
For details on the IFA, refer to "Integrated File Adapter (IFA)."

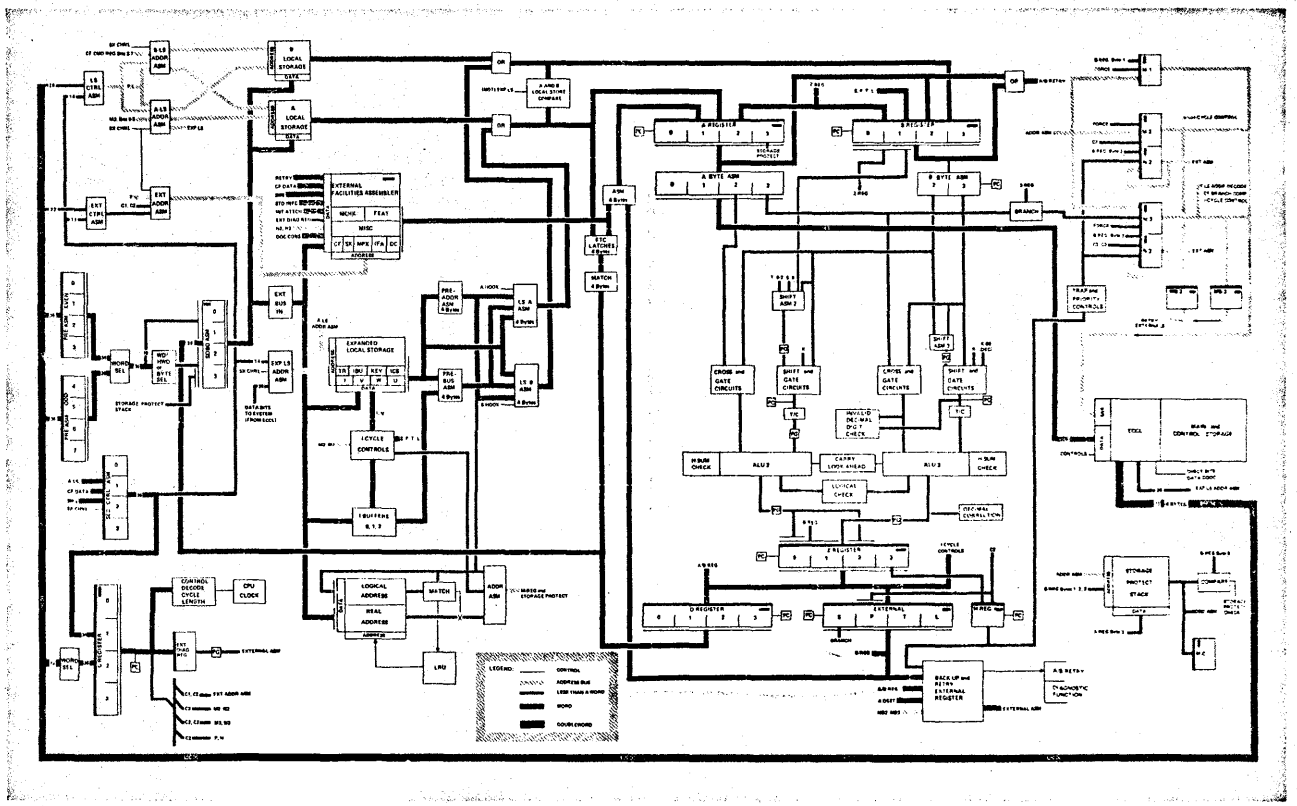
CPU HARDWARE

CONTENTS

The SPTL word is a special external register that has a word address of 04.

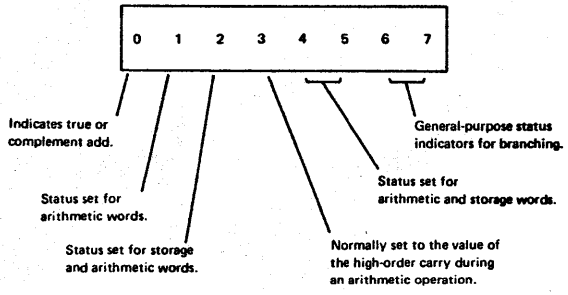
- Addressed directly by control-word bits.
- Has special data path to A- and B-register inputs.
- Only external that can be used as a B-source.
- Only facility (other than the H-Reg) that is destined in the same cycle.
- Composed of four byte registers: S, P, T, and L.
- Set by I-cycle controls during I-cycles.





Note: For details concerning how SPTL affects control-word operations and addressing, see the Microprogram (MIC) section.

S-REGISTER



S-register bits are used primarily to indicate the results of ALU operations or to specify how certain ALU operations are to be performed.

Branch fields in the control words can be set to specify branch testing of S-register bits. The results of the branch testing are used to determine a portion of the next control-word address. Therefore, a control-word sequence can be modified according to ALU results. For example, the following is frequently used in microprogram routines.

1. A control word specifying an ALU operation calls out set/reset of specific S-register bits, depending upon the ALU result.
2. A subsequent control word specifies branching on the same S-register bits.

Note: If the S-register is set with a control word and this same control word is branching on S-bits, the branch test is made on a previous S-register setting and not on the result of the current control-word operation.

3. The control word branched-to continues the microprogram sequence required by the ALU result.

The S-register can also be used as a general-purpose data register. For example, a control word can cause the S-register to be loaded with a byte for use in operations with subsequent control-word operations such that none of the following descriptions apply. Such use of the S-register is determined by the microprogrammer. The following listed S-register bit functions are not automatically performed. Any function must be explicitly specified in the control word for which the function is desired.

The data from the S-register is used in the M-register for branching.

Note: The duplicate S-registers are designated S-register A and B, respectively. The data from S-register B is displayable, and is gated to the A-register and B-register.

S0

The setting of S0 determines whether a true or complement add is to be performed in the ALU(s) when either a binary or decimal add is specified by the control word.

S0 Value	*Specifies*
0	True add
1	Complement add

The true/complement circuitry affects only the B-input (from the B-register or the K-assembler) to the ALU(s). The A-register input is always presented to the ALU(s) in true form, regardless of the value of S0.

In arithmetic word (type 10) shift operations, the value of S0 is shifted into the four high bits of the result word (shifted right) when the shift field of the arithmetic word specifies (SR4, S0). The bits (4-7) shifted out of the source-word byte 3 are set into T-register bits 0 through 3, respectively.

S1

1. In decimal operations, S1 is set to 1 if an invalid decimal digit is detected in the A or B inputs to the ALU. S1 is not changed if the decimal digits are all valid. A decimal digit greater than 1001 (binary) is invalid. The test on the inputs is made before the original decimal data is binarily added in ALU3; no such test is made on decimal data when it is being sent to ALU3 on a ±6 correction cycle.

   In order for S1 to function in this manner, the arithmetic control word (type 10) must specify both decimal addition (C, D+-, C), and the S12 status set.
2. In binary operations, S1 is set to the value of the carry-out of:
   a. Bit 1 in single-type ALU operations,
   b. Bit 1 of byte 0 in fullword ALU operations.

The control word must specify the S12 status set, along with the appropriate binary ALU operation, in order for S1 to function in either of these two ways.

S2

1. In byte operations, S2 is set to 1 if the Z-bus (ALU result byte 3) is not zero ($Z\overline{0}$). If the Z-bus is zero S2 is not changed from its prior setting. The arithmetic control word calling for the byte operation must specify the S12 status set in order for the S2 bit to function in this manner.
2. In fullword binary operations, S2 is set to 1 if the entire 32-bit result is nonzero. S2 is not changed if the 32-bit result is zero. A status set of S12 must be specified in the arithmetic control word (calling for the fullword operation) in order for S2 to function in this manner.
3. In fullword binary operation, S2 is set to 1 if:
   a. A Z24 status set is specified, and
   b. The low-order 24 bits (ALU result bytes 1, 2, and 3) are nonzero. If the low-order 24 bits are all zero, S2 is not changed.
4. When an S2 status set is specified in a storage word, S2 is set to 1 if the count field is not zero after the count is decremented; S2 is set to 0 if the count field is zero after decrementing. The decrement-count function is specified in the storage word to effect the following actions:
   a. The 24-bit address, in bytes 1, 2 and 3 of the even word (of an even/odd pair of local-storage words), is updated.
   b. The count field (low-order 16 bits of the odd word of the pair) is decremented.
   c. S2 is set according to the low-order 16-bit result of step b.

Note that the count value is updated by circuitry and that S2 is set/reset if the S2 status set is specified *even if* decrement count is not specified. In this case, however, the updated count value is not stored back into the count location.

S3

S3 is set to the value of the carry-out of bit-0 of the ALU operation. This function is used in both byte and word operations. The arithmetic operation field must contain a bit configuration designated by a statment that contains a C at the left, in order for S3 to function in this manner. For example, in the statement:

```
S0 = 0
C + 0
```

the C specifies that S3 is to be set/reset.

In fullword arithmetic operations, S3 is set to the carry-out of:

a. Bit 0 of byte 0 if an S12 or no status set is specified.
b. Bit 0 of byte 1 if a Z24 status set is specified.

S3 is not set/reset in storage-word address and count updates.

S4 and S5

In arithmetic words (types 10 and 11) that specify a status set of S45, S4 and S5 are set/reset according to the bit values of the ALU result byte as follows:

Bit	Value	Indicates Result Byte Bits
S4	0	0-3 not equal to 0000
S4	1	0-3 = 0000
S5	0	4-7 not equal to 0000
S5	1	4-7 = 0000

If a Z6 status set is specified:

Bit	Value	Indicates Result Byte Bits
S4	0	0-5 not equal to 000000
S4	1	0-5 = 000000
S5	0	4-7 not equal to 0000
S5	1	4-7 = 0000

The S45 and Z6 status sets can be specified in an arithmetic word, only if a single-byte ALU operation is called for; S45 and Z6 do not pertain to fullword arithmetic operations.

In storage-word operations, S4 and S5 functions are the same as in the arithmetic words and are specified in the same manner (S45 and Z6). S4 and S5 are set according to the value of the low-order byte of the count field after the count has been decremented. The count is in bytes 2 and 3 of the odd word of an even-odd pair of local-storage words. The address is in bytes 1, 2, and 3 of the even-word location.

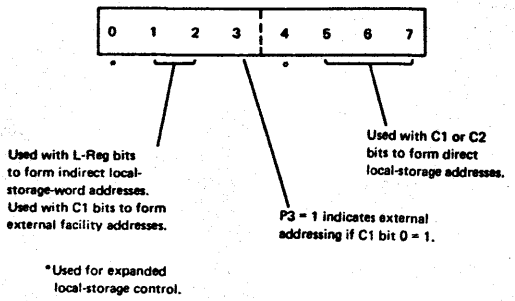
In an arithmetic word (type 10) that specifies an AB CK byte operation: S4 is set to 1 if a parity-check error is detected on the A input to the ALU(s).

Note: If an I24 status set is specified in a type 10 arithmetic word, no S-register bit setting occurs, regardless of any other specified status set. For example, if the operation specified is:

```
S0 = 0
C + 0
```

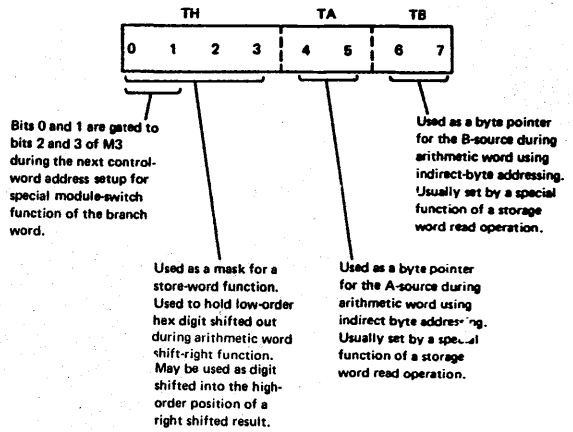
and I24 is also specified, then S3 and S0 are not changed even though the operation field calls for such set/reset functions.

P-REGISTER



The primary function of the P-register is to provide a base address during local-storage or external-register addressing. That is, the P-register is used to point to groups of external registers or areas of local storage; the remainder of the external/local-storage address is specified by the outputs of the C-register. In some cases, the L- and/or T-register contents are used to determine portions of the address.

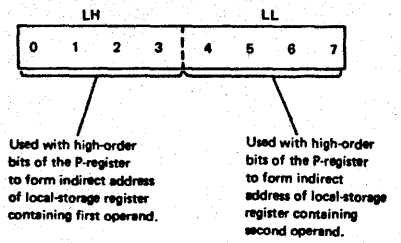
T REGISTER



The T-register is used in a variety of ways:

1. Bits 4 and 5 and/or 6 and 7 are used in indirect-byte addressing and branching operations.
2. Bits 0 and 1 are used to form a portion of the next-control-word address when a module-switching operation is specified in the branch word.
3. Bits 0 through 3 are used in arithmetic fullword shift operations.
4. In certain storage-word read operations, bits 4 and 5 or 6 and 7 are set to the value of the two low-order storage-address bits before the address is updated.
5. In certain storage-word store operations, bits 0 through 3 are used to specify which bytes of a source are to be stored and what constant, if any, is to be used to update the storage address.
6. May be used as a working register.

L-REGISTER



The primary purpose of the L-register is to hold the addresses of the general or floating-point registers, all of which are in local storage. The address of a general register can be specified by L0 through L3 or L4 through L7. For example, L0-3=0111 can specify general register 7. Note, however, that while this address corresponds to the hexadecimal address of general register 7, that register's address in local storage is determined by C, P, and L-register bits when the L-register is used in the addressing.

The address of a floating-point register is usually specified by L0-3 only (not L4-7).

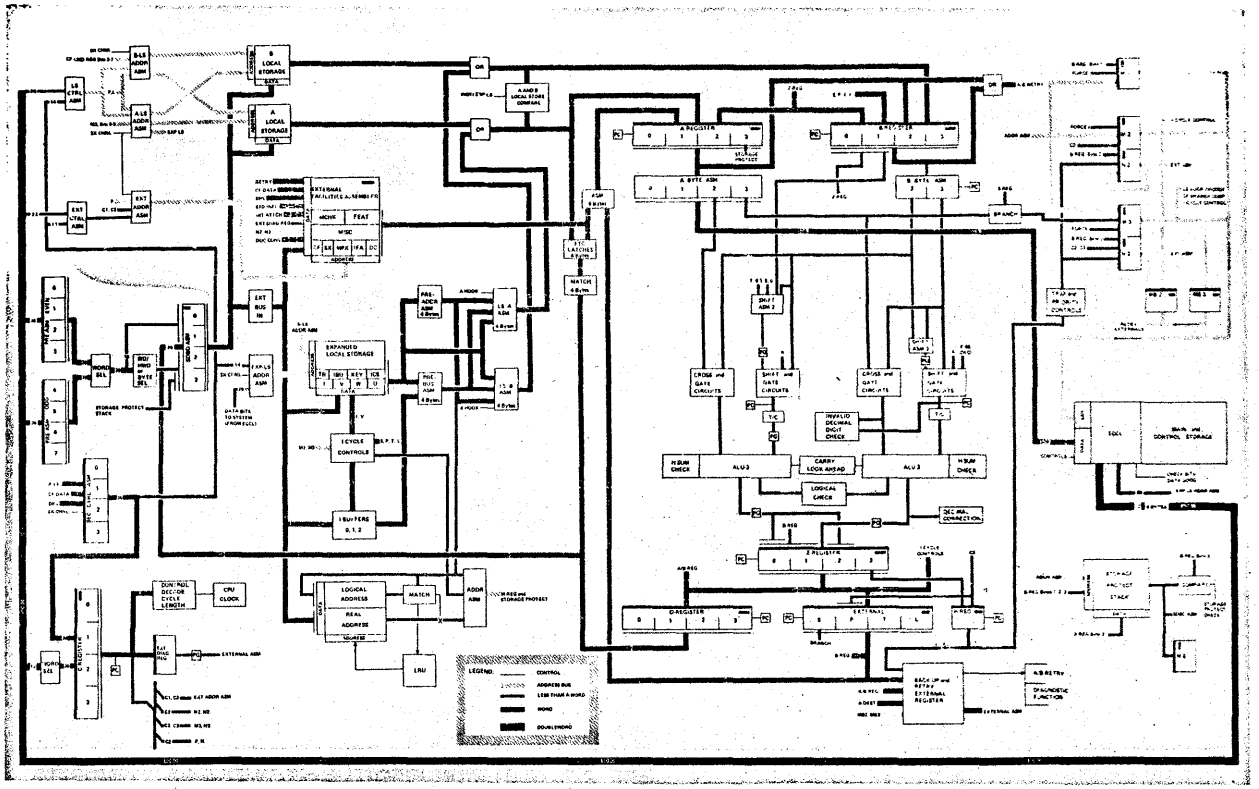
The L-register may be used as a working register.

SDBO PRE–ASM, ASM

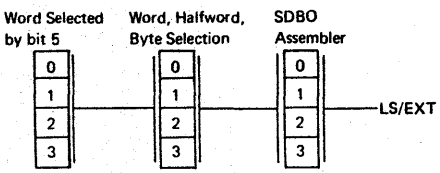
- The Storage Data Bus-Out (SDBO) preassembler receives a doubleword of data from internal or external storage.
- The output of the SDBO preassembler is gated by M3 bits 5, 6, and 7 to provide word, halfword, or byte selection.
- The SBDO assembler receives inputs from the SDBO preassembler, the storage-protect stack, and the D-register. The assembler provides an output that is used as data for External Bus-In (EBI) and local storage.

The selection of data fed to the SDBO assembler is accomplished by decoding M3 bits 5, 6, and 7. The decode of the M3 bits causes corresponding gating lines to be activated, which cause data from the SDBO preassembler to be routed to the SDBO assembler.

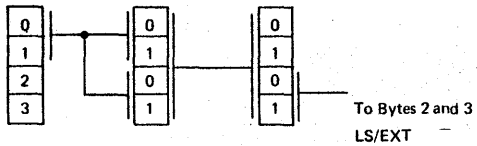
M3 Reg		Selects
bit 5	0	Even word bytes 0-3
	1	Odd word bytes 4-7
bit 6	0	Even halfword, bytes 0, 1
	1	Odd halfword, bytes 2, 3
bit 7	0	Even bytes 0, 2
	1	Odd bytes 1, 3



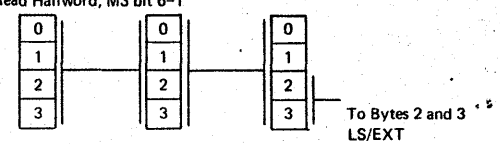
Read Word



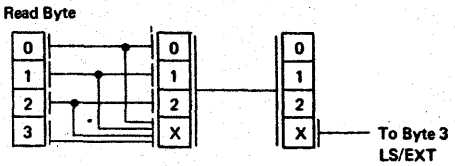
Read Halfword, M3 bit 6=0



Read Halfword, M3 bit 6=1

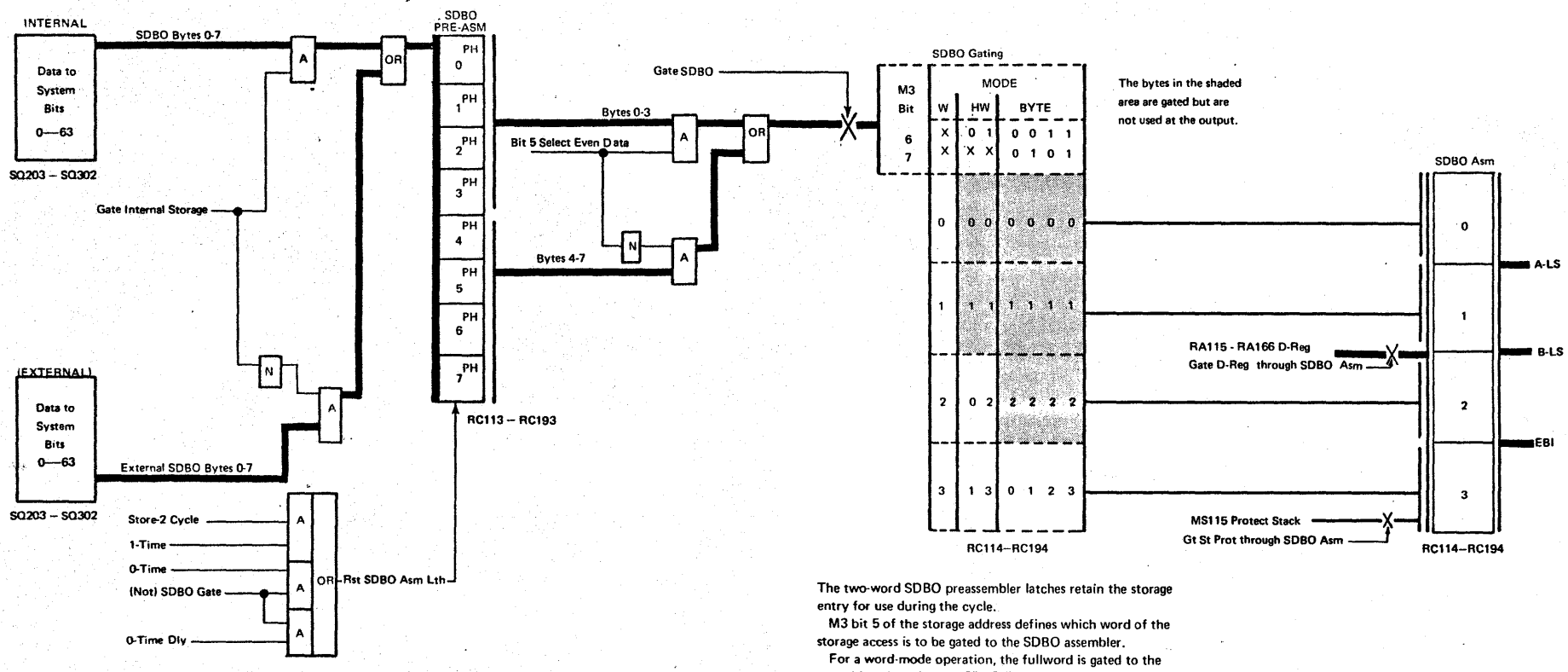


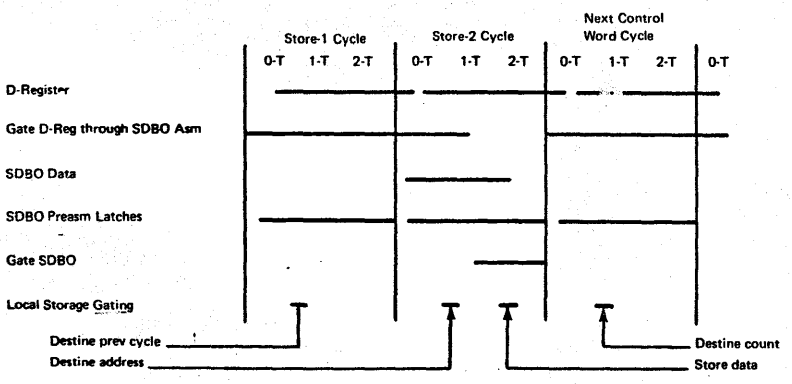
Read Byte



M3 bit 6,7=00, Byte 0 to X
M3 bit 6,7=01, Byte 1 to X
M3 bit 6,7=10, Byte 2 to X
M3 bit 6,7=11, Byte 3 to X

SDBO PRE-ASSEMBLER UNIT DATA FLOW





The two-word SDBO preassembler latches retain the storage entry for use during the cycle.

M3 bit 5 of the storage address defines which word of the storage access is to be gated to the SDBO assembler.

For a word-mode operation, the fullword is gated to the assembler when the gate SDBO line is raised.

For a halfword-mode operation, the upper or lower halfword, depending on M3 bit 6, is gated to bytes 2 and 3. Input bytes 0 and 1 continue to gate to output bytes 0 and 1.

For byte-mode operation, the byte defined by M3 bits 6 and 7 is gated to byte 3. The remaining input bytes are gated to their respective output positions.

The protect stack readout is gated to the SDBO assembler byte 3 during the ISK instruction to allow transfer to a GP-register.

The four-byte output of the D-register is gated to the respective outputs of the SDBO assembler when neither the gate SDBO line nor the gate st prot line are active.

LOCAL STORAGE

- Local Storage (LS) consists of two monolithic stacks of 64 words each (A-LS and B-LS).
- Destined data is written into both A- and B-LS so that both stacks contain the same information at any corresponding address. This permits comparison checking of LS data.
- LS is used by the microprogram as a high-speed buffer. Access time is 24 nanoseconds.
- Readout is nondestructive.
- Address range within each stack is 00 to 3F (Hex).
- Addressing is accomplished with combinations of control-word bits, P-register bits, L-register bits, T-register bits, selector-channel share-cycle forced bits, and console-file command-register bits.

LS has assigned locations for specified functions. Refer to "Local Storage Map (370 Microprogram in Control Storage)." Locations included are:

16 general registers
4 floating-point registers
Selector-channel work area
CPU work area

These locations are valid when the 370 microprogram is located in control storage. When diagnostics are running, another set of LS assignments is in effect.

LS is external to main and control storage. Each 64-word stack is located on two MST cards:

	Bytes	Card Location
A-LS	0 and 1	A-B4P2
	2 and 3	A-B4M2
B-LS	0 and 1	A-C4B2
	2 and 3	A-C4C2

Note: Do not remove or replace LS array cards with power on.

LOCAL STORAGE OPERATION

Read

- Either or both A-LS and B-LS can be accessed in one cycle.
- Data from A-LS is gated to the A-register.
- Data from B-LS is gated to the B-register.
- A-LS and B-LS sources can be different addresses.

Write/Read

- Data destined to LS always is written into both A- and B-LS.
- Data destined during any cycle is written during the next cycle.
- A write LS is always followed by a read LS. The read LS data is used for flush-through checking and A- and B-LS comparing.

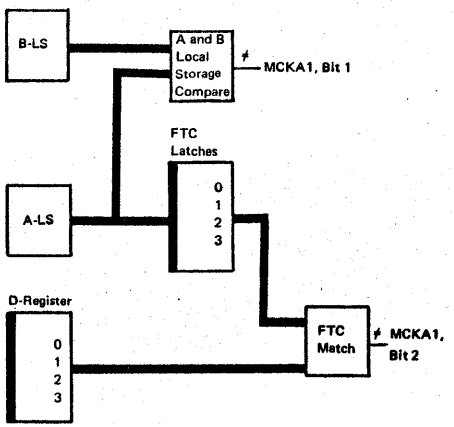
DATA CHECKING

Flush-Through Check (FTC)

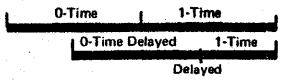
Data destined to local storage as a result of some control-word operation, other than a storage word read, is gated from the D-register through the SDBO assembler to local storage. The data in the D-register is compared with the data from the A-LS address that was the destination. If the compare is not equal, bit 2 of MCKA 1 sets to indicate an FTC error.

A and B Local Storage Compare

Data destined to local storage is stored at corresponding addresses in both local-storage stacks. The data is then read from these addresses and compared. If the compare is not equal, bit 1 of MCKA 1 sets to indicate the error.

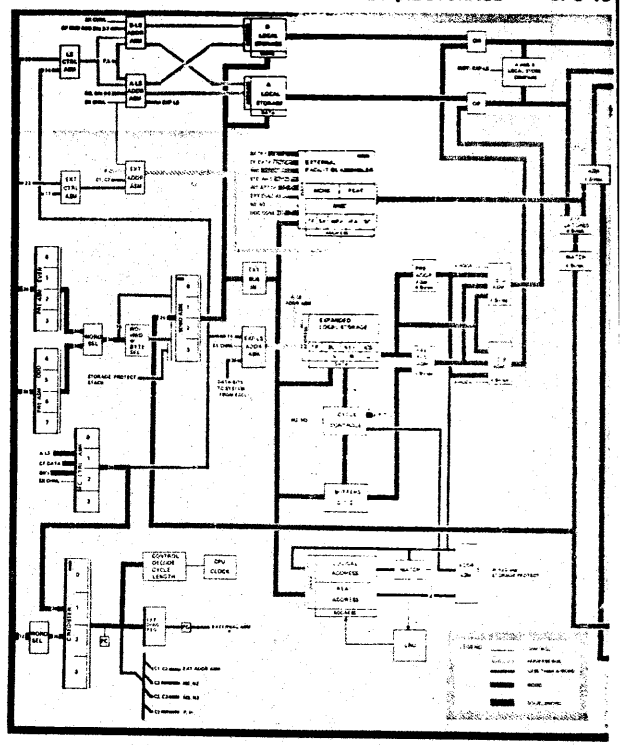


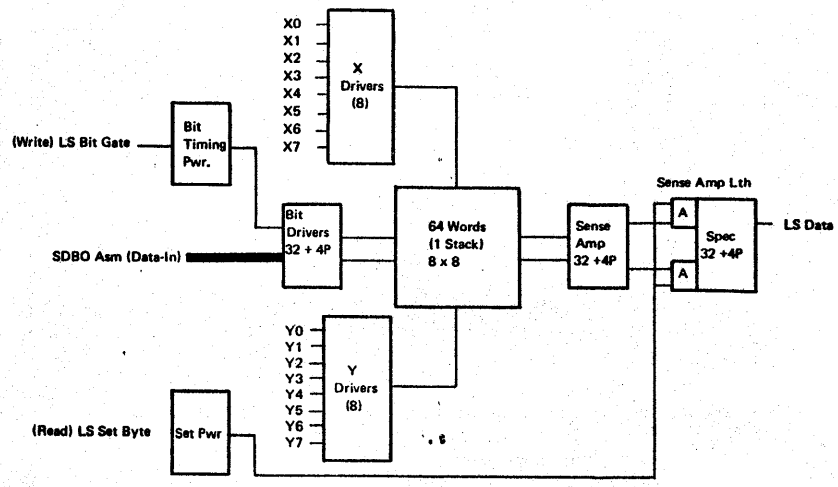
LOCAL STORAGE TIMING



Read	
Write/Read	
FTC	
A/B-LS Compare	

Write/Read, FTC, and A/B-LS compare occur during the cycle following the control-word cycle that the data was destined.



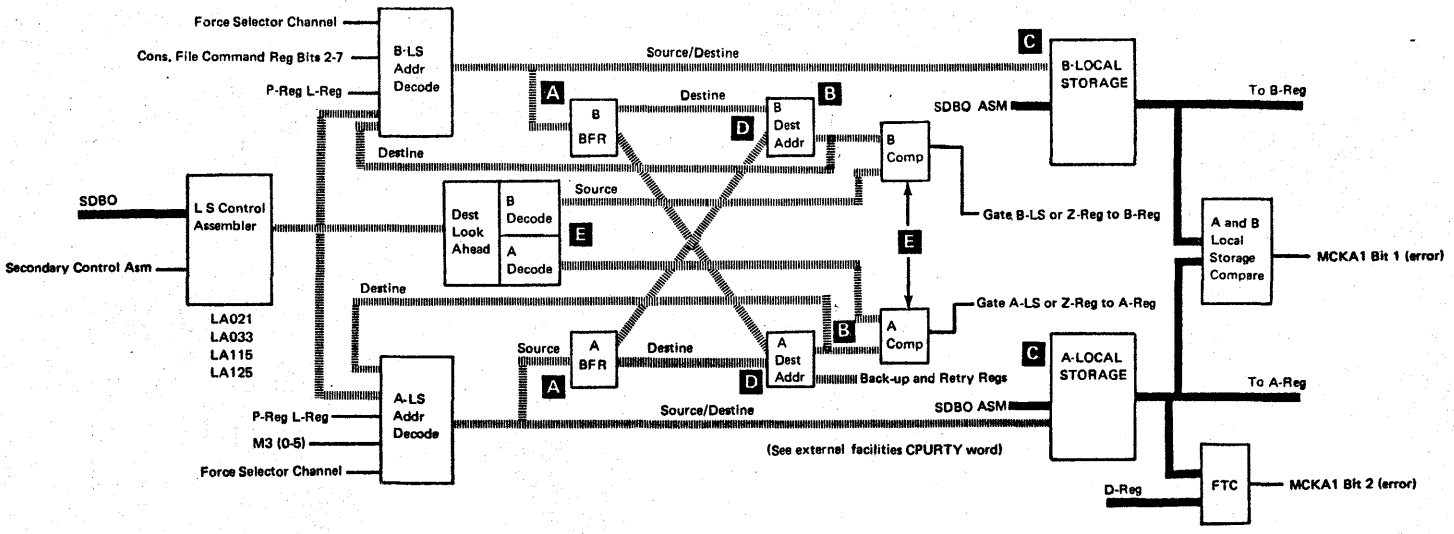


LOCAL STORAGE DESTINATION ADDRESSING

The type/form of the control word selects the source address (A or B) that is used for the destination address.

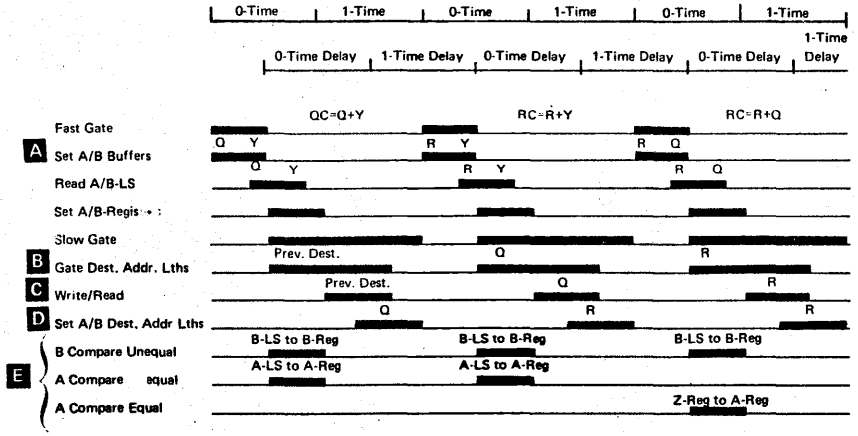
1. During fast gate, decoded source addresses are stored in the A and B buffers. A
2. At the beginning of slow gate, the previous control-word destination address is gated from the A and B destination address latches to the address decoders. B
3. The data destined during the previous cycle is written. C
4. The buffer (A or B) selected by the word type/form is gated to both the A and B destination address latches. D
5. At slow gate of the next cycle, this address is gated to the address decoders for destination write/read. B

LOCAL STORAGE ADDRESSING



Destination Look-Ahead E

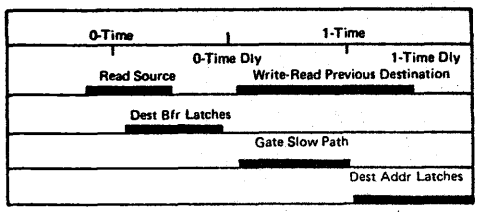
A new source address may be the same local-storage address as the previous destination address. When this condition occurs, the destined data in the Z-register is not stored into local storage in time to be accessed by the following control word as source data. Destination look ahead detects this condition by comparing A and B new source addresses with the previous destination address. An unequal compare (previous destination *not* new source) gates the new source data from local storage to the A- or B-register. An equal compare (previous destination *is* new source) gates the new source data directly from the Z-register to the A- or B-register. This compare is done on a byte basis.

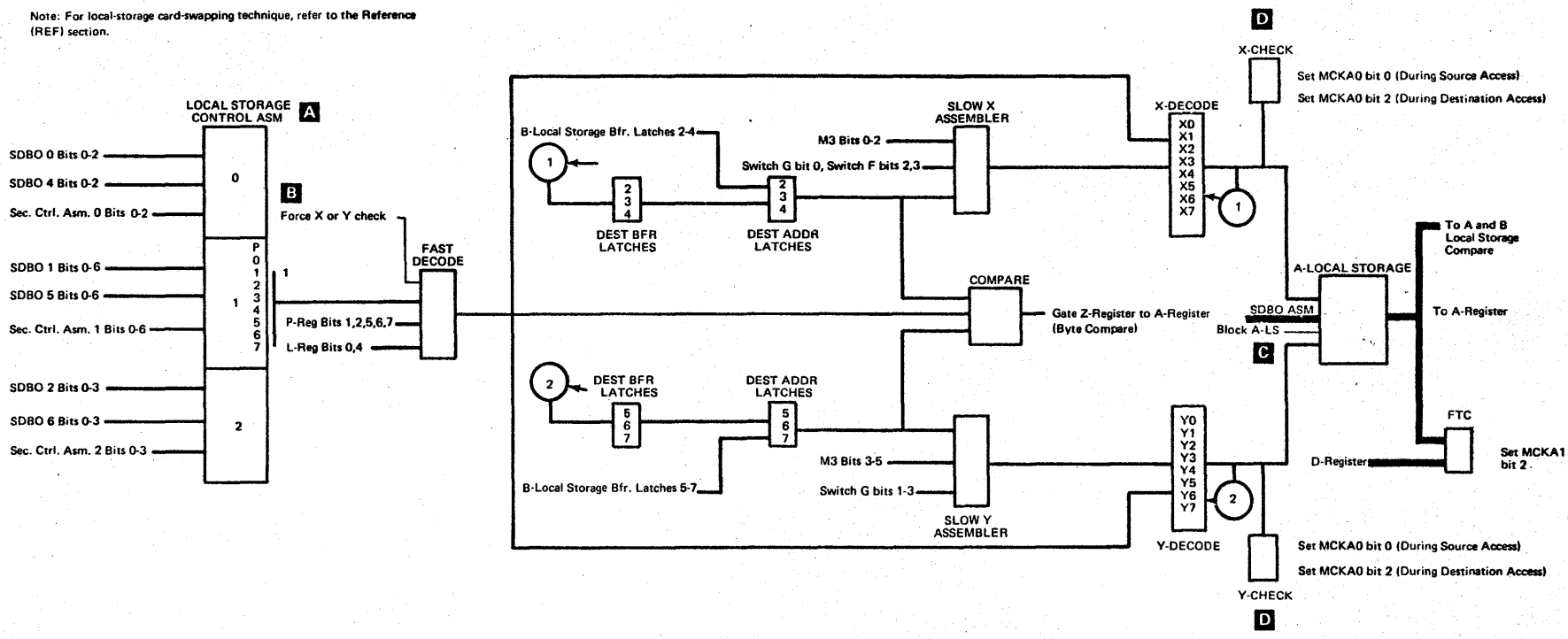


A-LOCAL STORAGE UNIT DATA FLOW

CIRCUIT	CARD	ALD
FAST DECODE	C4F2,C4N2,C4M2	LA011-LA018,LA021,LA111-LA117
DEST BFR LATCHES 2,3,4	C4G4	LA212
DEST BFR LATCHES 5,6,7	C4J4	LA222
DEST ADDR LATCHES 2,3,4	C4G4	LA212
DEST ADDR LATCHES 5,6,7	C4J4	LA222
SLOW X ASSEMBLER	C4G4	LA211
SLOW Y ASSEMBLER	C4J4	LA221
COMPARE	C4N2,C4L2	LA022-LA024, LA031-LA032
X DECODE	C4F2	LA015-LA016
Y DECODE	C4M2	LA114
X CHECK	C4G4	LA212
Y CHECK	C4J4	LA222
MONO BUFFER	B4P2,B4M2	LA311,LA327

Note: For local-storage card-swapping technique, refer to the **Reference** (REF) section.





A **LOCAL-STORAGE CONTROL ASSEMBLER**

A control word read from control storage (on the SDBO), or assembled in the secondary control assembler, is gated to the local-storage control assembler. In the local-storage control assembler, the word type is identified to specify the type of addressing needed.

B **DIAGNOSTIC FUNCTION:**

With DIAG0 bit 5 on, and RTY backup Asm byte 0 bit 5 off, an extra X-line is forced up to cause an X-check.

With DIAG0 bit 5 on, and RTY backup Asm byte 0 bit 6 off, an extra Y-line is forced up to cause a Y-check.

C **DIAGNOSTIC FUNCTION:**

With DIAG0 bit 4 or 5 on, and RTY backup Asm byte 3 bit 6 off, the storing of data into A local storage is blocked. This causes both a flush-through check and an A- and B-local store compare error.

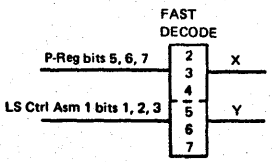
D **X- and Y-CHECKS**

The X- and Y-checks are tests to determine whether if at least one, but not more than one, X- and Y-line is active at a time.

A–LOCAL STORAGE ADDRESS ASSEMBLY

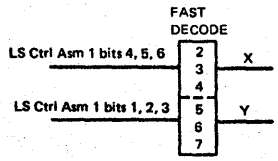
A-SOURCE DIRECT ADDRESS

When the A-source is called for by its symbolic name, or the actual address, is used, bit 0 of byte 1 of the control word is 0 to flag direct addressing. Bits 1, 2, and 3 of byte 1 of the control word form the Y-line, and bits 5, 6, and 7 of the P-Reg form the X-line.



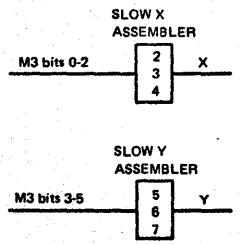
A-SOURCE DIRECT ADDRESS
(X- and Y- Lines carried in control word)

Two control words have the capability of carrying the X- and Y-lines in the bit structure of the control word. The Branch and Link or Return word carries the X- and Y-lines of the link register in bits 1-6 of byte 1. The word-move word carries the X- and Y-lines of the source or destination in byte 1.



LCSC MODE

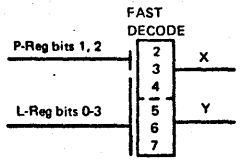
A special diagnostic function called LSCS (Local-Storage Control Storage) mode causes control words located in local storage to be read out and executed. The X- and Y-lines of the control word to be executed are formed from bits 0-5 of the M3-register, which is set up by the last word executed. Any of the other ways of addressing local storage may be used for data accesses during the execution of the control words read from local storage. For additional information about LSCS mode, refer to the Diagnostic Functions (DIAG) section under the heading "Basic Tests."



INDIRECT ADDRESSING

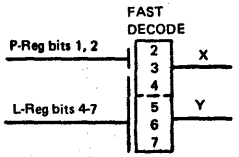
USING L HIGH (LH)

Two symbols in the microprogram language call for indirect word addressing. The symbol LH calls for local-storage addressing. The high bits of the L-register and bits 1 and 2 of the P-register form the X- and Y-lines.



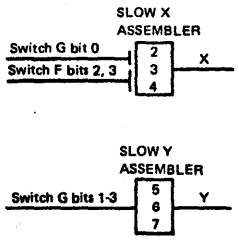
USING L LOW (LL)

The symbol LL calls for local-storage addressing. The low bits of the L-register and bits 1 and 2 of the P-register form the X- and Y-lines.



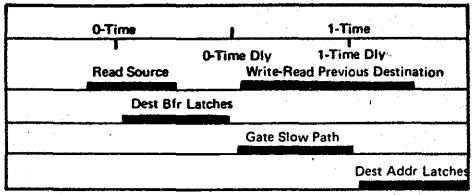
ADDRESSING FROM CONSOLE

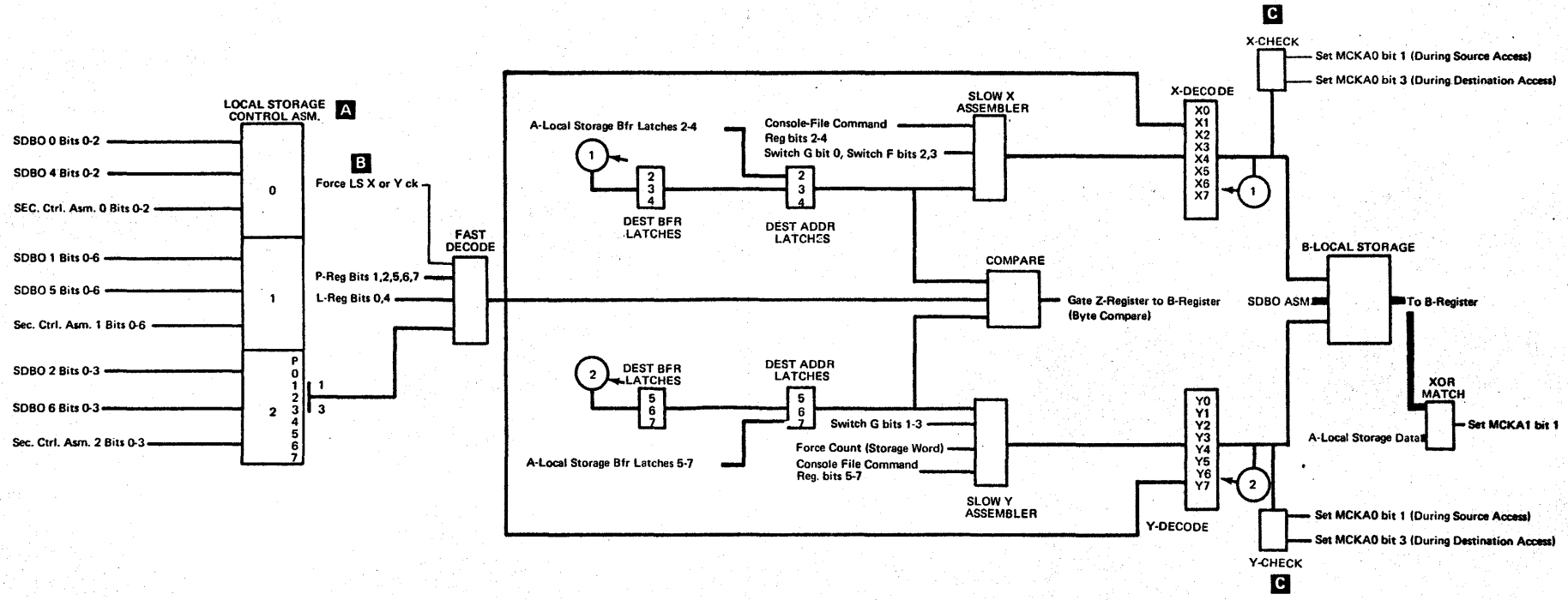
Local storage is also accessible from the operator's console. The X-line is formed from bits 2 and 3 of switch F and bit 0 of switch G. The Y-line is formed from bits 1-3 of switch G.



B-LOCAL STORAGE UNIT DATA FLOW

CIRCUIT	CARD	ALD
FAST DECODE	C4F2,C4N2,C4H2	LA011-LA018,LA021,LA121-LA127
DEST BFR LATCHES2,3,4	C4G2	LA232
DEST BFR LATCHES 5,6,7	C4J2	LA242
DEST ADDR LATCHES 2,3,4	C4G2	LA232
DEST ADDR LATCHES 5.6.7	C4J2	LA242
SLOW X ASSEMBLER	C4G2	LA231
SLOW Y ASSEMBLER	C4J2	LA241
COMPARE	C4N2,C4L2	LA022-LA024,LA031-LA032
X-DECODE	C4F2	LA017-LA018
Y-DECODE	C4H2	LA124
X-CHECK	C4G2	LA232
Y-CHECK	C4J2	LA242





A LOCAL-STORAGE CONTROL ASSEMBLER

A control word read from control storage on the SDBO or assembled in the secondary control assembler is gated to the local-storage control assembler. In the local-storage control assembler, the fields that indicate the source accessing are tested to determine the type of address formation needed to address local storage.

B DIAGNOSTIC FUNCTION

With DIAG0 bit 5 on, and RTY backup Asm byte 0 bit 5 off, an extra X-line is forced up to cause an X-check.
With DIAG0 bit 5 on, and RTY backup Asm byte 0 bit 6 off, an extra Y-line is forced up to cause a Y-check.

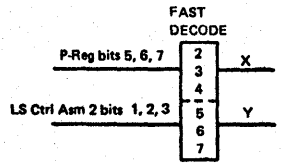
C X- and Y- CHECKS

The X- and Y-checks are tests to determine whether at least one but not more than one X- and Y-line is active at a time.

B-LOCAL STORAGE ADDRESS ASSEMBLY

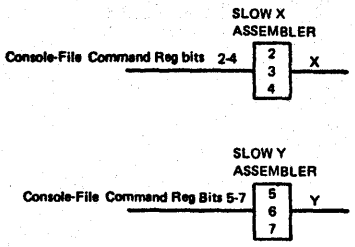
B—SOURCE DIRECT ADDRESS

When the B-source is called for by its symbolic name, or the actual address is used, bit 0 of byte 2 of the control word is 0 to flag direct addressing. Bits 1, 2, and 3 of byte 2 of the control word form the Y-line, and bits 5, 6, and 7 of the P-reg form the X-line.



ADDRESSING FROM CONSOLE FILE

Some console-file commands cause the control word currently in the C-register to be executed. The X- and Y-lines that would normally be formed by control word bits are formed from bits 2-7 of the console-file command register.

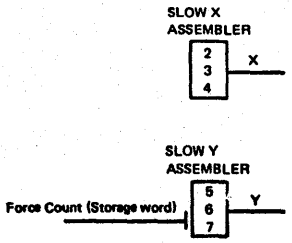


FORCE COUNT ADDRESS

When a storage word with the decrement count function is executed, an odd local storage address must be forced. The Y-line in effect, for addressing the source register, is assumed to be an even Y-line. The count to be accessed is in the next higher address; therefore, the next higher Y-line is forced.

For example: A storage word is addressing the GD register for selector-channel 2 (see "Local Storage Map (370 Microprogram in Control Storage"). The X Y-lines for GD are X4, Y0. To access the count, Y1 is forced, and the GC register is addressed.

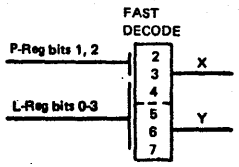
The X-line remains the same for the count access.



INDIRECT ADDRESSING

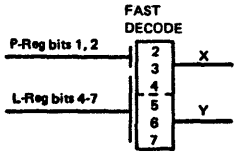
USING L HIGH (LH)

Two symbols in the microprogram language call for indirect word addressing. The symbol LH calls for local-storage addressing. The high bits of the L-register and bits 1 and 2 of the P-register form the X- and Y-lines.



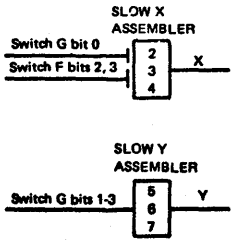
USING L LOW (LL)

The symbol LL calls for local-storage addressing. The low bits of the L-register and bits 1 and 2 of the P-register form the X- and Y-lines.



ADDRESSING FROM CONSOLE (B - LS)

Local storage is also accessible from the operator's console. The X-line is formed from bits 2 and 3 of switch F and bit 0 of the switch G. The Y-line is formed from bits 1-3 of switch G.



LOCAL STORAGE MAP (370 MICROPROGRAM IN CONTROL STORAGE)

Word Name	LS Location	Byte 0	Byte 1	Byte 2	Byte 3	X and Y Line	P low Direct Access	P high Indirect Access
	00		General Register 0			X0 Y0		
	01		General Register 1			X0 Y1		
	02		General Register 2			X0 Y2		
	03		General Register 3			X0 Y3	0	
	04		General Register 4			X0 Y4		
	05		General Register 5			X0 Y5		
	06		General Register 6			X0 Y6		
	07		General Register 7			X0 Y7		
	08		General Register 8			X1 Y0		0
	09		General Register 9			X1 Y1		
	0A		General Register A			X1 Y2		
	0B		General Register B			X1 Y3	1	
	0C		General Register C			X1 Y4		
	0D		General Register D			X1 Y5		
	0E		General Register E			X1 Y6		
	0F		General Register F			X1 Y7		
AX	10		SRTN Temp Link			X2 Y0		
DI	11		Alter/Display Log Link			X2 Y1		
RTX	12		Retry Link			X2 Y2		
DTX	13		Translate Link			X2 Y3	2	
X	14		Working			X2 Y4		
R	15		Working			X2 Y5		
Y	16		Working			X2 Y6		
Q	17		Working			X2 Y7		
MA	18		Working			X3 Y0		2
MBS	19		Working			X3 Y1		
MX	1A		Working			X3 Y2		
MC	1B		Working			X3 Y3	3	
MD	1C		Working			X3 Y4		
MF	1D		Working			X3 Y5		
MW	1E		Working			X3 Y6		
CX	1F	CPU	Link	Register		X3 Y7		

MPX Channel

	Word Name	IF A Name	LS Location	Byte 0	Byte 1	Byte 2	Byte 3	X and Y Line	P low Direct Access	P high Indirect Access
SX 2	GD		20					X4 Y0		
	GC		21			Count		X4 Y1		
	GM		22		Protect CCW Address			X4 Y2		
	GW		23					X4 Y3	4	
SX 3	GD		24					X4 Y4		
	GC		25			Count		X4 Y5		
	GM		26		Protect CCW Address			X4 Y6		
	GW		27					X4 Y7		4
SX 1	GD	FD	28					X5 Y0		
	GC	FC	29			Count		X5 Y1		
	GM	FM	2A		Protect CCW Address			X5 Y2		
	GW	FW	2B					X5 Y3	5	
SX 4	GD	FA	2C					X5 Y4		
	GC	FB	2D			Count		X5 Y5		
	GM	FS	2E		Protect CCW Address			X5 Y6		
	GW	FL	2F					X5 Y7		
			30		Floating-Point Register 0			X6 Y0		
			31		Floating-Point Register 0			X6 Y1		
			32		Floating-Point Register 2			X6 Y2		
			33		Floating-Point Register 2			X6 Y3	6	
			34		Floating-Point Register 4			X6 Y4		
			35		Floating-Point Register 4			X6 Y5		
			36		Floating-Point Register 6			X6 Y6		
			37		Floating-Point Register 6			X6 Y7		6
	SO		38					X7 Y0		
	PM		39	PE Control	PE Code Group After Mask			X7 Y1		
	DM		3A		Adjustment Factor			X7 Y2		
	RW		3B		Address Adjustment Working			X7 Y3	7	
	DP		3C		IF A Low-Priority Link			X7 Y4		
	LNK		3D		I-Cycle Link			X7 Y5		
	P4X		3E		SX-4 Link Register			X7 Y6		
	P3X		3F		SX-1, 2, 3 Link Register			X7 Y7		

Note: Words 28 through 2F are shown with selector channel designations.

Current PSW

Displayed in:

BC

System Mask 0–7	KEY 8–11	0MWP 12–15	Interrupt Code 16–31	ILC 32–33	CC 34–35	Program Mask 36–39	Instruction Address 40–63
External 10	EXP LS 50 Byte 0	EXP LS 53 Byte 1		EXP LS 53 Byte 0 Bit 0, 1	Byte 0 Bit 2, 3	EXP LS 53 Byte 0 Bits 4-7	EXP LS 50 Bytes 1, 2, 3

NOTE: The PSW can be manually displayed using the Console PR-KB. This procedure is contained in CPK.

EC

System Mask 0–7	KEY 8–11	1MWP 12–15	00 16–17	CC 18–19	Prog Mask 20–23	00000000 24–31	00000000 32–39	Instruction Address 40–63

SCOPE PROCEDURE FOR LOCAL STORAGE ADDRESSING

Use Tektronix* Type 454, or equivalent
10X Probes
Set Time/Div: .05 us
Set Channel 1 Volt/Div: 50 MV
Set Channel 2 Volt/Div: 50 MV

Store 385E6FC8 (using system console rotary switches A through H) in an unused location of control storage. This is a word-move word: Y = LNK, SF, STOP

Set the P-register = 02

The word-move word is the version 1 type that carries the X- and Y-lines of the Source register in byte 1 of the control word. The mask of F specifies that all four bytes of the source are to be moved to the destination register. The STOP function is active (bit 4 of byte 3 = 1); therefore, the word-move word is continually executed.

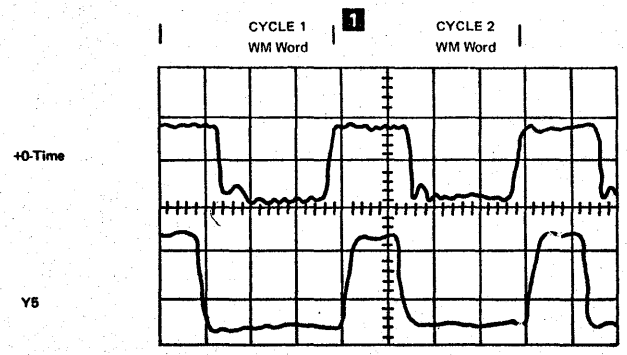
After storing the word and setting the P-register,

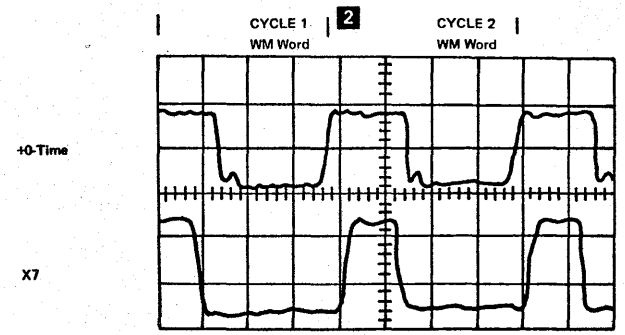
1. Dial the address of the word-move word into switches E through H.
2. Operate the control address set key.
3. Operate the start key.

Note: The manual indicator is on because the soft-stop condition is set by the STOP function of the word-move word.

4. Sync with channel 1 on +0-time (gate A C4 E2 G05).

With channel 2, scope the + A-LS ADDR Y EQUALS 5 line (C4 M2 J12).





With channel 2, scope the + A-LS ADDR X EQUALS 7 line (C4 F2 G12).

Scope pictures 1 and 2 show the X- and Y- lines for the source register LNK being activated early in the cycle of the word-move word.

By altering the P-register setting, and the address bits in the word-move word, all the local-storage registers may be addressed.

For example:

Original word—385E6FC8 Statement—Y = LNK, SF, STOP
Change word to—385E7FC8 Statement—Q = LNK, SF, STOP

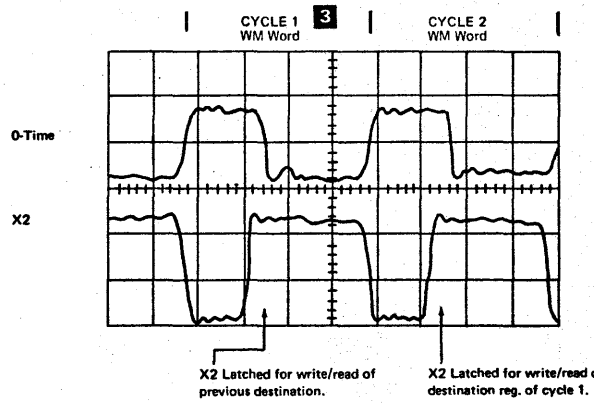
Change P-register setting to 03; leave the original control word in control storage. Statement—MN=LNK, SF, STOP

For the P-register settings necessary for addressing local storage, refer to the local-storage map in the CPU Hardware (CPU) section.

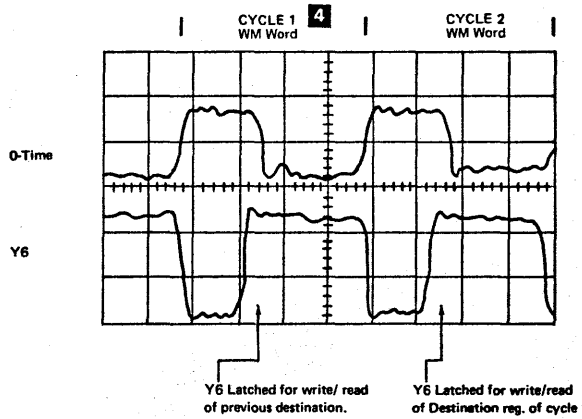
For variations of this word, refer to the bit definition of the word-move word in the Microprogram (MIC) section.

*Manufactured by Tektronics, Inc.

With channel 2, scope the + SLOW AX PATH DECODE 2 (C4 G4 B03)



With channel 2, scope the + SLOW AY PATH DECODE 6(C4 J4 D02)!



Scope pictures 3 and 4 show the X- and Y-lines for the destination Y-register being activated in the second half of the word-move word cycle. Destination addressing is always activated for the destining of the previous control word's results.

EXPANDED LOCAL STORAGE (EXPLS)

- Composed of hardware registers that are physically externals but are logically connected as local storage.
- Source addresses are formed through the expanded local-storage-address assembler.
- Destination addresses are formed through the local-storage-control assembler and the A-local-storage-address assembler.
- Expanded local-storage registers are not duplicated as are the local-storage registers.
- Used with I-cycles, selector channels, and address-adjustment circuits.

In order to access expanded local storage :

1. Mode register bit 1 must be on.
2. Direct local-storage addressing must be specified (C1 or C2 bit 0 = 0).

P-register bits 0, 3, and 4 control the expanded local-storage inputs to the A- and B-registers (shown on page CPU 20).

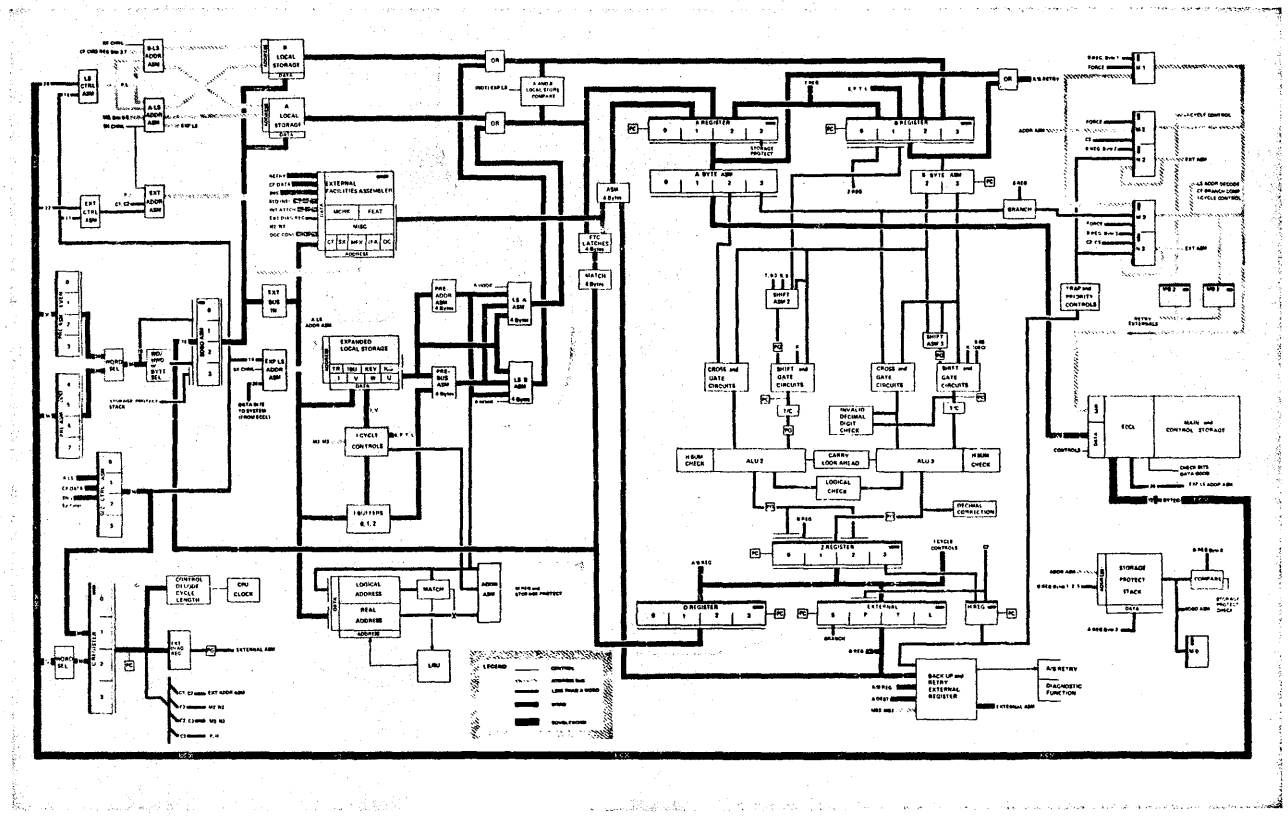
Only one expanded local-storage register can be accessed as a source in any one control word.

The branch and link/return word and the word-move word version 1 cannot access expanded local-storage registers.

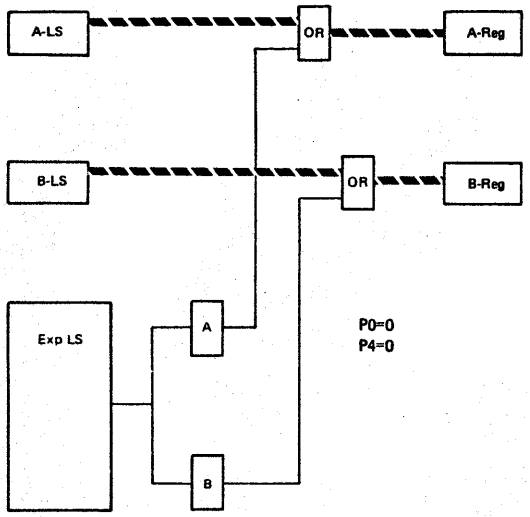
All expanded local-storage registers may be displayed from the console, but altered only from the console printer-keyboard.

The I, V, U, and W-registers are not under control of bits 0 and 4 of the P-register, but do require that P low = 2 or A.

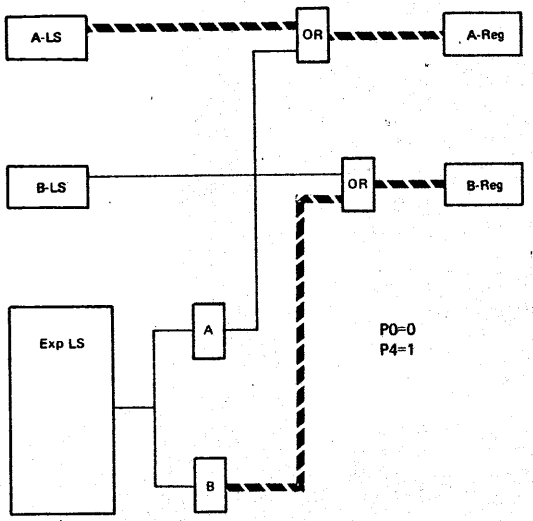
If different expanded local-storage addresses are called for with A and B inputs, the register designated as the A source is used for both A and B sources.



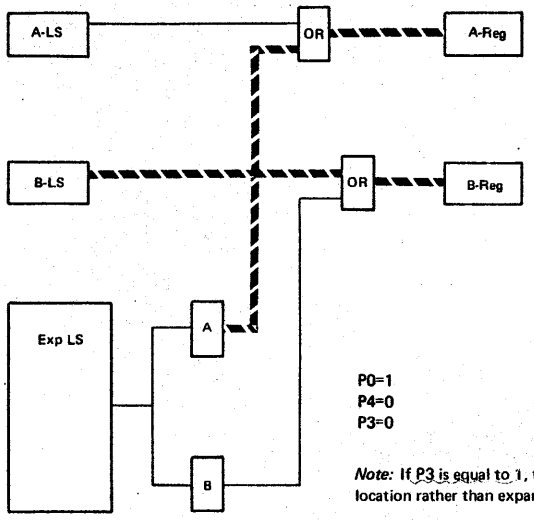
LOCAL STORAGE GATING



EXPANDED B-LOCAL STORAGE INPUT



EXPANDED A-LOCAL STORAGE INPUT



Note: If P3 is equal to 1, the address accesses an external location rather than expanded local storage.

I, V, U, and W are not dependent on the setting of P0 or P4.

EXPANDED LOCAL STORAGE MAP

Note: Expanded local storage may be altered from the printer-keyboard with the CE key on.

EXP LS	Word Name	Byte 0	Byte 1	Byte 2	Byte 3	X and Y Line
50	I	Key		I-Register		X2 Y0
51	V			V-Register		X2 Y1
52	W			W-Register		X2 Y2
53	U			U-Register		X2 Y3
54	IBU			IBU-Register		X2 Y4
55	TR			TR-Register		X2 Y5
56	ICS	I-Cycle Control Display				X2 Y6
		57 through 5F unassigned				
60	G2DRL	DATA ADDR (SX 2)				X4 Y0
61	G2DBRL	BACKUP DATA ADDR				X4 Y1
62						X4 Y2
63						X4 Y3
64	G3DRL	DATA ADDR (SX 3)				X4 Y4
65	G3DBRL	BACKUP DATA ADDR				X4 Y5
66						X4 Y6
67						X4 Y7
68	G1DRL	DATA ADDR (SX 1)				X5 Y0
69	G1DBRL	BACKUP DATA ADDR				X5 Y1
6A						X5 Y2
6B						X5 Y3
6C	G4DRL	DATA ADDR (SX 4)				X5 Y4
6D	G4DBRL	BACKUP DATA ADDR				X5 Y5
6E						X5 Y6
6F						X5 Y7
		70 through 77 unassigned				
78	SN	SEGMENT NUMBER GATE				X7 Y0
79	PN	PAGE NUMBER GATE				X7 Y1
7A	WK	Working Register				X7 Y2
7B	NP	*		CONTROL	CONTROL	X7 Y3
7C	DK	Local Addr. Reg.		TLB DESTINATION GATE		X7 Y4
7D	SS			SEG SIZE		X7 Y5
7E						X7 Y6
7F						X7 Y7

* SEGMENT AND PAGE NO. FROM PAA

I-Register (EXPLS 50)

When the instruction counter register is used as a destination, byte 0, 24 bits, or fullword-loading is possible.

Byte 0: If the I-Reg is destined, byte 0 is gated to the Key-Reg
Bytes 1,2, and 3 contain the instruction address

V-Register (EXPLS 51)*

Bytes 1, 2, and 3 usually contain the second operand address generated during I-cycles.

W-Register (EXPLS 52)*

Bytes 1, 2, and 3 usually contain the first operand address generated during I-cycles.

* If used in a storage word, the key register is gated as byte 0. The key register contains the storage protect key (bits 0-3, bits 4-7=0).
If V or W is used as a storage address in a storage word, the KEY reg is gated as byte 0. The KEY reg is always gated as byte 0 for I, TR, and IBU.

U-Register (EXPLS 53)

When used as a destination, byte 2 may not be changed (loaded via hardware)

Byte 0 bits 0-1 used for ILC
bits 2-3 used for CC
bits 4-7 used for program mask
Byte 1 bits 0-1 reserved for FLP mult. and divide
bit 2 Indicates that GRs 0-3 need restoring.
bit 3 indicates LEX MODE
bit 4-7 used for OWMP
Byte 2 used for Op-code
Byte 3 used for immediate byte information
Note: Byte 0 bits 0 and 1 and Byte 2 bits 0 through 7 are set only by hardware.

IBU-Register (EXPLS 54) *

Upon entering I-cycles, I-Reg bytes 1, 2, and 3 are set into IBU. If a retry condition is encountered during I-cycles, the instruction may be repeated (return to DF0C). In this event, IBU is moved to the I-Reg. This source-only register may not be used as a destination. IBU is loaded via hardware only from the I-register.

TR-Register (EXPLS 55) *

This register is the address of next doubleword after the address in the I-register. TR-Reg may not be used as a destination, but changes if the I-Reg is destined.

ICS (I-Cycle Status) Register (EXPLS 56)

This unique register is provided for manual display only, and is not accessible via microcode. While the specific signals occupy an expanded local-storage register address, no such register exists. Instead, various key signals from the I-cycle hardware (under hardware control) are gated via the register address to form a display. Interpretation of the display requires a fundamental knowledge of the functional operation of the I-cycle hardware.

Byte 0 forced to zeros
Byte 1 forced to zeros
Byte 2
bit 0 BR Read Latch
bit 1 Op Load Latch
bit 2 Op L2
bit 3 Op L1
bit 4 Prefetch required
bit 5 Prefetch inhibit
bit 6 FLP Long
bit 7 Op BR to DF
Byte 3
bit 0 I-Bfr 0 parity check latch
bit 1 I-Bfr 1 parity check latch
bit 2 Half Adder check latch
bit 3 IMM byte modifier parity check
bit 4 X=0
bit 5 B=0
bit 6 Set Control Address
bit 7 Low bit

ICS Bits: Functional Significance

Byte 2: **Bit 0** (BR Read Latch), when on, indicates that a RTN to I-cycles has forced an initial I-cycle address of DF0C.

Bit 1 (Op Load Latch), when on, indicates a hardware attempt to:
a) provide an initial I-cycle address of DF14 (if byte 2 bit 0 is off).
b) provide an address of DF14 for a further fetch of the instruction when within I-cycles and Set Control Address (Set CA) (byte 3 bit 6) is on.

Op Length		
1	2	Format
1	0	RR
0	1	RX SI
1	1	SS

Bit 2 (Op Length 2) from decode of two high-order bits of Op Reg, when on, denotes that the data currently in Op Reg is not an RR format.

Bit 3 (Op Length 1) from decode of two high-order bits of Op Reg, when on, denotes that the data currently in Op Reg is not of RX, RS, SI format.

Bit 4 (Prefetch required) when on indicates (as a function of I-Reg, I-Bfrs, and current instruction conditions) that the next instruction should be prefetched.

Bit 5 (Prefetch inhibit) When on, indicates that a Prefetch will not be allowed. This signal is also a function of I-Reg, I-Bfrs, and current instruction. Note that this signal does not take into account other functions, such as Real Instruction Address Compare mode.

Bit 6 (Floating Pt Long), when on, represents a partial decode of the data in the Op Reg. This signal is used with RR instruction format to determine a specific I-cycle path.

Bit 7 (Op Branch to DF), when on, indicates that the end of the hardware I-cycles branched to the read and align phase of I-cycles (the other half of the DF module instead of a CX module).

Byte 3: **Bit 0** (I-Bfr 0 parity check latch), when on, indicates an incorrect parity condition for I-Bfr 0.

Bit 1 (I-Bfr 1 parity check latch) when on, indicates an incorrect parity condition for I-Bfr 1.

Bit 2 (Half Adder check latch), when on, indicates that a check condition occurred in the half adder during an I-register hardware update.

Bit 3 (Immediate Byte Modifier parity check), when on, indicates a parity check of the immediate byte modifier register. This signal is the check latch input.
Note: The above four signals are combined to form the signal 'I-cycle hardware check'.

Bit 4 (X=0) Has particular significance when the op reg data is for an RX format. When this bit is off (and byte 3 bit 5 is off) double indexing is indicated for RX format instructions. Note that RS, SI, and SS format instructions force this signal on.

Bit 5 (B=0), when on, indicates that the data being gated through the base assembler of the I-Bfrs is zero. (refers to GPR00). This signal has no significance RR format instructions.

Bit 6 (Set CA), when on, indicates either that the current control store address is not within the hardware I-cycle range, or that a branch point within that address range has been encountered.

Bit 7 (Low Bit) when on, indicates that hardware-detected conditions are currently satisfied for deviating from the normal branch to the execution: the starting address is C (Op code) 4 instead of C (Op code) 4. This signal has significance for certain branch instructions, floating-point, and shift-double instructions.

Byte 3 bits 0-3 provide a further definition of the cause of an I-cycle hardware check. Byte 2, bits 0-6 and byte 3, bits 4-6 may be used, with discretion, to determine the starting address for I-cycle sequences. Byte 2 bit 7 indicates when the "read and align" phase is used; and Byte 3 bit 7 indicates the status of hardware tests for branch on condition, floating-point reg, specification, etc.

G2DRL, G3DRL, G1DRL, and G4DRL (EXPLS 60, 64, 68, and 6C)

These registers function as a pointer to the next storage location used for a share cycle.
BYTE 0 contains the protect key.

G2DBRL, G3DBRL, G1DBRL, and G4DBRL (EXPLS 61, 65, 69, and 6D)

Only bytes 1 and 2 exist.

SN-Register (EXPLS 78)

Byte 0	FF hex
Byte 1	00 hex

PN-Register (EXPLS 79)

Byte 0	FF hex
Byte 1	00 hex

WK-Register (EXPLS 7A)

Working Register
Byte 0 = FF

NP-Register (EXPLS 7B)

Bytes 0 and 1 Logical Address
Byte 2
Bit 0
Bit 1
Bit 2
Bit 3
Bit 4
Bit 5 Lex Mode
Bit 6
Bit 7 Reset Tables
Byte 3
Bit 0
Bit 1
Bit 2
Bit 3
Bit 4 Execute Instruction
Bit 5
Bit 6
Bit 7

See "Dynamic Address Translation NP2 and NPS Register," page CPU 155.

DK-Register (EXPLS 7C)

Byte 0 Bit 0 } Logical Address
Bit 1
Bit 2
Bit 3
Bit 4
Bit 5
Bit 6
Bit 7
Byte 1 Bit 0
Bit 1
Bit 2
Bit 3
Bit 4
Bit 5 0
Bit 6 0
Bit 7 0
Byte 2 Bit 0 } Real Address
Bit 1
Bit 2
Bit 3
Bit 4
Bit 5
Bit 6
Bit 7
Byte 3 Bit 0
Bit 1
Bit 2
Bit 3
Bit 4
Bit 5
Bit 6
Bit 7

EXPANDED LOCAL STORAGE: SOURCE GATING

When used as a source, expanded local storage is addressed by the expanded local-storage address assembler. The address bits from the control word are intercepted as the control word is being read from control storage, and gated to the expanded local-storage address assembler. Gating lines generated by the address assembler gate the proper expanded local-storage register to either the A- or B-register.

If expanded local storage is being gated to the A-register, the sense latches for A-local storage are blocked from being set. If expanded local storage is being gated to the B-register, the sense latches for B-local storage are blocked.

If the expanded local-storage register is also the destination, the destination latches in local storage are set to be used in the following control-word cycle.

Examples on the following pages show the various ways the expanded local-storage gates may be formed for source addressing.

Whether the expanded local-storage register is an A or a B source, it is gated to both the A-LS and B-LS assemblers. The source gating circuits then gate the A or B assembler to the A- or B-register.

There are control words in the microprogram listings that appear to be addressing two different expanded local-storage registers as sources in the same word. However, the B-source address defaults to the A-source. For example:

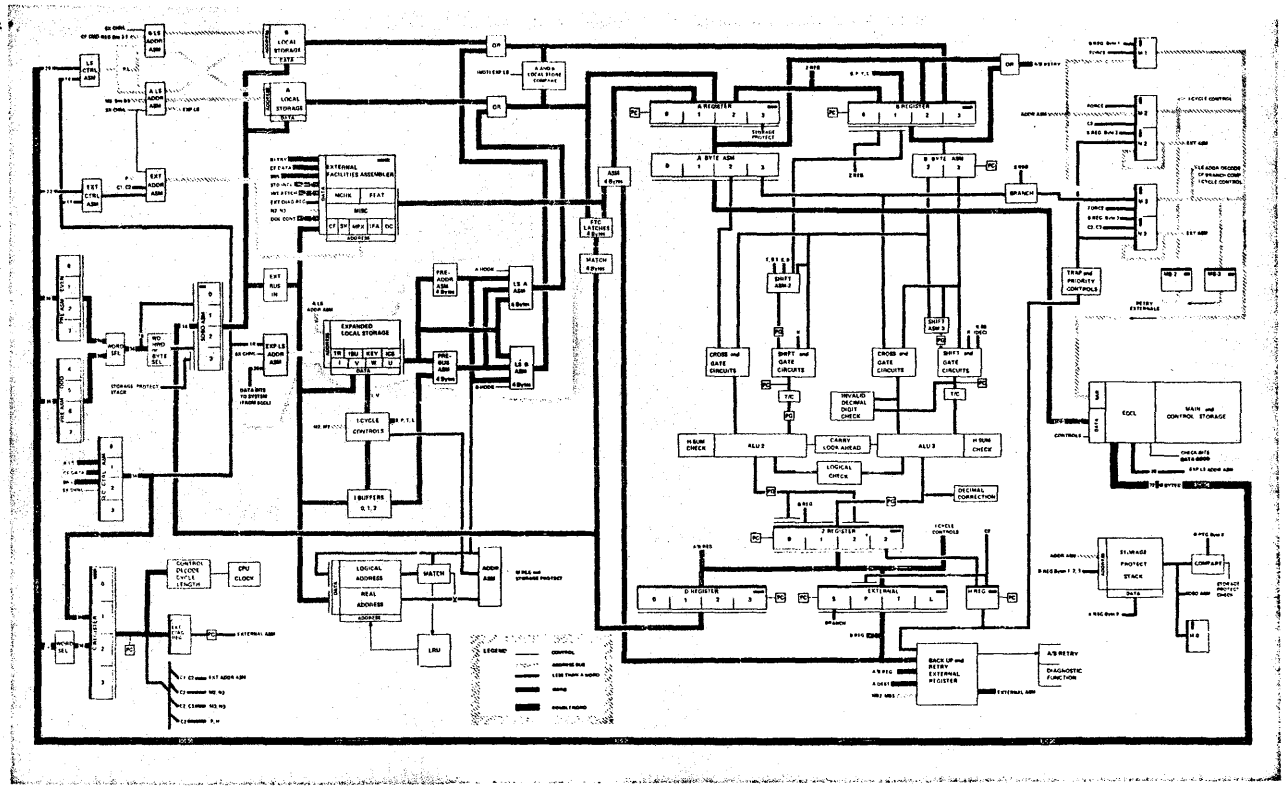
Control Word————WK1 = NP2, OE, WK1

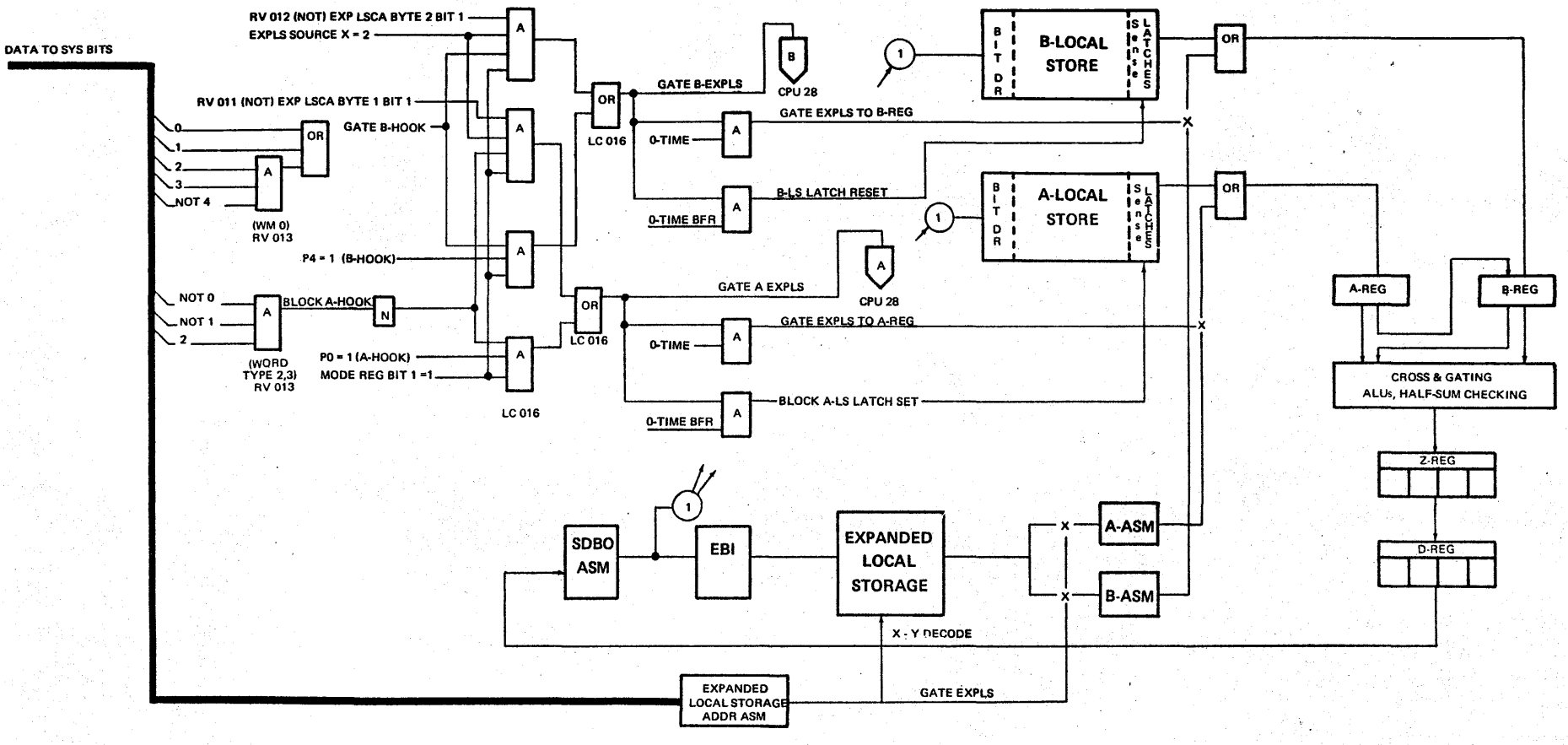
NP2 is the A-source
WK1 is the B-source

The decode of this word defaults to read effectively

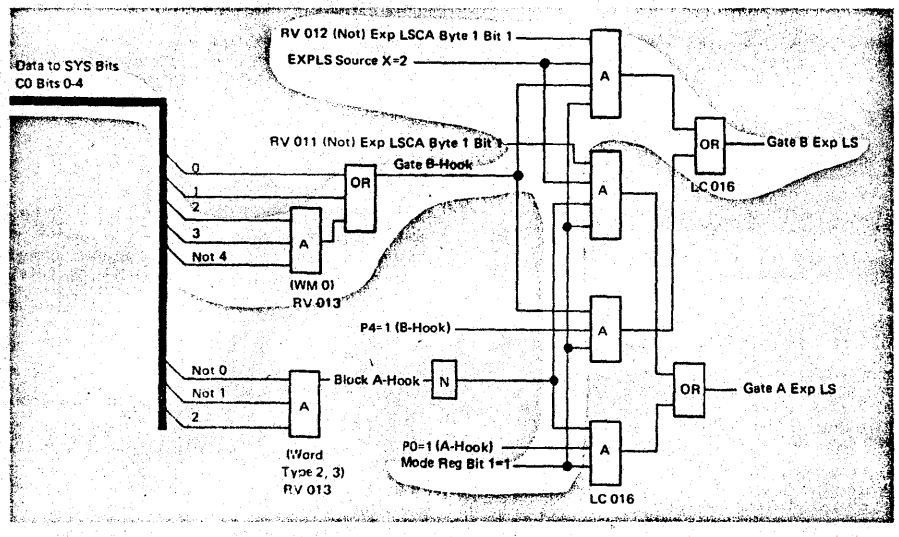
WK1 = NP2, OE, NP20

This type of control word is valid only if the B-source is being blocked from ALU entry.




DATA TO SYS BITS
RV 012 (NOT) EXP LSCA BYTE 2 BIT 1
EXPLS SOURCE X = 2
RV 011 (NOT) EXP LSCA BYTE 1 BIT 1
GATE B-HOOK
0
1
2
3
NOT 4
OR
A
(WM 0)
RV 013
P4 = 1 (B-HOOK)
NOT 0
NOT 1
2
BLOCK A-HOOK
N
(WORD
TYPE 2,3)
RV 013
P0 = 1 (A-HOOK)
MODE REG BIT 1 =1
LC 016
GATE B-EXPLS
B
CPU 28
0-TIME
0-TIME BFR
GATE EXPLS TO B-REG
B-LS LATCH RESET
1
BIT DR
B-LOCAL STORE
Sense LATCHES
A-LOCAL STORE
GATE A EXPLS
GATE EXPLS TO A-REG
BLOCK A-LS LATCH SET
X
A-REG
B-REG
CROSS & GATING
ALUs, HALF-SUM CHECKING
Z-REG
D-REG
SDBO
ASM
EBI
EXPANDED
LOCAL
STORAGE
A-ASM
B-ASM
X - Y DECODE
EXPANDED
LOCAL STORAGE
ADDR ASM
GATE EXPLS

EXPANDED LOCAL STORAGE: SOURCE GATING EXAMPLES



Word-Move Version 0

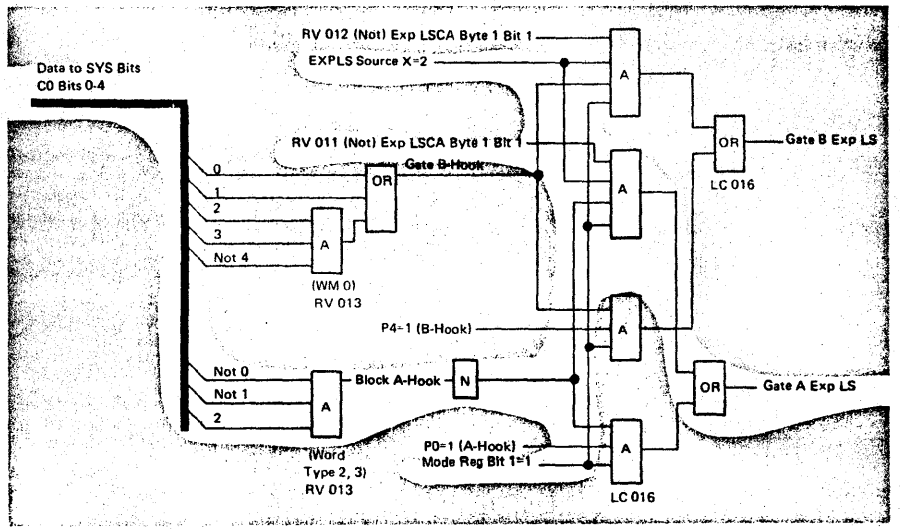
P Low=2 (X2 decode)

Statement C0 Bits 0-4=00110
C2 Bits 0-3=0010

Q=W,D7

In this word-move example, the expanded local-storage W-register is the B-source. W is one of the four expanded LS-registers that do not rely on P0 or P4 to bring up a gating line.

The upper AND circuit is activated by this control word. The line 'Gate B Exp LS' gates the W-register (from the B-Asm) to the B-local-storage bus-out. This gating line also blocks the B-local-storage sense latch set.



Arithmetic Word (A destination)

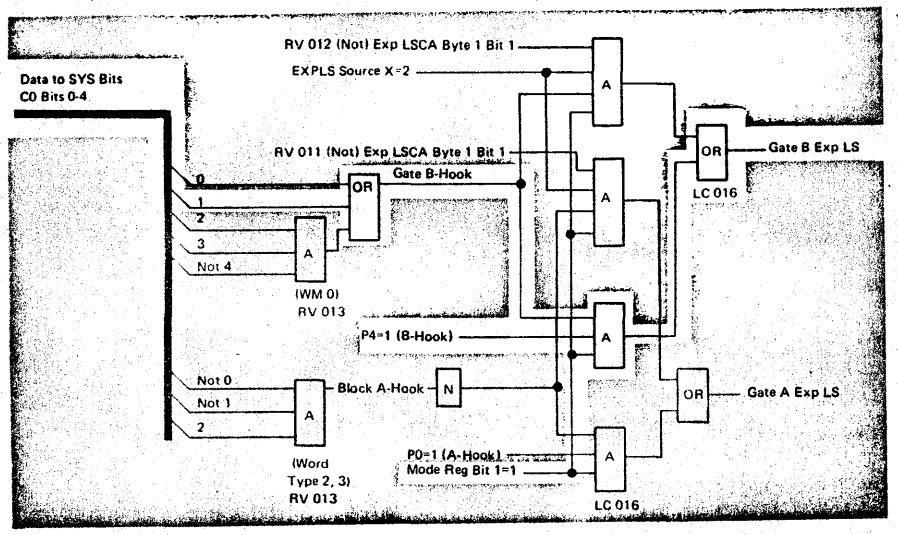
P Low=2 (X2 decode)

Statement C0 Bits 0-4=11000
C1 Bits 0-3=0001

V0=0

The decode of this arithmetic word specifies that the expanded local-storage V-register is to be accessed as the A-source and is also to be destination of the arithmetic result.

The 'Gate A Exp LS' line is activated through the AND circuit highlighted in the diagram. The X and Y decode for the V-register is also set up in the A-local storage destination latches for use in the following control word cycle.



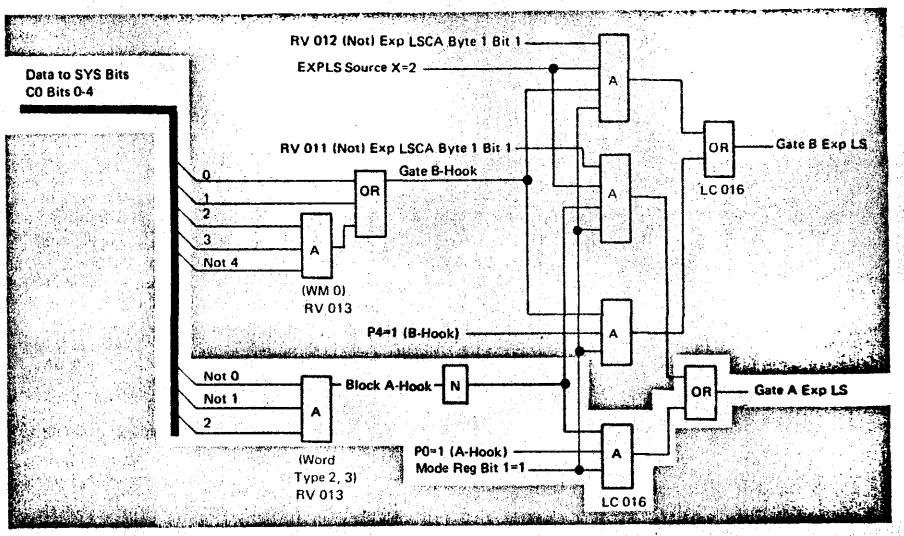
Storage Word (Read Word)

P Low = F (X7 decode)

Statement C0 Bits 0-4=01000
C2 Bits 0-3=0010

RDW RW WK, NOP

The expanded local-storage WK-register is the address source for this Read Word. The highlighted AND circuit brings up the 'Gate B Expls' line. The set to the B-local-storage sense latches is blocked to prevent any conflict on the B-register input.



Arithmetic Word (A destination)

P=87

Statement C0 Bits 0-4=10001
C1 Bits 0-3=0011

NP2=NP2, OR, K05

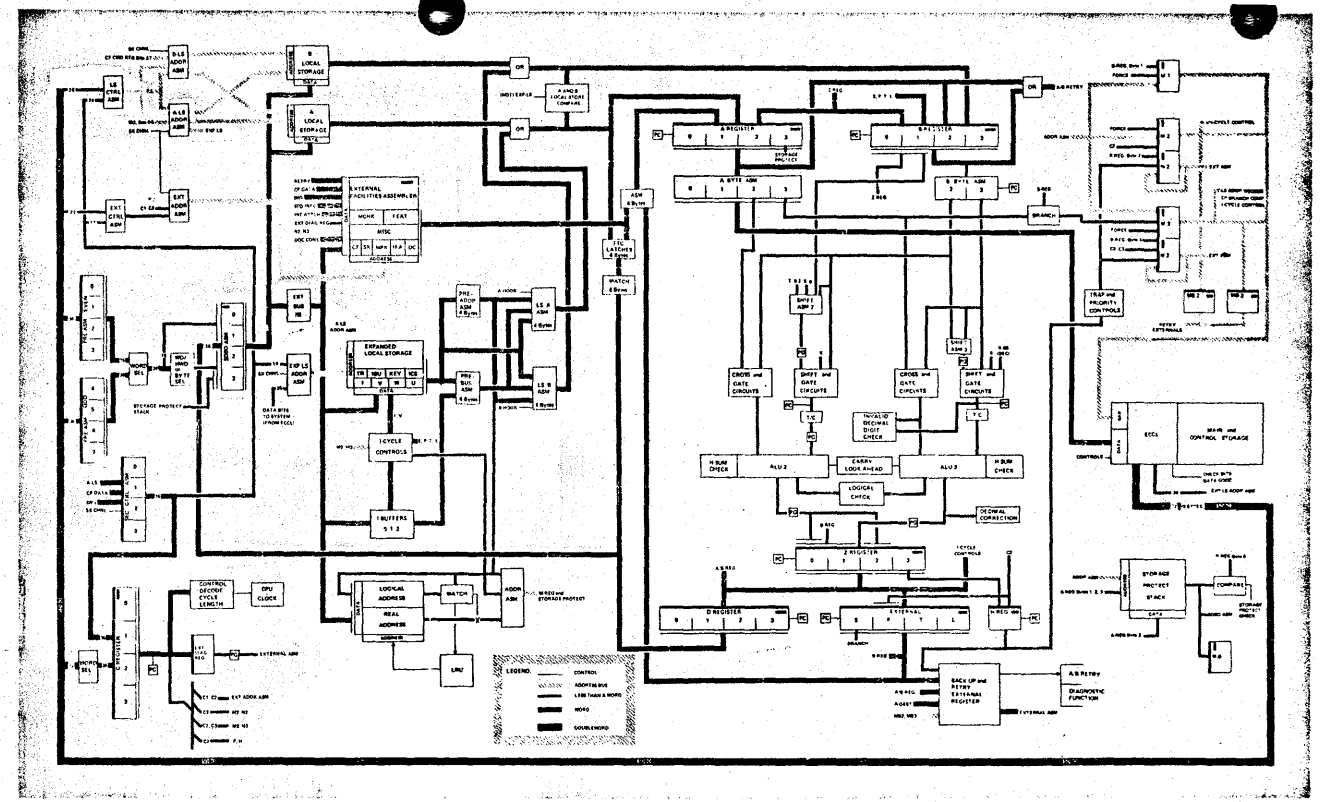
The expanded local-storage NP2 register is the A-source and destination in this arithmetic word. The 'Gate A Exp LS' line is brought up by the highlighted AND circuit.

The X- and Y-decode for NP is set up in the A-local storage destination latches for use in the following control word cycle.

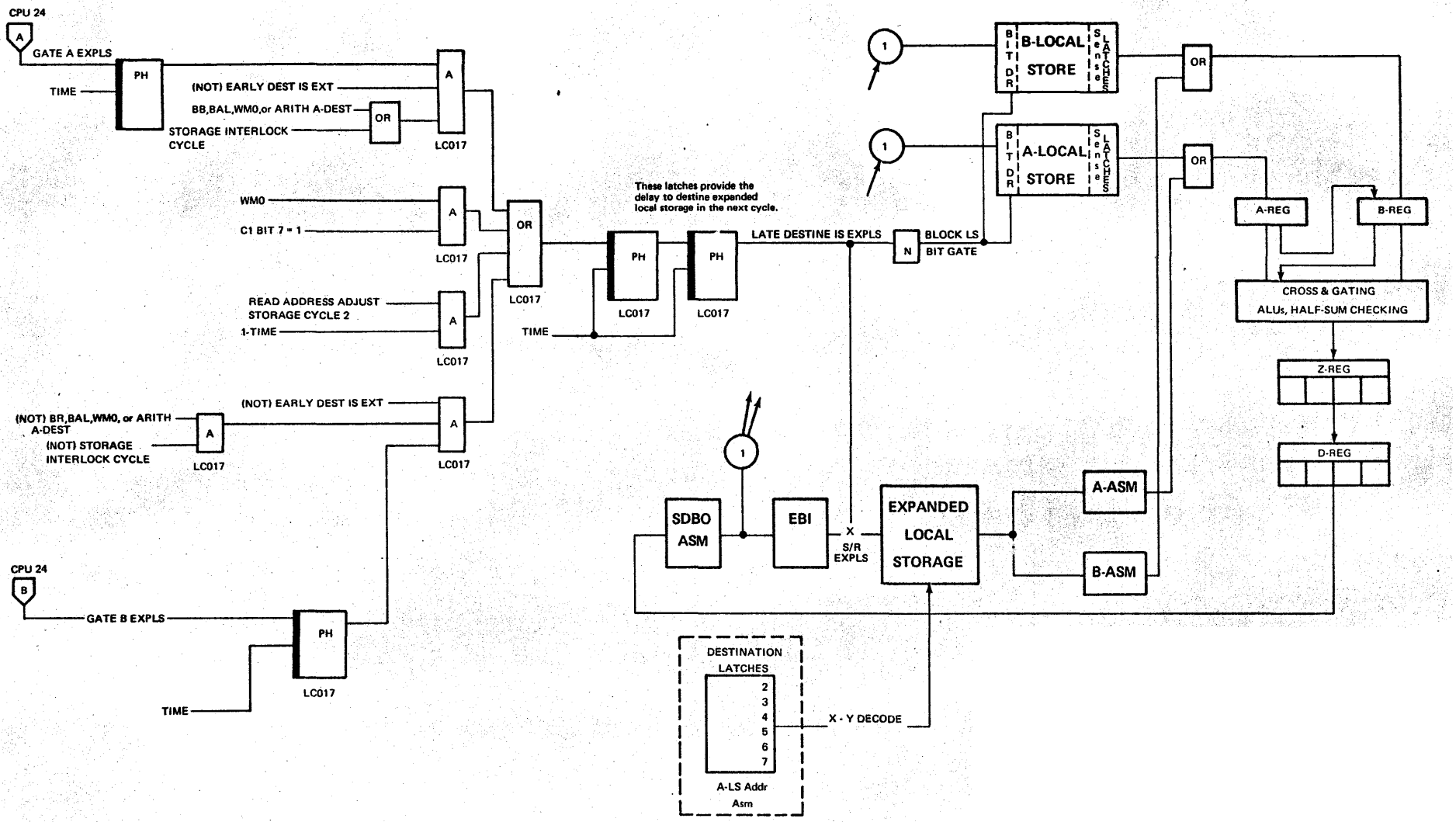
EXPANDED LOCAL STORAGE: DESTINING

When expanded local storage is addressed as a destination, the destination address latches of A-local storage are used to retain the address until the following cycle.

At destination time, the bit gates for both A- and B-local storage are blocked to prevent local storage from being set. The data destined to expanded local storage is gated from the D-register through the SDBO assembler on the EBI to the expanded local-storage register addressed by the A-local storage destination latches.



DESTINATION CONTROL



EXTERNAL FACILITIES

- External facilities are composed of registers, buses, status lines, and other circuitry that form the communications line between the microprogram and:
  - Channels
  - Console File
  - Console Printer-Keyboard
  - Checking facilities
  - Retry circuits
  - Integrated File Adapter
  - Features

- Addresses are formed from control words, console-file data, selector-channel circuits, console switches, retry information, and local-storage address data.
- Data from the externals enters the data flow through the external assembler to the A-Reg only. Externals cannot be gated to the B-Reg.
- Data to the externals is gated through the SDBO assembler on the External Bus-In (EBI).

External facilities have restrictions associated with them because of the manner in which they are used. For example, certain externals cannot be addressed as destinations for data, others cannot be sources of data.

EXTERNAL CONTROL ASSEMBLER A

- Receives data from the SDBO early in the CPU cycle to form the source gates necessary to gate-in external data to the data flow in time to be used in source-control-word operation.
- Receives data from the secondary control assembler to form addresses in conjunction with source selector channel, console file, or display operation.

X-Y DECODES B

The X- and Y-lines are brought up only for destination addresses. The X- and Y-combinations are routed to the various external hardware locations to bring up set and gating lines.

The X- and Y-lines are checked to assure that one and only one X-line, and one and only one Y-line is activated for a destination address. An X-compare check sets MCKA3 bit 0. A Y-compare check sets MCKA3 bit 1.

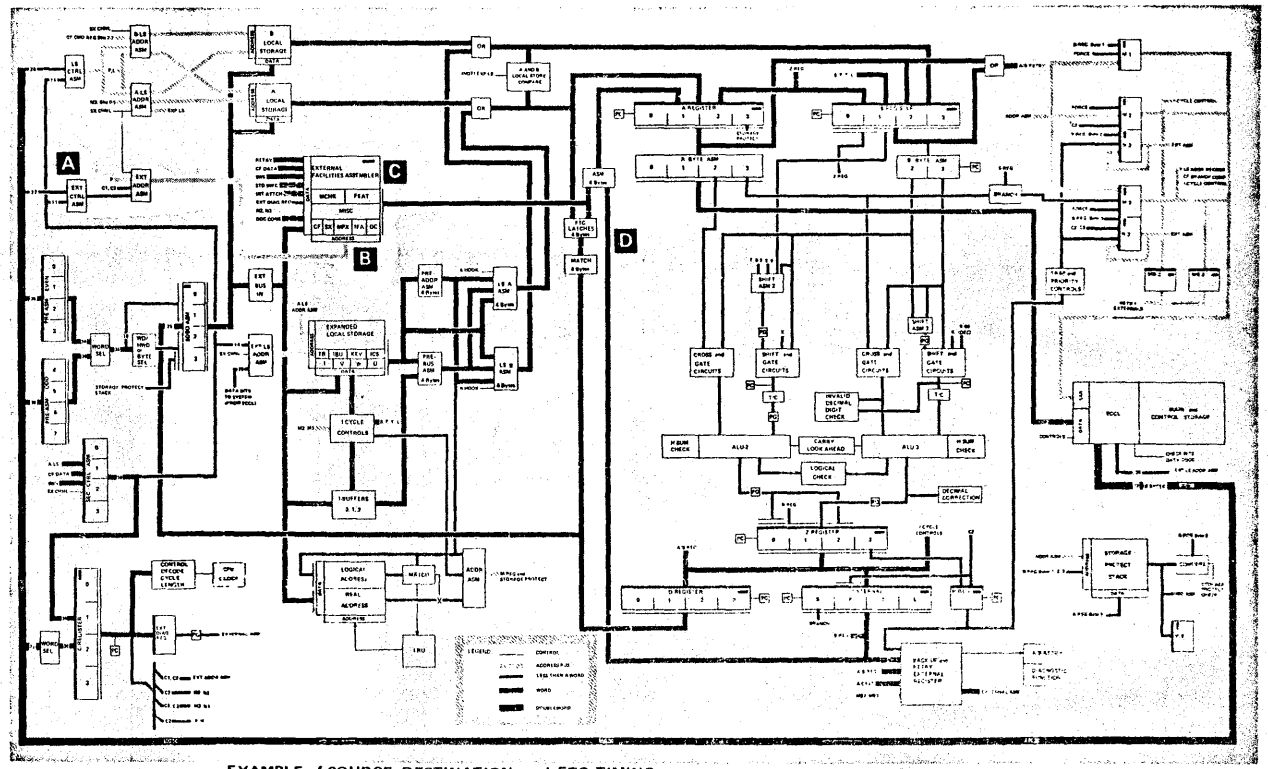
SOURCE ADDRESSING C

Source addressing is performed by generating gating lines that allow selected external buses to feed through the external or expanded external assembler into the CPU data flow.

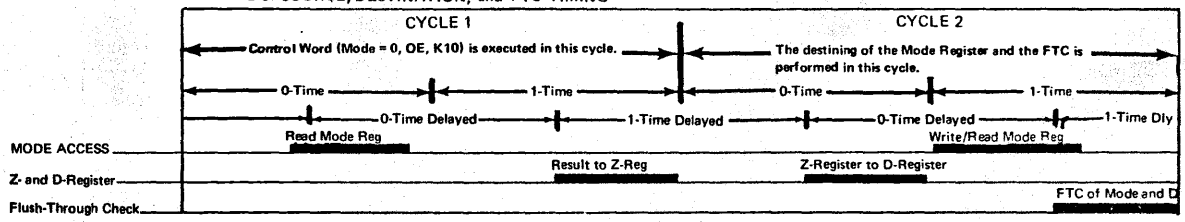
FLUSH-THROUGH CHECK D

Data destined to some external facility is gated from the D-register through the SD Bus-In to the selected external facility. The data from the D-register is gated to the flush-through-check match circuit where it is compared with the data from the external that was the destination. If the data does not compare, bit 2 of MCKA1 is set to indicate the error.

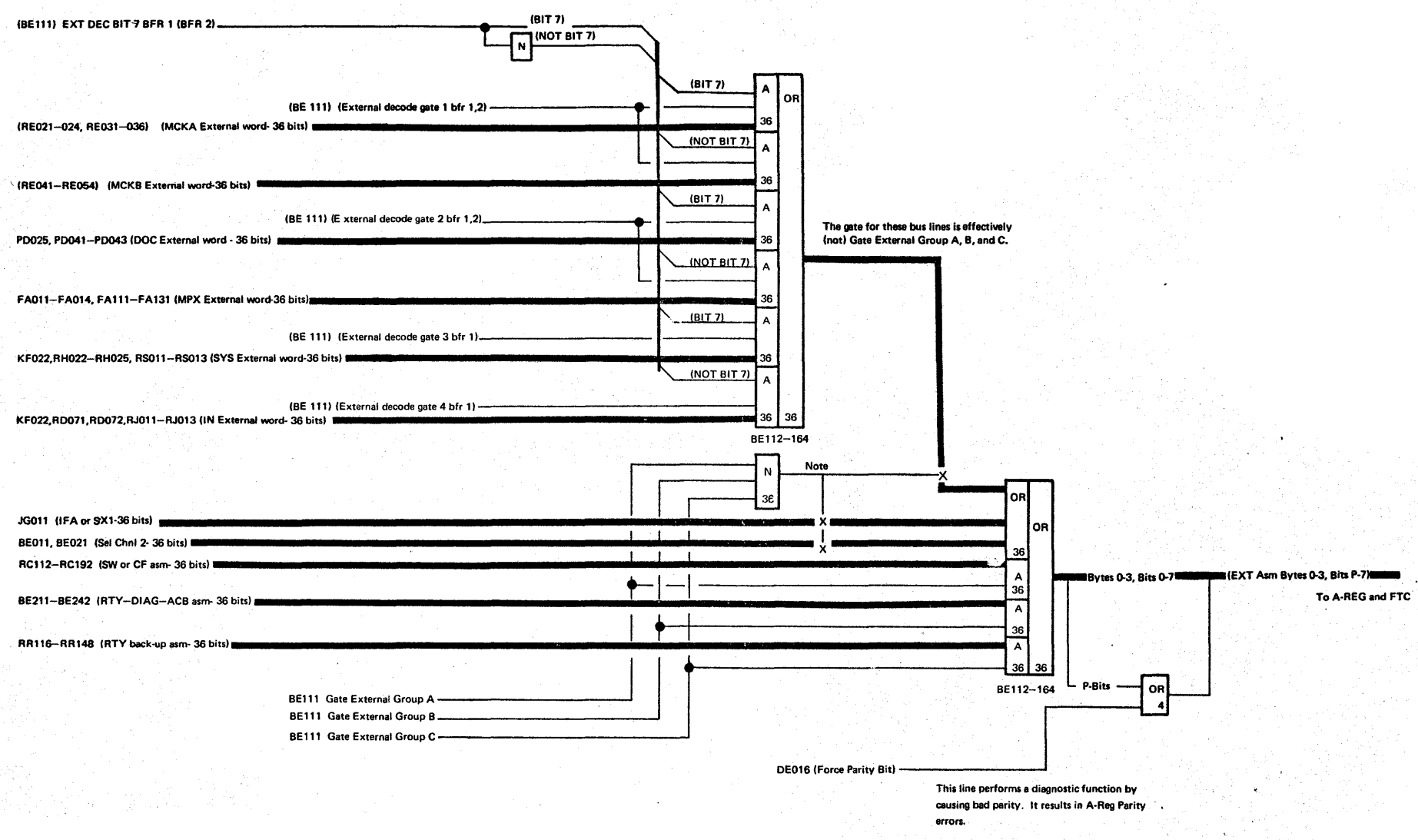
Data gated from the console file to the CFDR is not flush-through-checked.



EXAMPLE of SOURCE, DESTINATION, and FTC TIMING

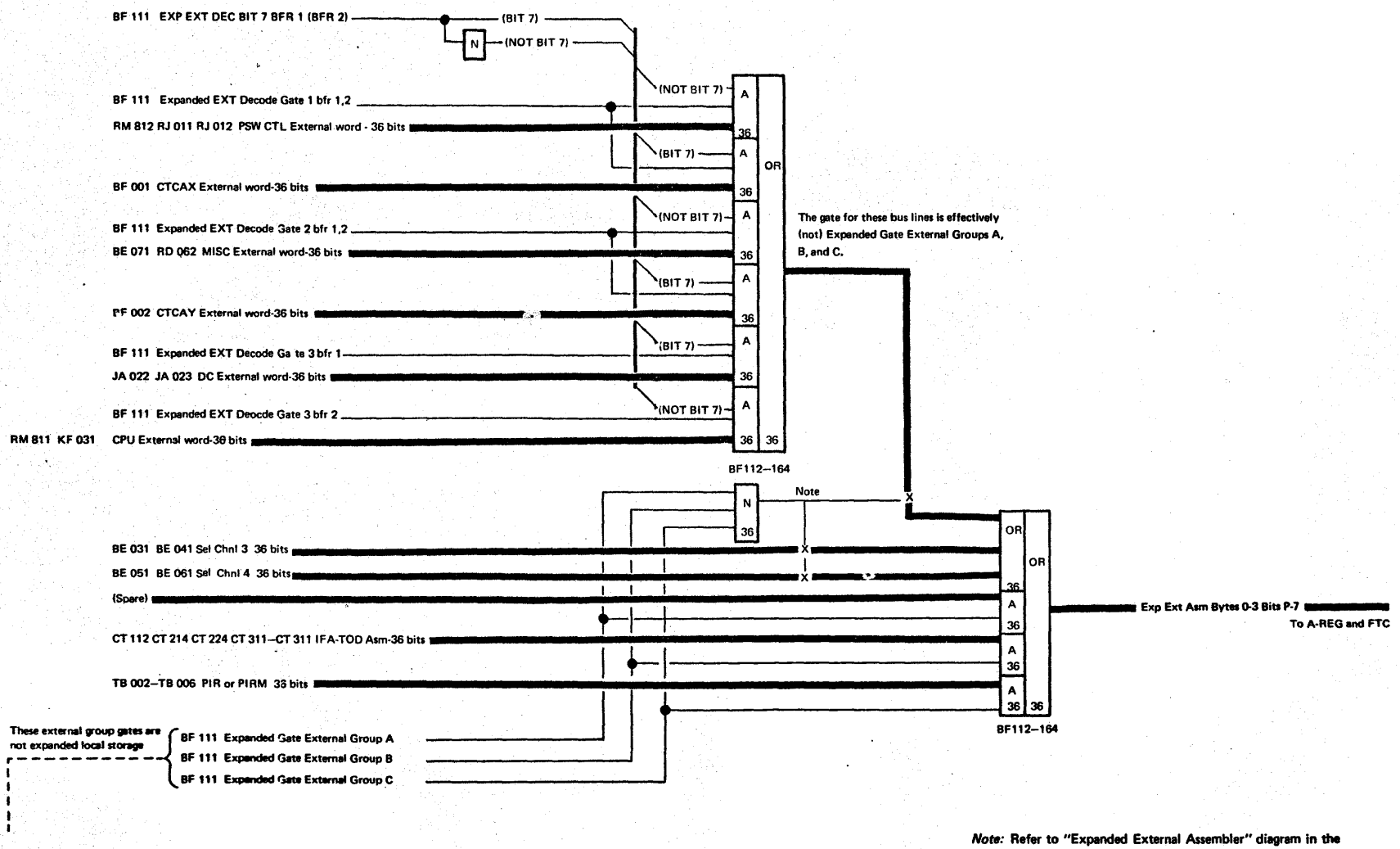


EXTERNAL ASSEMBLER DATA FLOW



Note: Refer to "External Assembler" diagram in the *3145 Processing Unit Maintenance Diagrams*, SY24-3580.

EXPANDED EXTERNAL ASSEMBLER DATA FLOW



Note: Refer to "Expanded External Assembler" diagram in the *3145 Processing Unit Maintenance Diagrams,* SY24-3580.

EXTERNAL ASSIGNMENT AND INDEX MAP

The one SPTL word and the one SYS word appear in every eight-word group. These two registers have direct-type addressing and are accessible with any P-register setting. SPTL is addressed when the hex digit C is specified in either the A-source or the B-source fields of a control word. SYS is addressed when the hex digit D is specified in the A-source field.

Bit MCKA and MCKB are set to zero when MCKA is used as a destination in a word-move word, with the NOREG as the source.

WORD ADDRESS	WORD NAME	BYTE 0	BYTE 1	BYTE 2	BYTE 3	X Y LINE
00	RTY	MB	MB 3	ECNT	RCNT	00
01	NOREG	NOREG0	NOREG1	NOREG2	NOREG3	01
02	DIAG	DIAG0	DIAG1	FEAT 2	FEAT 3	02
03	XXXXXXXX	XXXXXXXX	XXXXXXX	XXXXXXX	XXXXXXXX	03
04	SPTL *	S-REG	P-REG	T-REG	L-REG	04
05	SYS *	SYS0	SYS1	SYS2	H-REG	05
06	MCKB *	MCKB0	MCKB1	MCKB2	MCKB3	06
07	MCKA	MCKA0	MCKA1	MCKA2	MCKA3	07
08	CPU	MODE	CFDAR	LRUM	MATCH	10
09	CFDR	CFDR	CFDR	CFDR	CFDR	11
0A	ACB	ACB0	ACB1	XXXXXX	XXXXXXXX	12
0B	SW	SW0	SW1	SW2	SW3	13
0C	SPTL *	S-REG	P-REG	T-REG	L-REG	14
0D	SYS *	SYS0	SYS1	SYS2	H-REG	15
0E	MPX	MT0	MT1	MB1	MB0	16
0F	DOC	T1	TA	TT	TE	17
10	PSWCTL			MSKA	MSKB	20
11	CTCAX	CTCAX0	CTCAX1	CTCAX2	CTCAX3	21
12	MISC	EXTINT		EC LEVEL	SER 1	22
13	CTCAY	CTCAY0	CTCAY1	CTCAY2	CTCAY3	23
14	SPTL *	S-REG	P-REG	T-REG	L-REG	24
15	SYS *	SYS0	SYS1	SYS2	H-REG	25
16	IN	INTA	INTB	SER2	SER3	26
17	DC	DCB0	DCH1	TRBO	DCB1	27
18	ABRTY	ABRTY0	ABRTY1	ABRTY2	ABRTY3	30
19	SPTLB	SRTY	PRTY	TRTY	LRTY	31
1A	HMRTY		HRTY	MRTY2	MRTY3	32
1B	CPURTY	BYDST	RTYFLG	LSDST	EXTDST	33
1C	SPTL *	S-REG	P-REG	T-REG	L-REG	34
1D	SYS *	SYS0	SYS1	SYS2	H-REG	35
1E						36
1F						37

[redacted] May not be used as a destination.

*Not Flush-Through Checked.

Both MCKA and MCKB are set to zero when MCKA is used as a destination in a word-move word, with the NOREG as the source.

Address	Described In
00	REC
01	CPU
02	DIAG (Bytes 0, 1) CPU (Bytes 2, 3)
03	---
04	CPU
05	REC (Bytes 0, 1, 2) CPU (Byte 3)
06	REC
07	REC
08	CPU
09	CFA
0A	CPU
0B	CPU
0C	CPU
0D	REC (Bytes 0, 1, 2) CPU (Byte 3)
0E	CHNL
0F	CPK
10	CPU
11	FEAT
12	FEAT (Byte 0)CPU (Bytes 2, 3)
13	FEAT
14	CPU
15	REC (Bytes 0, 1, 2) CPU (Byte 3)
16	CPU
17	FEAT
18	REC and DIAG
19	REC
1A	REC
1B	REC
1C	CPU
1D	REC (Bytes 0, 1, 2) CPU (Byte 3)
1E	---
1F	---

WORD ADDRESS	WORD NAME	BYTE 0	BYTE 1	BYTE 2	BYTE 3	X Y LINE	
20	GBUF FBAK	GBO FWB	GB1 FCH	GB2 FCL	GB3 FOP	4 0	SX1/IFA
21	GBS FCND	GSP FDS	GBF FHC	GCT FED	GBD FMOD	4 1	
22	GSTAT FSTAT	GF FFL	GE FSC	GS FST	GL FGL	4 2	
23	GTAG FTAG	GTO FTO	GT1 FT1	GO FBO	GR FDR	4 3	
24	SPTL *	S-REG	P-REG	T-REG	L-REG	4 4	
25	SYS *	SYS0	SYS1	SYS2	H-REG	4 5	
26	FERR	FSB	FGT	FTS	FAT	4 6	IFA
27						4 7	
28	GBUF FRR	GBO FRRA	GB1 FRRC	GB2 FSC	GB3 FSR	5 0	SX4/IFA
29	GBS	GSP	GBF	GCT	GBD	5 1	
2A	GSTAT	GF	GE	GS	GL	5 2	
2B	GTAG	GTO	GT1	GO	GR	5 3	
2C	SPTL *	S-REG	P-REG	T-REG	L-REG	5 4	
2D	SYS *	SYS0	SYS1	SYS2	H-REG	5 5	
2E	ADJT	LOGICAL ADDR		REAL ADDR		5 6	
2F						5 7	
30	GBUF	GB0	GB1	GB2	GB3	6 0	SX2
31	GBS	GSP	GBF	GCT	GBD	6 1	
32	GSTAT	GF	GE	GS	GL	6 2	
33	GTAG	GTO	GT1	GO	GR	6 3	
34	SPTL *	S-REG	P-REG	T-REG	L-REG	6 4	
35	SYS *	SYS0	SYS1	SYS2	H-REG	6 5	
36	TODH	TODH0	TODH1	TODH2	TODH3	6 6	
37						6 7	
38	GBUF	GB0	GB1	GB2	GB3	7 0	SX3
39	GBS	GSP	GBF	GCT	SX3	7 1	
3A	GSTAT	GF	GE	GS	GL	7 2	
3B	GTAG	GT0	GT1	GO	GR	7 3	
3C	SPTL *	S-REG	P-REG	T-REG	L-REG	7 4	
3D	SYS *	SYS0	SYS1	SYS2	H-REG	7 5	
3E	TODL	TODL0	TODL1	TODL2	TODL3	7 6	
3F						7 7	

may not be used as a destination

*Not Flush-Through Checked

Address	Described In
20	CHNL or IFA
21	CHNL or IFA
22	CHNL or IFA
23	CHNL or IFA
24	CPU
25	REC (Bytes 0, 1, 2) CPU (Byte 3)
26	IFA
27	--
28	CHNL or IFA
29	CHNL
2A	CHNL
2B	CHNL
2C	CPU
2D	REC (Bytes 0, 1, 2) CPU (Byte 3)
2E	--
2F	--
30	CHNL
31	CHNL
32	CHNL
33	CHNL
34	CPU
35	REC (Bytes 0, 1, 2) CPU (Byte 3)
36	FEAT
37	--
38	CHNL
39	CHNL
3A	CHNL
3B	CHNL
3C	CPU
3D	REC (Bytes 0, 1, 2) CPU (Byte 3)
3E	FEAT
3F	--

NOREG Word

This fullword facility is not really a register. It is used to zero-out other locations. For example, if a word-move control word specifies that bytes 1 and 3 of the NOREG are to be moved to a local-storage location, then bytes 1 and 3 of that location are set to all zeros with odd parity.

Diag Word Byte 2

3145 Models FED, GE, GFD, H, HG, and I

(Feat 2) Byte 2, Bits 0-3 Main Storage Size
- 1 = 112k
- 2 = 160k
- 3 = 208k
- 4 = 256k
- 5 = 384k
- 6 = 512k

Bit 4 IFA

Bits 5, 6 Channels (Note: IFA counts as one channel)
- 00 = 1
- 01 = 2
- 10 = 3
- 11 = 4

Bit 7 Word Buffer

3145 Models H2, HG2, I2, IH2, J2, JI2, and K2

(Feat 2) Byte 2, Bits 0-3 Main Storage Size
- 1 = 768k
- 2 = 1024k
- 3 = 1536k
- 4 = 256k
- 5 = 384k
- 6 = 512k
- 7 = 2048k

Bit 4 Reserved

Bits 5, 6 Channels
- 00 = 1
- 01 = 2
- 10 = 3
- 11 = 4

Bit 7 Word Buffer

Diag Word Byte 3

(Feat 3) Byte 3, Bit 0 Model Configuration
- 0 = 3145 Models FED, GFD, H, HG, and I
- 1 = 3145 Models H2, HG2, I2, IH2, J2, JI2, and K2

Bit 1 3215

Bit 2 Second 3210

Bit 3 3145 Models JI2 and K2

Bit 4 Clock Comparator, CPU Timer

Bits 5, 6 Spare

Bit 7 Direct Control

CPU Word

MODE Register

Byte 0

Bit 0 Hard-stop latch (control register 14 bit 0)

Bit 1 Enable I-cycle and Adr Adj Ctrl and expanded local storage.

Bit 2 Enable hardware retry

Bit 3 Full recording mode for single-bit failures in main storage

Bit 4 Full recording mode for single-bit failures in control storage

Bit 5 Threshold mode for single-bit failures in control storage

Bit 6 Reserved

Bit 7 Reserved

Byte 1 CFDAR (Console-file data-address register) Track and sector address used by console file.

Byte 2 LRUM (Least Recently Used Matrix)

Bits 0-7 Indicate which adr adj table register was least recently used.

Byte 3 MATCH

Bits 0-7 Indicate which adr adj table register matches preaddress assembler, (useful only under diagnostic control).

SW Word (Console Switches)

SW0 through SW3 are the rotary console address/data switches:

SW Byte	Console Switches
SW0	AB
SW1	CD
SW2	EF
SW3	GH

PSWCTL Word

Byte 0 EPSWA Bit	Name
0	
1	
2	
3	
4	
5	Translation mode
6	I/O master mask
7	External master mask

Byte 1 EPSWB Bit	Name
0	
1	
2	
3	
4	
5	Machine-check mask
6	Wait state
7	Problem state

Byte 2 MSKA Bit	Name
0	Timer mask
1	Interrupt mask
2	External signal mask
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved

Byte 3 MSKB Bit	Name
0	MPX channel mask
1	Selector channel 1 mask
2	Selector channel 2 mask
3	Selector channel 3 mask
4	Selector channel 4 mask
5	Reserved
6	Reserved
7	Reserved

MISC Word Bytes 2 and 3

2. EC Level: External register 12 byte 2
   The last two digits in the 370 microprogram EC number are plugged. A test is performed before the go-no-go test to determine whether the disk being loaded is at the proper level.
3. Serial Number
   External register 12 byte 3 contains the first two digits of the six-digit serial number. These digits are always plugged as:
   01 = U.S. manufacture
   73 = German manufacture
   82 = Brazilian manufacture

IN Word (Interrupt Register)

An INTA or INTB (interruption) register bit is set on when the corresponding source has an interruption pending *and* the system mask is set to allow such an interruption. Bit names in the INT register are:

Byte 0 INTA Bit	Name
0	Spare
1	Spare
2	Timer
3	External signal
4	System control
5	CPU signal 0
6	CPU signal 1
7	Process stop

Byte 1 INTB Bit	Name
0	Multiplex channel
1	Selector channel 1
2	Selector channel 2
3	Selector channel 3
4	Selector channel 4
5	I/O interrupt
6	Timer update
7*	External

* External is set on if all of the following conditions exist:

1. An external interruption signal is on (that is, from the EXTINT register of the CPU signal from the SCPU register).
2. The external mask bit = 1 (bit 7 of the EPSWA register).
3. For the timer MSKA bit 0 = 1 or for externals 1 through 6, MSKA bit 2 = 1.

Bytes 2 and 3 contain the last four digits of the serial number.

ACB (Address Check Boundary) Register

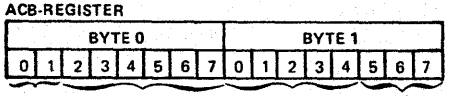
- The ACB-register is a two-byte hardware register that contains boundary information used to check main- and control-storage accesses.
- The ACB-register is loaded at IMPL and is reloaded each time the system reset routine (GRST) is executed.
- The ACB-register is addressed by the external address 0A and can be used as a source or a destination.

The ACB-register is set at IMPL with a specific value determined by the main-storage and control-storage configuration. Certain feature mixes may require additional control storage, above the 32k bytes that are standard. This expansion of control storage is made at the expense of main storage. The movement of the lower control-storage boundary into the main-storage area is done in 2k-byte increments. The change in the boundary location between main and control storage results in a different setting for the ACB-register.

Once the feature mix and control-storage size is established, the 370 microprogram disk generated at the plant contains the proper ACB setting for that configuration.

For each access of main or control storage, a comparison is made between the ACB-register and the M-register. If a main-storage access attempts to address the control-storage area, an address check occurs. If a control-storage access is made to a main-storage location, a machine check occurs.

ACB-REGISTER



Spares

Compared with M1 bits 3-7 on all main-storage accesses.

Compared with M2 bits 0-4 for all storage accesses. (Bits 0 and 1 may be altered for control-storage accesses.)

5=0 Internal storage only
- 67=00-16k boundary
- 67=01-32k boundary
- 67=10-48k boundary
- 67=11-64k boundary

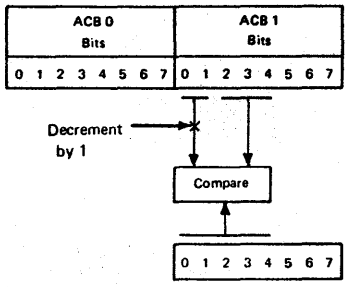
5=1 External storage attached
- 67=00-128k external storage or 1256k external storage
- 67=01-256k external storage or 1768k external storage
- 67=10-512k external storage
- 67=11-768k external storage

ACB SETTINGS

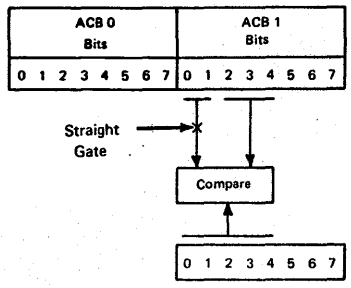
Main Storage	Control Storage	ACB	Main Storage	Control Storage	ACB	Main Storage	Control Storage	ACB	Main Storage	Control Storage	ACB	Main Storage	Control Storage	ACB
112k	32k	01C2	208k	32k	0340	384k	32k	0604	768k	32k	0C06	1536k	32k	1804
110k	34k	01BA	206k	34k	0338	382k	34k	05FC	766k	34k	0BFE	1534k	34k	17FC
108k	36k	01B2	204k	36k	0330	380k	36k	05F4	764k	36k	0BF6	1532k	36k	17F4
106k	38k	01AA	202k	38k	0328	378k	38k	05EC	762k	38k	0BEE	1530k	38k	17EC
104k	40k	01A2	200k	40k	0320	376k	40k	05E4	760k	40k	0BE6	1528k	40k	17E4
102k	42k	019A	198k	42k	0318	374k	42k	05DC	758k	42k	0BDE	1526k	42k	17DC
100k	44k	0192	196k	44k	0310	372k	44k	05D4	756k	44k	0BD6	1524k	44k	17D4
98k	46k	018A	194k	46k	0308	370k	46k	05CC	754k	46k	0BCE	1522k	46k	17CC
96k	48k	0182	192k	48k	0300	368k	48k	05C4	752k	48k	0BC6	1520k	48k	17C4
94k	50k	017A	190k	50k	02F8	366k	50k	05BC	750k	50k	0BBE	1518k	50k	17BC
92k	52k	0172	188k	52k	02F0	364k	52k	05B4	748k	52k	0BB6	1516k	52k	17B4
90k	54k	016A	186k	54k	02E8	362k	54k	05AC	746k	54k	0BAE	1514k	54k	17AC
88k	56k	0162	184k	56k	02E0	360k	56k	05A4	744k	56k	0BA6	1512k	56k	17A4
86k	58k	015A	182k	58k	02D8	358k	58k	059C	742k	58k	0B9E	1510k	58k	179C
84k	60k	0152	180k	60k	02D0	356k	60k	0594	740k	60k	0B96	1508k	60k	1794
82k	62k	014A	178k	62k	02C8	354k	62k	058C	738k	62k	0BBE	1506k	62k	178C
80k	64k	0142	176k	64k	02C0	352k	64k	0584	736k	64k	0B86	1504k	64k	1784
160k	32k	0281	256k	32k	0403	512k	32k	0805	1024k	32k	1007	2048k	32k	2005
158k	34k	0279	254k	34k	03FB	510k	34k	07FD	1022k	34k	0FFF	2046k	34k	1FFD
156k	36k	0271	252k	36k	03F3	508k	36k	07F5	1020k	36k	0FF7	2044k	36k	1FF5
154k	38k	0269	250k	38k	03EB	506k	38k	07ED	1018k	38k	0FEF	2042k	38k	1FED
152k	40k	0261	248k	40k	03E3	504k	40k	07E5	1016k	40k	0FE7	2040k	40k	1FE5
150k	42k	0259	246k	42k	03DB	502k	42k	07DD	1014k	42k	0FDF	2038k	42k	1FDD
148k	44k	0251	244k	44k	03D3	500k	44k	07D5	1012k	44k	0FD7	2036k	44k	1FD5
146k	46k	0249	242k	46k	03CB	498k	46k	07CD	1010k	46k	0FCF	2034k	46k	1FCD
144k	48k	0241	240k	48k	03C3	496k	48k	07C5	1008k	48k	0FC7	2032k	48k	1FC5
142k	50k	0239	238k	50k	03BB	494k	50k	07BD	1006k	50k	0FBF	2030k	50k	1FBD
140k	52k	0231	236k	52k	03B3	492k	52k	07B5	1004k	52k	0FB7	2028k	52k	1FB5
138k	54k	0229	234k	54k	03AB	490k	54k	07AD	1002k	54k	0FAF	2026k	54k	1FAD
136k	56k	0221	232k	56k	03A3	488k	56k	07A5	1000k	56k	0FA7	2024k	56k	1FA5
134k	58k	0219	230k	58k	039B	486k	58k	079D	998k	58k	0F9F	2022k	58k	1F9D
132k	60k	0211	228k	60k	0393	484k	60k	0795	996k	60k	0F97	2020k	60k	1F95
130k	62k	0209	226k	62k	038B	482k	62k	078D	994k	62k	0F8F	2018k	62k	1F8D
128k	64k	0201	224k	64k	0383	480k	64k	0785	992k	64k	0F87	2016k	64k	1F85

Note: The ACB setting for each 370 microprogram load may be found in the module chart in the back of the microlisting. Look up address FF08, the ACB setting is in bytes 0 and 1.

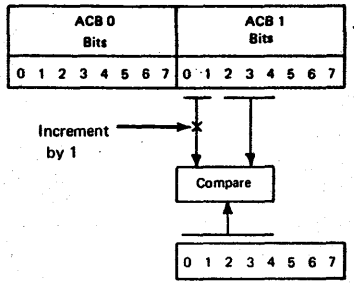
ACB Compare For Control Storage Access



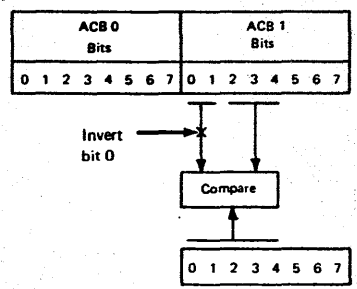
Main Storage Size 112k



Main Storage Size 160k



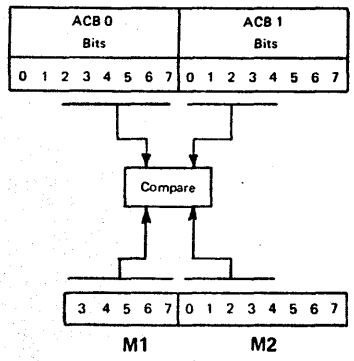
Main Storage Size 208k



Main Storage Size 256k and Above

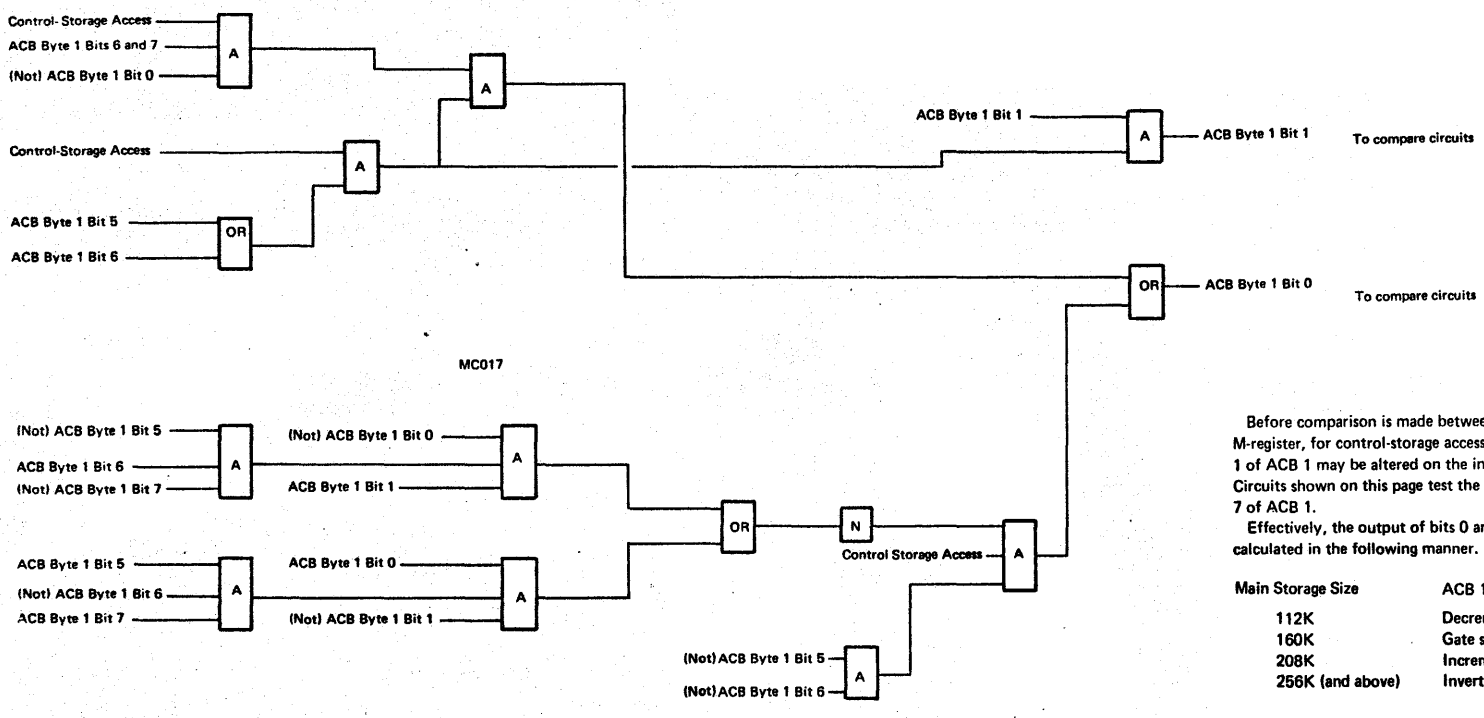
If the comparison indicates that the ACB value is more than the M2 value, a machine-check condition is specified.

ACB Compare for Main Storage Access



If the comparison indicates that the ACB value is equal to, or less than, the M-register value, an address check occurs.

ACB Byte 1, Bit 0 and 1 Gat[redacted]o Compare Circuits (Control Storage Only)



Before comparison is made between the ACB-register and the M-register, for control-storage accesses, the output of bits 0 and 1 of ACB 1 may be altered on the input to the compare circuits. Circuits shown on this page test the status of bits 0, 1, 5, 6, and 7 of ACB 1.

Effectively, the output of bits 0 and 1 to the compare circuits is calculated in the following manner.

Main Storage Size	ACB 1 bits 0 and 1
112K	Decrement by 1
160K	Gate straight
208K	Increment by 1
256K (and above)	Invert bit 0.

Note: Refer to "ACB and M1-Registers" in the 3145 *Processing Unit Maintenance Diagrams* manual SY24-3580, for high-level diagram of ACB compare.

SYS (System) Register

The system register gives the status or condition of the processor SYS reg is an exteranl hardware register located at word address 05.

Byte 0	
Bit 0	Machine-check interruption pending
1	Retry routine
2	Machine-check routine
3	Documentary console 2
4	Log present
5	Sub-block protection mode
6	Selector channel Start I/O latch
7	Force module 0 to LSCS
Byte 1	
Bit 0	Address contents
1	CPU interrupt force
2	SAR interrupt force
3	PSW restart
4	
5	System control interrupt
6	Timer interrupt force
7	Reserved
Byte 2	
Bit 0	Enable clear switch
1	IMPL
2	Load file wait bit
3	CE key in CE mode
4	00 System reset – **10 subsystem load (IPL)**
5	01 Power-on reset – **11 system load (multiprocess)**
6	Error in stop word
7	Instruction processing latch
Byte 3	H-Register
Bit 0	Machine-check trap
1	Retry trap
2	CPU high trap
3	Integrated file adapter (IFA) if installed Selector channels 1,2,or 3 if no IFA
4	Selector channels 1,2,or 3 if IFA installed Selector channel 4 if no IFA
5	Multiplexer channel
6	IFA if installed
7	Store-display

Priority Operations

Priority operations, which may be related to the current operation, can cause delay of the current microprogram routine. Most (not all) of the various priority operations are initiated by traps. A *trap* is basically a circuit-forced branch out of the current microprogram routine to a priority routine. After the priority routine is completed, a return can be made to the interrupted routine so that its execution can continue.

The interrupted routine is delayed further when several priority operations occur at the same time. Or, in some cases, the interrupted routine may be ended by the occurrence of a priority operation. For example, if an instruction address that specifies an unavailable main-storage location is used, an address-check priority operation occurs. The microprogram routine, in which the invalid address is used, is discontinued.

A hierarchy of execution preference exists within the priority-operation structure. Execution preference is exhibited in two instances:

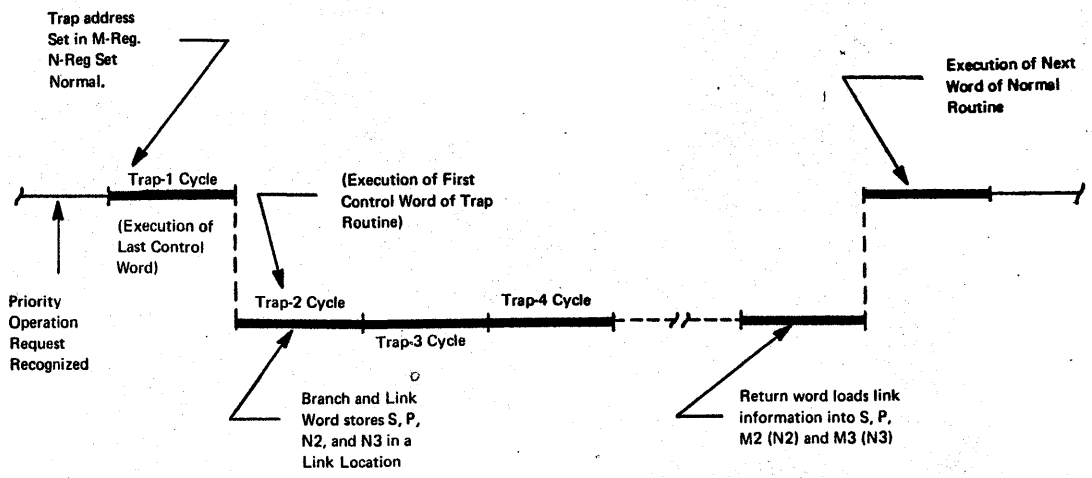
1. Circuit requests for two or more priority operations occur in the same CPU cycle. The highest-priority operation is executed first; the next highest second, etc.
2. During execution of a priority operation, a request for a *higher*-priority operation occurs. The higher-priority operation is executed; the lower-priority operation is delayed until completion of the higher-priority operation, subject to the rules of execution of priority operations specified later in this section.

Selector-channel data-transfer operations have the highest priority but do not use the trapping mechanism. All of the other priority operations use the trapping mechanism, which functions in the following manner.

1. In a CPU cycle, a priority operation request is recognized.
2. The control-storage address of the first control word in the priority microprogram routine is set into the M-register in what is known as the *trap-1 cycle*, the cycle following the one in which the request is recognized. (The normal next-control-word address, generated by execution of the word in progress, is set into the N-register.) The address set into the M-register is forced by circuitry and depends upon the priority operation for which the request is made.
3. The first word of the priority operation is read out of control storage and set into the C-register. Normally, this first word is a branch and link word.
4. The branch and link word stores the contents of S, P, N2, and N3 into a link location in local storage. (N2 and N3 contain the address of the word that would have been executed next if the trap had not occurred.) The cycle in which this occurs is called the *trap-2 cycle*.
5. The priority routine is executed. Normally, the last word in the priority routine is a return word. This word loads the link information back into S, P, M2 (N2) and M3 (N3).
6. Execution of the interrupted routine is resumed at the control word specified by the link (return) address in M2 and M3.

Trap Operation

TRAP-1 CYCLE	TRAP-2 CYCLE	TRAP-3 CYCLE	TRAP-4 CYCLE
Set By: 1. (Not) Inhibit Traps. 2. Request 0-9 3. 0-Time 4. (Not) Stg. 1 cycle	*Set By:* 1. Trap-1 INLK latch on. 2. 0-45 Time	*Set By:* 1. Trap-2 INLK latch on. 2. 0-45 Time	*Set By:* 1. Trap-3 INLK latch on. 2. 0-45 Time
Purpose: 1. Prevents any additional traps. 2. Sets TR 1 INLK latch at 90-135 time. 3. Prevents normal set to M-Reg. 4. Forces trap address to M2 & M3. 5. N2 and N3 set Normal 6. Set Block SPTL. 7. Execute last word of interrupted routine.	*Purpose:* 1. Forces module address to M2 (N2 buffer). 2. Normal M3 (N3 buffer) addr update. *3. BAL operation stores S, P, N2, N3 in link area (values that were set during the trap-1 cycle). 4. TR 2 INLK latch set at 90-135 time. *BAL is normally the first word of a trap Routine.	*Purpose:* 1. Normal address update to M2 2. TR 3 INLK latch set at 90-135 time. *Trap-3 and 4 cycles prevents continuous looping of Trap-1 and 2 cycles if an error is in the SPTL area.	*Purpose:* 1. Reset Block SPTL (Set at TR 1 cycle).



H-Register

Many priority operations cause an H-register bit to be set on in the trap-2 cycle. The priority operations and associated H-register bits are:

Operation	H-Register Bit	Trap Address
Selector share cycles	none	none
Machine check without I/O	H0	–
a. Normal		D000
b. H0 is already on		D004
c. One or more machine checks have already occurred (SYS0 Bit 2 = 1)		D008
d. H0 and SYS0 Bit 2 are already on		D00C
Machine check with I/O	H0	–
a. Normal		D010
b. H0 is already on		D014
c. One or more machine checks have already occurred (SYS0 Bit 2 = 1)		D018
d. H0 and SYS0 Bit 2 are already on		D01C
Retry	H1	–
a. Normal		D200
b. H1 is already on		D204
c. A retry trap operation is in progress (SYS0 Bit 1 = 1)		D208
d. H1 and SYS0 Bit 1 are already on		D20C
CPU High	H2	–
a. Set IC		D300
b. CA trap		D304
c. Address contents		D308
d. System reset		D30C
Integrated File Adapter	H3	–
a. Mini-Op end		D128
b. Error end		D12C
c. Index		D124
d. Gated Attn or D ADR		D120
Selector Channels 1, 2, 3 (without IFA)	H3	
a. Exceptional status trap		D120
b. Chaining (command or data)		D124
c. UCW handling		D128
d. D ADR trap		D12C
Selector Channels 2, 3/4 (with IFA:SX2, 3; without IFA:SX4)	H4	
a. Exceptional status trap		D100
b. Chaining (command or data)		D104
c. UCW handling		D108
d. D ADR trap		D10C
Multiplexer channel	H5	D400
Integrated File Adapter	H6	
a. Return low		D480
b. Unused		D484
c. Unused		D488
d. Diagnostic		D48C
Store/display	H7	–
a. Store/display		D840
CPU low without I/O	None	–
a. Spare		–
b. Storage protect		D804
c. Address check		D808
d. ADR ADJ exception		D80C
CPU low with I/O	None	–
a. Spare		–
b. Storage protect		D814
c. Address check		D818
d. Spare		–
Scan/Clear	None	–
a. Scan storage		D380
b. Clear storage		D384

The following rules apply to execution of priority operations:

1. A selector share cycle can break into any operation except the first cycle of a storage control word and during an ECC retry of a control word.
2. Any trap priority operation can not break into either:
   a. the first cycle of a storage word operation, or
   b. a trap-2 cycle operation. (That is, a trap-2 cycle can not be a trap-1 cycle for another trap.)
3. If H1 is on, all other priority operations (except H0—machine check—and selector share cycles) are prevented. If, however, a diagnostic trap occurs, it is executed, even though H1 is on. Also, if the system is in a single-cycle mode of operation, a store/display trap can be executed even if H1 is on.
4. If H3 is on, an H3, H4, H5, or H6 trap cannot be taken. If H4 is on, an H4, H5, or H6 trap cannot be taken. If H5 is on, H5 or H6 cannot be taken. If H6 is on, an H6 trap cannot be taken. In any of these cases, the H3, H4, H5, or H6 trap remains pending until after H3 (or H4 or H5) is turned off.
5. Selector share cycles can delay execution of other traps for a number of cycles, depending upon the rate at which share cycles occur.
6. Discounting the effects of the various non-H-Reg priorities (share cycles, CPU low, scan/clear), the following hierarchy applies.

H-Reg Bit	Blocks Trap Request for H-Reg Bit
H0	None
H1	H2,3,4,5,6,7
H2	H2
H3	H3,4,5,6
H4	H4,5,6
H5	H5,6
H6	H6
H7	H7

Share Cycle Priority Operation (Applies to: Selector Channel, Block Multiplexer, IFA).

Priority operation for selector-share cycles is as follows:

Without IFA

1. The priority sequence is selector channel 1, 2, 3, and 4.
2. A share-request for selector channel 4 is taken if no other request is pending.

With IFA

1. The priority sequence is selector channel 2, IFA, and selector channel 3.

Machine-Check Priority Operation (H0)

A machine-check trap occurs (if allowed by the machine-check bit in the PSW) because a series of retry operations has been unsuccessful. The number of retry attempts is determined by a hardware counter. Basically, a machine-check trap occurs either because errors are occurring faster than can be handled or because a hard error cannot be successfully retried.

An attempt is made to form logout information and initiate a machine-check interruption (depending upon the value of the machine-check bit in the PSW). The validity of such logout information may be unpredictable if the machine-check trap is called for.

Retry Priority Operation (H1)

The retry routine is entered through the retry priority operation (trap). The retry priority operation occurs when any machine check occurs if the retry counter is not full, retries are not masked off, and system register byte 2 bit 6 (indicates stop word error) is off. Depending upon the nature of the error and the word type, the error may be detected during execution of the failing microprogram word (Type 1), during execution of the following word (Type 2), or may be detectable but uncorrectable (Type 3).

CPU High Priority Operation (H2)

System Reset Microprogram

The system reset microprogram is executed after a circuit system reset has been performed. This action is initiated by operating:

1. The system reset key
2. The load key.

System reset causes various CPU registers and controls to be reset.

Integrated File Adapter High Priority Operations (H3)

Four trap addresses are provided for Mini-Op End, for Error End, for Index, and for Gated Attention or D ADR.

Selector Channels or Block-Multiplexer Channels 1, 2 and 3 (H3)

When IFA is not present, four trap addresses are provided for channels 1, 2, and 3: for Exceptional Status Trap, for Chaining (command or data), for UCW Handling, and to protect the next entry of the D ADR list.

Selector Channels and Block-Multiplexer Channels 2, 3/4 (H4)

Four trap addresses are provided: for Exceptional Status Trap, for Chaining (command or data), for UCW Handling, and to protect the next entry of the D ADR list. When IFA is preset, this trap is shared by channels 2 and 3. For non-IFA, this trap is for the sole use of selector channel 4.

Multiplexer Channel (H5)

This trap is for the sole use of the multiplexer channel for handling data, status, and chaining functions.

Integrated File Adapter Low Priority Operations (H6)

Four trap addresses are provided for Return Low, and for Diagnostics. The other two are not assigned.

Store/Display (H7)

Store/Display pertains to system control panel operations.

CPU Low (No H-Register Bit)

Address Check
This trap occurs when an access to an unavailable main-storage area is attempted.

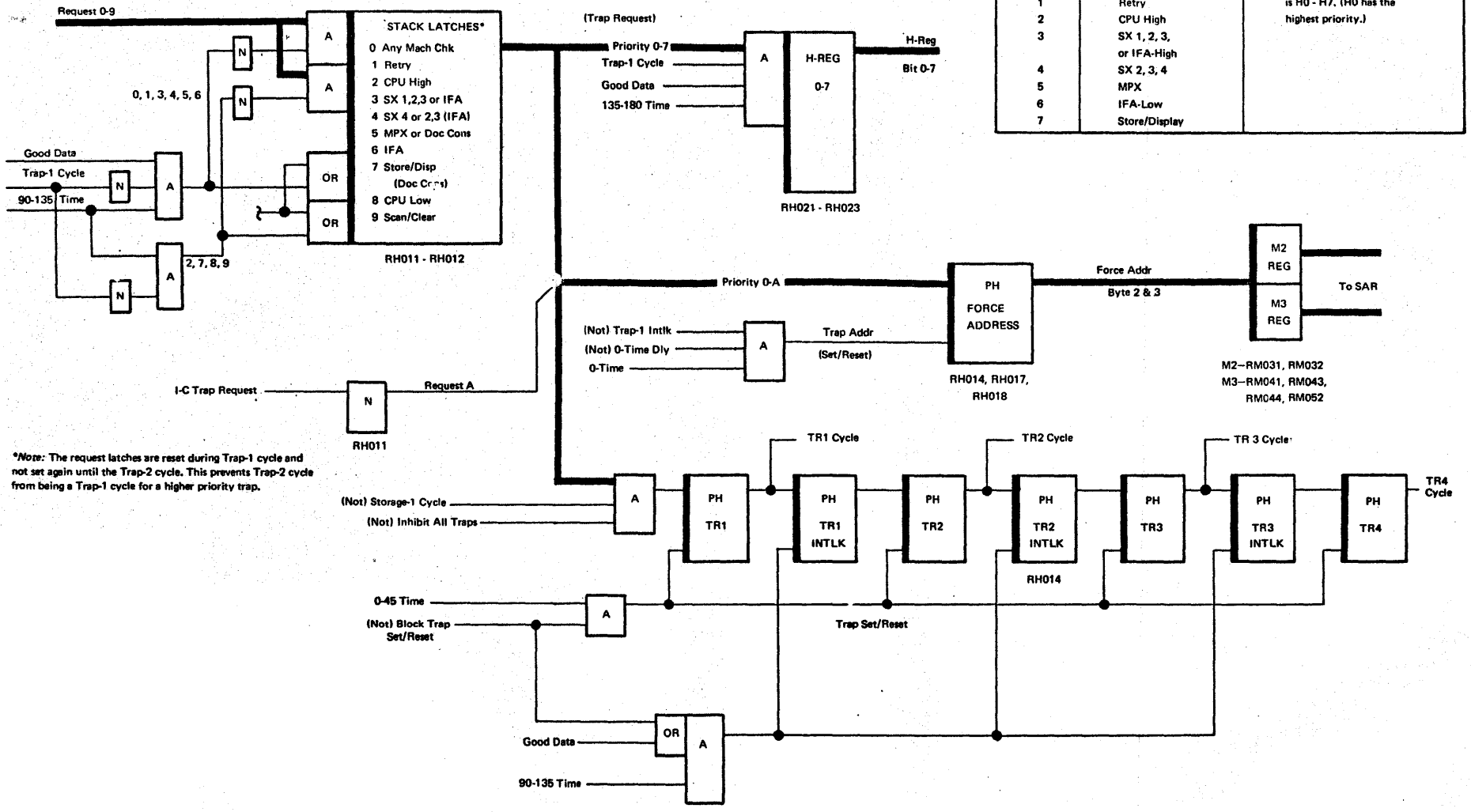
Storage Protection
This trap occurs because of a storage-protection violation.

Address Adjustment Exception
This trap is used with DOS emulator.

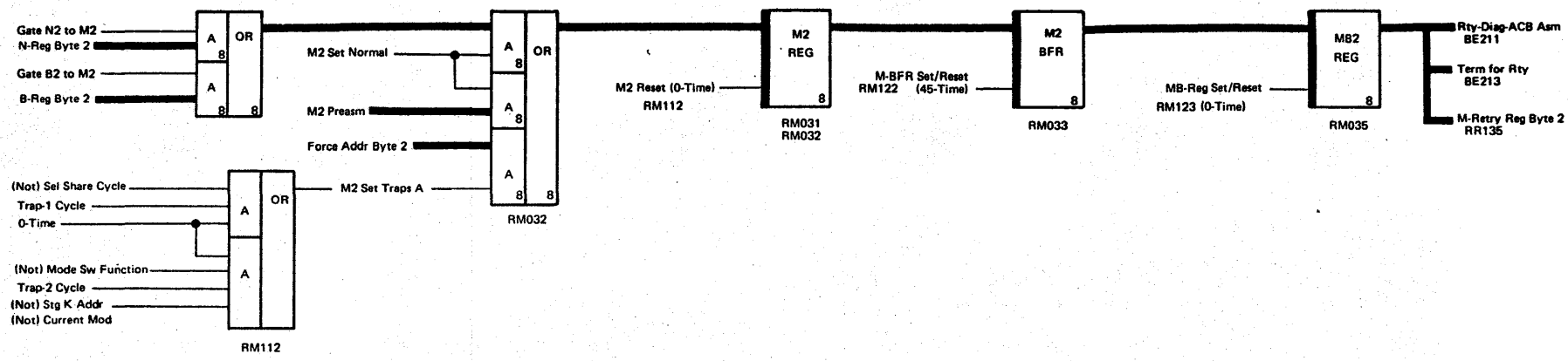
Scan/Clear (No H-Register Bit)
These traps are used for a clear-storage and a scan-storage operation.

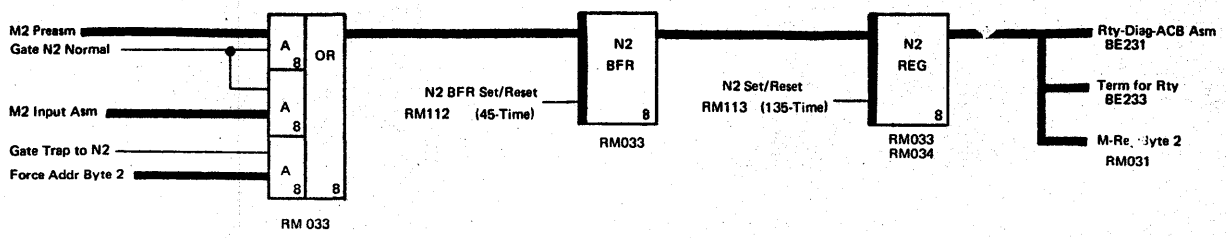
Priority and Trap Controls

H-REG	OPERATION	
Bit 0	Mach Check	*Note:* The priority sequence is H0 - H7. (H0 has the highest priority.)
1	Retry	
2	CPU High	
3	SX 1, 2, 3, or IFA-High	
4	SX 2, 3, 4	
5	MPX	
6	IFA-Low	
7	Store/Display	

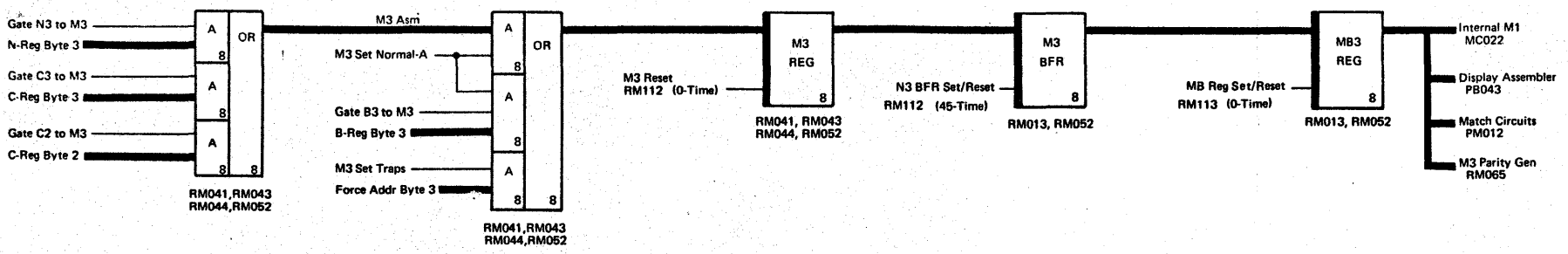


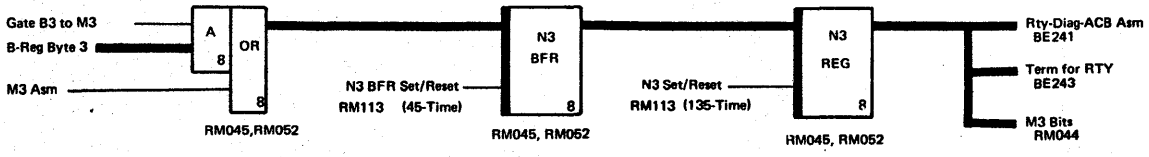
M2 Gating (Traps)





M3 Gating (Traps)





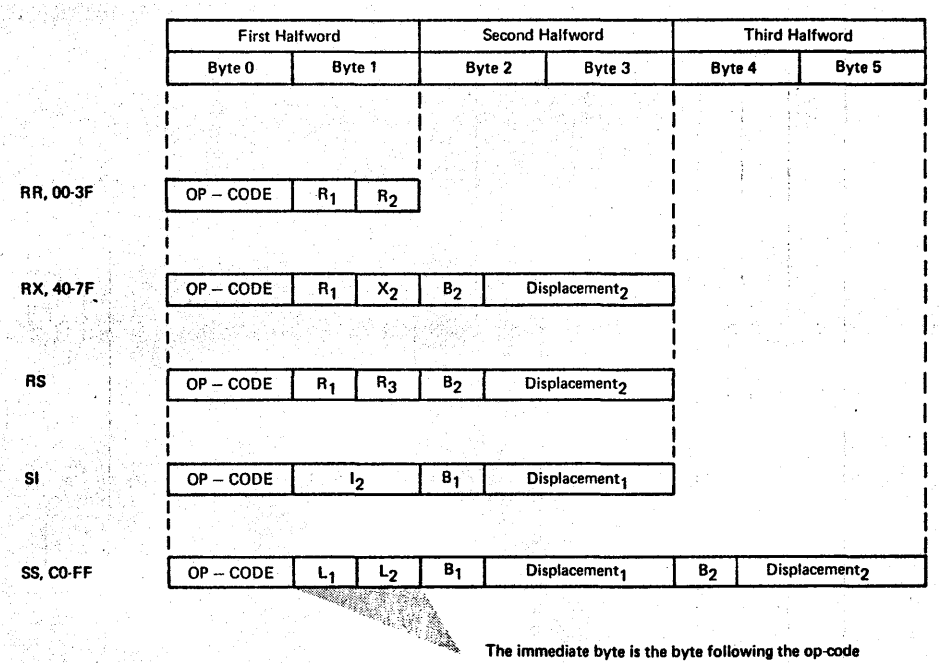
I–CYCLES

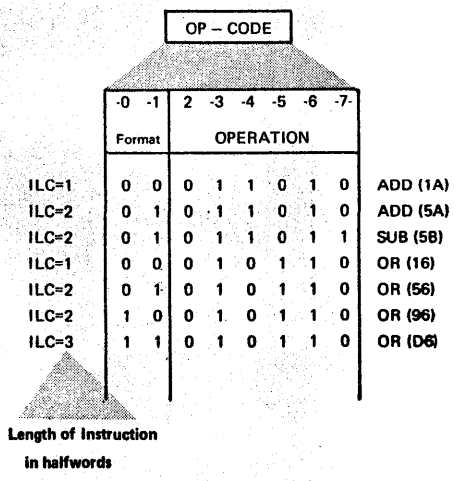
Processing a single software instruction may be divided into two parts: the I (instruction) phase and the E (execution) phase. Instructions are defined to be in different groups according to their format, length, and general form of execution.

The I-phase of processing performs the following basic functions:

- Fetch instruction
- Initialize the CPU facilities for the completion of the processing.

During the E-phase, the CPU performs the unique functions specified by the instruction op-code.





I-PHASE FUNCTIONS

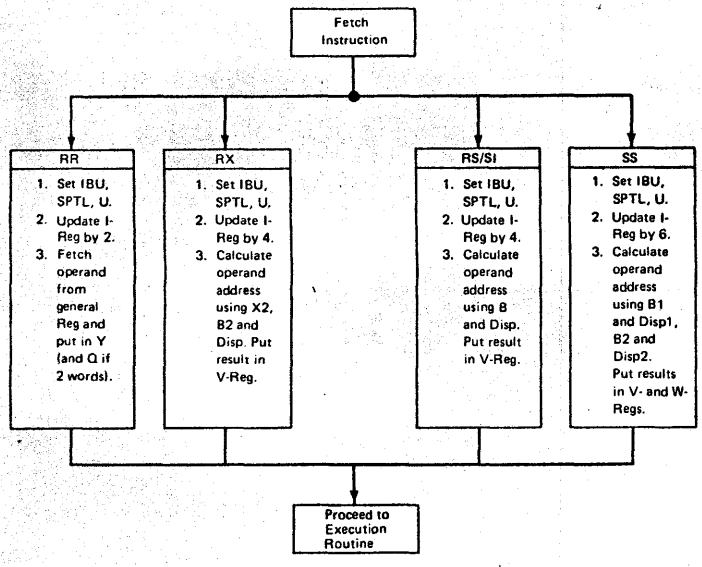
The initialization of CPU facilities for the E-phase depends partially upon instruction type. All instructions require an updating of the instruction counter, the setting of the specified CPU Regs, and a branch to the start of the execution routine. In addition, some instructions require the fetching of the second operand from a general register, or the calculation of operand addresses.

At this point, some observations may be made about the I-phase functions. For example, the RX and RS/SI functions are very similar. In fact, during the I-phase, an RS/SI instruction is handled exactly the same way as an RX instruction with the X2 field equal to zero. Also, some functions are identical, with only the data value depending upon the format and op code (SPTL, U, and I update).

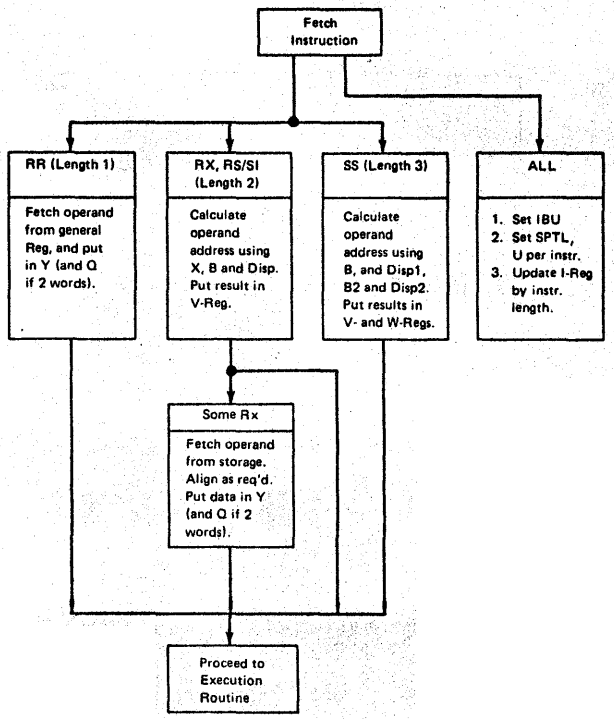
Note also that the E-phase for some instructions is identical; such as, AR and A, NR and N. The difference between these RR and RX types of instructions occurs only in the source of the second operand (general register or storage). Saving some control-storage words and time is possible by including the operand fetch as an I-phase function for such RX format op codes.

The I-phase functions may now be illustrated as follows:

I-PHASE FUNCTIONS BY INSTRUCTION



I-PHASE FUNCTIONS BY INSTRUCTION LENGTH



Hardware Functions

Each software instruction processed requires performing the previously mentioned I-phase functions. Minimizing this time and thereby reducing the time required to process a given instruction is desirable. To minimize the number of machine cycles required during the I-phase (that is, the I-cycles) some functions are performed by hardware. Additionally, some other characteristics of the machine are more fully exploited by hardware means.

First consider the previously defined I-phase functions which apply to all instructions. Hardware is used to perform the setting of IBU, SPTL, U, and the I-Reg update. These functions do not require microwords to be performed; hence, they do not require any additional time during I-cycles.

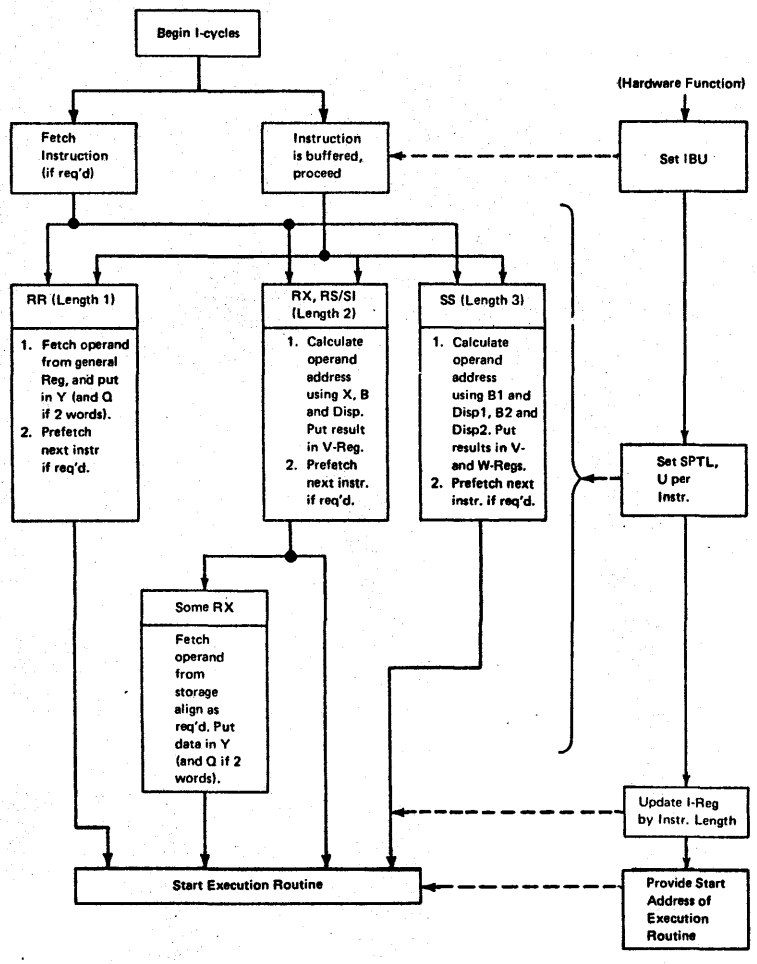
Now consider the function Proceed to Execution Routine. To perform a hardware-forced branch, define the starting control-storage address of each execution routine as a function of instruction op code. The hardware branch on the op code does not require any additional time, because no microword is used to perform the branch and module switch.

The interface between storage and the CPU provides a doubleword transfer of eight bytes of data. During a read type of microword, the SDBO assembler provides the selection of the addressed word (halfword, or byte) from the doubleword actually read. The I-cycle hardware provides for buffering the entire doubleword from storage, via a time-slotting of data from SDBO to EBI. When an instruction is fetched from storage, the addressed word is routed from SDBO to the I-buffer, via EBI, during the normal destination time in storage-2 cycle. During the next cycle time, the odd word is gated to EBI, and placed in the I-buffer. This time-slot action occurs when no decrement count function is specified by the storage microword. Therefore, up to two words of data from the instruction stream can be buffered when fetching one instruction from storage. Upon completion of the instruction being processed, the next instruction may be available in the buffer and, therefore, need not be fetched from storage.

A savings in processing time becomes obvious, especially if the doubleword being buffered represents four RR instructions. The concept of buffering a portion of the instruction stream can now be extened to include pre-fetching. Although the buffer does speed subsequent instructions, the fetch of the first (current) instruction does require some time. Having the current instruction resident in the buffer is always desirable. To get to this I-buffer condition, the instruction must have been obtained at some point during the previous instruction. This function of reading the next instruction from storage to the I-buffer is termed prefetching and is performed during I-cycles.

As described in "Expanded Local Storage," the TR Reg always contains a value representing the next doubleword address beyond the current I-Reg value. The TR-Reg is always used as the storage address during a prefetch, and the SDBO time-slotting is forced to provide the even word; then the odd word. This guarantees that the I-buffers are loaded with sequential data.

The I-phase functions at this point are:



Microcode-Hardware Functions

The complexity of the I-cycle functions are increasing. On a previous diagram (CPU 49) selecting one of three paths after fetching the instruction was necessary. Here also required is to:

- Determine source of instruction (storage vs. buffer).
- Select path if instruction is in the I-buffer.
- Determine whether a prefetch is required.

Also minimizing the time required to perform each of these functions and the basic I-phase functions is desirable.

Microcode branch operations requires CPU time. I-cycle hardware can force a control-storage address to the M-Reg. (for example branch on the op code). This facility is expanded to include all addressing within I-cycles. The I-cycle hardware provides the starting address of the I-cycle routine to the M-Reg, as a function of I-buffer status, instruction format and prefetch requirement. When the CPU encounters a RTN word (to I-cycles) this address is set into the M-Reg and the I-phase of the instruction begins. Thereafter, except for some parts of the RX-align routine, the I-cycle hardware provides the next control-storage address and a gating signal to the M-Reg, until the execution routine has begun. The I-cycle hardware then initializes for the start of the next I-phase.

The minimization of time spent performing the basic I-phase functions requires more hardware control. The I-cycle hardware controls the data inputs and setting of the SPTL Regs and uses this facility to select general registers from local store. By setting the P-Reg to a value of 02 or (62) and gating a portion of the instruction to the L-Reg (R2, X, or B), the microcode can indirectly address any general register, including floating-point registers. This gating to SPTL is done by hardware, and is, therefore, transparent to the microcode. Note that setting SPTL to the desired value one machine cycle before use is necessary.

Furthermore, the I-cycle hardware can force and/or block gating of data through a portion of the Expanded Local Store. This capability is utilized as follows: a microword is executed performing the arithmetic operation of V = LL+V. During the previous machine cycle, a value is set into the P- and L-Regs of: P = 02, and L = R1B2 (for an RX format instruction). The underlined data value gives the general register specified by the B2 field of the instruction as the data source of the A-Reg. Although the microword is attempting to source the V-register, the I-cycle hardware blocks this Expanded Local Store source, and forces the Disp 2 field through the gating to the B-register. The microword function of adding the A- and B-Regs is then completed, with the result destined to the V-register. Thus, the microcode and I-cycle hardware are combined to perform the function of:

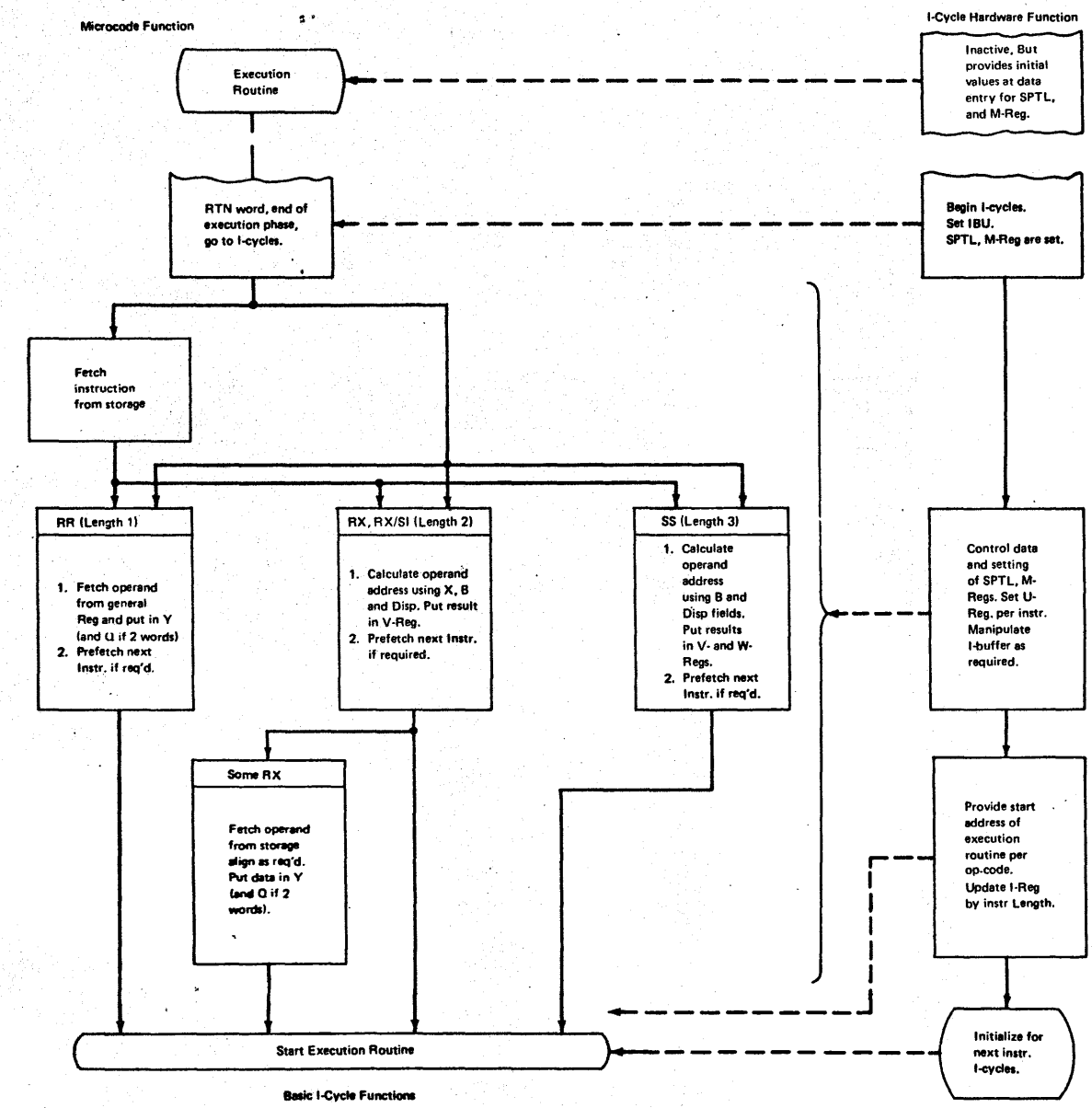
V = Base + Displacement

Most of the I-cycle microcode/hardware functions is performed this way.

Another significant interaction of microcode and hardware occurs when prefetching and calculation of an operand address are performed simultaneously. The microword executed during a prefetch is of the form: RDW LL ADJ, V + 4. First, the V-Reg is blocked as an address source. Then, the TR-Reg is substituted as the address source for the M-Reg, with gating performed via the PAA, and I-cycle/ADR ADJ path to the M-Reg. At the same time, the Disp field is gated from the I-buffer through the Expanded Local Store to the B-Reg. The A-Reg data source is an indirectly addressed general register, as previously described. This form of microword normally performs an update of the B-register value; however, this function is blocked and changed to an A + B operation. The function, V = Base + Displacement, is, therefore, performed during the storage-1 cycle. During storage-2 cycle, the destination of data to local store is blocked, and SDBO is time-slotted to the I-buffer, as previously described.

Because the I-cycle hardware and microcode are expected to operate simultaneously, the microcode must consist of specific microwords at fixed addresses. This is obvious because the hardware is providing the control-storage address for the microwords, and then performing hardware functions coincident with the microword execution. What has not been obvious is how the hardware remains in sync with the microcode. This function is performed by routing the M-Reg output back to the I-cycle hardware. Thus, when the M-Reg contains a value corresponding to the control-storage location of an I-cycle microword, the I-cycle hardware can determine what functions are to be performed at the next 0-time (that is coincident with the microword execution).

This introduction has provided the basic functional concepts of the hardware I-cycles. Significant omissions include trapping, share cycles, correction cycles, error conditions, etc. The hardware description in the rest of this section has sufficient explanation for these conditions. The basic I-cycle functions are described "Microcode-Hardware Relationship," page CPU 51.

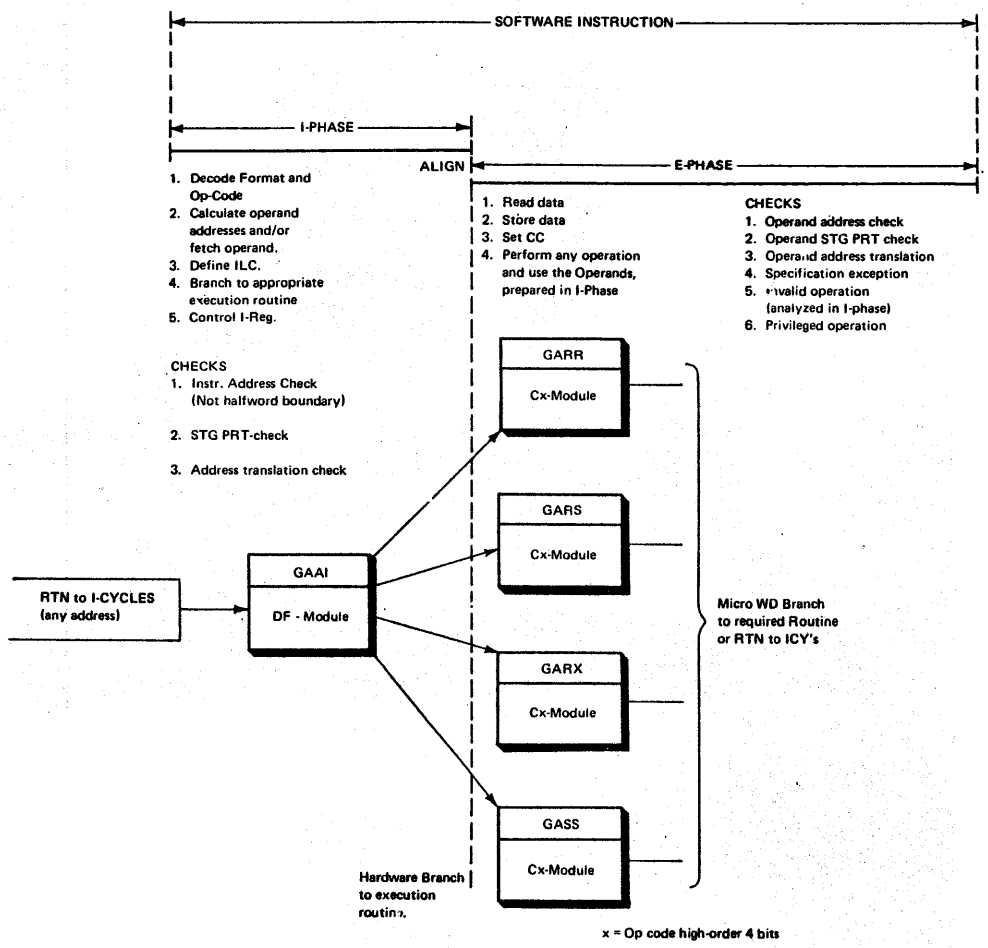

Microcode Function
I-Cycle Hardware Function
Execution Routine
Inactive, But provides initial values at data entry for SPTL, and M-Reg.
RTN word, end of execution phase, go to I-cycles.
Begin I-cycles. Set IBU. SPTL, M-Reg are set.
Fetch instruction from storage
RR (Length 1)
1. Fetch operand from general Reg and put in Y (and Q if 2 words)
2. Prefetch next Instr. if req'd.
RX, RX/SI (Length 2)
1. Calculate operand address using X, B and Disp. Put result in V-Reg.
2. Prefetch next Instr. if required.
SS (Length 3)
1. Calculate operand address using B and Disp fields. Put results in V- and W-Regs.
2. Prefetch next Instr. if req'd.
Control data and setting of SPTL, M-Regs. Set U-Reg. per instr. Manipulate I-buffer as required.
Some RX
Fetch operand from storage align as req'd. Put data in Y (and Q if 2 words).
Provide start address of execution routine per op-code. Update I-Reg by instr Length.
Start Execution Routine
Initialize for next instr. I-cycles.
Basic I-Cycle Functions

I-Cycles Microcode Module Assignment

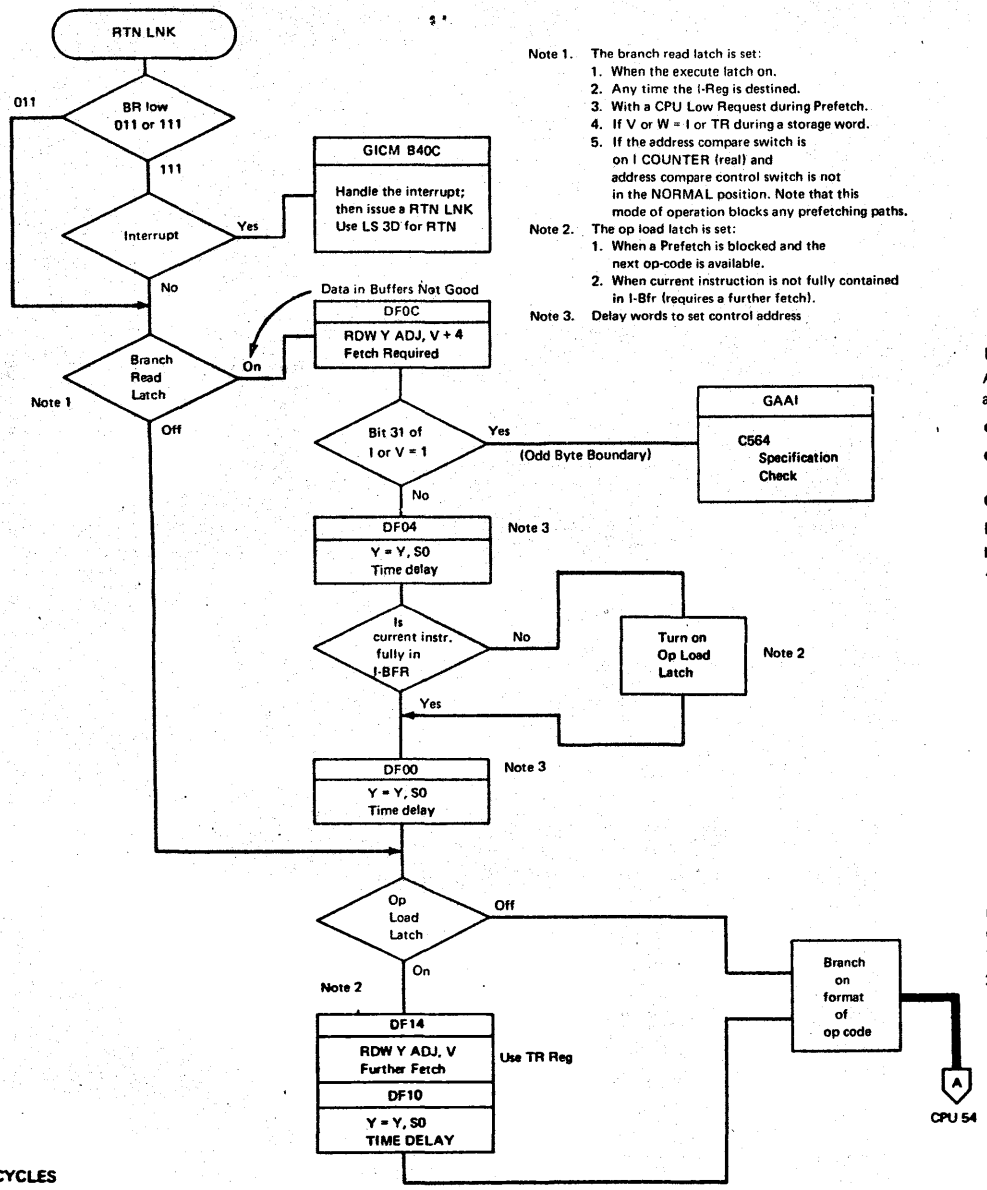
The I-cycle microcode routine (GAAI) resides in the DF module of control storage. The control-storage address, hardware functions generated, and microcode are directly related. Specifically, the hardware generates control signals (and a next control-storage address) from the contents of the M-register, so as to be active (and coincident) when the control word executed from that address.

The DF module is active if bit 1 of the Mode Reg (external address 08) is on.

MICROPROGRAM MODULE ASSIGNMENT



I-Cycles Microcode and Control Hardware-Loading of I-Buffers



Note 1. The branch read latch is set:
1. When the execute latch on.
2. Any time the I-Reg is destined.
3. With a CPU Low Request during Prefetch.
4. If V or W = I or TR during a storage word.
5. If the address compare switch is on I COUNTER (real) and address compare control switch is not in the NORMAL position. Note that this mode of operation blocks any prefetching paths.

Note 2. The op load latch is set:
1. When a Prefetch is blocked and the next op-code is available.
2. When current instruction is not fully contained in I-Bfr (requires a further fetch).

Note 3. Delay words to set control address

I-cycle Entry

I-cycles may be entered (and the I-cycle controls enabled) by two RTN words:

- Conditional: testing for interrupt. C3 bits 5-7 are 111. If an interrupt is pending, the RTN word is executed normally, and does not go to the I-cycle routine. If no interrupt, the return is to I-cycles.
- Unconditional: goes directly to I-cycles when C3 bits 5-7 are 011.

The M-register controls the decode of the return and accesses control storage using the address inputs to the M-register from I-cycles. The I-cycles inputs to the P- and L-registers are also gated. Data is maintained on these inputs by the I-cycle hardware when not in I-cycles (except when performing a storage address adjustment access).

Initial I-Cycle Address

All I-cycle addresses are in the DF module. The initial address for a given instruction is determined by:

- The instruction itself, and
- The requirement for prefetching the next instruction.

Current Instruction Not Fully Contained in I-Bfrs

First, consider the two cases when the instruction is not completely contained in I-Bfr.

1. The first case (and highest address priority) occurs when the branch read latch is on. This occurs not only from the most obvious case of macroprogram branch, (detected by the I-register being loaded from EBI), but also from:
   - Program modification (detected by storing within the present, or next, storage doubleword address as compared to the I-register).
   - A prefetch condition that was not filled during the last I-cycle phase.
   - Blocking a trap during a prefetch in the last I-cycle phase.
   - Being in real instruction address compare mode.
   - Performing an execute macro-instruction.

   These examples are summarized as "whenever the instruction must be read from storage." When the branch read latch is on, all other initial addresses are blocked and an address of DF0C is sent to the M-register input.
2. The second case (and next highest address priority) occurs when the op load latch is on. The condition for setting this latch occurs when part of the instruction is in I-Bfr, but the remainder is in storage. The latch is set during the previous I-cycle phase if it is determined that a prefetch is required, but blocked, and only part of the instruction is in I-Bfr. When this latch is on, all other initial addresses are blocked and an address of DF14 is sent to the M-register input.

Current Instruction Fully Contained In I-Bfr

With the instruction fully contained in I-Bfr, the next condition considered for an initial address is prefetching. The rule for prefetching is that a prefetch is performed if the:

- Present instruction ends at a doubleword boundary (the next op code is not available) or,
- Next instruction crosses a doubleword boundary. This is determined by hardware as a function of:
  a. The halfword address of the present instruction within the doubleword,
  b. The length of the present instruction, and
  c. The length of the next instruction.
- A prefetch is blocked if the present instruction is decoded to be a branch type, or a special addressing case of an SS instruction, at an address of 6 or E

The initial address is gated to the M-register according to the following table:

Instruction	Without Prefetch	With Prefetch
RR (but not flt. pt. long)	DF20	DF34
RR (flt. pt. long only)	DF24	DF3C
RX (double index only)	DF4C	DF5C
RS, SI, RX (not double index)	DF48	DF58
SS	DF6C	DF7C

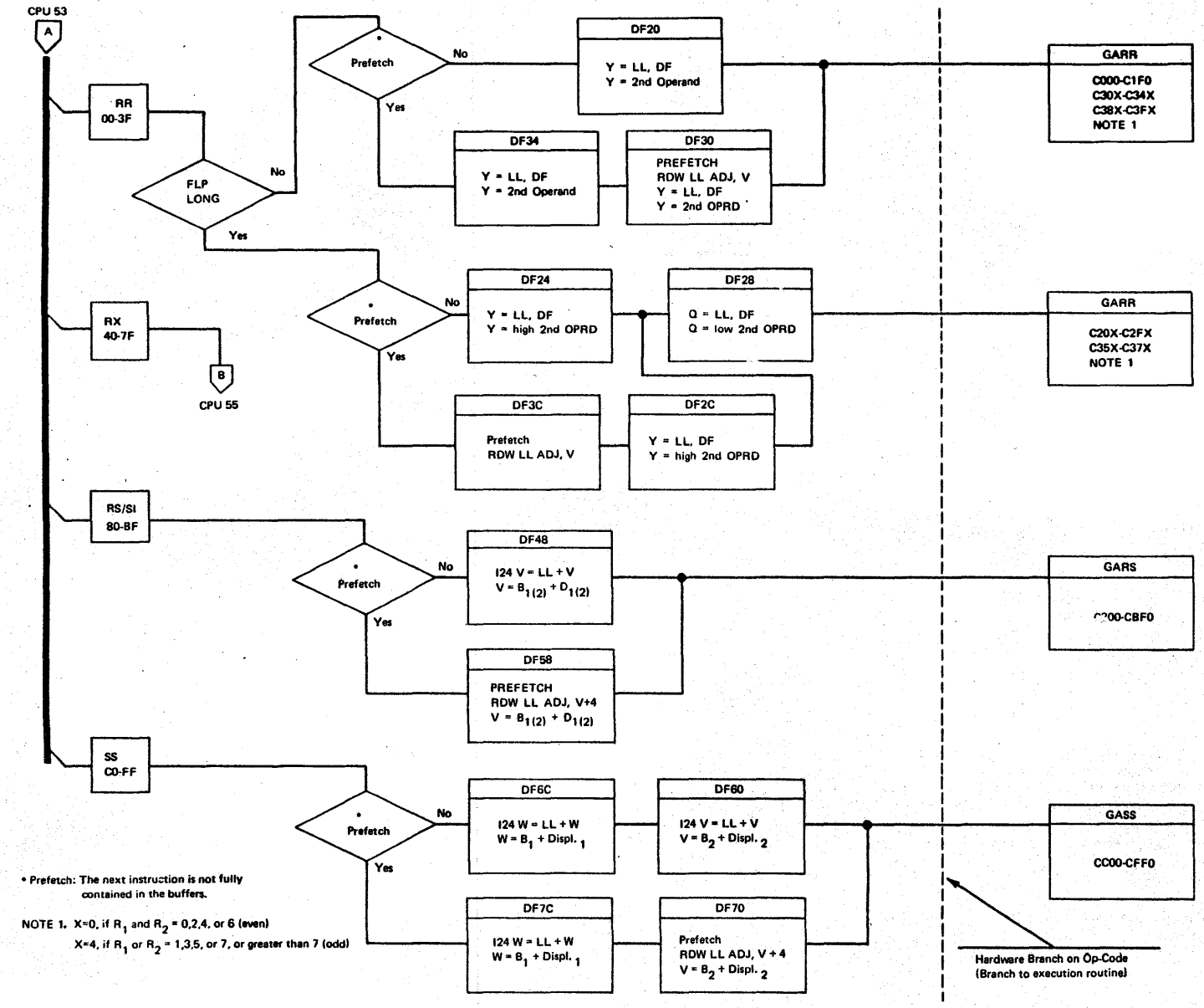
I-BFR SET CONDITIONS FOR MOVE I-BFRs

1. RR op and even halfword	=do nothing	
2. SS op and odd halfword	=S/R 0,1	S/R 1
3. Neither of the above conditions	=S/R 0	S/R 1
4. Prefetch and the next instruction is not fully contained in the I-BFRs	=S/R 0, 1	S/R 1

I-BUFFER SET CONDITIONS FOR STORAGE WORDS (begin in Stg 2 cycle)

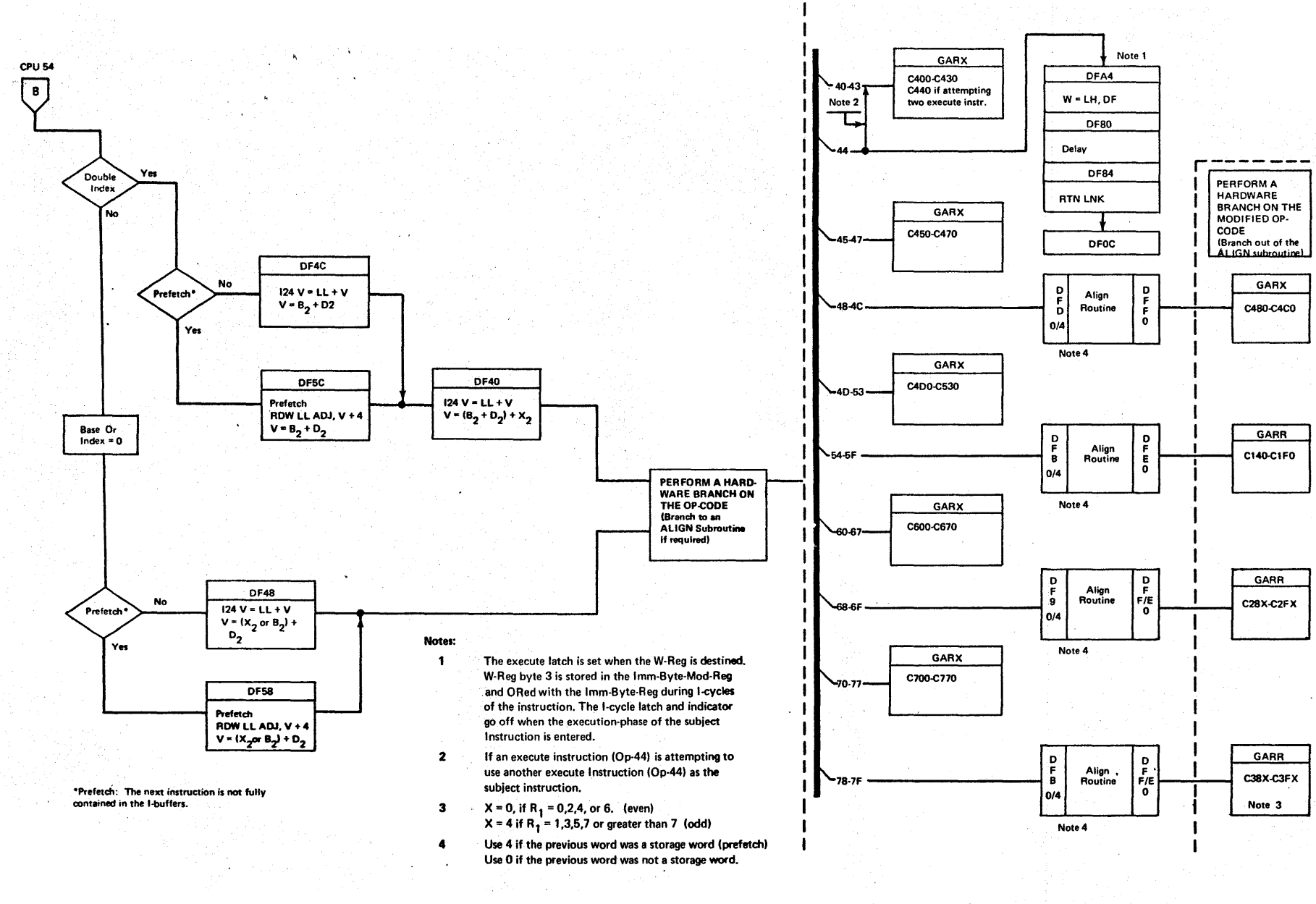
1. Prefetch and the next op is not available	=S/R 0, 1, 2	S/R 1.2
2. Prefetch and the next op is available	=S/R 1,2	S/R 2
3. Branch load latch	=S/R 0, 1, 2	S/R 1, 2
4. Op load latch	=S/R 1, 2	S/R 2

I-Cycles Microcode and Control Hardware-Calculate Operand Address and Perform Prefetches



• Prefetch: The next instruction is not fully contained in the buffers.

NOTE 1. X=0, if R_1 and R_2 = 0,2,4, or 6 (even)
X=4, if R_1 or R_2 = 1,3,5, or 7, or greater than 7 (odd)

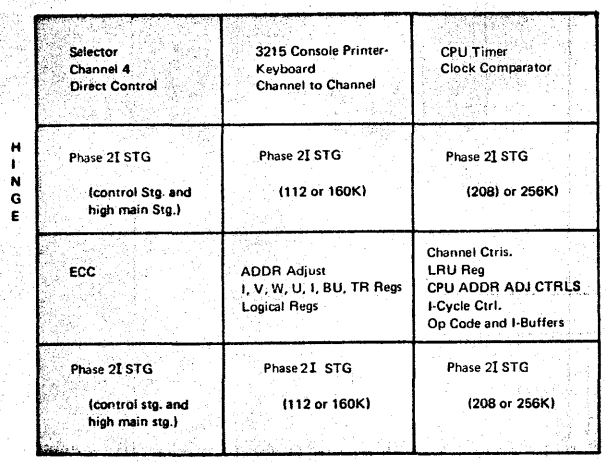


*Prefetch: The next instruction is not fully contained in the I-buffers.

Notes:

1. The execute latch is set when the W-Reg is destined. W-Reg byte 3 is stored in the Imm-Byte-Mod-Reg and ORed with the Imm-Byte-Reg during I-cycles of the instruction. The I-cycle latch and indicator go off when the execution-phase of the subject Instruction is entered.
2. If an execute instruction (Op-44) is attempting to use another execute Instruction (Op-44) as the subject instruction.
3. X = 0, if R_1 = 0,2,4, or 6. (even)
   X = 4 if R_1 = 1,3,5,7 or greater than 7 (odd)
4. Use 4 if the previous word was a storage word (prefetch)
   Use 0 if the previous word was not a storage word.

I-Cycle Hardware Locations



H I N G E

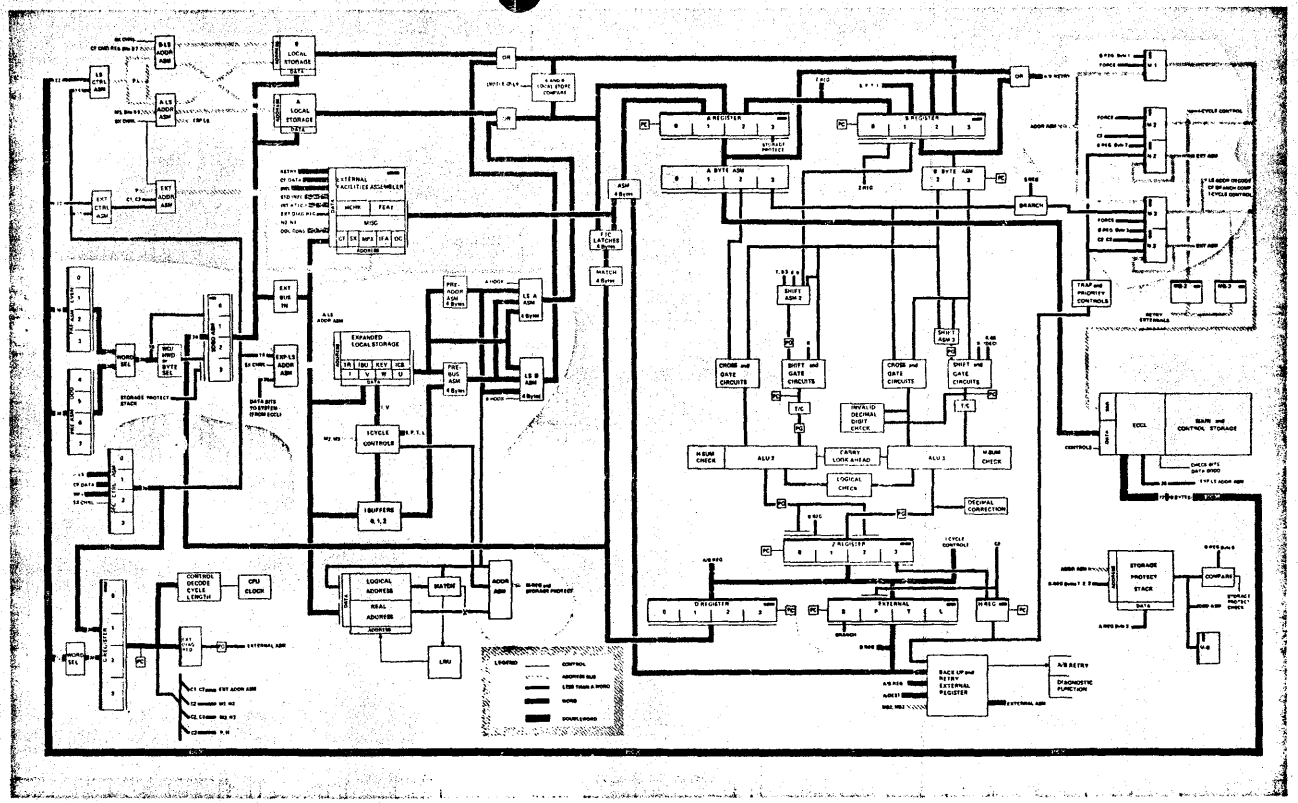
B GATE (CARD SIDE)

CARD LOCATION & TYPE		ALD PAGE	
B-C3D2	Type 8551	RU011	Op-U2 Reg-Op Decode
		RU012	Op-U2 Reg-Op Decode
		RU013	Op-U2 Reg-Op Decode
		RU014	Op-U2 Reg-Op Decode
		RU015	Op-U2 Reg-Op Decode
		RU016	Op-U2 Reg-Op Decode
B-C3E2	Type 8552	RU021	Imm Byte-U3 Regs
		RU022	Imm Byte-U3 Regs
		RU023	Imm Byte-U3 Regs
		RU024	Imm Byte-U3 Regs
		RU025	Imm Byte-U3 Regs
		RU026	Imm Byte-U3 Regs
B-C3F2	Type 8553	RU031	I-Cycles Generation
		RU032	I-Cycles Generation
		RU033	I-Cycles Generation
		RU034	I-Cycles Generation
		RU035	I-Cycles Generation
B-C3G2	Type 8554	RU041	I-Buff Ctrls and Gates
		RU042	I-Buff Ctrls and Gates
		RU043	I-Buff Ctrls and Gates
		RU044	I-Buff Ctrls and Gates
		RU045	I-Buff Ctrls and Gates
B-C3H2	Type 8558	RU051	PAA Latches
		RU052	PAA Latches
		RU053	I-Cycles Controls
		RU054	I-Cycles Error Latches
B-C3B2	Type 7771	RU111	I-Buffer
		RU112	I-Buffer
		RU113	I-Buffer
		RU114	I-Buffer
		RU115	I-Buffer
		RU116	I-Buffer
		RU117	I-Buffer
		RU118	I-Buffer
B-C3C2	Type 7771	RU121	I-Buffer
		RU122	I-Buffer
		RU123	I-Buffer
		RU124	I-Buffer
		RU125	I-Buffer
		RU126	I-Buffer
		RU127	I-Buffer
		RU128	I-Buffer

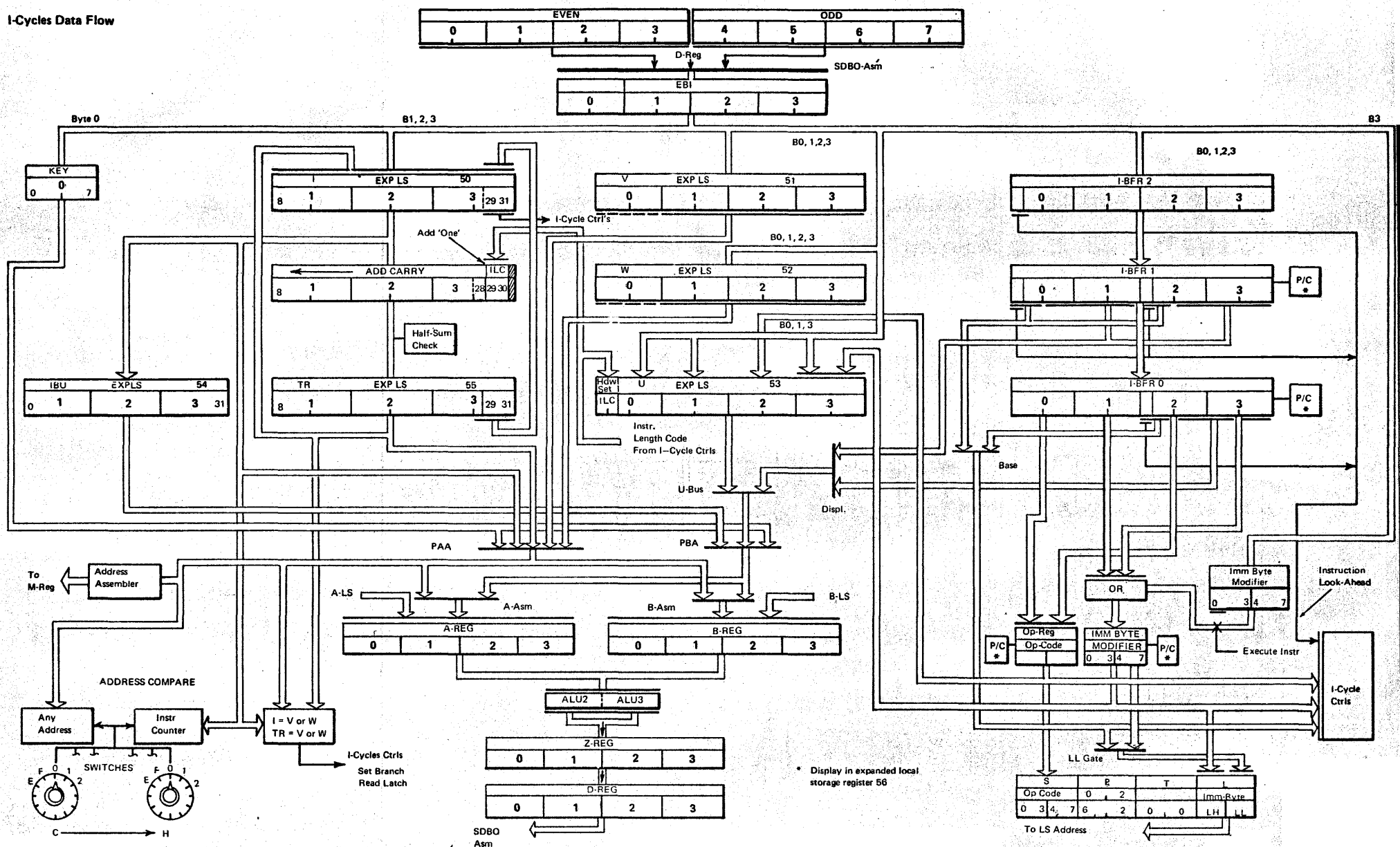
I-Cycle Hardware Description

The hardware I-cycle concept increases CPU performance by:

- Buffering instructions and prefetching (using a hardware generated address for the next doubleword storage location) instructions while calculating an operand address.
- Performing hardware controls concurrent with micro-program execution.
- Instruction decoding via a hardware forced branch on the eight-bit Op code.
- Windows in Data Flow Diagram show the hardware used by I-cycles.



I-Cycles Data Flow



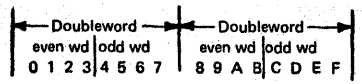
* Display in expanded local storage register 56

A I-BFR 2, I-BFR 1, and I-BFR 0

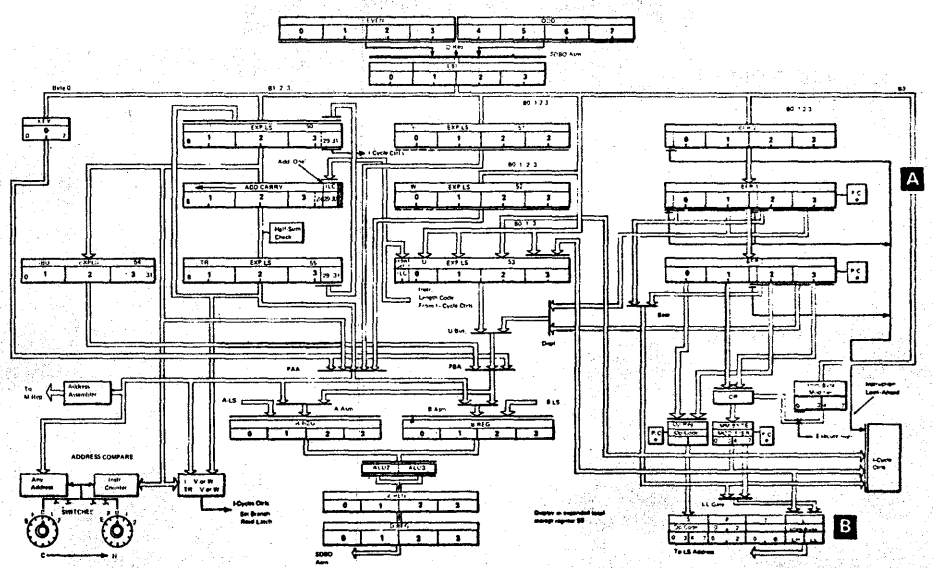
Instruction Buffer (I-Bfr): three one-word registers are used to hold the present instruction and next doubleword (where possible). Loading of the registers is from EBI.

Instructions are assembled on a halfword basis to obtain the op-code and immediate byte. The base and displacement fields are gated from the I-Bfr through separate assemblers as required.

When the TR-Reg is used, the even/odd time-slot of data is forced. When the I-Register is used, (DF0C) M3 bit 5 controls the gating.
Bit 5 off = even/odd
Bit on = odd/odd.



M3 bit 5=1 indicates that the odd word information is to be loaded into all 3 I-buffers.



I-BFR SET CONDITIONS for MOVE I-BFRs

1. RR op and even halfword	=do nothing	
2. SS op and odd halfword	=S/R 0, 1	S/R 1
3. Neither of the above conditions	=S/R 0	S/R 1
4. Prefetch and the next instruction is not fully contained in the I-Bfrs	=S/R 0, 1	S/R 1

I-BUFFER SET CONDITIONS for STORAGE WORDS (begin in Stg 2 cycle)

1. Prefetch and the next op is not available	=S/R 0, 1, 2	S/R 1, 2
2. Prefetch and the next op is available	=S/R 1, 2	S/R 2
3. Branch load latch	=S/R 0, 1, 2	S/R 1, 2
4. Op load latch	=S/R 1, 2	S/R 2

Low-Order Address Position

ILC	IBFR	1st Halfword 0/8	2nd Halfword 2/A	3rd Halfword 4/C	4th Halfword 6/E
1 (RR)	2				
	1				
	0				
2 (RX, RS, SI)	2				
	1				
	0				
3 (SS)	2				
	1				
	0				

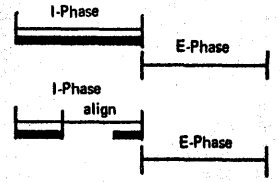
LEGEND:
- = The end of the doubleword in the I-Bfrs
- = Content of Bfr not significant.
- = Remainder of Doubleword
- = One IBFR Position
- Instruction being operated on by I-cycle hardware.

The above chart represents the contents of the I-Bfrs for the 12 different combinations of instruction length and instruction location within a doubleword.

B SPTL Registers

S-Reg Set from the op-reg for further use by the execution routine.

Set time for SPTL.



P-Reg Set to a value of 02 (or 62 for floating point) to allow addressing of local-storage areas containing general registers. CPU working area, and floating-point registers.

T-Reg Reset to zero for use in an align routine (if required).

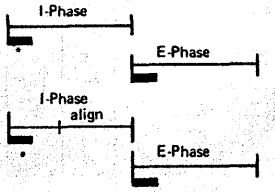
L-Reg Set from the immediate byte register or base field assembler to allow indirect addressing of the general registers and floating-point registers.

C

Op-Register

The op-register is used to hold the present instruction op code during I-cycles. The output is gated to the U- (2) and S-registers, and is also used for I-cycle decoding and branching to the instruction execution routine.

The decode of the op-register is used to build the ILC, which is gated to the U-register byte 0, bits 0 and 1.

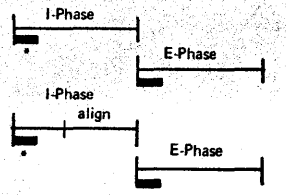


D

Immediate Byte Register

The Imm byte reg holds the second byte of the present instruction during I-cycles. The output:

- Loads the U3-register.
- Is assembled with the base field to be gated to the L-register.
- Is gated to the I-cycle controls.



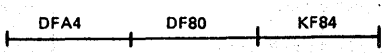
*Only with branch-read (DF0C)

E

Immediate Byte Modifier Register

- Used only by the Execute software instruction to modify the second byte of the subject instruction (if the R_1 field of the Execute instruction is not zero).
- The register is set when the W-reg is destined, with data from byte 3 of the GR specified by the R_1 field. (Refer to "Execute Phase (I-Cycles).")

W=LH, DF

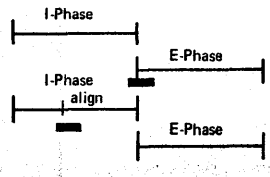


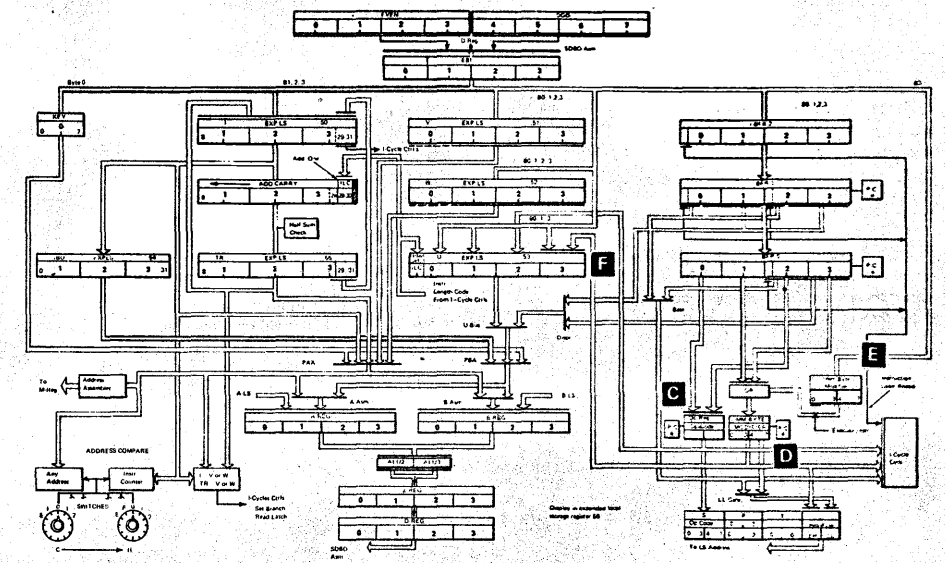
F

U-Register (Exp LS 53)

Part of this register is set only by hardware. The two-bit Instruction Length Code (ILC) is set to a value determined by the op-reg decode. The condition code (two bits) is used by the I-cycle controls to determine whether a branch-on-condition code instruction branches. Byte 2 is set to the op-code by hardware only. Byte 3 is initialized to the immediate byte by hardware. Bytes 0 (except bits 0, and 1), 1, and 3 may be loaded from EBI by microcode.

Byte 0	Bits 0-1	Instr. length code (hardware set only)
	Bits 2-3	Condition code
	Bits 4-7	Program mask
Byte 1	Bits 0-3	Special CPU use
	Bits 4-7	0MWP bits
Byte 2	Bits 0-7	Op code (hardware set only)
Byte 3	Bits 0-7	Immediate byte





G

Special address-matching function--on a doubleword basis (bits 8-28).

- I-reg is compared to PAA (V or W). TR-reg is compared to PAA (V or W). (These matches are required to determine whether program modification is taking place in that part of the instruction stream that may have been loaded in the I-Bfrs.)
- If the comparison is equal during a store operation, the BR read latch is set. (This forces a new loading of the I-buffer op-reg and imm byte reg.)

H

V-Register (EXP LS 51)*

Bytes 1, 2, and 3 usually contain the second operand address.

I

W-Register (EXP LS 52)*

Bytes 1, 2, and 3 usually contain the first operand address.

*If used in a storage word as a storage address, the key register is gated as byte zero.

J

Key Register

The key register acts as byte 0 of the I-register when the I-register is destined. The key register contains the storage protect key in bits 0-3. Bit 4 is the fetch protect bit. Bit 7 is 0. With the DAT feature installed, bit 5 is the reference bit and bit 6 is the change bit. For a more detailed description of bits 5 and 6, see "Reference and Change Bit Recording" under DAT in the CPU section.

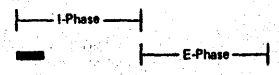
If the V-register or W-register is used in a storage word as a source, the key register is gated as byte 0 of the V or W-register. If the I-register, IBU-register, or TR-register is used, the key register is gated as byte 0 of that register. The key register is set when the I-register is destined.

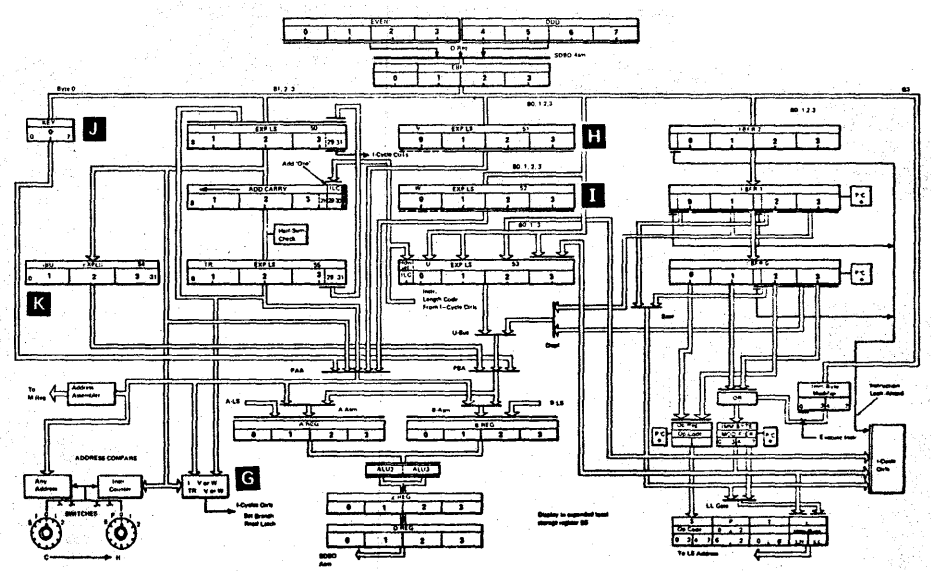
K

IBU-Register (EXPLS 54)*

Upon entering I-cycles, I-reg bytes 1, 2, and 3 are set into IBU. If a retry condition is encountered during I-cycles, the instruction cycles may be repeated (return to DF0C). In this case, IBU is moved to the I-reg by the retry microprogram.

IBU is set from I-reg as follows:

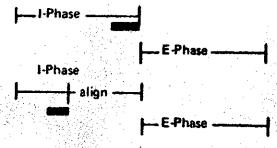




L I-Register (EXPLS 50)

Instruction counter register, byte 0: if the I-reg is destined, byte 0 is gated to the key register. Bytes 1, 2, and 3 contain the instruction address.

The I-reg is updated as follows:



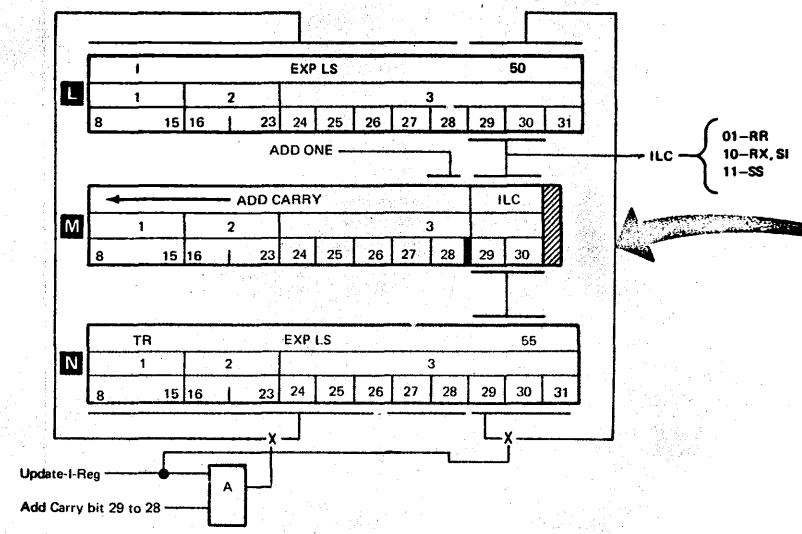
M Add Carry (Adder)

The two low-order adder positions (24-bit adder) add the instruction length to the I-register bits 29-30.

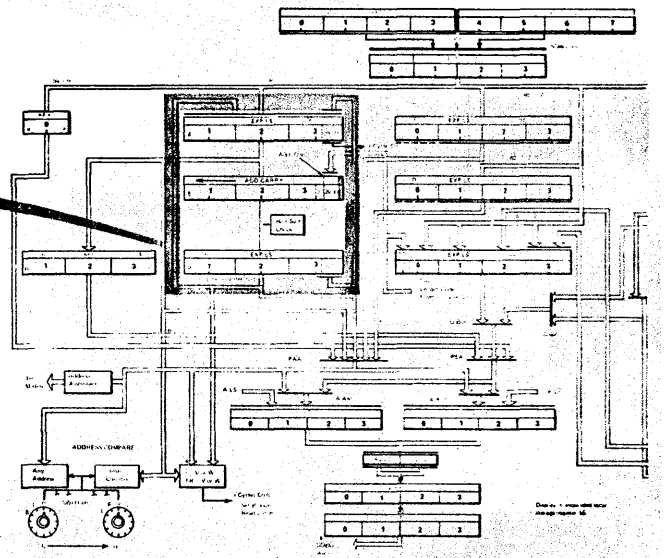
- Adder (except the low two bits) adds a "1" to I-reg bit position 28. This sum represents the next doubleword storage location.
- A carry-out of position 28 is added to bits 27 to 8.
- A carry-out of position 29 indicates that TR bits 8 to 28 are to be loaded into I for a hardware update.
- An adder check turns the I-cycle hardware indicator on.

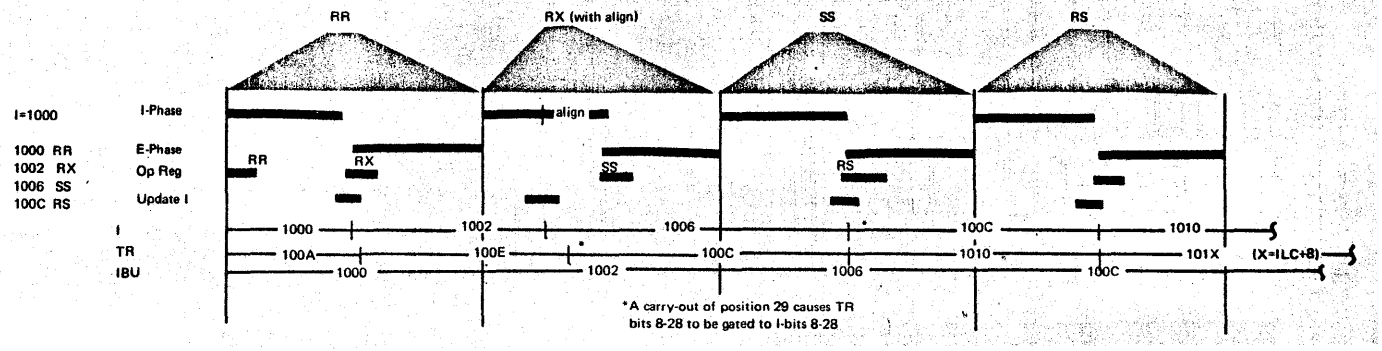
N TR-Register (EXPLS 55)

- The TR-reg consists of bits 8 to 30, with bit 31 always forced to a zero. (Bit 31 has no latch.)
- Contains an address within the next doubleword after the address in the I-register; used when prefetch or further fetch (DF14) is required.
- Used to buffer the adder output during hardware updating of the I-register.
- I-hardware update, consists of loading I bits 29-31 from TR 29-31. Bit 31 of the I-register is not gated through the adder. I bits 8 to 28 from TR 8 to 28 if the Adder bit 29 had a carry-out. (TR = I + ILC + 8)



Carry outs from add carry bit 29 to bit 28 are not allowed, but are remembered. At update-I time, a carry remembered causes all of TR to be gated back to I. Otherwise, only bits 29 and 30 are used to update I-reg.

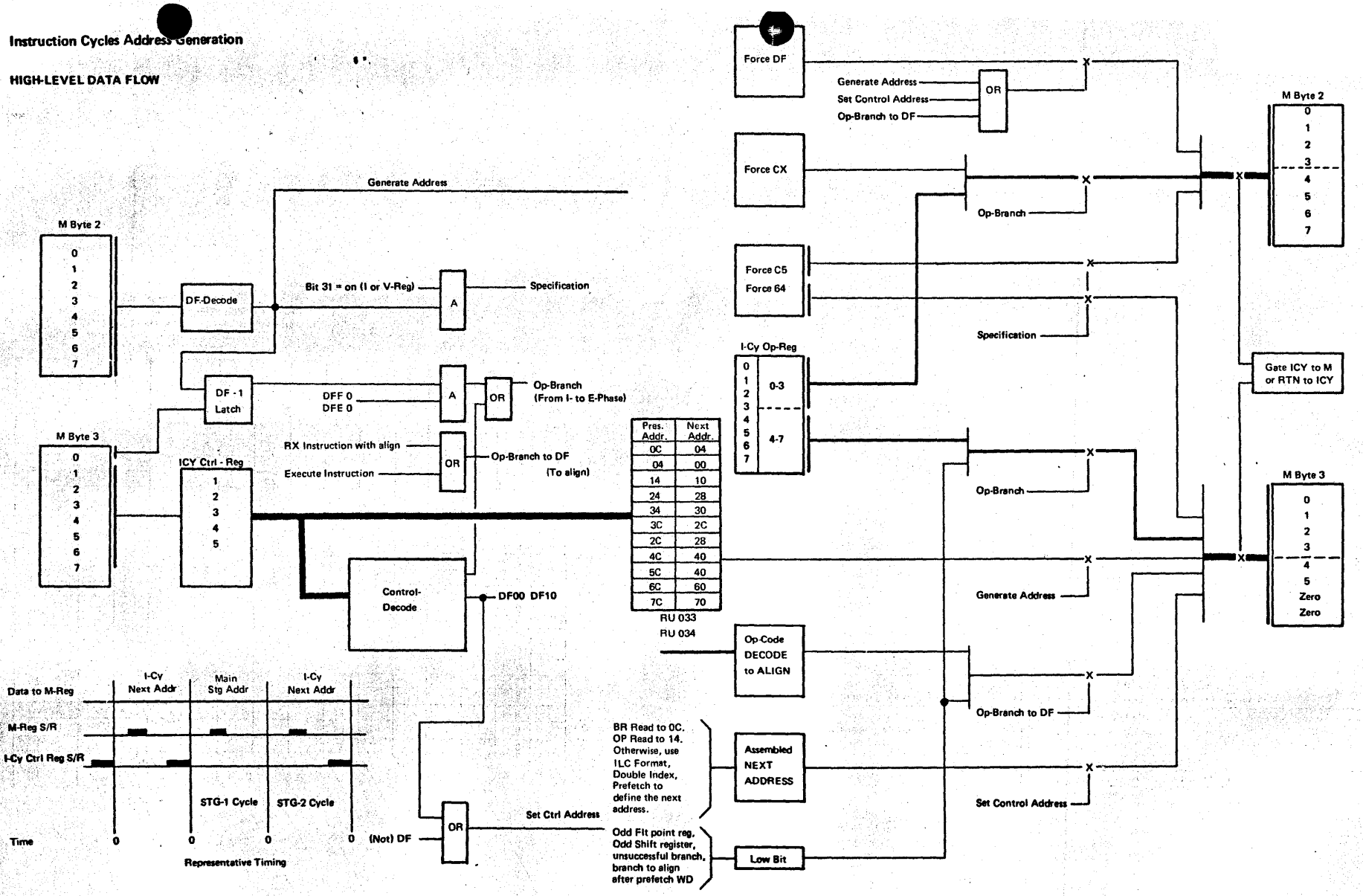




*A carry-out of position 29 causes TR bits 8-28 to be gated to I-bits 8-28

Instruction Cycles Address Generation

HIGH-LEVEL DATA FLOW



I-Cycles Control Line Generation

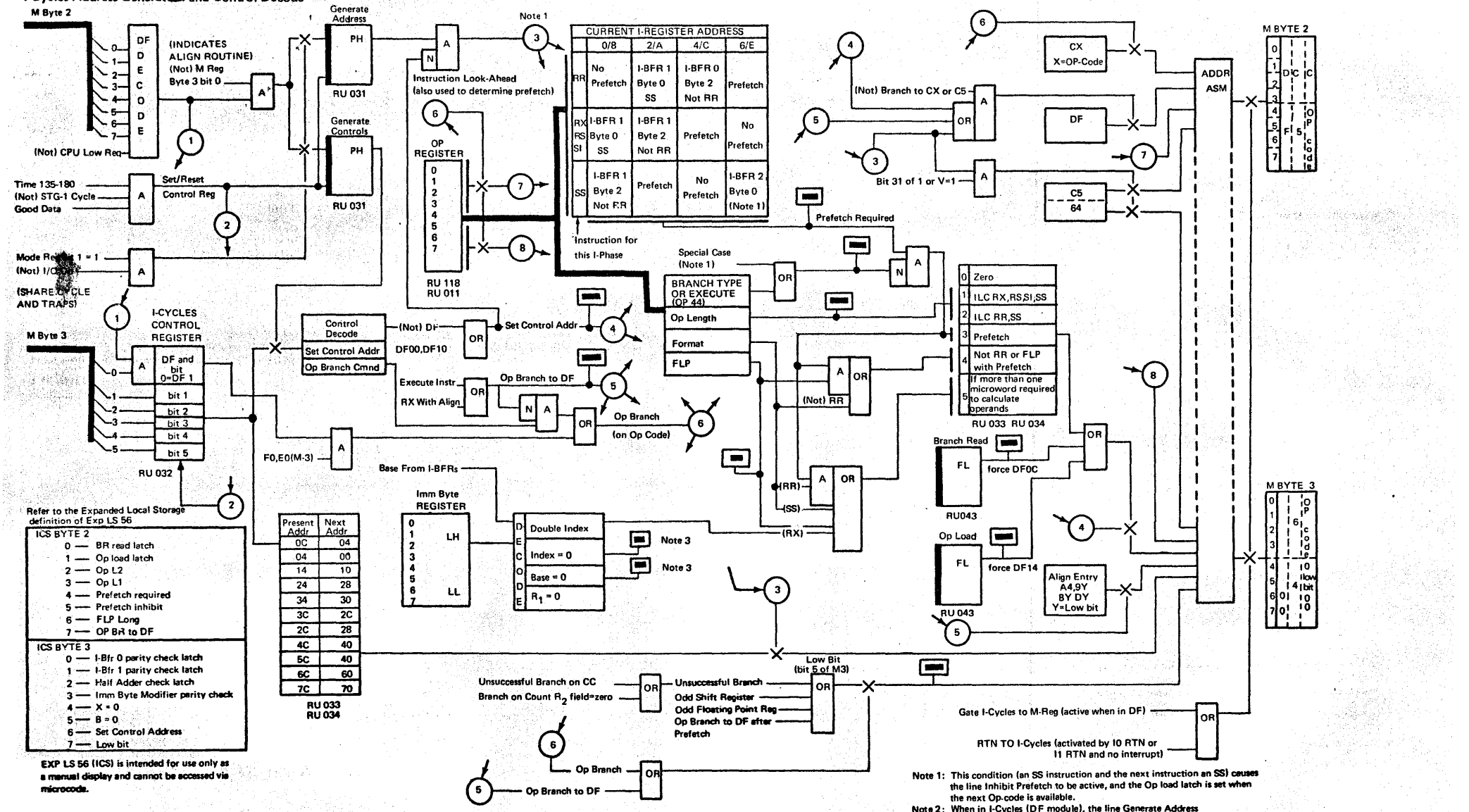
ADDRESS	Command Branch Load	Force I to B	Command Op Load	Force TR	Command Prefetch	OP Branch Command	L plus one	Load Op-Reg Imm Byte	Command Move I-Bfrs	Gate D_1 and B_2	Gate D_2	Set Control Address	Set SPL Reset T	Gate Imm Byte to L
DF00								2				2	1 6	
DF04								2					1 6	
DF0C	2	2						2 5					1 6	
DF10												2	1 6	
DF14			2	2									1 6	
DF20						2			2				1 6	
DF24							2		2				1 6	
DF28						2							1 6	
DF2C							2						1 6	
DF30				1 3	2	2			4				1 6	
DF34													1 6	
DF3C				1 3	2				4				1 6	
DF40						2							1 6	1
DF48						2			2	2			1 6	1
DF4C									2	2			1 6	
DF58				1 3	2	2			4	2			1 6	1
DF5C				1 3	2					2			1 6	
DF60						2			2		2		1 6	1
DF6C										2			1 6	
DF70				1 3	2	2			4		2		1 6	1
DF7C										2			1 6	
(Not) DF												1		
RTN to I-Cycles													1 7	

1. These controls are not the result of the control register decode.
2. The control line is activated by the corresponding address.
3. This line is activated by command prefetch.
4. The set/reset of the I-Bfrs is also controlled by command prefetch.
5. This control line is activated, but not used.
6. From DF00 through DF7C SPTL is controlled by I-cycles. DFE0 or DFF0 activates this control line again to restore SPTL after an align.
7. Set P-set LL.

CONTROL REGISTER DECODE

I-CYCLES CTRL REG BITS 1	2	3	4	5	CONTROL LINE	FUNCTION
0	0	–	1	–	Command Branch Load	Load I-Bfr 0 (I-Bfr 1 in next cycle)
0	0	–	1	–	Force I	Gate I-Reg to B (and Address Adjust)
0	0	1	0	1	Command Op Load	Load I-Bfr 1 (I-Bfr 2 in the next cycle)
0	0	1	0	1	Force TR	Gate TR-Reg to B (and Address Adjust)
–	1	1	–	0	Command Prefetch	Activate Prefetch
1	0	1	1	–		Force TR to ADR ADJ Asm
0	1	1	1	–		(and B if DF30 or DF3C)
–	1	–	–	0	Op Branch Command	Use op-reg to define the next Cxxx address; or go
1	–	–	–	0		to the align routine
0	1	0	–	1	L plus one	Force bits 7 and P of data being gated to L-reg to be inverted for FLP long
0	0	0	–	–	Load Op, Imm Byte	Initial load of Op and Imm Byte
–	1	0	0	–	Command Move I-Bfr	Used with I-reg to activate the set
1	0	0	1	–		or reset of the I-Bfrs
1	–	–	1	–	Gate D_1 and B_2	Used to gate the correct base or displacement
1	1	–	0	–	Gate D_2	field from the I-Bfrs 0 and 1 to L-low or B-Reg
0	0	–	–	0	Set Control Address	Set next address for I-cycles sequence

I-Cycles Address Generation and Control Decode



Note 1: This condition (an SS instruction and the next instruction an SS) causes the line Inhibit Prefetch to be active, and the Op load latch is set when the next Op-code is available.

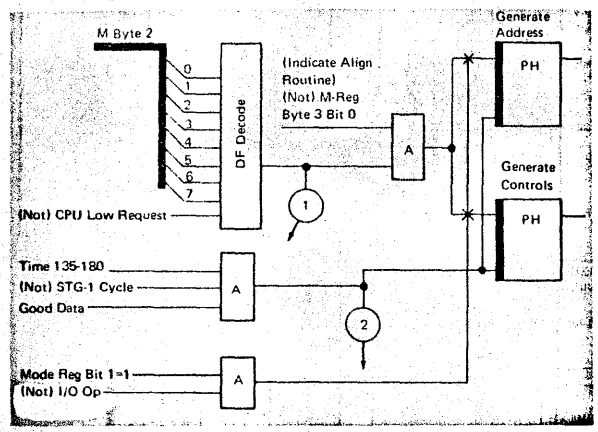
Note 2: When in I-Cycles (DF module), the line Generate Address is functionally overridden during addresses DF00 and DF10.

Note 3: If both Index and Base indicators are off, double indexing takes place.

Unique Conditions During I-Cycles

Share Cycle

If a share cycle is attempted during I-cycles, the I-cycles hardware is deconditioned. This occurs when the line 'Not I/O Op' becomes inactive. (See "I-CY Address Generation and Control Decode.") With '(Not)I/O Op' inactive, the generate address, and generate controls latches are deconditioned.



Trap (Not Machine Check)

In the event of a trap during I-cycles, the M-reg is set to the appropriate trap address, and the trap is taken. When the trap address enters the M-reg, the DF Decode turns off, and suspends I-cycle operation. '(Not) I/O Op' is also deconditioned whenever H1, 3, 4, 5, 6, or 7 are on.

Trap (Machine Check)

If a machine-check trap occurs during I-cycles, the failing instruction is retried eight times before entering the hard machine-check routine.

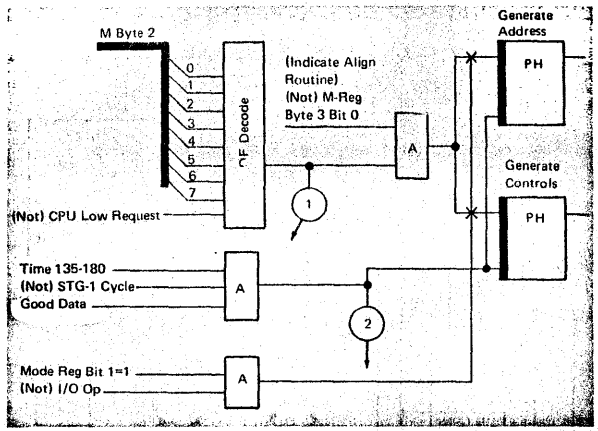
The Retry Microroutine takes the contents of IBU (I-Reg Backup; that is, the current instruction address) and places it in the I-Reg. The retry procedure is to then return to I-cycles to begin processing the instruction again.

I-Cycle Error Conditions

Parity-check errors for I-Bfr 0, I-Bfr 1 Op and Imm Byte regs as well as half-sum check errors are indicated in expanded local-storage register 56 (ICS) I-cycle control display.

Storage Correction Cycle

If a storage correction cycle occurs during I-cycles, the latches Generate Address and Generate Controls are not S/R. The line 'good data' blocks the clock pulse. (See "I-CY Address Generation and Control Decode.")

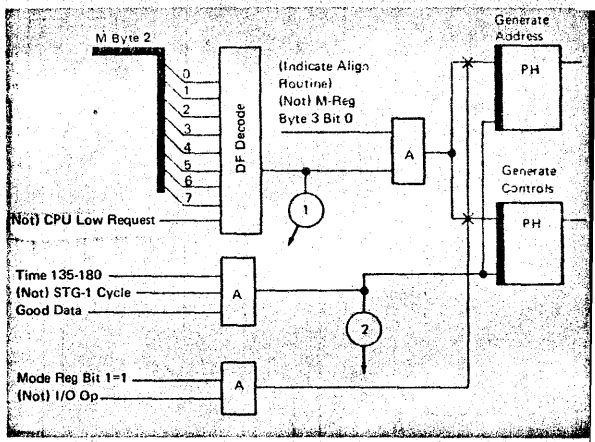


CPU Low Request (Not Prefetch)

The line 'CPU Low Request' may be activated by:

- Storage-protect check (caused by a mismatch of the storage keys)
- Address check (TR pointing to the doubleword above the top of storage)
- Address translation trap

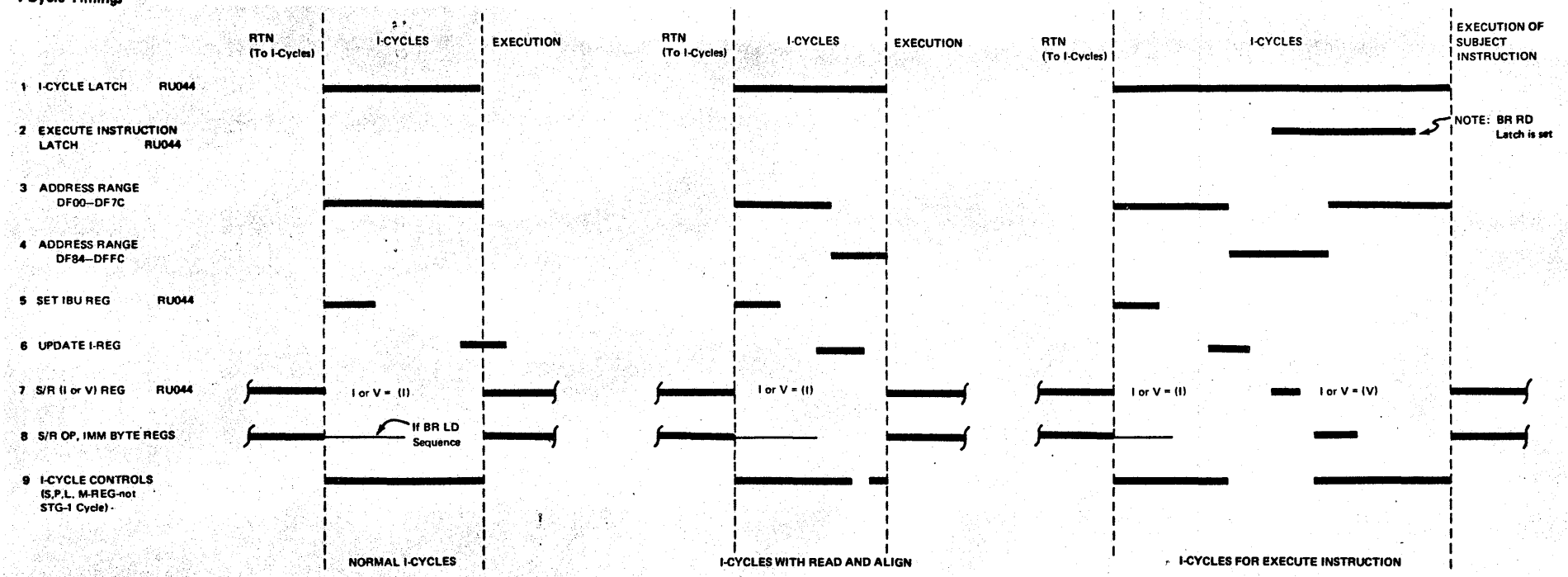
The two check conditions may be considered as program errors that cause a program check and go to the GICM routine. The address translation trap may cause a similar check, depending upon the availability of the addressed area.



CPU Low Request (Prefetch)

CPU low request is blocked during Prefetch. If a storage check occurs during a Prefetch, it sets the branch read latch. Upon return to I-cycles, the I-Reg is used (instead of the TR-Reg) to refill the I-Buffer. If a second CPU low request occurs, it indicates a program check.

I-Cycle Timings



I-Cycle Operational Description

Software instruction decoding on the 3145 is accomplished by a unique interaction of microprogram and hardware.

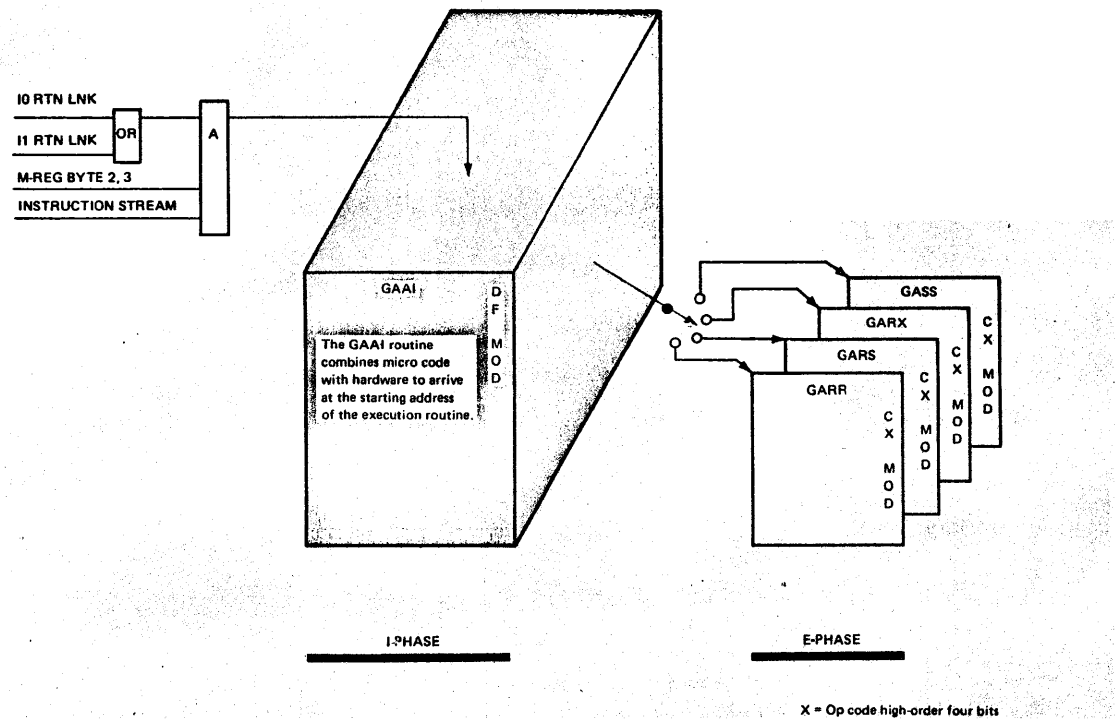
The microprogram used for instruction decoding is the GAAI routine, which resides in the DF module of control storage.

I-Phase

During I-phase, the instruction is read out of storage and placed in I-buffers. Certain determinations may then be made concerning format, Op-code, and instruction length code. The purpose of the I-phase is to ensure that the correct data is available for use during the E (execute) phase. Upon exit from the I-phase, the Op-code is used to point to the next control-word address of the microroutine for that format. (for a 1A add instruction, address C1A0 is used as the entry to the GARR routine).

E-Phase

In the E (execute) phase, data is read, stored, and the correct condition code (CC) is set. The operations indicated in I-phase are performed by using the operands fetched during I-phase. The address sent to the M-reg is that of the entry to the execution routine.



Fetch Operations

Main Storage
Address 1000 1004

(Ignore) D2 0F 1D 01

Fetch

FETCH

- I-Bfrs are initialized.
- May have occurred by a program branch to the instruction at address 1004.

Note: See Fetch Sequence and Loading I-Buffers for additional information.

Main-Storage
Address
1004

D2 0F 1D 01 | 2D 02 47 41 05 01 1A 5A

Further fetch required because the current instruction is not fully contained in the I-Bfrs.

FURTHER FETCH

- Required when the current instruction is not fully contained in the I-Bfrs.
- Recognized by:
  1. The decoding of the current Op-code, which determines the length of the current instruction.
  2. The I-register points to an address within the doubleword that the first byte of the present instruction is located.
- Accomplished by: (storage word at DF14)
  1. Setting the Op load latch during DF04 and forcing DF00 to branch to DF14 with the gating line Set Control Address.
  2. DF14, a storage word, forces the TR-Reg to the B-Reg, thus fetching the next doubleword from storage by using S/R I-buffers 1 and 2 with the even word late in the storage-2 cycle of DF14. During DF10 (early in the cycle), I-buffer 2 is S/R with the odd word.

Note: See "Fetch Sequence and Loading I-Buffers" for additional information.

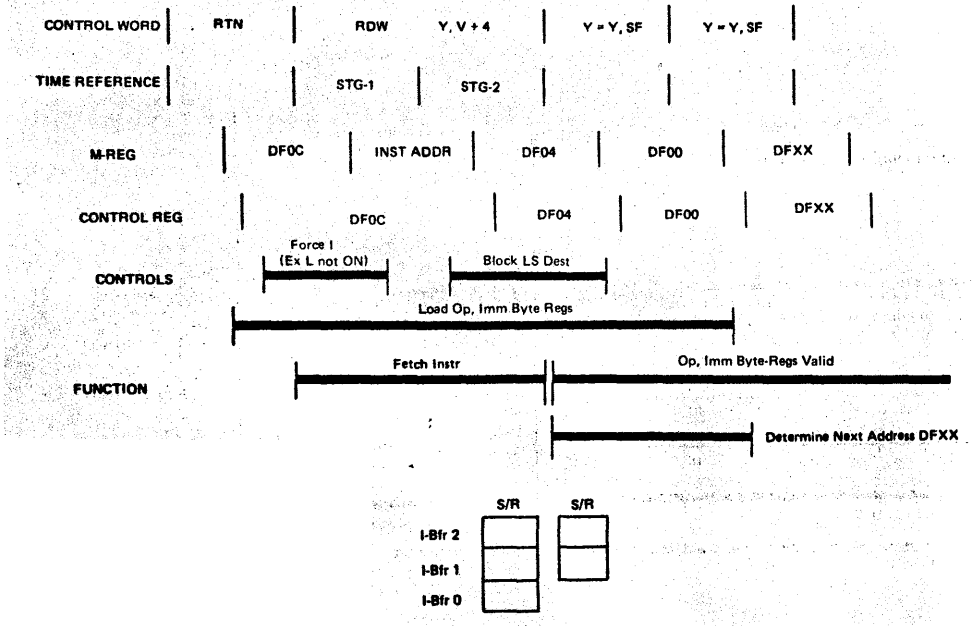
1A 5A | 47 30 A0 20 47 F0 A2 50 |

Prefetch

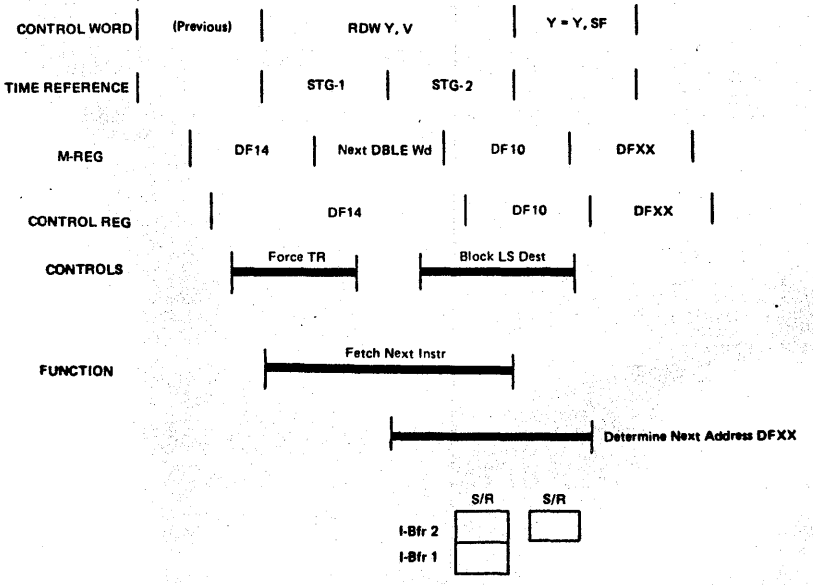
PREFETCHING

- Required when the next instruction is not fully contained in the I-Bfrs.
- Recognized by a combination of decoding:
  1. The I-Reg points to where the instructions come from within the doubleword.
  2. The Op code indicates the length of the present instruction.
  3. Instruction look-ahead knows the format of the next instruction and, therefore, knows the length of the next instruction.
- Prefetch is blocked under the following conditions:
  1. All branch-type instructions.
  2. Two SS instructions in succession (TR is pointing to the doubleword just fetched).

FETCH SEQUENCE
(BRANCH LOAD SEQUENCE)

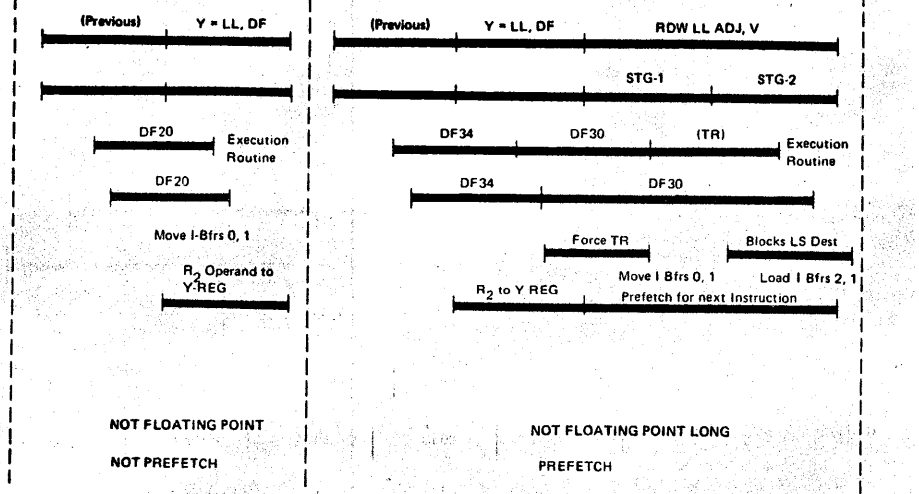


FURTHER FETCH SEQUENCE
(OP LOAD SEQUENCE)

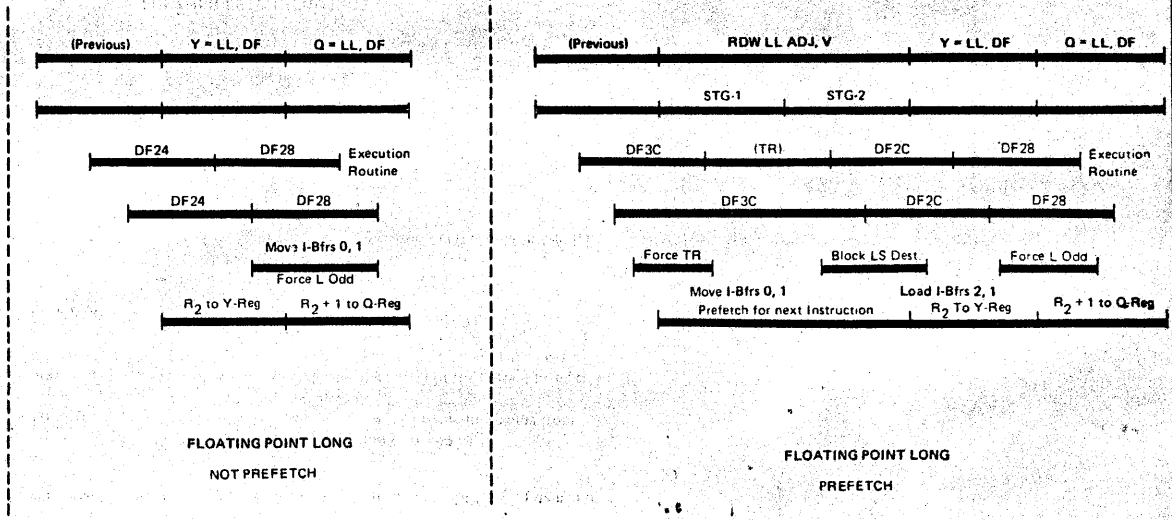


RR SEQUENCES

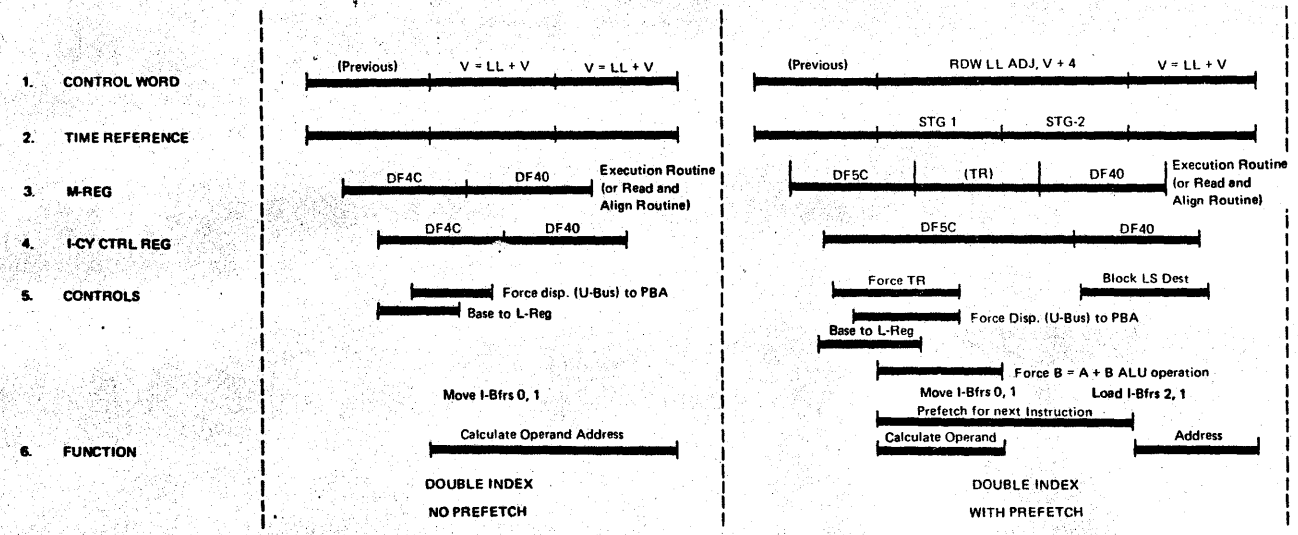
1. CONTROL WORD
2. TIME REFERENCE
3. M-REG
4. I-CY CTRL REG
5. CONTROLS
6. FUNCTION



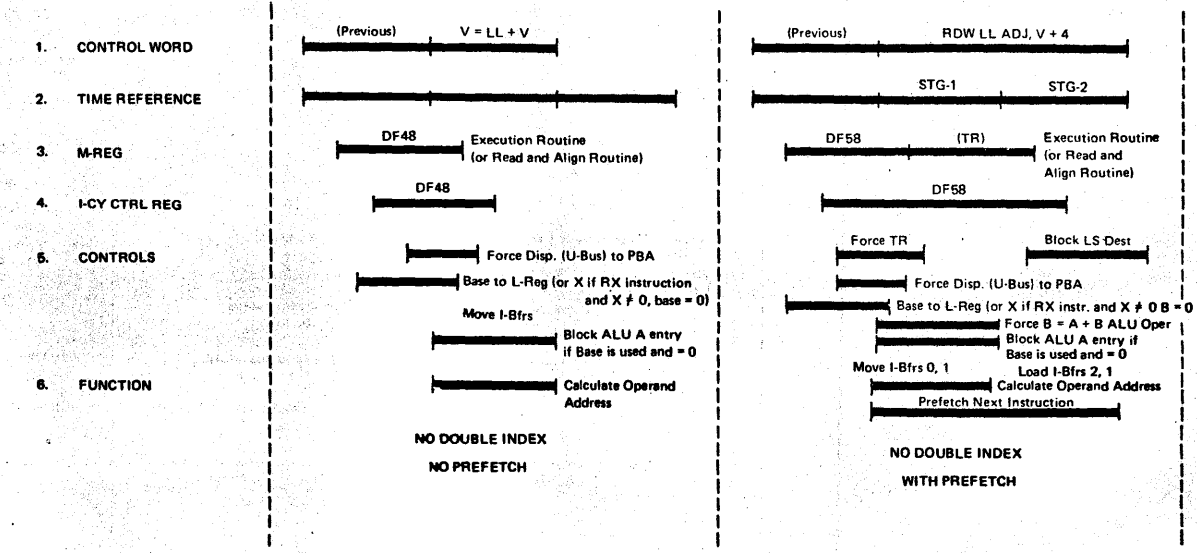
1. CONTROL WORD
2. TIME REFERENCE
3. M-REG
4. I-CY CTRL REG
5. CONTROLS
6. FUNCTION



RX SEQUENCES

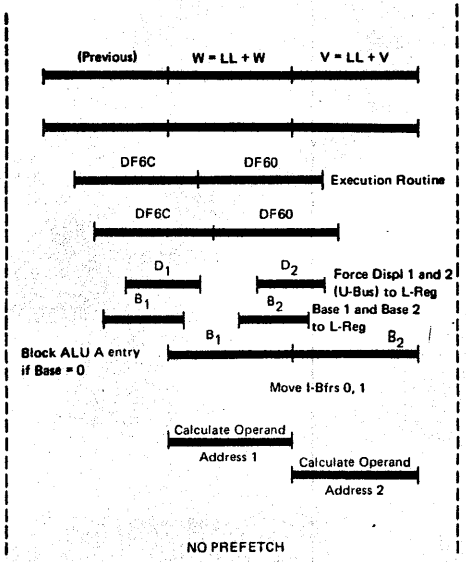


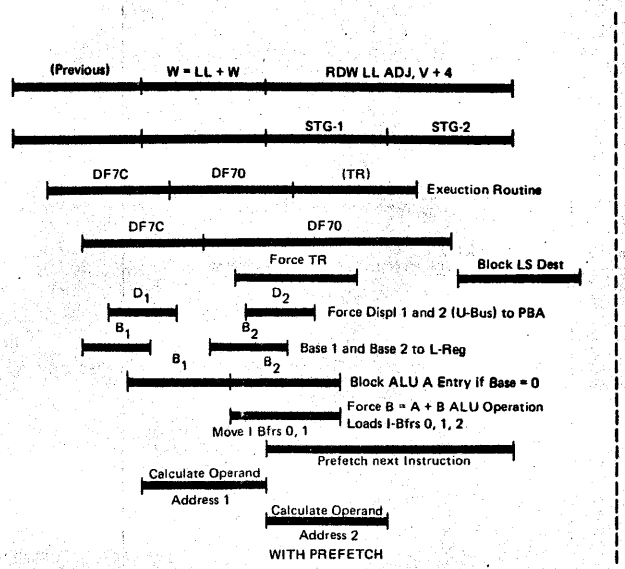
RX, RS, SI SEQUENCES



SS SEQUENCES

1. CONTROL WORD
2. TIME REFERENCE
3. M-REG
4. I-CY CTRL REG
5. CONTROLS
6. FUNCTION

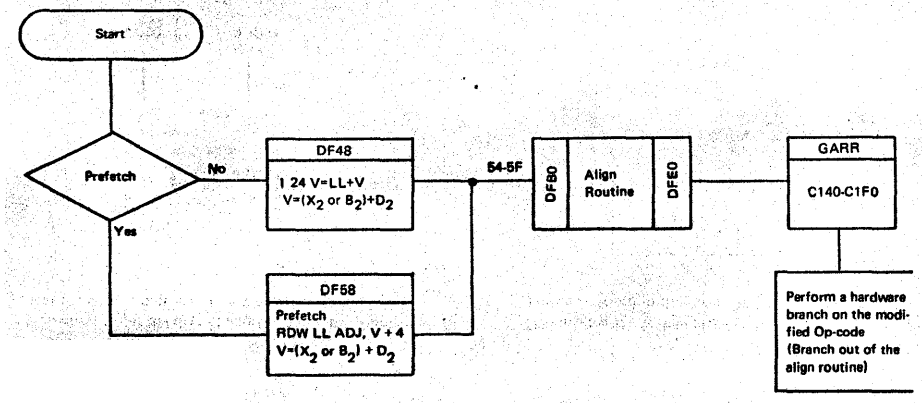


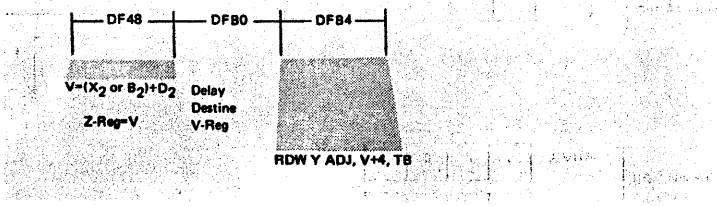


I-Cycles Alignment R[illegible]

I-Cycle Entry with No Prefetch

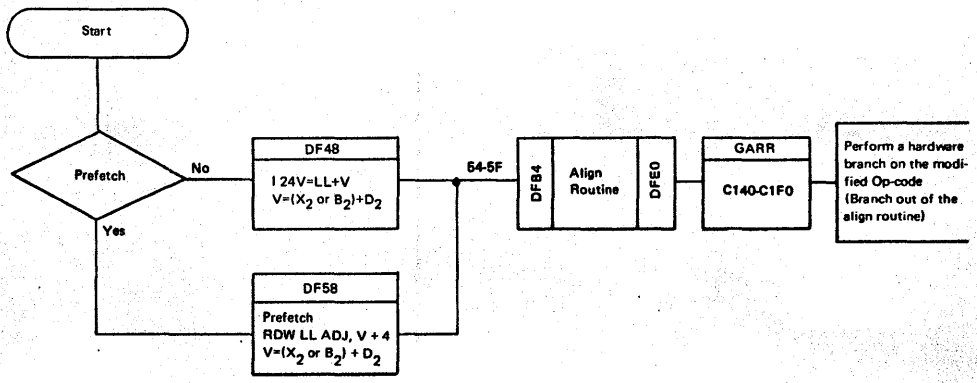
(Previous word was not a storage word.)
The align routine is entered at address DFB0, which is a delay word. The delay word allows the destine of the V-Reg.

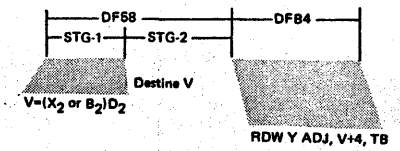




[illegible]le Entry with Prefetch

(Last word was a storage word, requiring a storage-1 and storage-2 cycle.)
The align routine is entered at address DFB4, thereby eliminating the delay word. The delay word is not necessary in this instance because the storage-2 cycle allows sufficient time for the V-Reg to be destined.



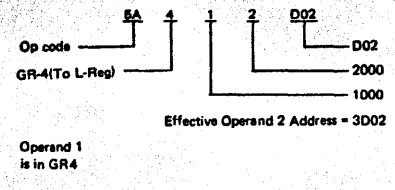


I-Cycle Alignment

The align routine enables data in main storage not on a word boundary to be aligned to appear on a word boundary.

Consider the following example:

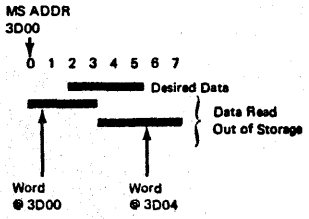
GR 1 = 00 00 10 00
GR 2 = 00 00 20 00



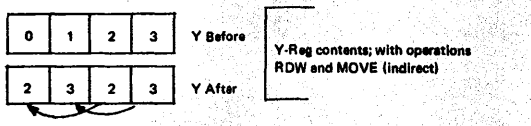
Operand 1
is in GR4

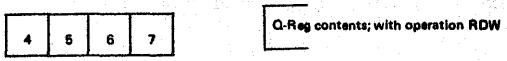
The data at main-storage address 3D02 is to be added to the contents of GR-4 and the result placed in GR-4.

In order to get the data at address 3D02, the problem arises that the desired data is not on a word boundary.

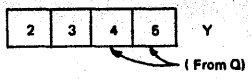


During the storage access, the bytes 0, 1, 2, and 3 are placed in the Y-Reg. Through the use of the TA-TB control-word function, bytes 2 and 3 are moved to bytes 0 and 1.





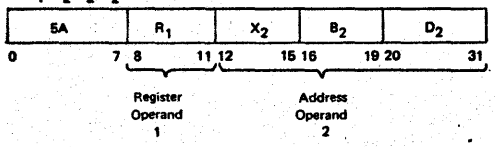
The information is then placed in the Y-Reg (via an indirect move operation) in the form;



I-Cycle Exit from the Align[redacted] Routine

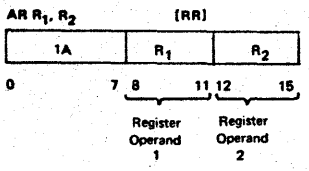
In some instances, an RX instruction can place the operand 2 data into a local-storage register. Once in the register, the operand 2 information may use the RR format-execution routine. Consider the following example:

A $R_1, D_2(X_2, B_2)$ [RX]

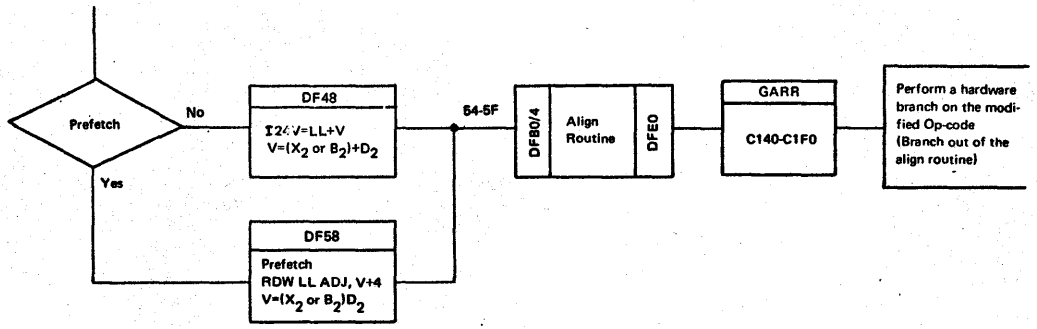


An RX '5A' add instruction has two operands, consisting of R_1 and $X_2 + B_2 + D_2$. If the operand two fields are combined, the instruction would then appear as a '1A' RR instruction. The operand 2 data for both formats is placed into the Y-register before entering the execution phase.

AR R_1, R_2 [RR]



During the align routine, the ope[redacted]2 data is placed in a local-storage register. Upon exiting the align routine, the RR execution routine may be used.

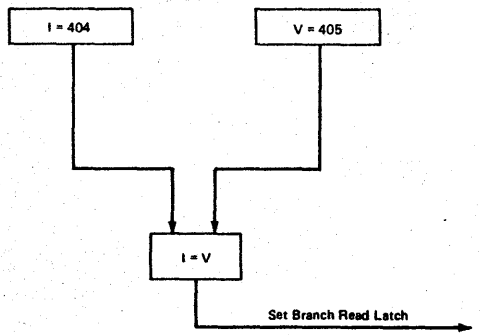


I-Cycles Program Modification

Program modification is detected by a store operation that uses V or W as a storage address within the present, or next storage double-word address, as compared to the I- or TR-Reg. Only bits 8-28 are compared.

When program modification is detected, the branch read latch is set, and the modified instruction is loaded into the I-Bfrs.

In this example, the Move Immediate Op-code causes the modification of a portion of storage that is already loaded in the I-Bfrs. To continue operating on the old information in the I-Bfrs would cause the wrong result. To replace the old information, the branch read latch is set, and the modified information is loaded into the I-Bfrs early in the next I-phase.



I - BFR 2

D2	00	F1	24

I - BFR 1

D2	00	F1	24

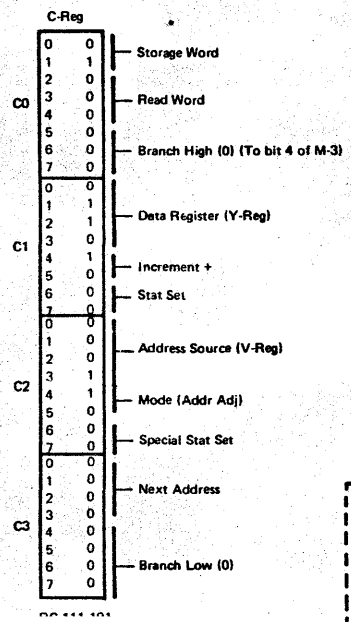
I - BFR 0

92	FF	04	05

```
ADDR.   INSTR.          LABEL
400     92FF0405                .MVI FF, INOUT + 1
404     D200F124F700    INOUT    MVC OUTPUT (00), INPUT
```

Control Word DF0C

ADDRESS	WORD	STATEMENT
DF0C	40 68 18 00	RDW Y ADJ, V + 4



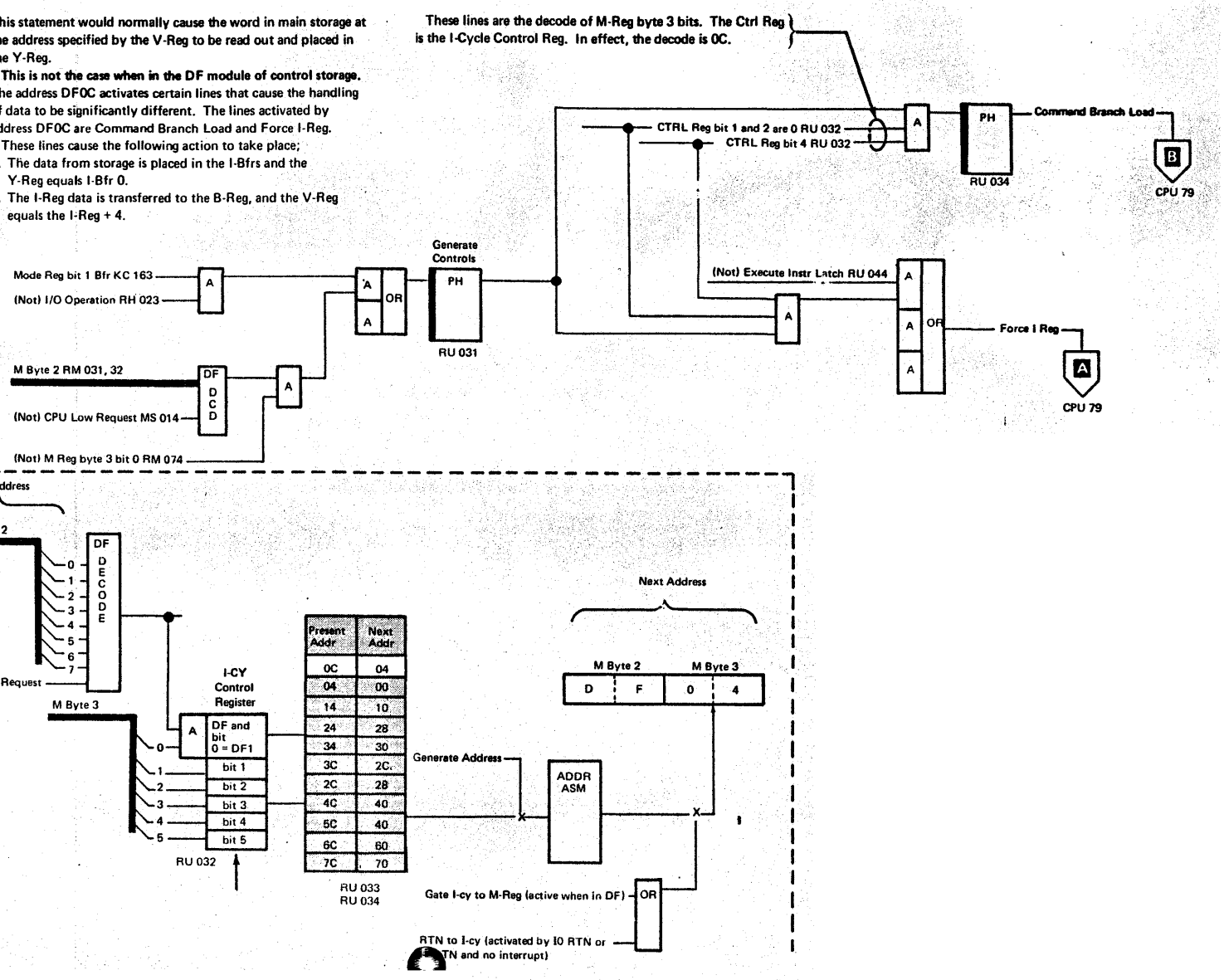
This statement would normally cause the word in main storage at the address specified by the V-Reg to be read out and placed in the Y-Reg.

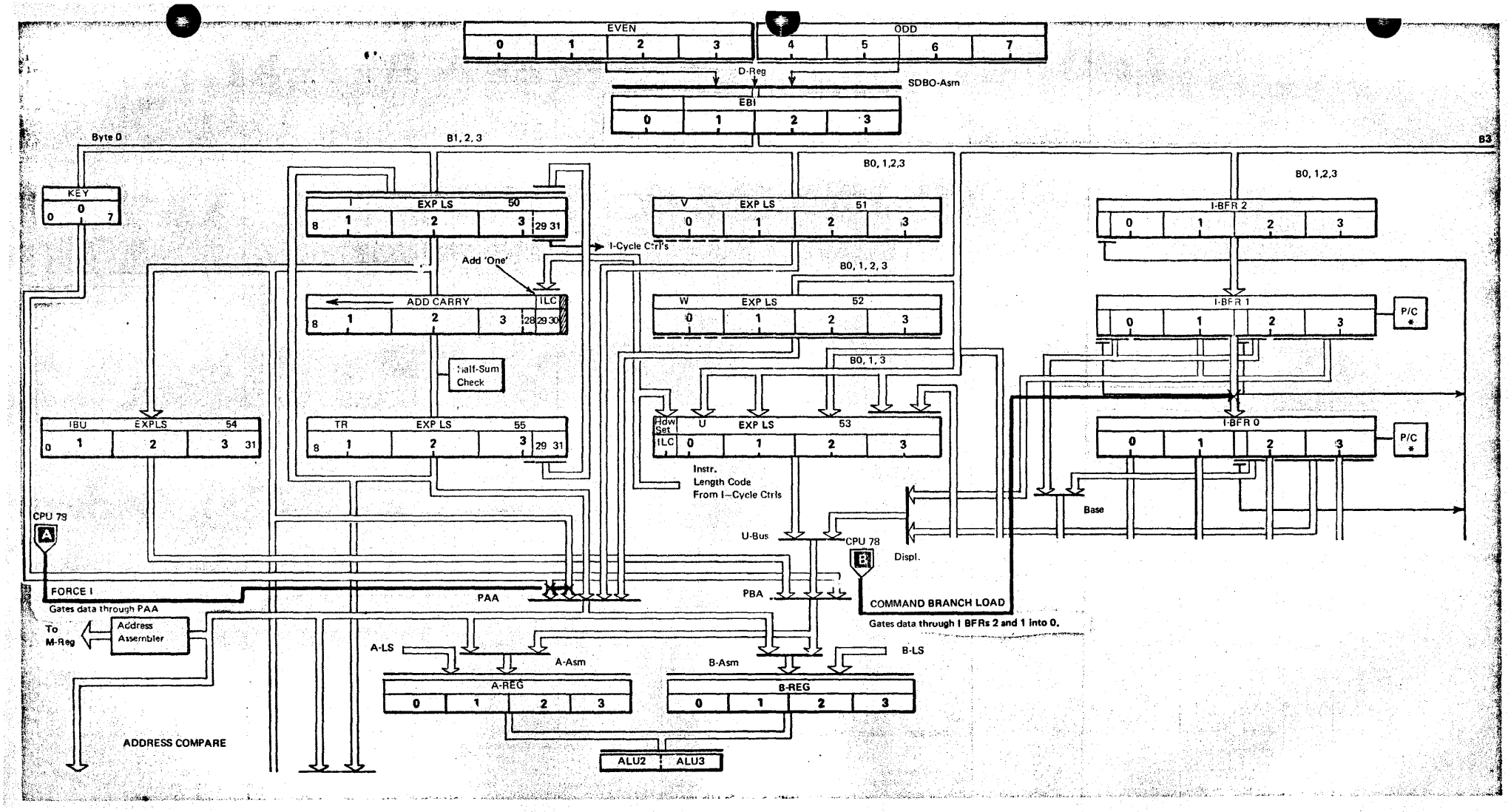
This is not the case when in the DF module of control storage. The address DF0C activates certain lines that cause the handling of data to be significantly different. The lines activated by address DF0C are Command Branch Load and Force I-Reg.

These lines cause the following action to take place;

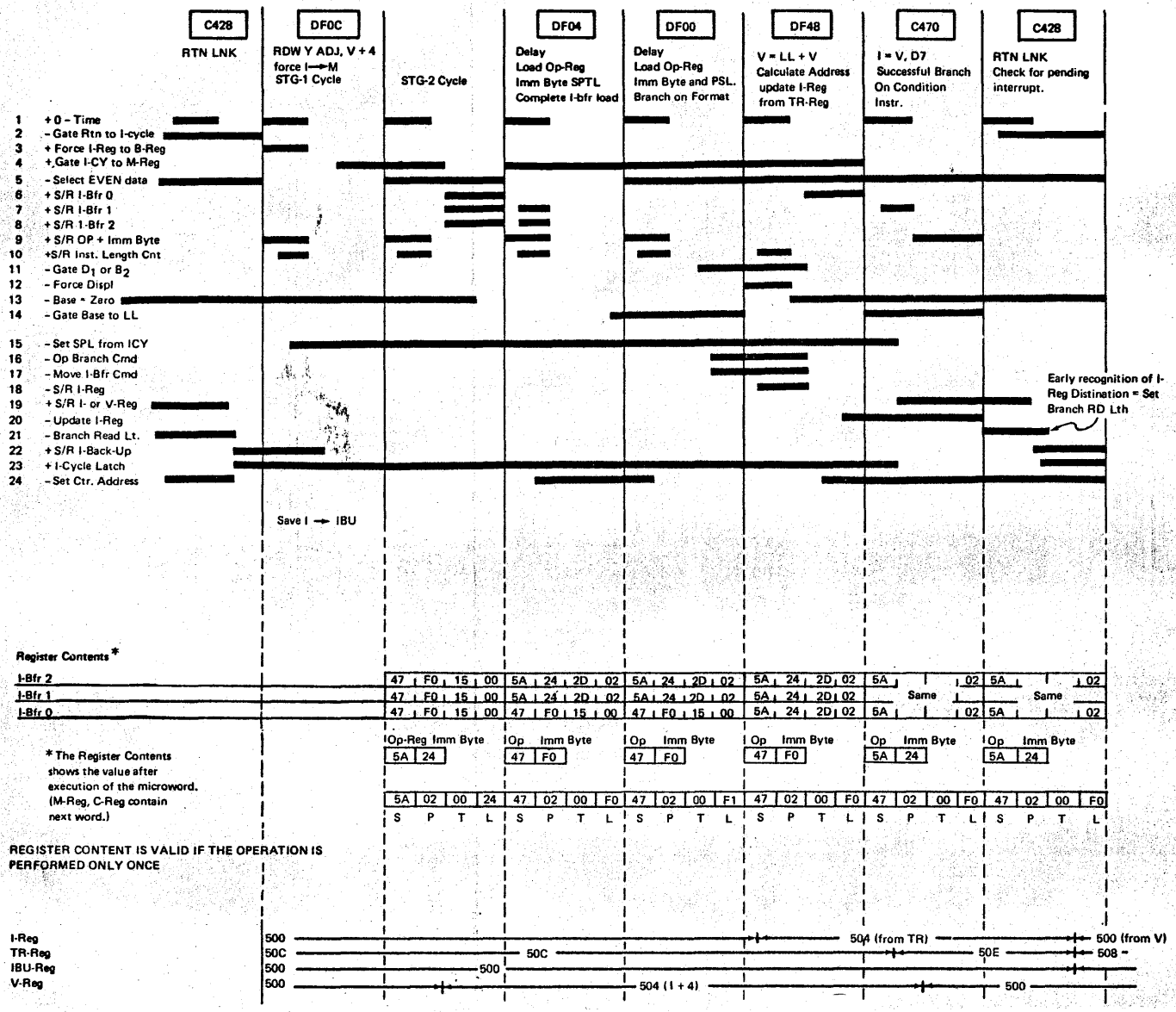
1. The data from storage is placed in the I-Bfrs and the Y-Reg equals I-Bfr 0.
2. The I-Reg data is transferred to the B-Reg, and the V-Reg equals the I-Reg + 4.

These lines are the decode of M-Reg byte 3 bits. The Ctrl Reg is the I-Cycle Control Reg. In effect, the decode is 0C.




EVEN
0
1
2
3
ODD
4
5
6
7
D-Reg
SDBO-Asm
EBI
0
1
2
3
Byte 0
B1, 2, 3
B3
KEY
0
7
I
EXP LS
50
8
1
2
3
29 31
I-Cycle Ctrl's
Add 'One'
ADD CARRY
ILC
8
1
2
3
28 29 30
Half-Sum Check
IBU
EXPLS
54
0
1
2
3
31
TR
EXP LS
55
8
1
2
3
29 31
B0, 1,2,3
V
EXP LS
51
0
1
2
3
B0, 1, 2, 3
W
EXP LS
52
0
1
2
3
B0, 1, 3
Hdw Set ILC
U
EXP LS
53
0
1
2
3
Instr. Length Code From I–Cycle Ctrls
B0, 1,2,3
I-BFR 2
0
1
2
3
I-BFR 1
P/C
0
1
2
3
I-BFR 0
P/C
0
1
2
3
Base
Displ.
CPU 79
A
CPU 78
B
U-Bus
FORCE I
Gates data through PAA
PAA
PBA
COMMAND BRANCH LOAD
Gates data through I BFRs 2 and 1 into 0.
To M-Reg
Address Assembler
A-LS
A-Asm
B-Asm
B-LS
A-REG
0
1
2
3
B-REG
0
1
2
3
ADDRESS COMPARE
ALU2
ALU3

Branch Loop Example

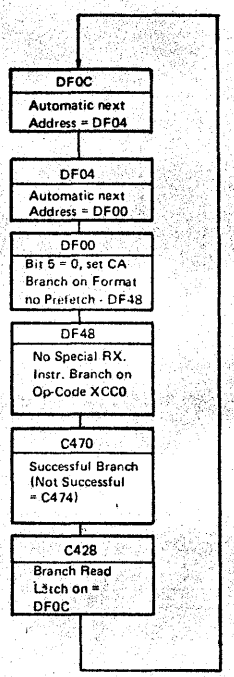


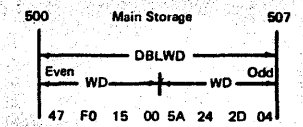
Register Contents*

I-Bfr 2	47	F0	15	00	5A	24	2D	02	5A	24	2D	02	5A	24	2D	02	5A	Same		02	5A	Same		02							
I-Bfr 1	47	F0	15	00	5A	24	2D	02	5A	24	2D	02	5A	24	2D	02															
I-Bfr 0	47	F0	15	00	47	F0	15	00	47	F0	15	00	5A	24	2D	02	5A			02	5A			02							
	Op-Reg		Imm Byte		Op		Imm Byte		Op		Imm Byte		Op		Imm Byte		Op		Imm Byte		Op		Imm Byte								
	5A	24			47	F0			47	F0			47	F0			5A	24			5A	24									
	5A	02	00	24	47	02	00	F0	47	02	00	F1	47	02	00	F0	47	02	00	F0	47	02	00	F0							
	S	P	T	L	S	P	T	L	S	P	T	L	S	P	T	L	S	P	T	L	S	P	T	L							

* The Register Contents shows the value after execution of the microword. (M-Reg, C-Reg contain next word.)

REGISTER CONTENT IS VALID IF THE OPERATION IS PERFORMED ONLY ONCE





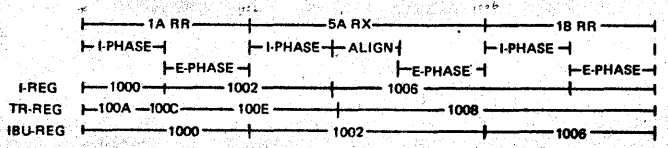
GR1 = 00000000

The base register is assumed to be zero for this example. This causes an unconditional branch to the first instruction (47). The ADD instruction (5A) is used only to provide data for the registers.

Partial Instruction Stream Cycles Example

The following instructions are contained in main storage.

1A 32 *5A* 41 2D 02 *1B* 46 (IC set to address 1000)



Add (1A) Instruction

The 1A instruction starts at address 1000 in main storage. To start processing at that address, assume that the set IC key is pressed. During the set IC microroutine, the I-Reg is destined. (This sets the instruction counter to address 1000.) The IBU-Reg address 1000 and the TR-Reg contains address 100A, an address within the next doubleword. Destining the I-Reg as a result of the set IC microroutine causes the branch read latch (RU 043) to set. Upon completion of the set IC routine, the start key must be preseed. This initiates the start micro-routine. At the end of the start microroutine is a I0 RTN LNK microword. The combination of the I0 RTN LNK and the M-Reg not being at address DFXX along with branch read latch (previously set on) forces the M-Reg to address DF0C. The M-Reg addresses control storage and reads out the control word at address DF0C.

CONTROL WORD AT DF0C (RDW Y ADJ, V + 4)

1. M-reg is at DF0C.
2. This is a RDW, force I to B and read the doubleword at MS 1000 and put it on SDB0.
   a. This doubleword (1A32 5A41 2D02 1B46) is in the SDB0 Pre-Asm early in storage-2 cycle (storage-1 cycle was addressing MS).
   b. Late in Stg-2 cycle, the even word from the SDB0 Pre-Asm is gated to the SDB0 Asm through EB1 to the I-Bfrs. I-Bfrs 0, 1, and 2 are "set/reset" (S/R). At the end of storage-2 cycle, all three buffers contain the same information (the even word).
   c. During the control word DF0C, the M-reg was at address DF0C. Through hardware, the M-Reg is forced to DF04 by using the gating line 'Generate Address' (*RU 031*).
   d. The M-Reg reads out the control word from control-storage address DF04.

CONTROL WORD AT DF04 – (Y = 0)

1. Early in the cycle, the odd word of the doubleword read out during DF0C is gated from the Pre-Asm to the SDB0 Asm through EB1 to the I-Bfrs. This time *only* I-Bfrs 1 and 2 are 'S/R'. I-Buffers 1 and 2 contain the same information. The odd word I-Bfr 0 contains the even word.

I-BUFFERS

2	2D02	1B46
1	2D02	1B46
0	1A32	5A41

2. The Op-code (RU 118) and the Imm byte (RU 128) are set into the Op-Reg and Imm byte Reg, respectively. (I-Reg indicates which byte to gate to the Op-Reg.)
3. The I-CY controls decode the Op-code and set the ILC in U0 Bits 0 and 1. The ILC is also made available to the add-carry. (The ILC is available to the add-carry at the time the I-Reg is updated.)
4. The Op-Reg decode and the I-Reg are used to determine whether the present instruction is fully contained in the buffers. The Op-code indicates the length of the current instruction, and the I-Reg indicates what part of the doubleword the current instruction came from. Therefore, through this combination of decoding the need for further fetching can be determined. (*Further fetch* is defined as the condition when the present instruction is not fully contained in the I-buffers.) In this case, no further fetching is required. The present instruction is fully contained in the I-Bfrs. The Op-load latch (RU 043) is not turned on.
5. The present instruction is fully contained in the I-Bfrs. The I-Bfrs must be decoded to see whether the next instruction is fully contained in the I-Bfrs. Instruction Look-Ahead is used to decode the format of the Op-code that is following the current instruction. The decoding of this format indicates the length of the instruction that follow the current instruction. The I-Reg keeps track of the address that the data came from within the doubleword. Through this combination of decoding whether the next instruction is fully contained in the I-Bfrs can be determined. If not, a prefetch is performed. (A *prefetch* is defined as the condition when the next instruction is not fully contained in the I-Bfrs.)The next doubleword is loaded using the TR-register contents as an address. If the I-Bfrs do not have enough room for the next doubleword, the prefetch is blocked. Prefetch is also blocked on all branch type O-codes. (In this case, a prefetch is not required.)

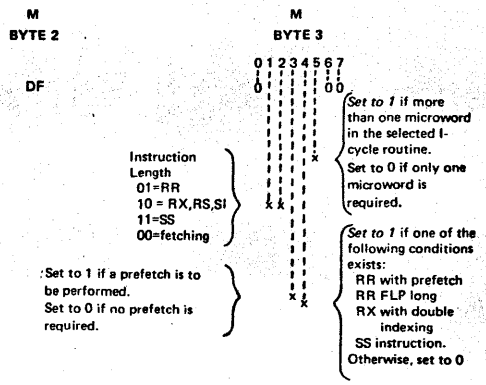
   The S- and L-Regs are set from the Op-Reg and Imm Byte Reg. This occurs during I-CY and when a cycle is taken. LL = operand 2. LH = Operand 1.
6. During the control word DF04, the M-Reg was at address DF04. The M-Reg is forced via hardware to address DF00, by gating line 'Generate Address' (RU 031).

   M-Reg at DF00, the M-Reg reads out the control word from control storage address DF00.

CONTROL WORD AT DF00 (Y = Y, S0)

1. This is a delay word that allows the hardware to develop the next address.
2. The gating line 'Set Control Address' (RU 032) is needed because a new sequence of control words is being started. Note that DF20 does not always follow DF00. In this case, the Op-code has been decoded to 1A and causes a branch to DF20.
3. During the control word DF00, the M-reg was forced to DF20.

Once the instruction is contained within the I-buffers, the next address is formed as follows:



M-Reg at DF20, the M-Reg reads out the control word from control-storage address DF20.

CONTROL WORD AT DF20 (Y = LL, DF) (the last cycle in I-Phase)

1. The word Y = LL, DF. (LL points to operand 2.)
2. The contents of GR2 are placed in the Y-Reg.
3. At the end of this cyle, the value to be destined to the Y-Reg is in the Z-Reg.
4. U2 and U3 are set from the Op-Reg and Imm-Byte-Reg, respectively, so that the information is available to the microprogram during E-phase.
5. Update I-Reg (I = 1002) (*Note:* TR is updated after the S/R to the I-Reg.)
6. At the conclusion of I-phase, the Op-code has been decoded (in this case 1A) and the next address is know (C1A0).
7. The M-Reg is forced to C1A0.

With M-Reg at C1A0, the M-Reg reads out the control word from control-storage address C1A0.

EXECUTE PHASE (I-CYCLES)

The control word at address C1A0 is read out (the first cycle of E-phase of the 1A instruction).

CONTROL WORD AT C1A0 - (LHC = LH + Y)

A. The following hardware functions are completed for I-cycles.
   1. S/R the Op-Reg and the Imm Byte Reg from the next Op-code and Imm Byte. The I-Reg was updated just before leaving I-phase; therefore, the I-Reg is pointing to the next instruction.
   2. In E-Phase, the Op-Reg and Imm byte Reg cannot be set into the S- and L-Regs. The system must be in I-cycles to gate the Op-Reg and Imm byte Reg to the S- and L-Regs.
   3. The TR-Reg is updated to point to the next double-word.

B. The control word read from address C1A0 performs the following.
   1. LH is pointing to GR3.
   2. Y has not been destined (Z = Y).
   3. At source time, the Z-Reg is gated back to the B-Reg controlled by Destination Look-Ahead, and the contents of GR3 are gated to the A-Reg. At the end of this cycle, the sum of GR3 and 2 is in the Z-Reg and is destined to GR3 in the next cycle.
   4. The 1A add instruction is now complete (during E-phase of the add instruction):
      a. The S-Reg is changed by Stat sets.
      b. The U2- and U3-Regs still contain the Op-code and Imm byte of the current instruction.
      c. The L-Reg still contains the address of operations 1 and 2.
      d. The microcode sets the condition code, test overflow, etc.
      e. While in E-phase of the 1A instruction, the I-CY controls are decoding the next Op-code to determine the controls required for the next I-phase operation and the first address that will be used upon returning to I-CY.

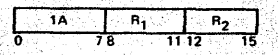
Example 2B Add (5A) Instruction (Double Indexing with Alignment)

1A 32 5A 41 2D 02 1B 46

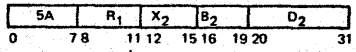
Operand 1 = GR4
Operand 2 = main storage addr 3D02

Add

AR R_1, R_2 (RR)



A $R_1, D_2(X_2, B_2)$ (RX)



The second operand is added to the first operand, and the sum is placed in the first operand location.

RX With Double Indexing and Alignment

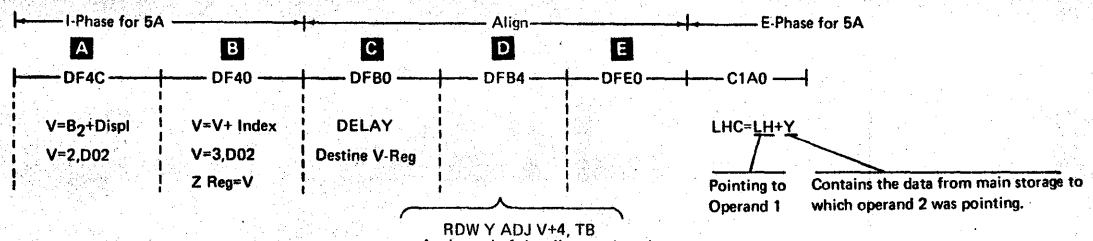
- The definition of *double indexing* is: Neither base nor index fields are using GR0 (zero). Another control word is needed to calculate the operand.
- Alignment: (operand 2 is pointing to a storage address; the data at that address is to be added to the contents of the GR that operand 1 is pointing to).

  Fetch data from storage (using operand 2).

  Align data in the GR.

  Both operands are now in local storage.

  The same routine may be used as in the RR Add to add the two operands because both operands are in local-storage registers.



*B2 gated to LL on a RTN LNK

At the end of the align routine, the data to which operand 2 was pointing is fully aligned in the Y-Reg. The V-Reg, which contains the main-storage address, must be taken through the address Asm because Destination Look-Ahead can not be used at this time. The DFB0 delay word allows time for this.

A *Developing address DF4C after E-Phase of the 1A Op-code.*
During E-phase of the 1A op code, the op-reg contains the next Op-code 5A. During this time, I-CY controls are setting up for the next address in the SF module by decoding the next op code 5A. The next address 4C is developed by the lines ILC, RX, RS, SI, SS, and IF MORE THAN ONE M-WORD. (The line 'if more than one M-word' is developed from the decode of double index.) After the execution phase of the Add (1A) operation is complete, the RTN LNK microword develops the gate line 'set control address' (not DF). 'Set control address' gates the developed address 4C to the M3 assembler, and gates DF to the M2 assembler. Rtn to I-CY gates the assemblers to the M-Reg, setting the M-reg to the next address DF4C.

At the end of DF4C, the V-Reg = Base + Displacement (2D02).

B The transition from address DF4C to DF40 is accomplished by using the decode of the I-CY Ctrl Reg 4C and gating line 'Generate Address'.

At the end of DF40, the V-Reg contains X + B + D (3D02).

C Address DF40 to address DFB0 (going to the Alignment Routine):

1. 'RX with alignment' develops the gating line 'Op-Branch to DF'.
2. 'Op Branch to DF' gates 'Align Entry' (in this example, the last control word in I-CY is not a prefetch. Therefore, the 'Low Bit Y' is zero and the Op code 5A develops a B) to the M3 assembler and DF to M2 Assembler.

'Gate I-CY to M-Reg' gates the assemblers to the M-Reg. The M-Reg contains DFB0, which is the first address in the align routine.

Being in the align routine deactivates the I-CY controls. This is done by M3 bit 0 being on, which deactivates 'generate address' latch and 'Generate Controls'. With the I-CY controls deactivated, the control words in the align routine develop the next address. (The exception to this is the first and last addresses of the align routine, which are developed by I-CY controls.)

Note the setting of U2-U3, updating I-Reg, etc., when going from DF40 to DFB0 (Op-Reg and Imm byte are set in the first control word of E-phase of the 5A Op-code.)

D The first control word in the align routine is a delay word to allow the destining of the V-Reg. This is necessary because the next control word 'DFB4' in the alignment routine is RDW Y ADJ, V + 4, TB.

The V-Reg is addressing storage, and 'ADJ' (hardware adjustment) specifies that the V-Reg must be taken through the Addr Asm to address storage. Destination Look-Ahead may not be used to get data to the Addr Asm. Therefore, DFB0 must be used to allow destining of the V-Reg so that it is available to the address Asm during 'DFB4'.

At the end of the alignment routine, the data to which operand 2 was pointing is fully aligned and is in local storage in the Y-Reg .

E The last address in the align routine is DFE0. M3 bit 0 being on and the line *'F0, E0 (M3)'* activate the gating line 'Op Branch' (on Op-code). Upon exiting the align routine, the Op code 5A is still in the op-code register. The gating lines 'Op Branch' (on Op-code) cause a hardware branch. LH is pointing to operand 1 (GR4), and the data to which operand 2 was pointing is aligned and in the Y-Reg. Branching to the RR add routine and adding the two locations in local storage is now possible. The first control word found in the RR add (at address C1A0) is LHC = LH + Y. Hardware takes the present op-code 5A and 'minus 4' from the left hex digit of the present op-code. Therefore, the present op-code (5A) is changed to op-code of 1A. This is how the I-CY controls set up the next address when leaving the alignment routine of some RX instructions and op-branch (on op-code) to the execute routine for a RR instruction.

MVC (D2) Instruction Example

MAIN-STORAGE
ADDRESS
1004

D2 0F 1D 01 | 2D 02 *47* F1 05 01 *1A* 5A

Further fetch required because the current instruction is not fully contained in the I-Bfrs.

I-Phase for MVC (D2) Requiring Further Fetch

- Set I/C to 1004.
- DF0C: (M-3 bit 5 is on, forcing the odd/odd time slot) late in storage cycle 2, I-Bfrs 0, 1, and 2 are S/R with the odd word.
- DF04: early in the cycle, I-Bfrs 1 and 2 are again S/R with the odd word. The Op-Reg, Imm byte, ILC and S and L are set. If the current instruction is not fully contained in the bfrs the 'Op latch' turns on.
- DF00: delay 'set control address'.
- DF14: 'Op load' latch on and 'set control address' caused generation of DF14. RDW: force TR to B-Reg (fetch next doubleword). Storage-2 cycle S/R Bfrs 1 and 2 comes from even word. Next, the 'generate address' gating line is needed.
- DF10: (delay) S/R Bfr 2 from odd word. Format the branch. A new sequence of words is being started in I-CY. Therefore, the gating line 'set control address' is needed. The op-code in the next address '6C' is developed by:
  a. ILC, and SS develop the '6'.
  b. More than one control word and the control line 'not RR or FLP with prefetch' develops the "C".
- DF6C W = LL + W (add base 1 + displ 1 to W-Reg) Block 'W' as a source and gate Displ. 1 from Bfr 0 to the B-Reg. LL points to base register. At the end of this cycle, the Z-Reg contains operand 1. B2 is gated from Bfr 1 to LL. (B2 must be in LL for the next address to calculate operand 2.) To develop the next address, the gating line 'generate address' is activated.
- DF60 V = LL + V, Block V as a source, gate displ. 2 from Bfr 1 to the B-Reg. LL is pointing to Base Reg 2. During this cycle, the W-Reg was destined. At the end of this cycle, the Z-Reg contains operand 2. 'Op Branch' to CD20. Leaving I-phase to E-phase, sets U2 U3. Update the I-Reg.(Update I-reg before the bfrs are moved, and set Op-Reg and Imm byte.) Move Bfrs (this overlaps into first cycle of E-phase).
- CD20: Set Op-Reg and Imm byte with Op-code and Imm byte of the next instruction. (The I-Reg indicates where the Op-code and Imm byte are in the Bfrs.)
- CD20: Set ILC (from next Op-code). Update TR. The S and L Regs are set from the Op and Imm byte because the addressing range is out of I-CY (DFXX Module).
- During E-phase of the MVC, the microprogrammer has the Op-code and Imm byte available in U2 and U3. The length is decremented in U3. The condition code is set in U0 bits 2 and 3. The S-Reg is changed by Stat sets. Move characters. During E-phase, the I-CY controls are decoding the next Op-code 47 to develop the first address that is to be used when returning to I-CY after E-phase of the MVC Op-code.
- XXXX - RTN LNK I1 - The I-CY controls provide an I-cycle starting address of DF48. If no interrupt is pending, this is the data value set into the M-register.
- DF48 - V = LL + V - Block V as a source, gate displ 2 from Bfr 1 to the B-Reg.LL is pointing to base Reg 1. 'Op Branch' to C470. Leaving the I-phase, set U2U3. Update the I-Reg and move the I-buffers.
- C470: Beginning of E-phase for branch instruction. V-Reg is destined with sum Base + Displacement.
- Note that a software branch has occurred. Refer to example 1 for any further explanation.

Execute 44 Instruction Example

The execute instruction causes one instruction in main storage to be executed out of sequence without actually branching to the object instruction. For example, assume that a move (SI) instruction is located at address 3820, with format as follows:

Machine Format

Op-Code	I_2	B_1	D_1
92	66	C	003

0 7 8 15 16 19 20 31

Assembler Format

Op-Code	D_1	B_1	I_2
MVI	3 (12),		X'66'

where register 12 contains 00 00 89 16.

Further assume that at storage address 5000, the following execute instruction is located:

Machine Format

Op-Code	R_1	X_2	B_2	D_2
44	1	0	A	000

0 7 8 11 12 15 16 19 20 31

Assembler Format

Op-Code	R_1	$D_2(X_2,B_2)$
EX	1,	0(0,10)

where register 10 contains 00 00 38 20, and register 1 contains 00 0F F0 99.

When the instruction at 5000 is executed, bits 24-31 of register 1 are ORed inside the CPU with bits 8-15 of the instruction at 3820:

Bits 8-15: 0110 0110 = 66

Bits 24-31: 1001 1001 = 99

Result: 1111 1111 = FF

causing the instruction at 3820 to be executed as if it originally were:

Machine Format

Op-Code	I_2	B_1	D_1
92	FF	C	003

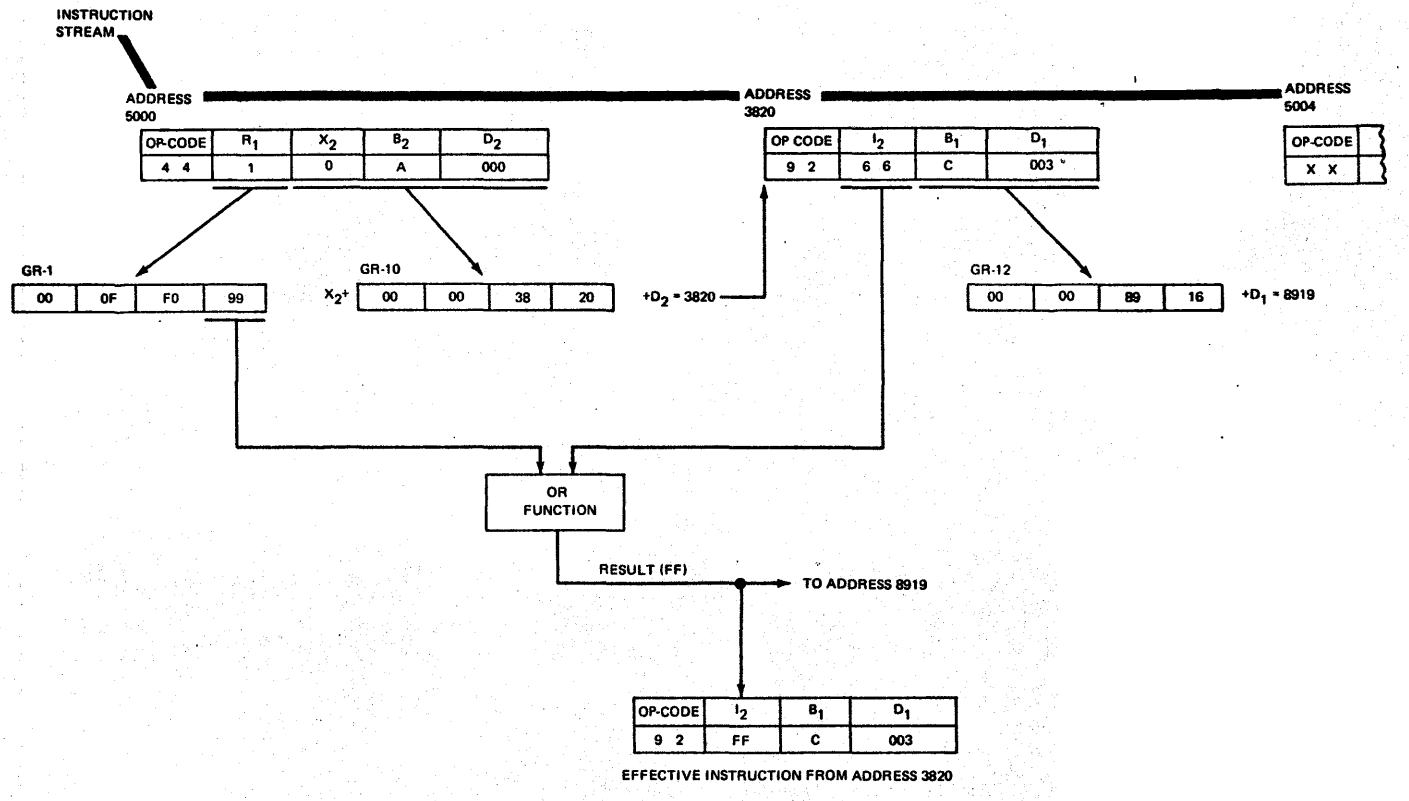
0 7 8 15 16 19 20 31

Assembler Format

Op-Code	D_1	B_1	I_2
MVI	3 (12),		X'FF'

However, after execution:

Register 1 is unchanged.
The instruction at 3820 is unchanged.
Storage location 8919 contains FF.
The CPU next executes the instruction at address 5004
(PSW bits 40-63 contain 00 50 04)



1. a. Assume that the instruction being processed had prefetched storage location 5000.
   b. During the E-phase, the I-cycle starting address of DF48 is provided as data input to M-Reg.
2. 10 for Instruction O-Code '44'

- Base 1 was gated to LL during the RTN to I-CY.
- I-Reg = 5000
- During E-phase of the previous instruction, I-CY controls recognized that the current and the next instructions are fully contained in the I-Bfrs. Therefore, the first address of I-CY for Op-code 44 is DF48.
- DF48 'set control address' and the lines 'ILC, RX, RS, SI, SS, and not RR or FLP with prefetch' cause address DF48 to be gated to the M-Reg.

  Note: Op Branch to DF. The three control words used for the 'Execute Instr' are referred to as *align*. That is, they are in the DF module. When entering these three words, the I-CY controls are inactive. The I-Reg is updated to 5004 (next sequential instruction).
- DFA4: 'Op Branch to DF gates out align entry A4.
- The word at DFA4 moves the GR specified by the R1 field to the W-Reg, and doing so:
  a. Sets the execute latch on (I- or V-Reg = V)
  b. Loads the immediate byte modifier reg and ORs it with Imm byte reg because R1 is not GR0. (This is done during I-phase of the instruction.)
- RTN LNK (DF84): The first address in I-CY is DF0C because the 'branch read' Latch was found on during DF84.

I-Phase for the Subject Instruction

- DF0C: The 'execute' latch is on, indicating that the V-Reg should be used in place of the I-Reg to address storage. The V-Reg contains the address of the subject instruction that was calculated as operand 2 during I-phase of the execute instruction.
- V + 4, used if further fetch is required to load the subject instruction.
- After E-phase of the subject instruction, return to DF0C and reload the Bfrs.

The I-Reg was not updated during the subject instruction; therefore, the next sequential instruction is loaded from the instruction stream (from address 5004).

A-REGISTER, B-REGISTER, AND ALUs

Arithmetic and logic operations in the CPU are processed by two one-byte arithmetic and logic units (ALUs).

ALU operations and program symbols:

Symbol	Operation
,A,	AND
,OR,	OR
,OE,	Exclusive OR
+	True ADD
–	Complement ADD
,D+–,	Decimal ADD
+–	Binary ADD
,A–,	Complement AND

The two input operands for the ALUs are entered into the A-register and the B-register and gated through their output assemblers.

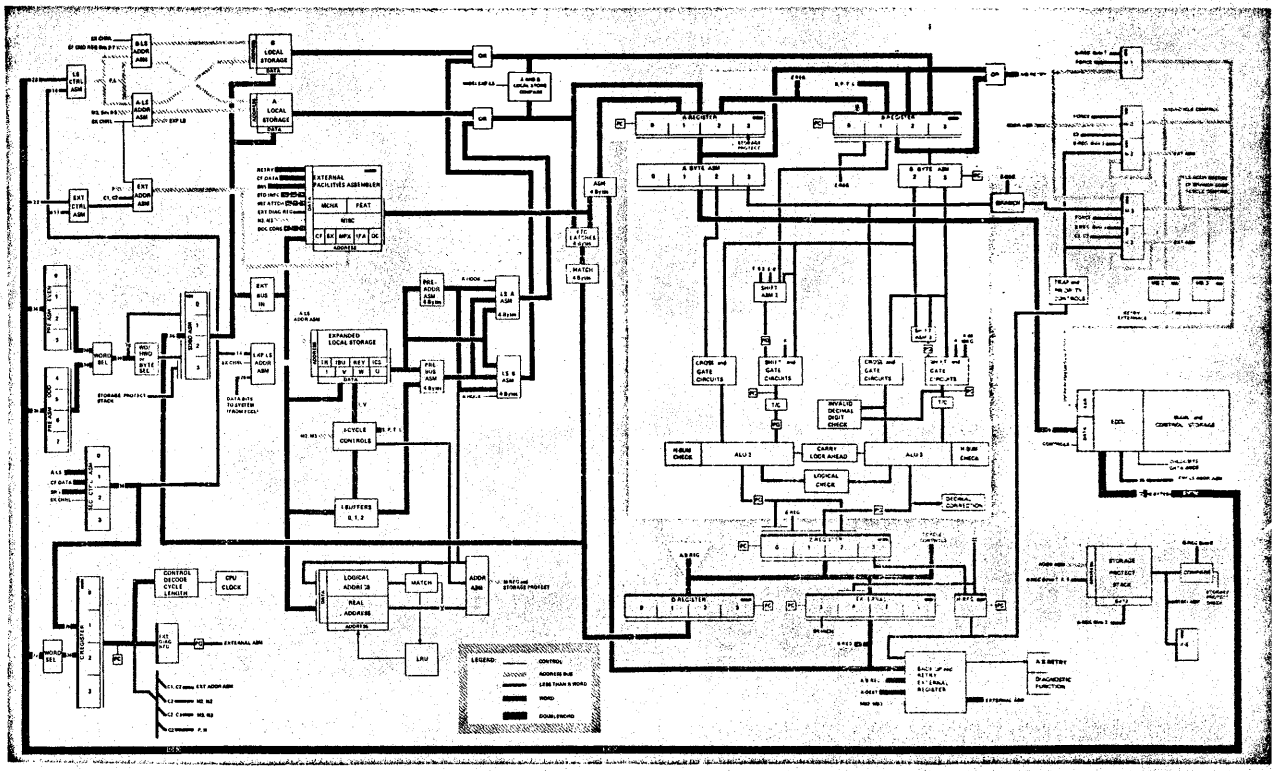
Two ALUs are provided to allow simultaneous operation in halfword operations and to provide checking during logic-byte operations.

By using two ALU cycles during a CPU cycle, the ALU system can binary-add two fullword operands.

Normal operations are for either byte operands or fullword operands as defined by the control word.

Special gating controls are provided in the entry of each ALU to allow manipulation of byte operands for logic and arithmetic operations.

The description of ALU operation in control words is given in the CPU Hardware (CPU)section.



A-REGISTER AND A-BYTE ASSEMBLER

The A-register is the A-source entry for the ALU system and enters data into main storage through SDBI.

Input data for the A-register comes from the A-local storage, the external registers, or previous ALU output from the Z-register.

An error condition is reported in MCKA byte 2 bit 4.

The A-register is a fullword (four-byte) register that normally enters the fullword.

The A-byte assembler provides the means to present the A-register bytes to the ALU system and to the SDBI.

The A-byte assembler has a four-byte output to allow assembling a fullword to feed the SDBI.

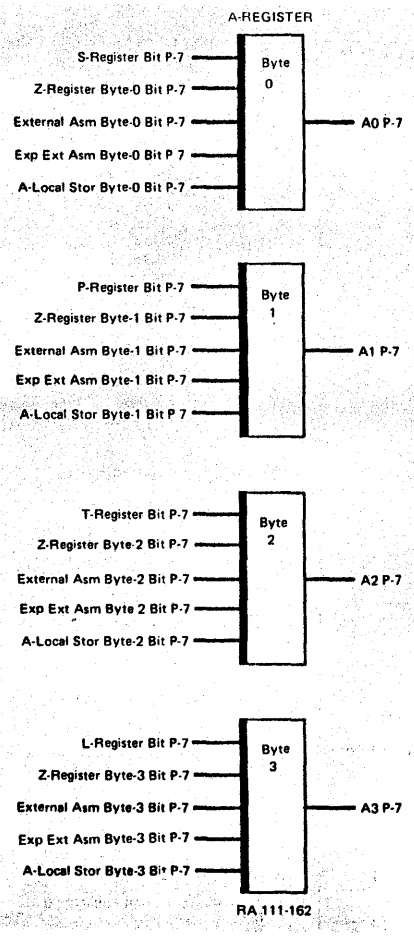
Only the byte 2 and byte 3 assemblers feed the ALU system and have gating to enter any of the four A-register bytes into either output.

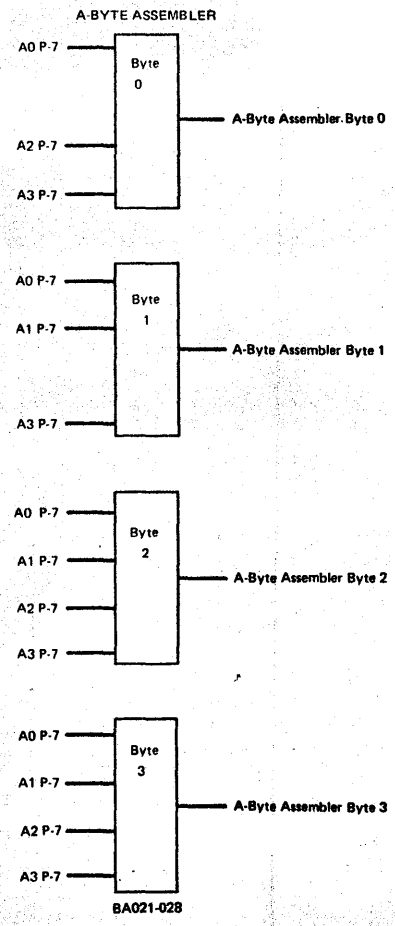
Byte 0 assembler cannot gate the A-register byte 1, and byte 1 assembler cannot gate the A-register byte 0 into their outputs, because they are not required in defined operations.

Parity is checked on all four bytes of the A-register during arithmetic fullword cycle, storage-word, and word-move cycles. On CPU byte cycles, only the byte being specified is checked at the A-byte assembler.

The high-order three bytes of the A-byte-assembler do not have a parity check but depend on the receiving area checks.

The byte 3 assembler is parity-checked because of its use in decimal adding, and the error condition is reported as an A-register error.





B-REGISTER AND B-BYTE ASSEMBLER

The B-register serves as the B-source entry for the ALU system and the data address entry for the M- and N-registers.

Input data for the B-register comes from the B-local storage, the A-register, the SPTL external, or a previous ALU output from the Z-register.

An error condition is reported in MCKA byte 2 bit 5.

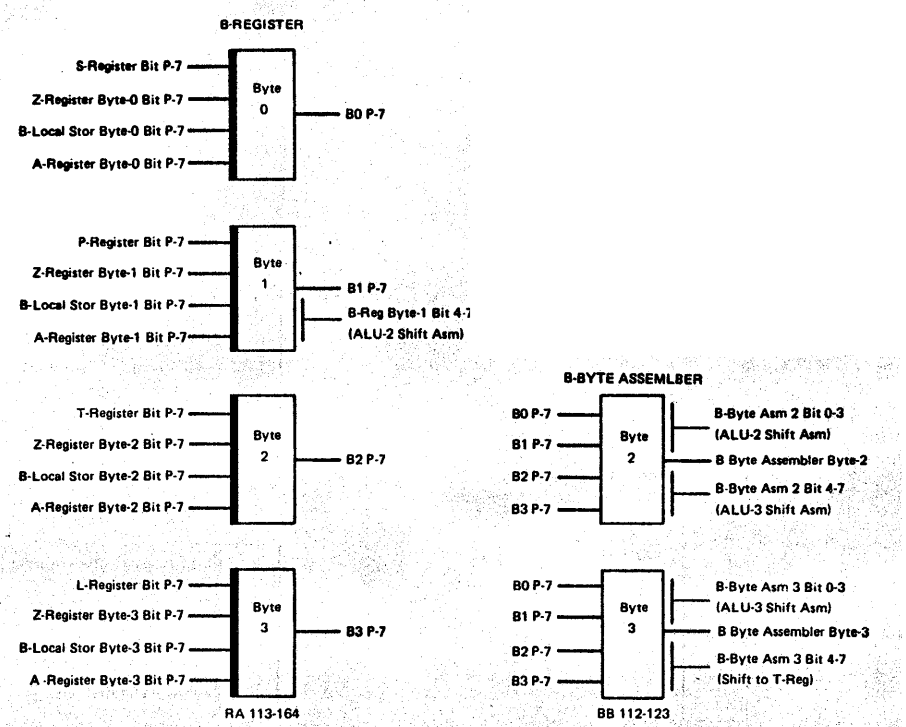
The B-register is a fullword (four-byte) register that normally enters a fullword.

The B-byte-assembler provides the means to present the B-register bytes to the ALU system.

The B-byte-assembler can gate any byte of the B-register into both byte 2 and byte 3 outputs to feed the respective ALUs.

Parity is checked in all four bytes of the B-register during arithmetic fullword, storage word, and word-move cycles. On CPU byte cycles, only the byte being specified is checked.

The B-byte assembler has a parity check on both outputs, and an error condition is reported as a B-register error.



ALU A-ENTRY GATING

The logic requriements of the system make it necessary to be able to block or transpose (cross) a part or all of the A-entry byte operand.

The high (0-3) and low (4-7) portions of the operand are separately gated to allow moving a portion straight to the ALU or to cross high and low in entry to the ALU.

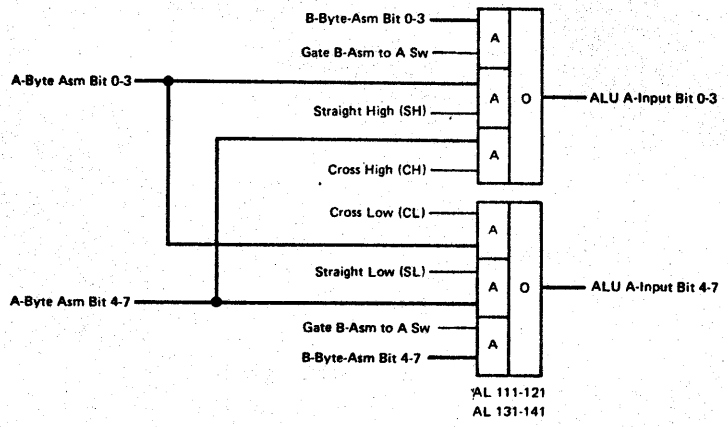
When none of the ALU entry gates are activated, the entry is blocked and the operand is presented to the ALU as zeros.

For a normal entry, both straight gates are raised and the full byte is entered.

For a crossed or transposed entry, both cross gates are raised.

For operations requiring only one portion of the operand, the appropriate gate is raised to enter the desired portion into the specified position of the ALU with zeros in the ungated portion.

ALU A-ENTRY GATING



Op Sym	Operation	Entry HHHH	Byte LLLL	Gate SH	CH	CL	SL
BS	Straight	HHHH	LLLL	x			x
BS0	Block High and Low	0000	0000				
BSH	Gate High; Block Low	HHHH	0000	x			
BSL	Gate Low, Block High	0000	LLLL				x
BSX	Cross High and Low	LLLL	HHHH		x	x	
BSXH	Cross; Gate High; Block Low	LLLL	0000		x		
BSXL	Cross; Gate Low; Block High	0000	HHHH			x	

ALU B-ENTRY GATING

The logic requirements of the system make it necessary to be able to block all or part of the B-entry byte operand and to enter the shift and K factors.

The high (0-3) and low (4-7) portions of the operand are separately gated to the ALU entry in order that one or both portions may be blocked and entered as zeros for the operand.

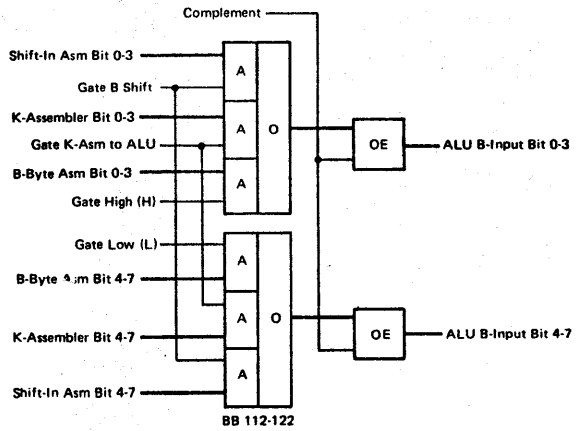
For a normal entry, both gates are raised and the data is transferred straight to the ALU.

When the normal entry is blocked, the byte developed in either the shift assembler or the K-assembler can be gated.

The ALU B-entry has a true/complement gating that reverses the binary bit levels of the operand byte when the complement line is raised.

The complement line is under control of the minus operation sign or the presence of the S0 bit when the operation sign is ± in the control word.

ALU B-ENTRY GATING



Op Sym	Operation	Entry HHHH	Byte LLLL	Gate H	L
BS	Straight	HHHH	LLLL	x	x
BS0	Block High and Low	0000	0000		
BSH	Gate High; Block Low	HHHH	0000	x	
BSL	Gate Low; Block High	0000	LLLL		x

SHIFT GATING

Special ALU entry gating is provided to allow the two-cycle right shift (four bits) operation.

During both cycles, the low-order four bits of byte 2 and the high-order four bits of byte 3 are assembled as a byte for the B entry of ALU-3.

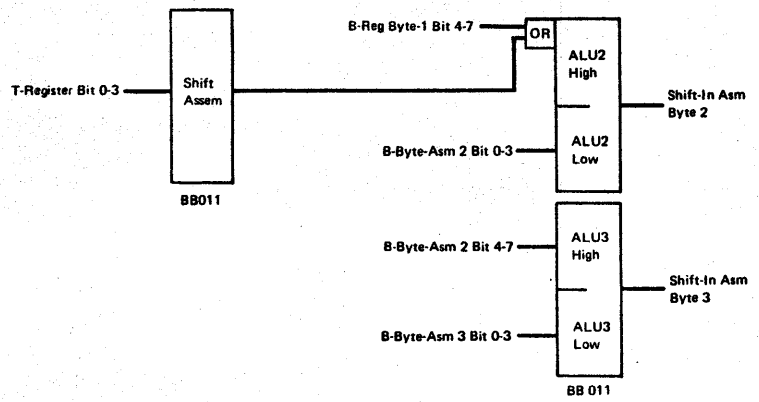
During the first cycle, the high-order four bits of the T-register are set into buffer latches in the shift assembler, and the low-order four bits of byte 3 enter the T-register.

During the first cycle, the low-order four bits (4-7) of byte 1 and the high-order four bits (0-3) of byte 2 are assembled as a byte for the B-entry of ALU-2.

During the second cycle, the buffered T-register bits and the high-order four bits (0-3) of byte 2 are assembled as a byte for the B-entry of ALU-2.

The A-entries for both ALUs are blocked and enter zeros for both cycles.

SHIFT ASSEMBLER



ALU K-ASSEMBLER

The K-assembler for the ALU gates fixed constant and constants defined by the control word to the B-entries of the ALUs.

Byte 2 of the control word in the C-register can be gated directly as a full byte to the ALUs.

The low-order four bits (4-7) of byte 2 in the C-register can be gated to the low-order (4-7) of the K-byte with zeros in the high-order (0-3).

The low-order four bits (4-7) of byte 2 in the C-register can be gated to the high-order (0-3) of the K-byte with zeros in the low-order (4-7).

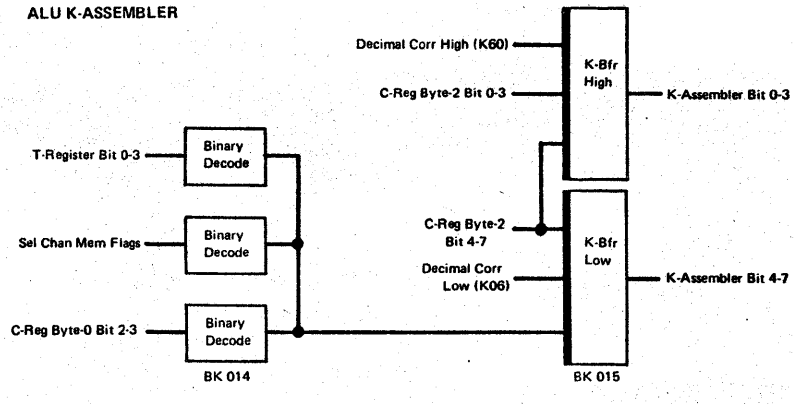
The high-order four bits (0-3) of the T-register define the bytes to be read or stored are converted to a binary count byte for adjusting the address and the count.

The four memory flag bits from the selector-channel buffers are converted to a binary count byte for adjusting the address and count during share cycles.

When C-register byte 0 bits 2 and 3 are used to define the size of the data transfer in a storage word, these bits are converted to a binary count.

For a decimal operation requiring binary-to-decimal adjustments, K60 and K06 are used as inputs on the second pass. The resulting output developed is K60, K06, or K66.

ALU K-ASSEMBLER



ARITHMETIC AND L[illegible]C UNIT (ALU)

The arithmetic and logic unit (ALU) performs the logic manipulation and adding operations in the CPU.

Two ALU units are provided to allow halfword binary and word-move operations in one pass.

Two ALU passes can be made during a CPU cycle to complete a fullword binary or word-move operation.

Each ALU consists of the A- and B-entry gates, the logic and arithmetic circuits, and output gating to position the output byte in the Z-register.

ALU-3 entry lines are checked for invalid decimal digits to ensure a correct decimal output with errors reported in S1 bit.

A carry look-ahead circuit is shared by the ALUs to allow simultaneous arithmetic processing of two successive digits or bytes.

During logical operations, both ALUs are fed with the same data, and a logical check circuit compares the results and reports errors through MCKA2 bit 2

The ALU logic circuits develop four outputs for any input presentation, and gates defined by the control-word operation select the appropriate output to set the Z-register.

Three of the ALU circuits provide the logic AND, OR, and OE outputs; the fourth combines these with the carry inputs to develop a full-sum output used for binary arithmetic.

The A-logic function is performed by raising the complement line in the B-entry and gating the AND output.

A parity prediction (generator) circuit on each ALU develops an output parity based on the inputs and the operation and is proven by the parity check of the Z-register.

In word operations the ALU-2 output is set into Z0 and Z2 and ALU-3 output is set into Z1 and Z3 on the first pass with an update of Z0 and Z1 on the second pass.

For byte operations ALU-3 output is set into all Z-register bytes to allow the SDBO byte assembler to enter the byte into any position.

Half-Sum Checking

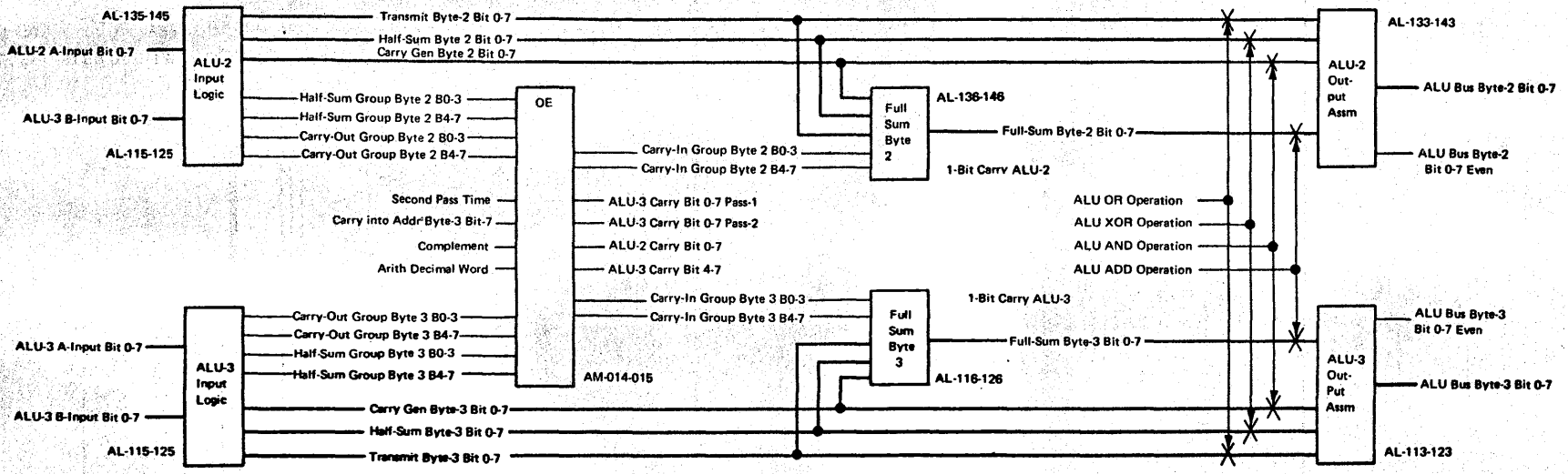
The half-sum lines (OE) developed from the two ALU operands are tested against the parity bits of the two operands to check the parity of the entry data.

Each half-sum line indicates the odd/even relation of a bit position within the operands.

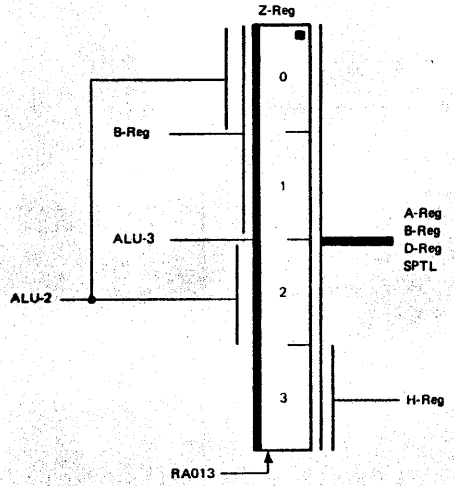
Developing an odd/even count from the eight half-sum lines indicates the level of the entry data bits (16) that should check with the level of the entry parity bits.

A separate latch is set for each pass and each ALU, but the ALU-2 indications are blocked for decimal operations.

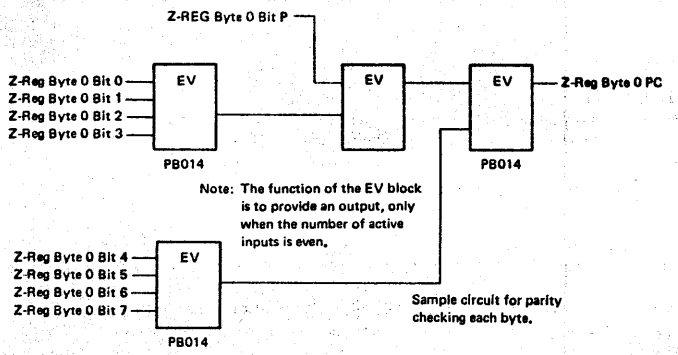
A detected error sets an ALU check indicator in MCKA2 (bit 0 for ALU-2; bit 1 for ALU-3).



Z-REGISTER and D-REGISTER



Z-REGISTER PARITY CHECKING

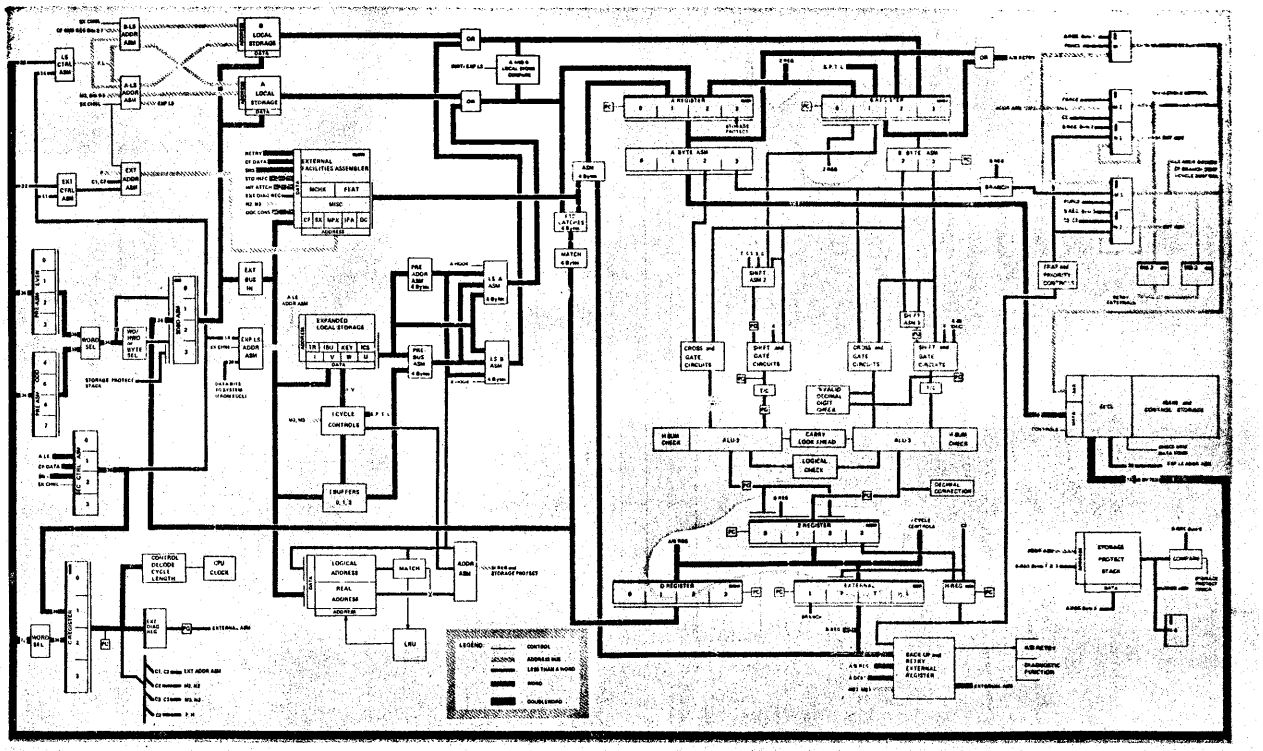


Z-REGISTER

The ALU results are set into the four-byte Z-register. The ALU result data can then be routed to:

- The S, P, T, or L-register
- The A- and B-registers
- The D-register

Z-register data (ALU result) is tested, if so specified in the control word being executed, to set or reset S-register bits.



The direct path from the Z-register to the S, P, T, and L-registers permits the setting of these externals earlier than other external facilities. This capability is necessary because these registers are frequently used by the next control word for status information and for local-storage and external-register addresses. Data destined to all other facilities is set into the D-register and destined during the next control-word cycle.

ALU data set into an external register (other than S, P, T, or L) in one control-word cycle is not available as source data for the next control-word operation.

One-byte Ops feed all four bytes of the Z-Reg with the same data.

A path is provided from the Z-register to the A- and B-registers. This path allows ALU data destined to local storage (not externals) by one control word to be used as source data by the next control word.

Example:

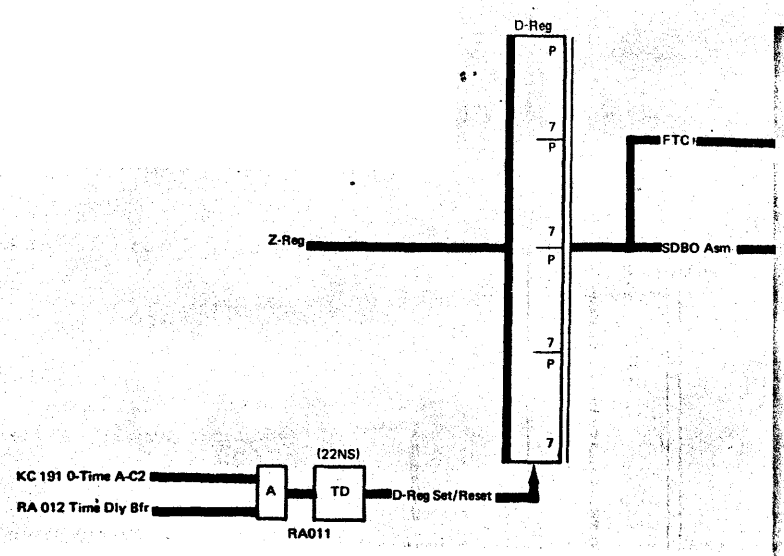
A control word is executed, and the result is gated to the R-register.

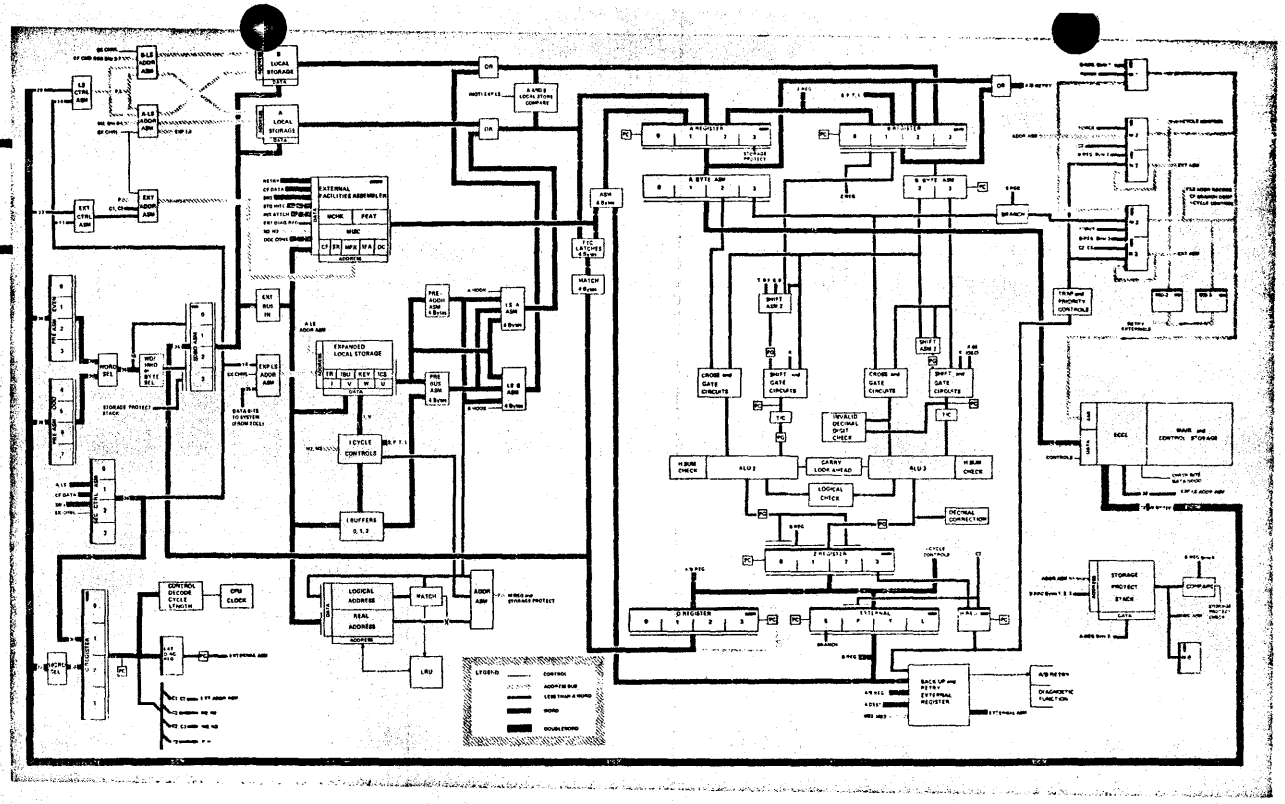
The next control word requires the R-register as the B-source.

The Z-register to B-register gate is activated because the result of the previous control word has not been set into the R-register. The Z-register is also gated to the D-register in normal fashion.

Refer to "Local Storage, Destination-Look-Ahead" on CPU 11 Storage.

Any combination of byte selection is possible. For example: if only byte 2 of a local-storage location is altered, then only that byte is routed to the A- and B-registers from the Z-register. Bytes 0, 1, and 3 come from the local-storage location addressed by the control word being executed. Consequently, no matter which bytes of a local-storage location are altered and destined by a control-word operation, the altered data is available as source data for the next control-word operation. Address information derived from the control word being executed determines whether Z-register data is routed to the A-register, the B-register, or both.

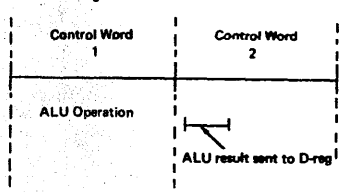




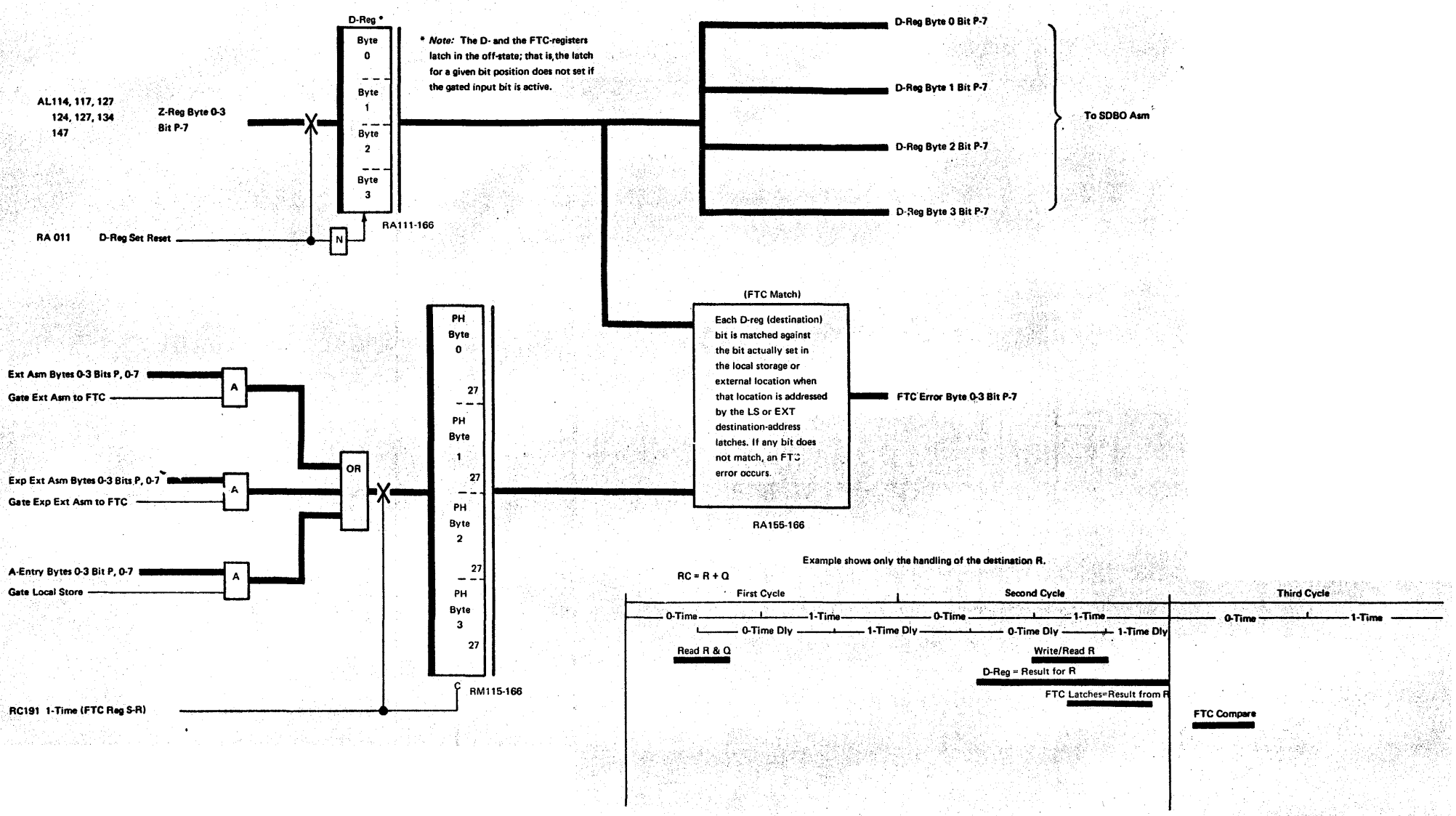
D-REGISTER

The four-byte D (destination) register is set to the ALU result data (from Z-register) early in the next control-word cycle.

The D-register data is sent to the SDBO assembler, where it can be routed to externals (via external bus-in) or to local storage.



D-REGISTER and FLUSH-THROUGH-CHECK (FTC) REGISTER



Example shows only the handling of the destination R.

C-REGISTER (CONTROL WORD DECODE)

C-REGISTER

The C-register contains 32 bits plus four parity bits. The purpose of the C-register is to decode the control word and provide control and gating of CPU functions. Once read out of control storage and gated to the C-register, the control word is decoded to determine:

- Word type
- CPU function
- CPU clock cycle and length

The functions of the bytes of the C-register are:

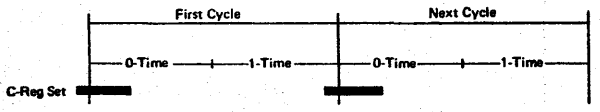
C0 Define word type and format Branch High Address (M3 B4)

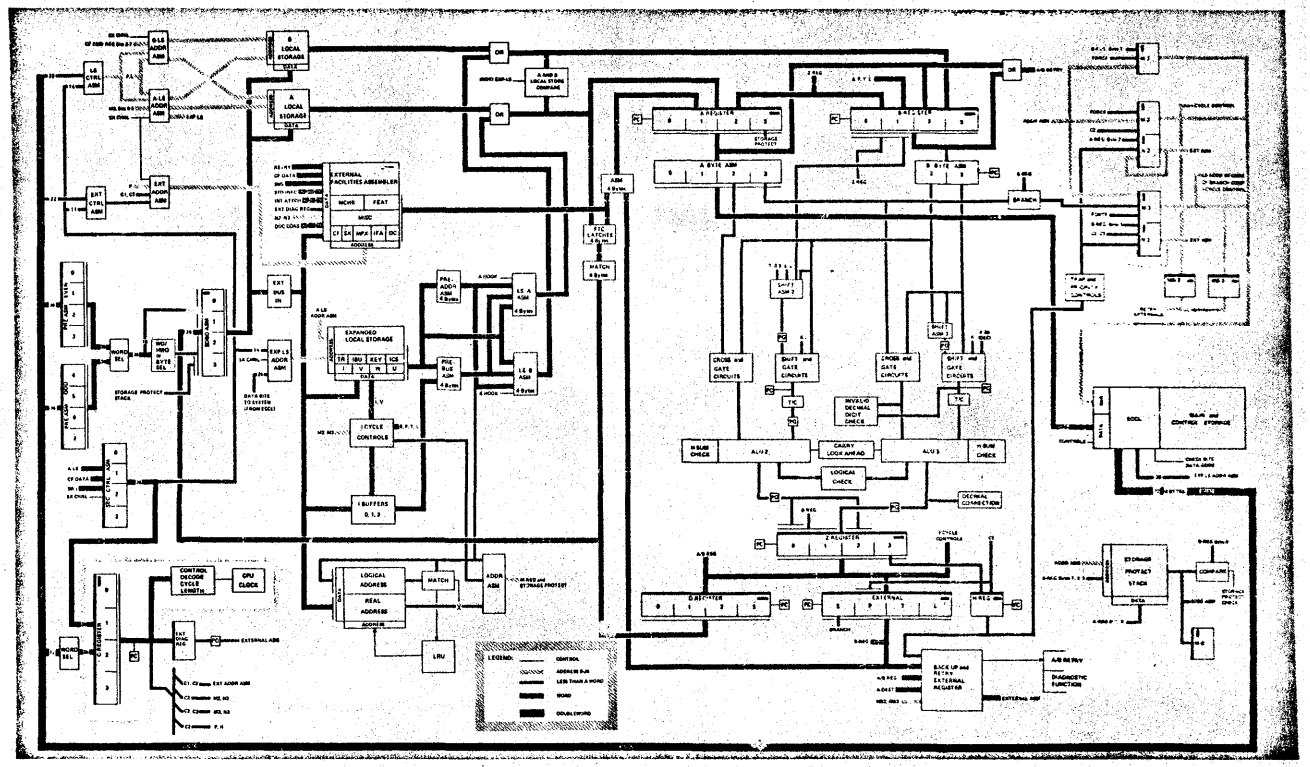
C1 Specify A-source or destination (an external register or an A-local-storage address). Stat sets and/or special functions.

C2 Specify a B-local-storage source or destination. Contains Mask or K values special Stat sets.

C3 Specify next control-word address Branch Low Address (M3 B5).

The C-register is set by 0-time of each control cycle.





CPU CLOCK

CPU CLOCK OSCILLATOR

Timing for the 3145 is developed from a 22.222-MHz crystal-controlled oscillator. This oscillator is fed to an oscillator control card, where it goes through one frequency divider stage and is controlled for distribution to each board clock. From this, each board clock then develops six basic timing signals to time the CPU circuitry.

CPU CLOCK TIMING

This clock is a variable-cycle clock that is designed to operate 180-, 225-, and 270-nanosecond cycles, with each cycle having the further capability to extend by 22.5-nanosecond increments (referred to as pauses). This then allows (with one pause) 202.5 ns from a 180-ns cycle; 247.5 ns from a 225-ns cycle; and 292.5 ns from a 270-ns cycle. With two pauses, a cycle may be extended 45 ns as in the case of the Storage-1 Cycle Write. (270 becomes 315.) If the cycle after a Storage-2 cycle is a selector channel share cycle, the pause in storage-2 cycle is eliminated (270-ns cycle).

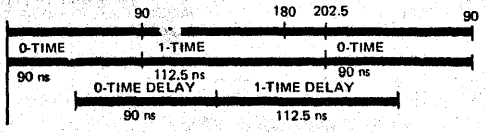
The CPU control word decodes determine cycle length. They provide 180-, 225-, and 270-cycle control signals that determine the cycle with which the clock should operate and the number of pauses the cycle should contain.

The CPU clock runs for one cycle under control of the clock start latch. The clock start latch has many input controls (start switch, set IC, CF clock start, etc.).

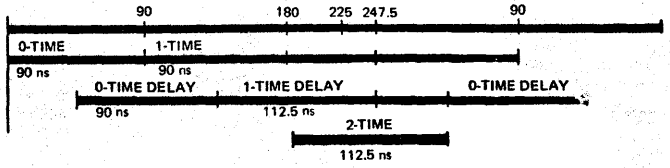
The clock consists of six latches operated in an overlapped configuration to produce six timing pulses.

180-Cycle (202.5)	0-Time 0-Time Delay 1-Time 1-Time Delay
225-Cycle (247.5)	0-Time 0-Time Delay 1-Time 1-Time Delay 2-Time
270-Cycle (292.5)	0-Time 0-Time Delay 1-Time 1-Time Delay 2-Time 2-Time Delay

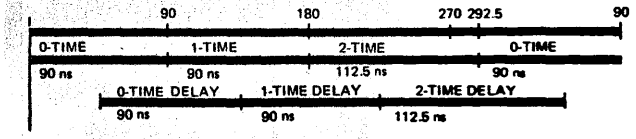
180-CYCLE 202.5 ns



225-CYCLE 247.5 ns

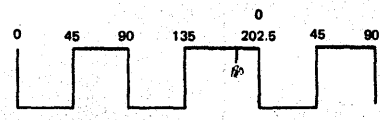


270-CYCLE 292.5 ns

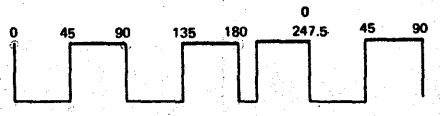


OSCILLATOR SIGNAL FROM OSCILLATOR CONTROL CARD

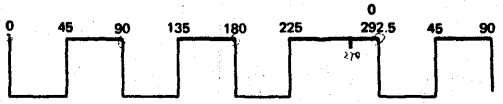
180-CYCLE 202.5 ns



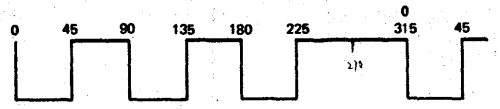
225-CYCLE 247.5 ns



270-CYCLE 292.5 ns



STORAGE-1 CYCLE WRITE 315.0 ns



CPU Clock Checks and Adjustments

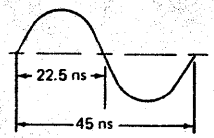
Equipment Required

Tektronix* type 454 oscilloscope, or equivalent. Oscilloscope probes of equal length and equal attenuation.

Note: All measurements should be made with the displays centered on the scope face.

Oscillator

Oscillator frequency is 22.5 MHz. Symmetry and frequency are checked at 01AB2L2S05.



Clock

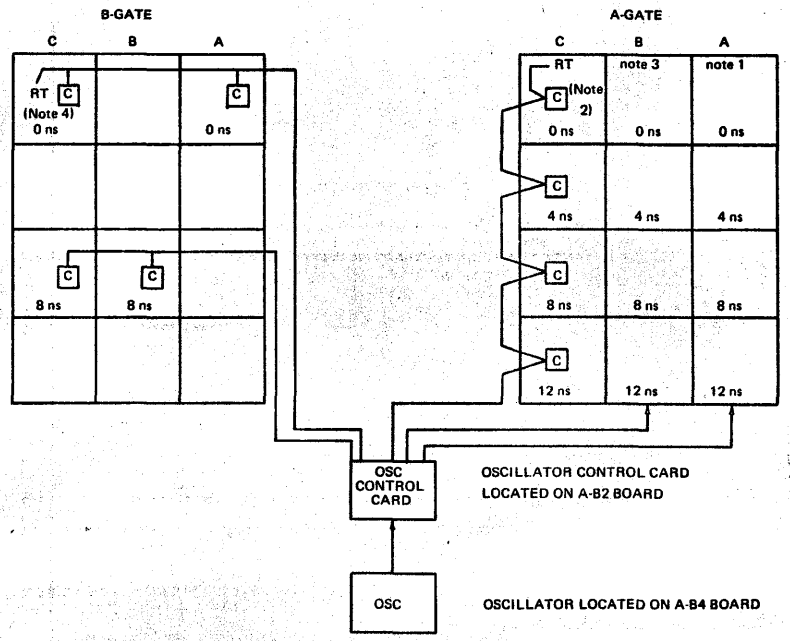
The CPU clocks are initially synchronized at the factory and should be readjusted only when additional boards are installed, a feature is added to a board, or a clock card is replaced.

Each clock has a programmable delay-line adjustment. Pin G10 of each clock card (6735) is the oscillator test point. To ensure oscillator synchronization during the following adjustment, sync negative on the G10 test point.

The oscillator signal should now be synchronized for the system. Check the oscillator test point on each board. If the signal is found to be more than ±2.0 ns out of sync, resynchornize the clock on the card that is out of sync.

1. To determine 'late clock' sync:
   a. Set rate switch to SINGLE CYCLE HARD STOP (CLOCK STOP indicator on).
   b. Ensure that zero delay is plugged in the clock cards in boards A-A1, A-B1, and A-C1.
   c. Oscilloscope settings
      (1) A sweep Time/Div = .02 us.
      (2) X10 multiplier on.
   d. Sync channel 1 (minus) on clock card at A-C1G2G10 and display the signal.
   e. Using channel 2 to display the G10 pins of the clock cards in A-A1 and A-B1 boards, determine the latest (in time) of the three clocks.
   f. Place the channel 1 probe on the latest of the three clocks. Channel 1 is now synchronized on 'late clock'.

Note: By swapping the input signals at the oscilloscope, verify that the oscilloscope is calibrated. The relationship between the two signals must not change. If signal relationship does change, use another oscilloscope before continuing with this adjustment.



Notes:
1. These numbers are initial programmable delay settings.
2. Select has a separate oscillator signal.
3. For IFA version 003 machine, clock is located in B1B4 socket on the A-Gate.
4. Clock-card position is feature-sensitive. Refer to the KC ALD pages.

2. With channel 1 sync and display on 'late clock', display all other clocks. If any clock is more than 1.0 ns earlier or later than 'late clock', change the programmable delay line for that clock to being it to within ± 1 ns of 'late clock' (negative pulse at G10).

CPU Clock Locations

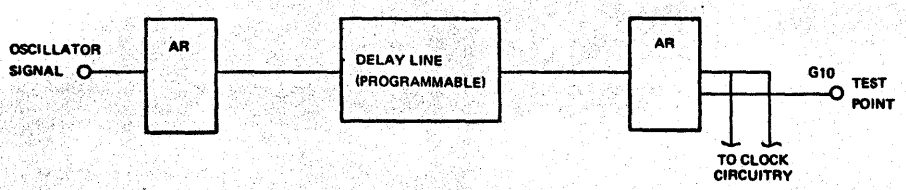
Gate A-C1G2 C3G4 B1C4 (note 3) B3H4 A1K2 A3C4
C2J2 C4E2 B2M2 B4K2 A2C4 A4Q4
Gate B-A1C4 B1 C1M2 (note 4) B3V3 C3J2

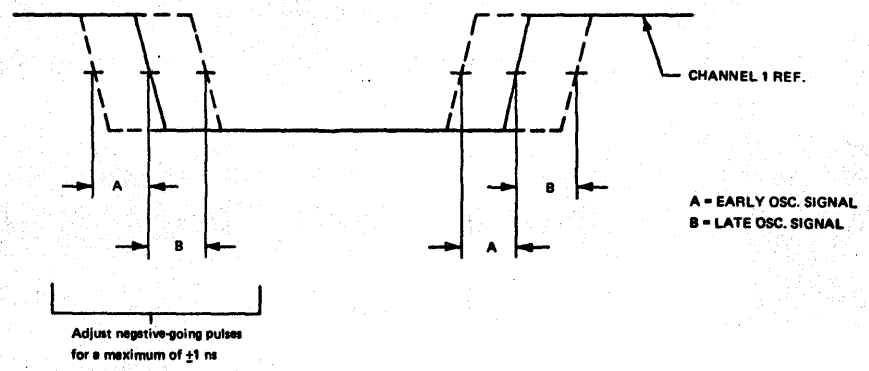
Oscillator Location:

Oscillator and oscillator control card location: A-B4A3
A-B2C2

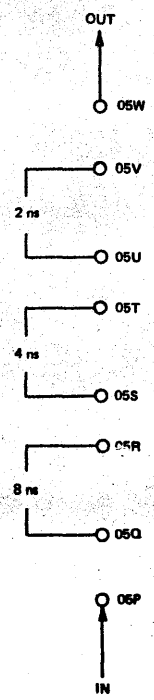
*Trademark of Tektronix, Inc.

CIRCUITRY LOCATED [redacted]ACH CLOCK CARD AND CLOCK-START CONTROL CARD





PROGRAMMABLE-DELAY LINE LAYOUT



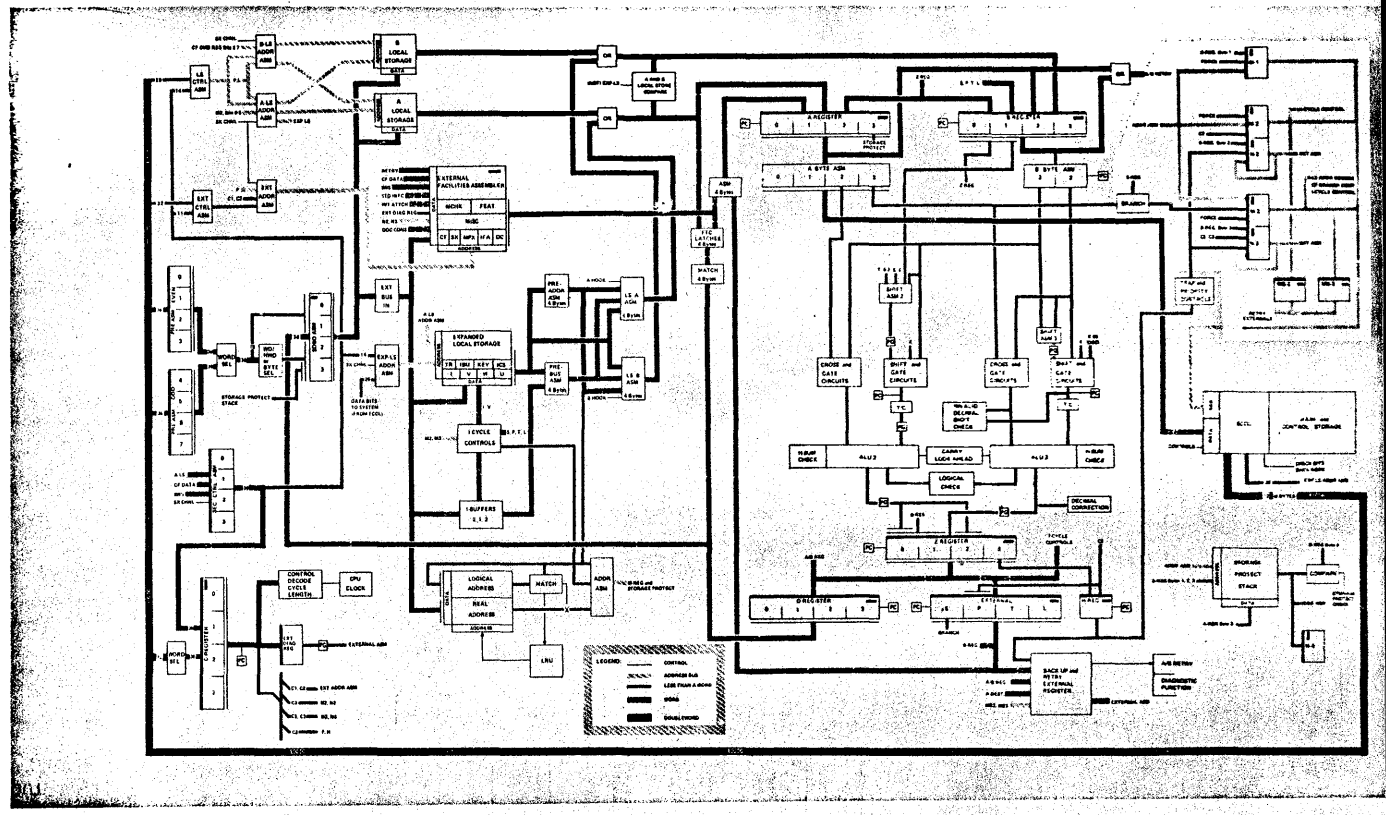
CLOCK CARD (6735)

M, N, and MB-REGISTERS

M-REGISTER

- Addresses main and control storage.
- Feeds storage address buses (SAB).
- Made up of M1, M2, and M3, which provide a 21-bit (plus 3 parity bits) storage address.
- M1, M2, and M3 address both main and control storage. Storage is read out on a doubleword boundary and stored on a word boundary.
- M3 bits 5,6, and 7, and the storage word being executed, provide the following selections for storage-word operations.

Read	Store
Odd/Even word, Halfword, or Byte	Word, Halfword, or Byte *Note:* Information may be stored under mask (any byte selected).



Setting M-Register for [redacted] Storage Addressing

- M1, M2, and M3 are set from the ADR/ADJ circuits or B-Reg bytes 1 (bits 4-7), 2, and 3.
- M1, M2 may be forced to zero for direct main-storage addressing.

Setting M-Register for Control Storage Addressing

- M1 is set to zero for display purposes. A line address CTRL store is sent to the ECC board.
- M2 selects the control-storage module and is set from C2, trap circuits, or N2 (no module-switch function). M2 may be forced to FF for direct control-storage addressing.
- A module is defined as a group of 64 words in storage. This increment of storage has particular significance in the control storage addressing structure. The address in M2 selects the specific module. Any word in the 64 word module can be specified by the address in M3 bits 0-5. Most control word operations cause only M3 to be changed.
- M3 selects one of 64 words within the module selected by M2. M3 is set from C2 (K-adr STW), C3, or trap circuits.

N-REGISTER

- Made up of N2 and N3.
- Backup register for control-storage addressing.
- N2 is set with the same information as M2 and is changed only when the control word being executed performs a module-switch function.
- N3 is set with the same information as M3.
- N is not changed when a trap occurs.
- When a trap occurs, the M-register is set to the trap address. The trap routine stores the contents of N (the N-Reg contains the next address that would have been used had the trap not occurred). At the end of the trap routine, M and N are restored to their original value so that the control-word sequence may continue as if there had not been any trap.
- N2 sets M2 for every control-storage word access except when a module switch occurs.

MB-REGISTER

Made up of MB2, MB3.

Set with the control-word address in M2, and M3 from M2 BFR and M3 BFR.

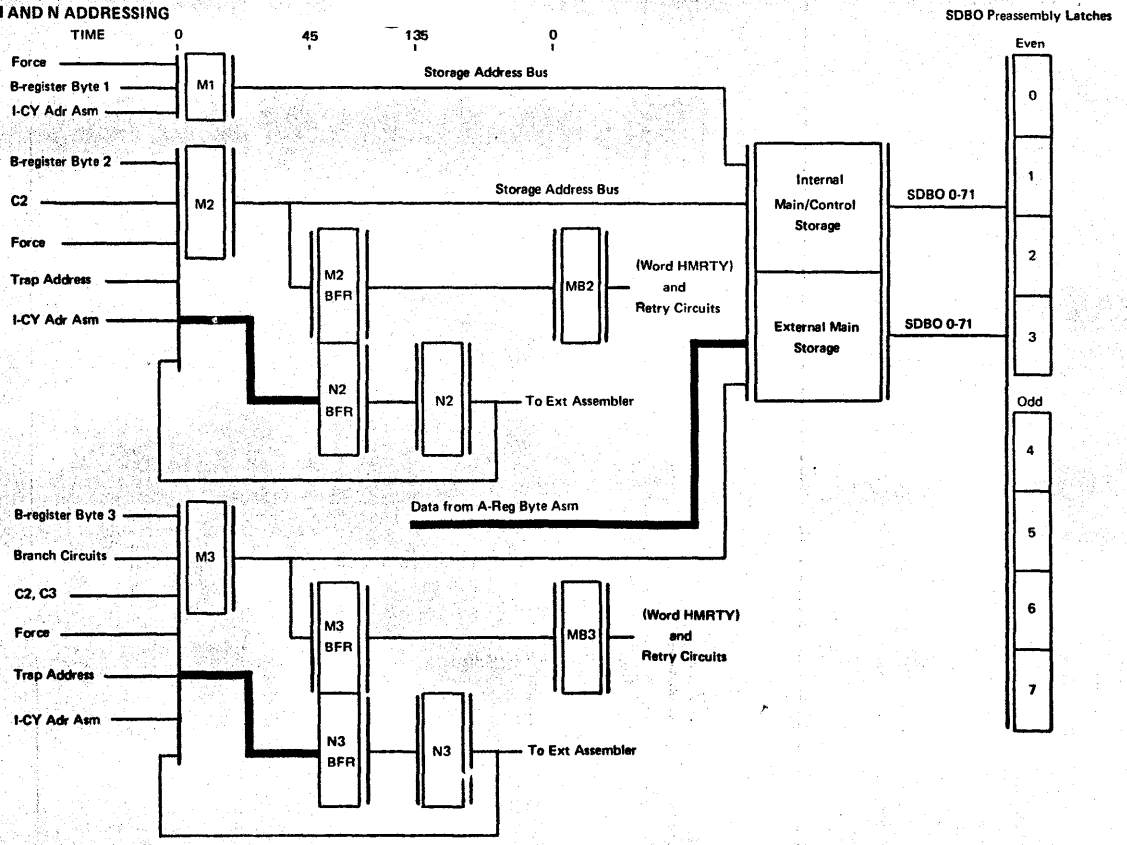
When the CPU clock is stopped, MB contains the address of the last word executed.

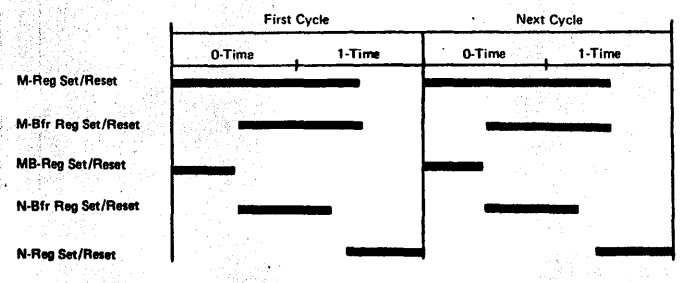
MB-reg output is available to the retry and backup circuits as well as the external assembler (word RTY).

Buffer Registers

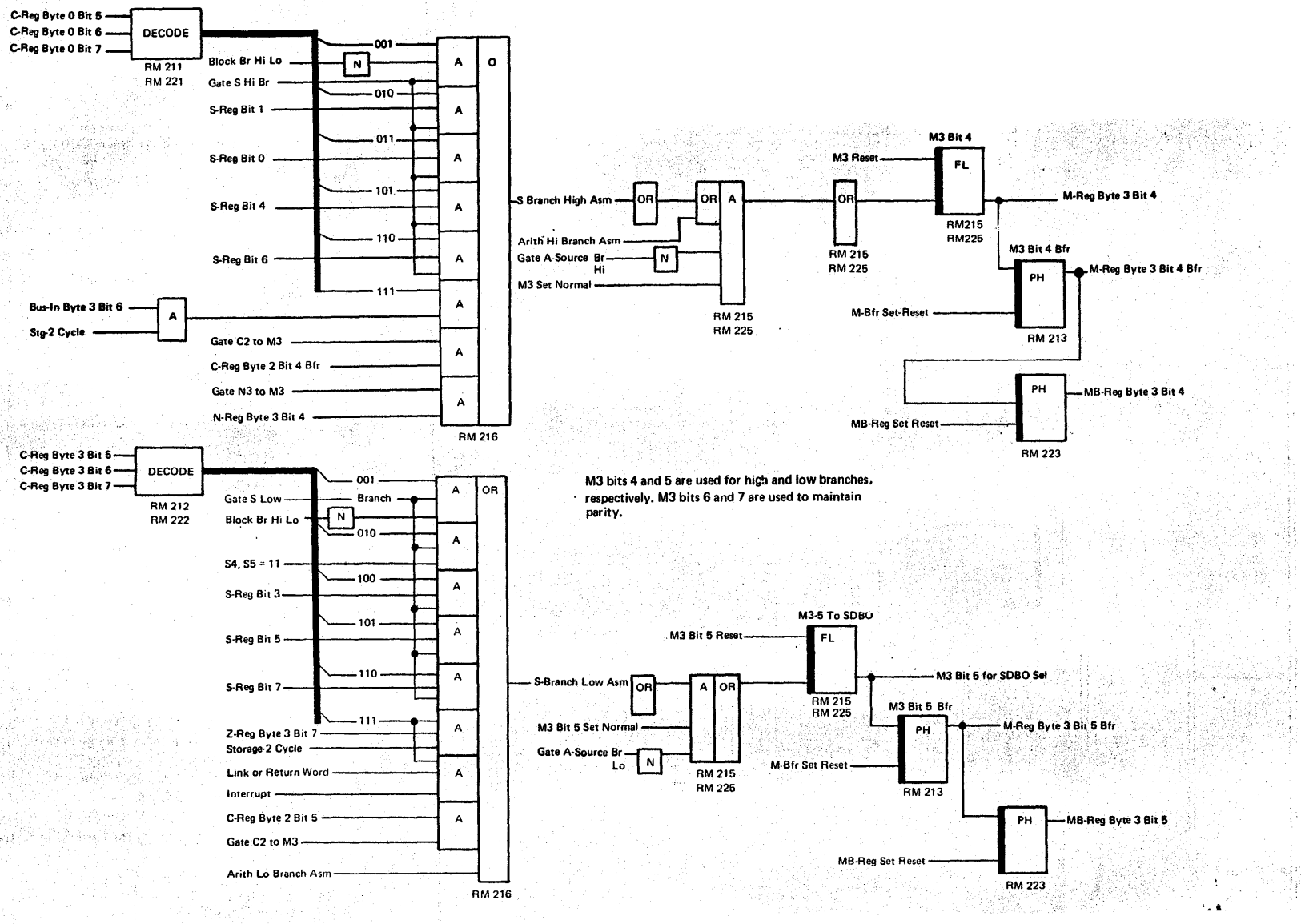
The M-buffer registers are an interim set of latches between the M-register and the MB-register. This allows cycle-to-cycle communications.

The N-buffer registers perform a similar function.





M-, N-, and MB-REGISTERS BRANCH CIRCUITS



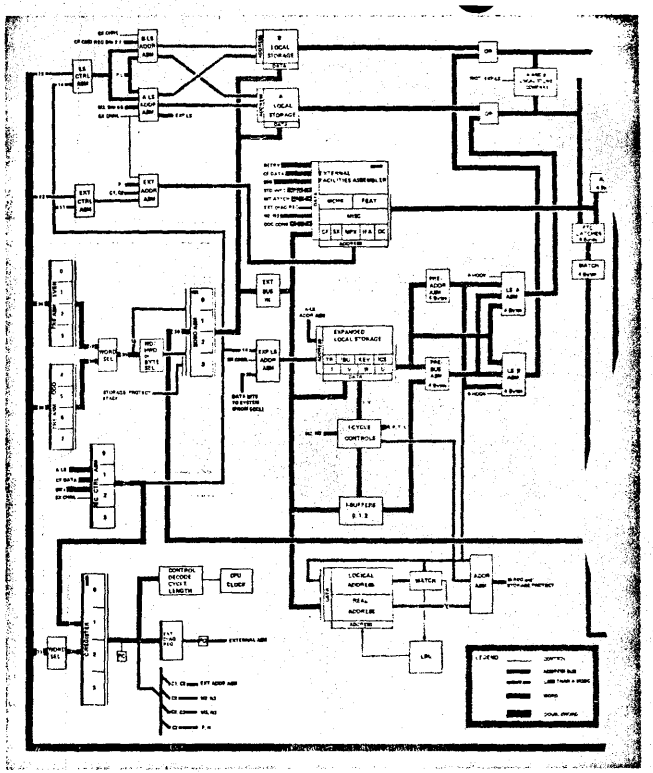
M3 bits 4 and 5 are used for high and low branches, respectively. M3 bits 6 and 7 are used to maintain parity.

SECONDARY CONTROL ASSEMBLER

The Secondary Control Assembler provides a direct path for the microprogram and the console file to move data and addressing information.

From:	To:
Console-file data register	C-Register
A-Local storage	Local-storage control assembler
Console switches ABCDEFGH	Expanded local-storage address assembler
Selector-channel force C-register	External control assembler

Logic Pages	
Bytes 0, 1, 2, and 3.	
Bit	Page
P	RC112
0	RC122
1	RC132
2	RC132
3	RC152
4	RC162
5	RC172
6	RC182
7	RC192



Note: In Local Storage Control Storage (LSCS) mode, diagnostic hardware forces A-local storage to act as control storage. Control words are read out of A-local storage, loaded into the C-Reg, and executed.

Refer to "CPU Diagnostic Hardware" in DIAG section.

Note: Only the bits used by the microprogram are shown for each application of the Secondary Control Assembler.

CONTROL REGISTERS

DESCRIPTION

The control registers provide a means for maintaining and manipulating control information and are an extension to the EC PSW. The 3145 has sixteen 32-bit registers that are located in control storage at addresses F480 through F4BF.

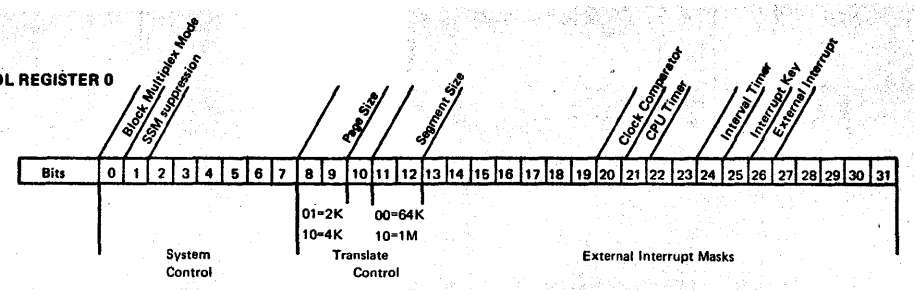
Two instructions, Load Control and Store Control, move data to and from the control registers. The Load Control instruction provides a means for loading control information from main storage into control registers; whereas Store Control permits information to be transferred from control registers to main storage. These instructions operate in a manner similar to Load Multiple and Store Multiple.

Details of the register assignments are contained in the sections that discuss the features using the control registers.

CONTROL REGISTER ASSIGNMENTS

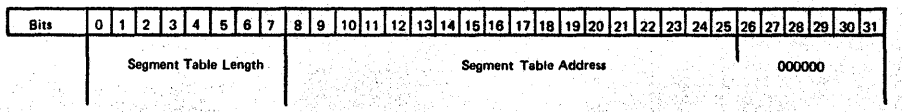
	0	8	16	24	31
0	System Control	Translate Control	External Interrupt Masks		
1	Segment Table Length	Segment Table Origin Address			
2	Channel Masks				
3	Reserved				
4	Reserved				
5	Reserved				
6	Unassigned				
7	Unassigned				
8			Monitor Mask		
9	PER Event Masks	00000000	PER General Register Alteration Mask		
10	00000000	PER Starting Address			
11	00000000	PER Ending Address			
12	Unassigned				
13	Unassigned				
14	Error Recovery Control and Masks				
15	00000000	MCEL Address			

CONTROL REGISTER 0



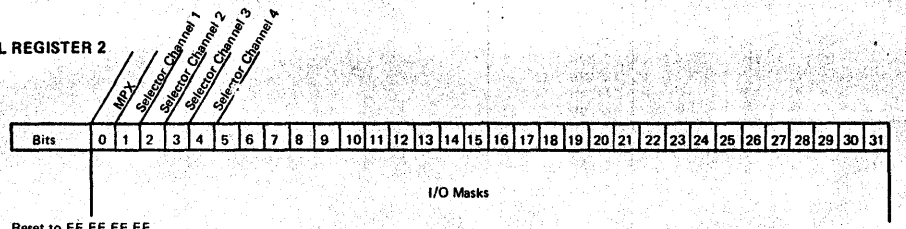
Reset to 00 00 00 E0

CONTROL REGISTER 1



Reset to 00 00 00 00

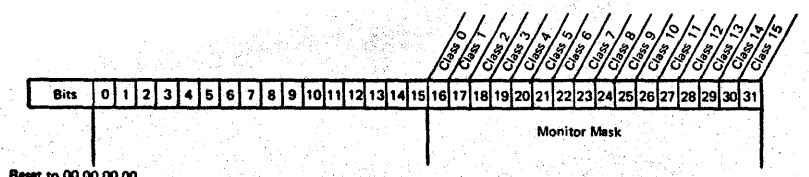
CONTROL REGISTER 2



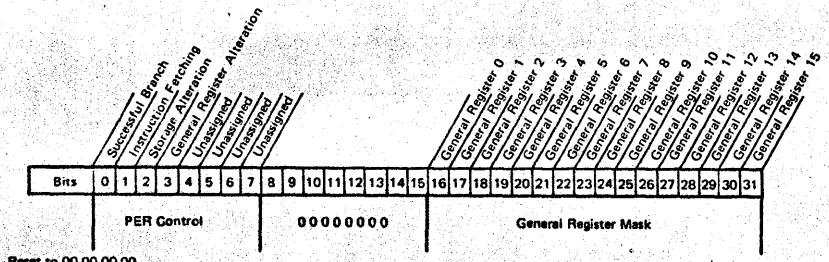
Reset to FF FF FF FF

Note: Initial Value of unassigned positions in all registers is unpredictable but is assumed to be 0.

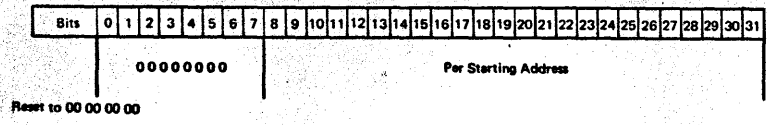
CONTROL REGISTER 8



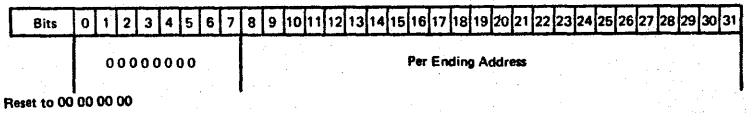
CONTROL REGISTER 9



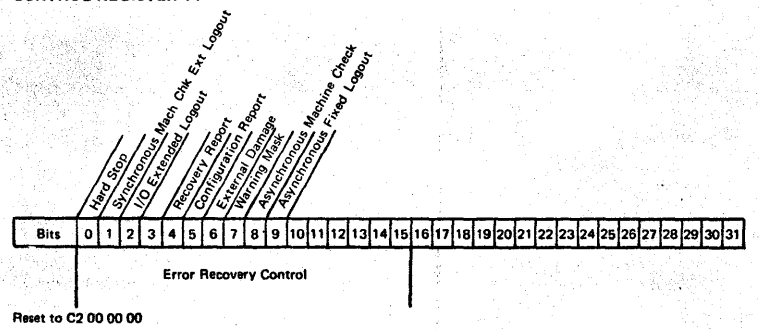
CONTROL REGISTER 10



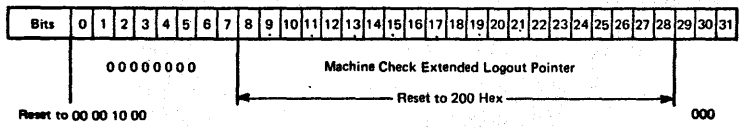
CONTROL REGISTER 11



CONTROL REGISTER 14



CONTROL REGISTER 15



Note: Initial value of unassigned positions in all registers is unpredictable but is assumed to be zero.

TIME-OF-DAY CLOCK

The time-of-day (TOD) clock provides a consistent measure of time suitable for elasped time and time-of-day indications. The cycle of the clock is about 143 years when started from zero as an elapsed time measure. To provide a consistent time-of-day indication, the zero point must be defined to a calendar date. IBM programming systems have established this date as January 1, 1960, 0 AM Greenwich Mean Time.

Setting the TOD clock on the basis of a synchronization signal given by the operator introduces errors in the fractions of a second. This error is usually of small consequence in defining the time relating to human reaction. The error does not enter elapsed-time calculations because the difference between two time readouts does not consider the initial setting of the clock. For many TOD applications, only the high-order 32 bits need be considered. Position 31 of the counter is advanced every 1.048576 seconds. Operation in this mode still requires entering some value for the TODL destination, or the clock does not start.

The clock is a binary counter with a two-word format (64 bits) numbered 0 to 63 corresponding to the bit positions of a fixed-point number of double precision. Time is measured by incrementing position 51 of the counter every microsecond. Only the high-order 52 positions of the counter are used for this configuration. The remaining low-order positions are not used for time indication and are normally set to zeros except for three positions that define the status of the clock.

The program is not signaled of an overflow condition when the counter is advanced to the point of carry from either position 1 or position 0. At the point of carry out from position 0, the counter goes to zero and continues to count from that value.

The clock can be inspected by the instruction store-clock. The current value of the clock counter is stored in main storage. The clock can be set to a specific value by the instruction set-clock. The operand specified by the instruction replaces the current value in the clock counter. The set-clock instruction can be executed only when the clock security switch on the system control panel is set to enable.

The operation of the time-of-day clock is not affected or inhibited by any normal activity or event in the system other than turning off the CPU power or running the TOD clock diagnostics. The clock runs when the CPU is in wait state, stopped state, or instruction step mode, and its operation is not affected by system reset or the IPL procedure.

The 3145 time-of-day clock stops when the CPU power-off switch is operated. Execute the set-clock instruction each time the system is started after powering up or after running the TOD clock diagnostics.

Physical Description

The time-of-day clock is driven by a 1-MHz oscillator feeding a binary-coupled trigger to produce a 1-MHz output. These circuits also develop a 75.46-KHz output that drives the interval timer.

The clock counter functions as a binary-connected counter but is modified with a set of adder latches that allow holding the output of the basic latches until such time that the CPU sampling is complete. The counter can be loaded by destining the appropriate binary values to the TODL and TODH externals after executing a loading sequence. The counter starts to advance, immediately following the loading by gating the 1-MHz drive signal to the low-order position of the counter. The counter advance is checked by predicting parity and then comparing the predicted value with the parity generated from the counter. The counter output with parity bits for each byte is available to the CPU through the store clock instruction that causes externals TODL and TODH to be transferred to main storage.

The TODH and TODL externals may be called out as a destination at any time, but the contents of the clock are not changed unless TODL byte 3 bit 0 and the clock-run latch have been reset. The FTC (flush-through check) is blocked by the TODL byte 3 bit 0 to prevent error signals for this condition.

The control bits are associated with the clock readout to convey information to the user as to whether the clock value is a true measure of elapsed time since the last time the clock was set. These control bits are stored as the three high-order bits of byte 3 of the low-order word (TODL). Bit 0 indicates that the clock is running. Bit 1 indicates that the clock was set. Bit 2 indicates that an error occurred. During the processing of the store-clock instruction, the indicators control the condition code to be set.

TODH EXT 36

Byte 0	Byte 1	Byte 2	Byte 3
0	8	16	24 31

TODL EXT 3E

Byte 0	Byte 1	Byte 2	Byte 3
32	40	48	56 63

Bit definition:

- TODH Bits 0-31 Binary Counter
- TODL Bits 32-51 Binary Counter
- TODL Bits 52-55 Spare Counter Positions (forced to zeros)
- TODL Bit 56 Run Bit. This bit is set during the destining of the TOD word following the destining of the other TOD word. This bit must be reset by microcode with the security switch in the enable-set position to reset the TOD Run latch and enable the TOD clock hardware to accept the TODL or TODH as a destination. Power-on reset also resets this bit to allow the power-on reset routine to set the TOD clock to a consistent value.
- TCDL Bit 57 TOD Clock Security Switch. This bit on signifies that the security switch is in the enable-set position. This bit must be on to allow TOD Clock destinations, except during the power-on reset routine.
- TODL Bit 58 Validity Latch. This bit on indicates that the clock is valid. The TOD validity latch is set by the AND of TODL byte 3 bit 0 and TODL byte 3 bit 1. This bit is reset by resetting TODL byte 3 bit 0, power-on reset, or TOD clock error. This bit off lights the TOD invalid light on the console.
- TODL Bits 59-63 CPU Identification.

TODH may be displayed through EXT 36.
TODL may be displayed through EXT 3E.

Clock Security Switch (TOD CLK)

The clock security switch (TOD CLK) provides an interlock with the set-clock instruction to guard against inadvertent change of the clock value. The switch is spring-returned to the secure position. When the switch is in the enable-set position, execution of the set-clock instruction sets the clock to the value of the designated operand. When the switch is in the secure position, the set-clock instruction does not change the value of the clock. The switch does not have any other effect on the operation of the clock.

Clock Validity Indicator (TOD CLK INVAL)

The clock validity indicator (TOD CLK INVAL) is used to indicate when the time-of-day clock current value is not a true measure of the elapsed time since the last time the clock was set. The validity indicator is turned off when the set-clock instruction is executed with the TOD CLK switch in the enable-set position and no exceptions are encountered. The indicator is turned on whenever the clock misses a time increment or stops. This may result from a power failure or a malfunction in the clock circuits. When the indicator lights for an error condition, the machine-check indicator is set and an interrupt is requested. If the clock is started by the power-on reset routine and used for elapsed-time indications, the indicator remains lighted because it is invalid as a TOD indication. In this case, the indicator being lighted does not mean that an error has occurred.

Error Detection

The time-of-day clock checks its advance operation by a check on the progressive parity conditions. Before the advance, the counter value of each byte is fed into a parity predictor circuit to develop the updated parity bits. These parity bits are compared with parity bits generated from the updated value in the counter. Any failure within the counter advance circuits results in a difference between the predicted parity bits and the resultant parity bits. The difference signal sets the TOD clock check latch. The latch output sets the MCKA3 bit 6 and resets the bit 2 and bit 1 latches as indicators. A class 3 machine-check interrupt is requested. The setting of the MCKA latch resets the TOD clock check latch. The reset of the bit 2 latch lights the TOD invalid indicator on the system control panel.

Clock-Setting Sequence

The clock is set to zero and started by the power-on-reset sequence. With this start the clock output can be used to indicate running time or elapsed time. Under this mode of operation, the TOD-Invalid indicator on the system control panel remains lighted because the clock output is not TOD.

To obtain time-of-day output, the set-clock instruction must be executed to start the clock at the current time. When the set-clock instruction is executed, the correct TOD is assumed and the TOD Invalid indicator on the system control panel is not lighted. If a clock error is detected during operation, the indicator is lighted to warn the operator.

The set-clock instruction must have the TOD CLK switch on the system control panel held in the enable-set position to allow execution. If the switch is not operated, the clock value and its operation are not affected. The first step in the clock-set sequence is to reset TODL byte 3 bit 0 along with the 'clock run' latch and the clock controls. With the 'clock run' latch reset, the TODH and TODL externals can be destined. The value destined for TODL must have byte 3 bit 0 set to 1. When both load latches have been set, the TODL byte 3 bit 0 line is developed, the 'clock run' latch is set, and the 'clock set' latch is set. At this point, the clock is set and starts to run with the next 1-MHz pulse.

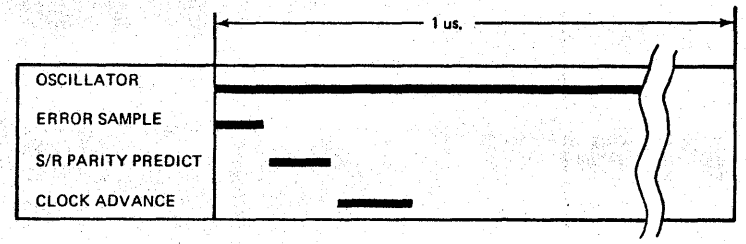
The power-on-reset start differs only in that the TOD CLK switch does not need to be operated, and the first TODL destined micro-step is not required.

For microdiagnostic operation, the enable-set output of the switch is forced, but the set sequence is the same as for the set-clock instruction.

TOD Clock Update Sequence

Because of the asynchronous operation of the TOD clock, the advance pulse from the 1 MHz oscillator cannot be used directly to advance the clock. The clock readout must not change during the A-register set/reset time. The CPU 90-135 time is gated with the 1-MHz oscillator to develop the advance pulse. If the CPU clock is stopped, the CPU oscillator provides the timing. The advance pulse sets the start-update latch whose output provides a series of delayed outputs that control error sample and advance of the clock.

TOD CLOCK UPDATE SEQUENCE



TOD Manual Set

This procedure is available for manual setting of the TOD clock.

1. Hold TOD CLK switch at ENABLE SET.
2. Store in EXT 3E hex 00. Hex 00 is stored into all 4 bytes of TODL. This value resets the run latch.
3. Release TOD CLK switch.
4. Store in EXT 36 hex FF (see Note 1).
5. Hold TOD CLK switch to ENABLE SET.
6. Store in EXT 3E hex 80. This value sets the run latch.
7. Release TOD CLK switch.
8. Display either TODH (EXT 36). Observe that bit 31 increments approximately every second, or

   TODL (EXT 3E). Observe that counter is rippling much faster.

Note 1: Any value put into switches A and B may be stored. This value is propagated to all 4 bytes (example: hex 12 stored from switches A and B is stored as 12 12 12 12). FF is inserted because the first time the clock is incremented, the counter flips to zero and starts from there.

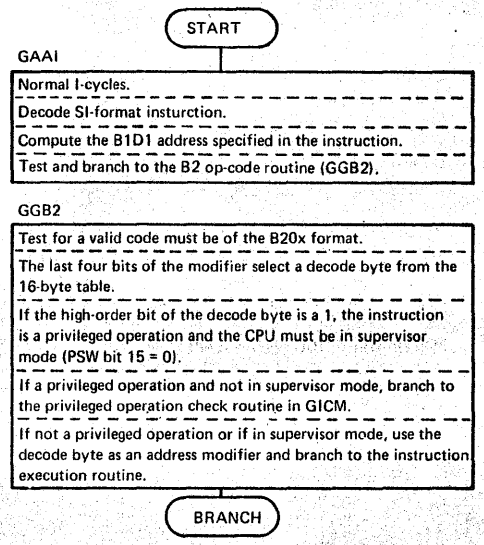
TOD Clock Instructions

The time-of-day clock has two instructions:

- Set Clock used to set the initial time.
- Store Clock used to enter the current clock value into main storage.

Both instructions are SI-format instructions modified so that byte 2 is an extension of the operation code (byte 1) instead of the immediate operand. Byte 2 specifies the exact function of the operation code.

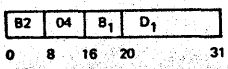
Both instructions are decoded as operation code B2 in the GAAI routine. The operation branches to the GGB2 routine to decode and validate the modifier. A test for privileged operation is made when required. Then the operation is branched to the routine of the function specified. The common data flow for both instructions is:



Set Clock Instruction

Set-Clock Instruction

SCK $D_1(B_1)$ [SI]



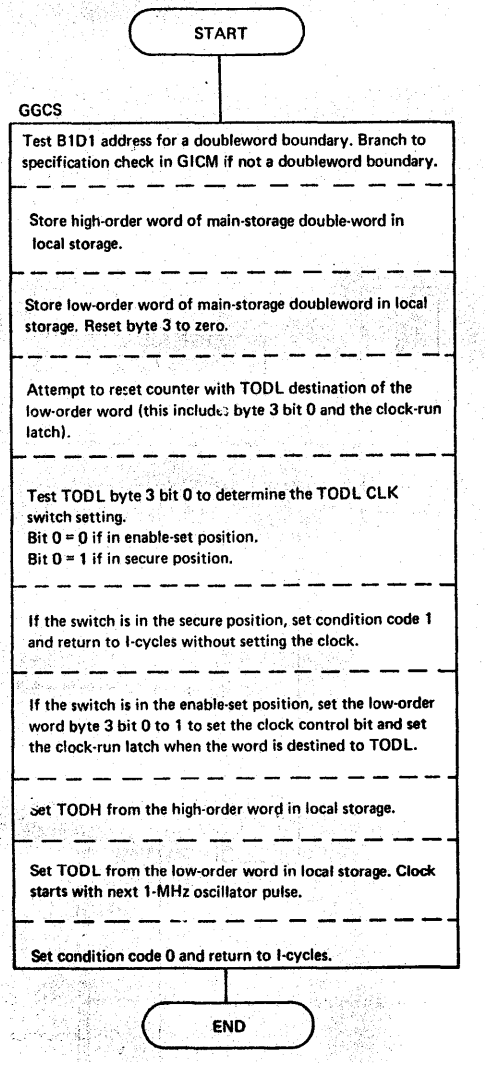
B2 = Operation Code
04 = Set-Clock Function
B1D1 = Storage address of an eight-byte field. Must be on a doubleword boundary. Bits 52-63 of the field are ignored and are not used in the clock value.

The set-clock instruction is a privileged operation used to place a value into the time-of-day clock. The location of the value is specified by the B1D1 portion of the instruction. The implied length of the value is eight bytes (two words). The address must be located on a doubleword boundary. Only the high-order 52 bits of the doubleword are used to set the clock counter. The remaining bits (52 to 63) are ignored by the operation in setting the clock value.

The value in the time-of-day clock is replaced by the designated value if the set-clock instruction is executed while the TOD CLK switch is in the enable-set position. If the TOD CLK switch is in the secure position when the set-clock instruction is executed, the value in the clock is not replaced and the condition code is set to indicate:

Condition code settings:

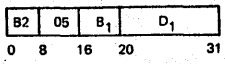
0 = Clock value set.
1 = Clock value secure (TOD CLK switch in secure position; therefore, clock value was not changed).
2 = Not used by the 3145.
3 = Not used by the 3145.



Store-Clock Instruction

Store-Clock Instruction

STCK $D_1(B_1)$ [SI]



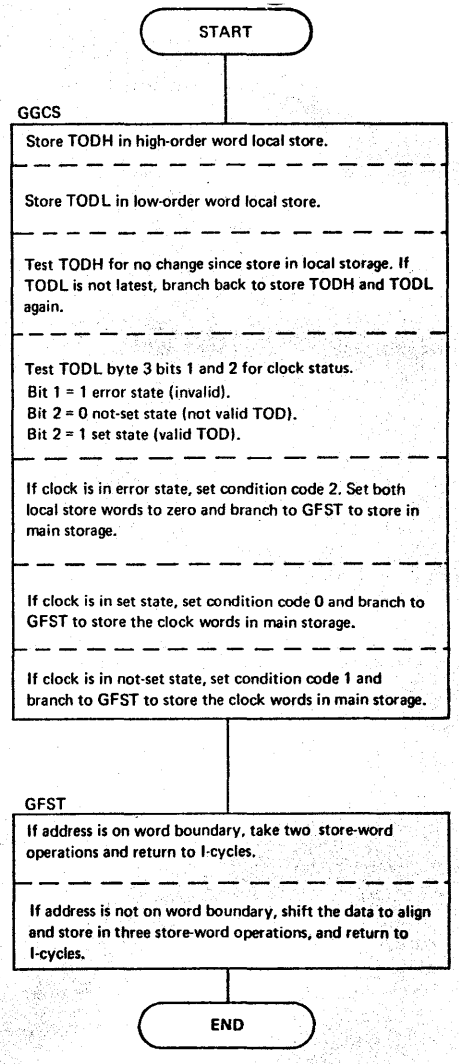
B2 = Operation code
05 = Store-clock function
B1D1 = Storage address of an eight-byte field. May be located on a byte boundary. Bits 52-63 of field are:

52-55 = Set to zero.
56-58 = TODL Ctrl Bits 0-2.
59-63 = CPU Identification bits.

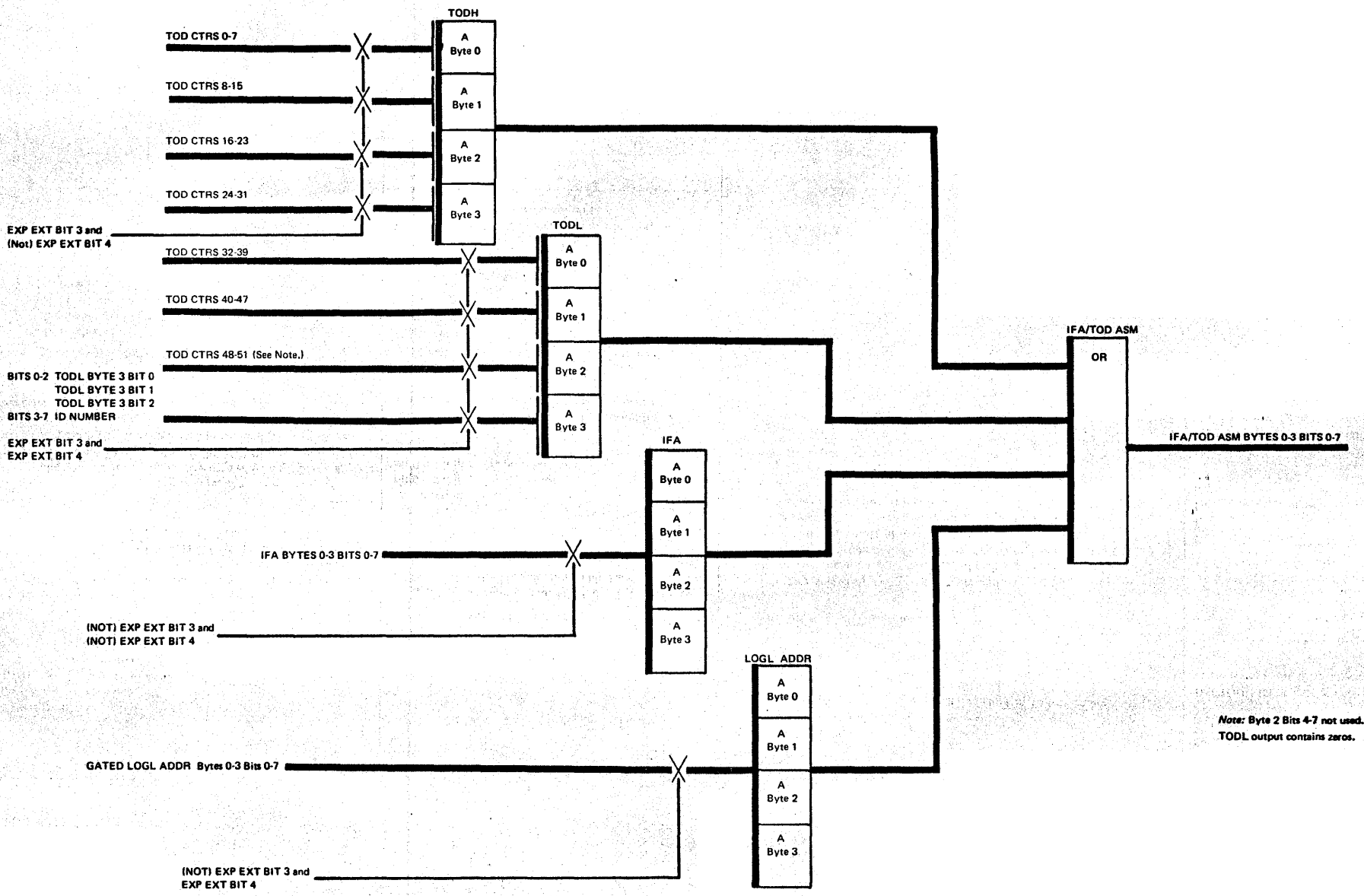
The store-clock instruction is used to place the current time-of-day clock value in the eight-byte field of main storage designated by the B1D1 portion of the instruction. The 52-bit clock value stores in the high-order of the doubleword assignment. The low-order byte stores the three clock control bits and the five-bit CPU pluggable identification. If the clock value is invalid, the doubleword is stored with all zeros.

Condition code settings:

0 = Clock in set state.
1 = Clock in not-set state.
2 = Clock in error state.
3 = Not used by the 3145.



TOD Clock Output Assembler



CIRCUIT CARD LOCATION: A1L2

LOGIC/ALD PAGE:

CT011
- Osc Drive

CIRCUIT CARD LOCATION: A1N2

LOGIC/ALD PAGE:

CT111
- Time-of-Day Counter Advance Controls
- Interval Timer Osc Drive

CT112
- TOD Counter Set Controls
- Condition Codes
- Identification
- IF A/TOD ASM BYTE 2 BITS 2-3

CT113
- Error Detection

CT114
- TOD Ctrs 18-19
- TOD Ctrs 50-51
- TOD Asm Byte 2 Bits 2-3

CT115
- Lock-Load Timing Generation

CIRCUIT CARD LOCATION: A1P2

LOGIC/ALD PAGE:

CT211
- TOD Ctrs 0-1-2
- TOD Ctrs 32-33-34
- TOD Asm Byte 0 Bits 0-1-2

CT212
- TOD Ctrs 3-4-5
- TOD Ctrs 35-36-37

CT212 Continued
- TOD Asm Byte 0 Bits 3-4-5

CT213
- TOD Ctrs 6-7-8
- TOD Ctrs 38-39-40
- TOD Asm Byte 0 Bits 6-7
- TOD Asm Byte 1 Bits 0

CT214
- IFA/TOD Asm Byte 0 Bits 0 through 7
- IFA/TOD Asm Byte 1 Bit 0
- Gating Controls

CT215
- Parity
- Parity Predict
- Parity Asm for Byte 0

CT216
- TOD Termination

CIRCUIT CARD LOCATION: A1Q2

LOGIC/ALD PAGE:

CT221
- TOD Ctrs 9-10-17
- TOD Ctrs 41-42-49
- TOD Asm Byte 1 Bits 1-2
- TOD Asm Byte 2 Bit 1

CT222
- TOD Ctrs 11-12-13
- TOD Ctrs 43-44-45
- TOD Asm Byte 1 Bits 3-4-5

CT223
- TOD Ctrs 14-15-16
- TOD Ctrs 46-47-48
- TOD Asm Byte 1 Bits 6-7
- TOD Asm Byte 2 Bit 0

CT224
- IFA/TOD Asm Byte 1 Bits 1 thru 7
- IFA/TOD Asm Byte 2 Bits 0-1

CT225
- Parity
- Parity Predict
- Parity Asm for Byte 1

CT226
- TOD Termination

CIRCUIT CARD LOCATION: A1R2

LOGIC/ALD PAGE:

CT311
- TOD Ctrs 20-21-22
- IFA/TOD Asm Byte 2 Bits 4-5-6

CT312
- TOD Ctrs 23-24-25
- IFA/TOD Asm Byte 2 Bit7
- IFA/TOD Asm Byte 3 Bits 0-1

CT313
- TOD Ctrs 26-27-28
- IFA/TOD Asm Byte 3 Bits 2-3-4

CT314
- TOD Ctrs 29-30-31
- IFA/TOD Asm Byte 3 Bits 5-6-7

CT315
- Parity Asm for Bytes 2 and 3
- Gate Buffers

CT316
- Parity Predict for TOD Bytes 2 and 3

CT317
- TOD Asm Byte 3 Bits 3-4-5-6-7
- CPU Identification Number

INTERVAL TIMER

The interval timer provides program interruption on a program-controlled time basis. Interval timer applications include:

- Job accounting
- Monitoring for perpetual program loops
- Time stamping
- Polling at timed intervals

The storage area allocated for the interval timer feature is in main storage locations 50-53 hex. If the interval timer switch is set to NORM (normal), any value stored at this location is decremented by the hardware.

The program being processed can be interrupted by an external interruption (if PSW bit 7 and control register 0 bit 24 are on) when the interval timer word changes from a positive to a negative value. The interruption is identified by the appropriate external interrupt register bit.

Description

- Has a 32-position counter
- Stores last timer value in MS 50
- Contained on 3 cards:

A-B2K4	Controls	ALD CH031-032
A-C1T2 and A-C1U2	Timer	ALD CH211-225

- Uses same oscillator as TOD clock (High Resolution Timer 75.46-KHz oscillator)

Interval Timer Operation

Enable or disable timer

Decrement timer

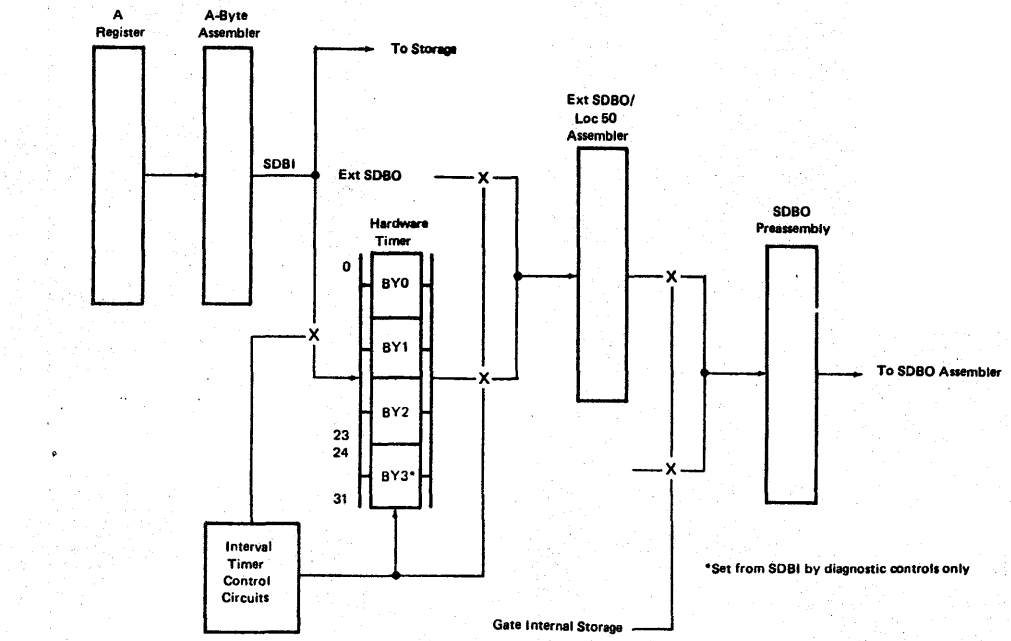
Set timer

Read timer

Display Timer

Interval timer interrupt

INTERVAL TIMER BLOCK DIAGRAM



Enable Timer

To enable the interval timer:

1. Set interval timer switch to NORMAL.
2. Set rate switch to PROCESS, and
3. Execute a return word (RTN with Br Lo = 111) to set the timer run latch.

Decrement Timer

The hardware circuits gate a signal to the counter. This signal is synchronized with the oscillator which has a 13-us. time interval. At 45 to 135-time in the CPU clock cycle at the start of the 13-us. interval counter position 31 is decremented. This update is inhibited each time the interval timer is being set to a new value.

Position 23 of the counter is decremented every 3.3 ms. To obtain this degree of resolution, the ability to store in byte 3 (positions 24-31) is inhibited by the timer hardware. Therefore, these positions are used functionally to assure that position 23 is updated each 3.3 ms by a signal which is developed every 13 us.

Whenever an interrupt occurs, the timer value is stored in main storage location 80. If interrupts occur at time intervals greater than 2.048 ms the TOD clock carry-out of bit position 41 forces a branch to the section of microprogram (GICM) that transfers the timer value to main storage location 80.

Disable Timer

The interval timer is disabled when:

1. The interval timer switch is in DISABLE
2. The rate switch is *not* in PROCESS
3. A timer error occurs.

Set Timer

The timer value is set into the hardware and main storage location 80 at 0 to 45-time of the storage-2 cycle of a storage word that specifies:

1. A not k-addressable storage word, and
2. A store word into location 80.

Read Timer

The contents of the counter cannot be displayed directly. To enable the timer value to be checked the timer contents are transferred by the 370 microprogram A

Display Timer

The manual display of location 80 shows the timer value of the last update.

Interval Timer Switch

NORMAL

This switch position enables the interval timer control circuits, which allows the hardware to be decremented continually.

DISABLE

This switch position disables the interval timer control circuits.

Regardless of the interval-timer switch position, a store/display of location 80 displays only the contents of main storage location 80.

Interval Timer Interrupt

Once enabled, the timer decrements continually. The instant that the value changes from positive to negative, an interrupt condition is indicated. The interrupt routine (GICM) tests for this timer value change in LS Y (LS 16). When LS 16 bit 0 changes from zero to one, the microprogram branches when Y0 bit 0=1 B and sets C the External Interrupt register bit 0 (Ext 12, byte 0 bit 0) on.

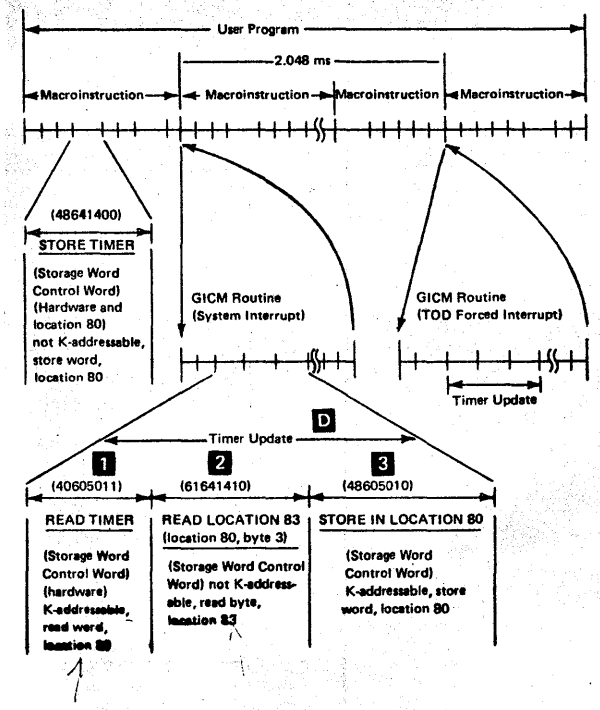
During program execution, whenever both PSW bit 7 and MSKA bit 0 are on, the timer interrupt request is gated to the interrupt latch. The system uses this to execute the interrupt handling routine.

A

Sample Routine:			
Label	**Next Label**	**Statement**	**Comments**
		PX̄0, OE, 52	Set P for interrupt routine
		VC = 0 + K50	V = 0000050
		RDW Y DM, 50 1	Read timer hardware
		RDW X V, NOP 2	Read location 80
		Y3 = X3	Preserve byte 3
		STW Y DM, 50 3	Update location 80
	TIMR 1, B0 B	Y0	Current value negative?
C TIMR 00	TIMR 1, 0	EXTINT, OR, K80	Set timer interrupt request
TIMR 10	MISC B1	SYS1	Continue GICM
TIMR 11	TIMR B0, 0	X0	Was last value negative?

Notes: 1. These statements A are found at the beginning of the GICM routine.

2. The timer update D occurs only when either a system interrupt or a TOD forced interrupt occurs.



OS/DOS COMPATIBILITY

Introduction

- Consists of the OS/DOS emulator program and the hardware and microprogramming needed for execution.
- Two new instructions: Execute Local (EXL) and Adjust CCW String (ACCW), are used by the emulator-program.
- The DOS Emulator and the DOS system being emulated are located in main storage above the OS area.
- The minimum storage area needed for the DOS emulator and the DOS system is 38K bytes.
- The OS/DOS emulator operates in the same manner as any OS job.

When operating in local mode (DOS programs being executed), all addresses pertaining to the DOS area are adjusted by the address-adjustment hardware.

When local mode is terminated, addressing is performed in the standard manner.

Refer "OS/DOS Functional Units", and "Real Address Computation Example" for an explanation of the address-adjustment hardware.

Address translation is needed for DOS emulation because the addresses of the DOS supervisor (SV) are basically fixed and the DOS supervisor is located in an area of storage not normally used by DOS SV. All references to the fixed addresses of the DOS SV must be adjusted to reflect the real location of these fixed areas.

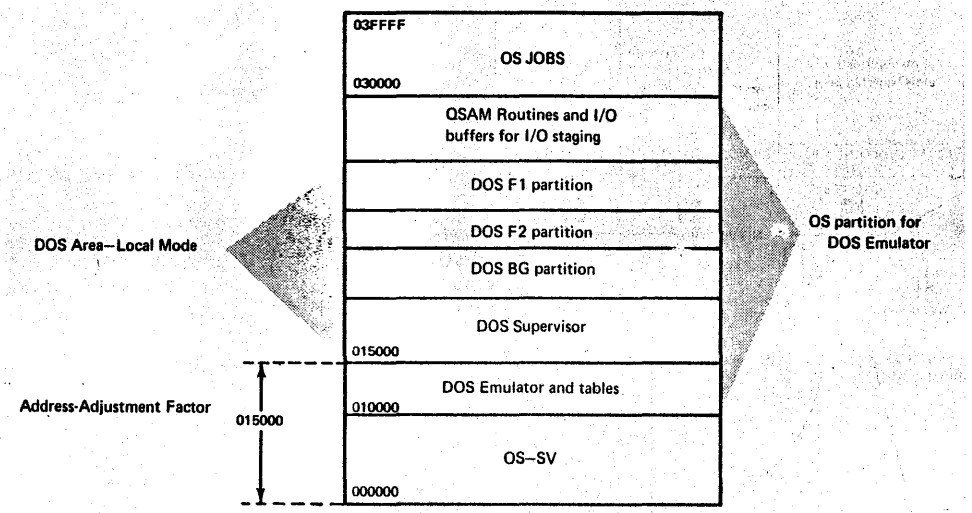
The EXL instruction operates with the LEX List, which is a table that the emulator program loads before the execution of the DOS area. This list is located in the emulator area and is used to handle entry to and exit from the local mode of operation.

The ACCW instruction operates with the Adjust CCW list (ACCW List). The ACCW list is loaded and maintained by the emulator program for use in the adjustment of CCW data addresses.

The OS/DOS emulator and the DOS system being emulated (DOS supervisor and up to three processing program partitions) execute together in an MFT partition or MVT region, which must be a minimum of 38K. The OS/DOS emulator program and tables require 22K plus another 4K if I/O staging is used. Additional OS/DOS emulator program storage may be required, depending on the I/O devices used. Up to ten I/O devices are supported in 22K, and 250 bytes are required for each additional device. The I/O staging requirement of 4K supports unblocked reader, printer, and punch records and residence of the required QSAM routines in the OS/DOS emulator partition or region.

The DOS system being emulated can be 16K, 24K, or 32K and up, in 4K increments. The OS/DOS emulator is scheduled to operate in the same manner as any other OS job. Several OS/DOS emulator jobs can execute concurrently with OS jobs if enough I/O devices and processor storage are available. In addition, the Model 145 OS 1401/1440/1460 and 1410/7010 Emulator programs can execute concurrently with the OS/DOS emulator if enough resources are present.

EXAMPLE STORAGE ASSIGNMENT for 256K MODEL 145



Translate Look-Aside Buffer (TLB)

- Eight 26-bit registers contain the local and real addresses used during the accessing of the local area when operating in local mode.
- The local address occupies bytes 0 (bits 0-7) and byte 1 (bits 0-3) of the TLB.
- The local address is gated to the registers from PAA.
- The real address occupies byte 2 (bits 0-7) and byte 3 (bits 0-3) of the TLB.
- The real address is gated to the registers from EBI.
- The register to be loaded is addressed by the LRU.
- The TLB can be displayed through EXT 2E with switch H selecting 1 of 8 registers.

Least Recently Used (LRU) Matrix

- The LRU is an address matrix that keeps track of the use of the translate look-aside buffer.
- The LRU addresses the least recently used translate look-aside buffer whenever a computed real address must be loaded.
- Composed of 28 latches and associated circuits.
- The LRU is reset to zero before the OS/DOS operation.
- The LRU can be displayed through EXT 08 byte 2.

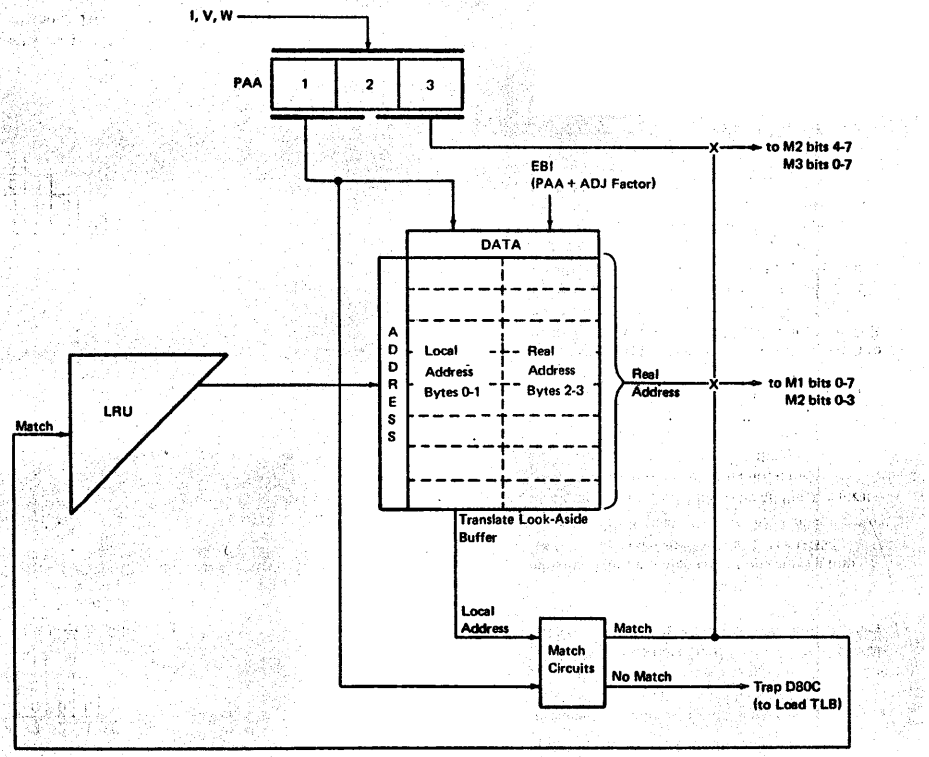
When a mismatch occurs between the local address portion of the PAA and the local address portion of the TLB, the LRU determines the register to be loaded with the computed real address. In the GGST microroutine, the computed real address is loaded into TLB bytes 2 and 3 and the local address causing the mismatch is loaded into TLB bytes 0 and 1 addressed by the LRU.

The status of the LRU is changed each time a match occurs between the local address portion of the PAA and the local address portion of the translate look-aside buffer. This constant changing assures that the least recently used register is addressed when a mismatch occurs. Refer "LRU Operation Example."

Match Circuits

- Perform the comparison of the local address portion of the PAA and the local address portion of the translate look-aside buffer bytes 0 and 1.
- Output of the match circuits sets and resets specified combinations of the LRU.
- The match circuit can be displayed through EXT 08 byte 3. When displaying the match circuits, the PAA must contain valid information.

There is a match circuit for each translate look-aside buffer. When a match occurs, the corresponding LRU row is set to ones and the corresponding column is reset to zeros. The resulting status of the LRU provides the means of addressing the least recently used register.



LRU Operational Example

Assume: Storage access attempted for local address 0A000.
Adjustment factor = 14000
Translate look-aside buffers are set to values indicated.
Table buffer Regs are set to values indicated.

Objectives: Provide a real address to the M-register to access the local area specified as address 0A000.

Description: When the storage access is attempted in local mode, the local address portion of the PAA and the local address portion of the translate look-aside buffers are compared.

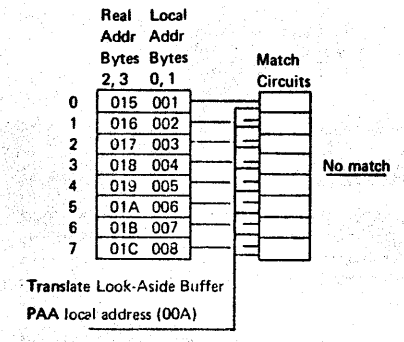
A no-match condition results from the match circuits. This no-match condition causes a trap to control-storage address D80C. The GGST microroutine is executed. The GGST microroutine:

- Computes the real address by adding the local address and the adjustment factor,
- Re-executes recently used of the translate look-aside buffers with the real and local address.
- Re-executes the storage word that caused the mismatch.

Re-executing the storage word causes a match to occur from the number 4 TLB. the number 4 TLB was loaded in the GGST microroutine.

The match line from the number 4 register sets row 4 and resets column 4 of the LRU. The status of the LRU now indicates that TLB 6 is the least recently used. Should a mismatch occur on the next storage access, the computed real address is loaded into TLB 6.

FIRST STORAGE ACCESS ATTEMPT

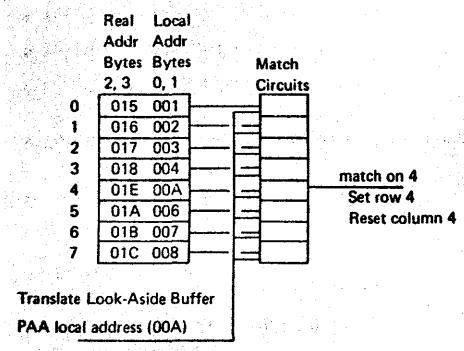


LRU

	7	6	5	4	3	2	1	0
0	1	1	1	1	0	0	0	
1	1	1	1	1	0	0		
2	1	1	1	1	1			
3	1	1	1	1				
4	0	0	0					
5	1	1						
6	0							
7								

4 is least recently used
(row 4=0s and column 4=1s)

SUCCESSFUL EXECUTION OF STORAGE WORD



LRU

	7	6	5	4	3	2	1	0
0	1	1	1	0	0	0	0	
1	1	1	1	0	0	0		
2	1	1	1	0	1			
3	1	1	1	0				
4	1	1	1					
5	1	1						
6	0							
7								

6 now least recently used
(row 6=0s and column 6=1s)

New Instructions for [redacted]/DOS Emulator

- The new instructions are Execute Local (EXL), and Adjust CCW string (ACCW).
- The Op code for both these instructions is B2.
- The immediate byte determines which of the two instructions is to be executed.

Execute Local

B2	0E	B1	D1

(EXL)

- This instruction addresses the LEX list, performs certain initialization functions, and sets local mode for system operation.

When the EXL instruction is executed, the condition code, program mask, and instruction address in the current PSW are replaced by values from the LEX list. General registers 14 and 15 are loaded from the LEX list and the CPU is placed in Local mode.

During the execution of the EXL instruction, the modified PSW is not checked for program interruptions. Any such checks occur as part of the next instruction execution.

Condition Code

Upon completion of the EXL instruction, the condition code is set according to the condition code loaded from the LEX list.

Program Interruptions

Addressing: The address of the LEX list is invalid. The address formed by the addition of the origin address and the local address exceeds the maximum address allocated to the emulator program.
The operation is suppressed.

Operation: The instruction is not installed. The operation is suppressed.

Protection: The LEX list is protected for fetching or storing. The operation is suppressed.

Specification: The first operand address does not specify a 64-byte boundary.
The origin address is not a multiple of 4096.
The local limit address is not one less than a multiple of 4096.
The operation is suppressed.

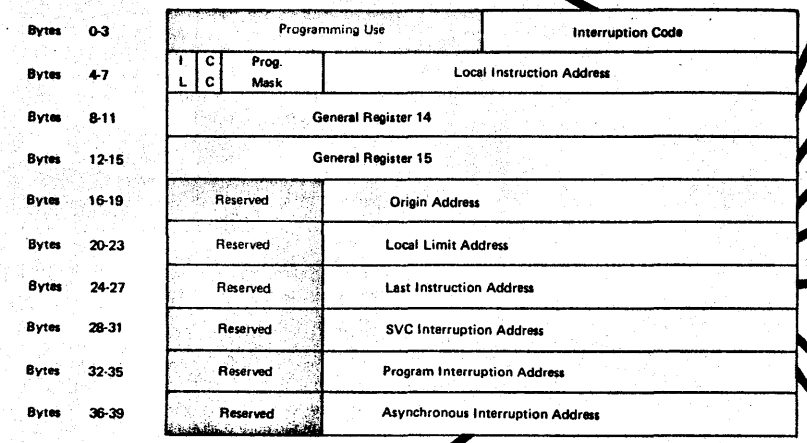
Special Operations: If the EXL instruction, the ACCW instruction, the Monitor Call instruction, or the emulator instruction is encountered while in local mode, the operation is suppressed. The interruption is reflected in the LEX list of the program that placed the CPU in local mode.

A privileged operation is any privileged instruction encountered while in local mode.

Local List Format and Definition

Bytes 0, 1 Reserved for emulator program use. This area is not addressed by the EXL instruction.

Bytes 2, 3 Upon termination of local mode by a program or supervisor call interruption, the 16-bit interruption code describing the interruption is placed in this field.



Bytes 36-39 When local mode is terminated by an asynchronous interruption (external, I/O, or machine-check), the address located in bytes 37-39 is placed in the PSW. The high-order byte is reserved and should be set to zero. The adjusted PSW is stored in the corresponding low storage old PSW and an interrupt is taken to the OS supervisor.

Bytes 4-7 Updates the current PSW when the EXL instruction is executed. The instruction address in bytes 5-7 is the address of the next instruction within the DOS area. Whenever local mode is terminated by an interruption, this area is updated. The ILC field is unpredictable when local mode is terminated by an asynchronous interruption.

Bytes 8-11 The value in this field is loaded into general register 14 when the EXL instruction is executed. When local mode is terminated by an interruption, the current contents of general register 14 are stored into this field.

Bytes 12-15 The value in this field is loaded into general register 15 when the EXL instruction is executed. When local mode is terminated by an interruption, the current contents of general register 15 are stored into this field.

Bytes 16-19 The address contained in bytes 17-19 corresponds to the address adjustment factor and points to the zero address of the emulated DOS supervisor. This address must be a multiple of 4096, or a specification interrupt occurs. The high-order byte of this field is reserved and should be set to zero.

Bytes 20-23 The address contained in bytes 21-23 specifies the upper address of the emulated environment. The address must specify one less than a 4096 boundary, or a specification interrupt occurs. The high-order byte of this field is reserved and should be set to zero.

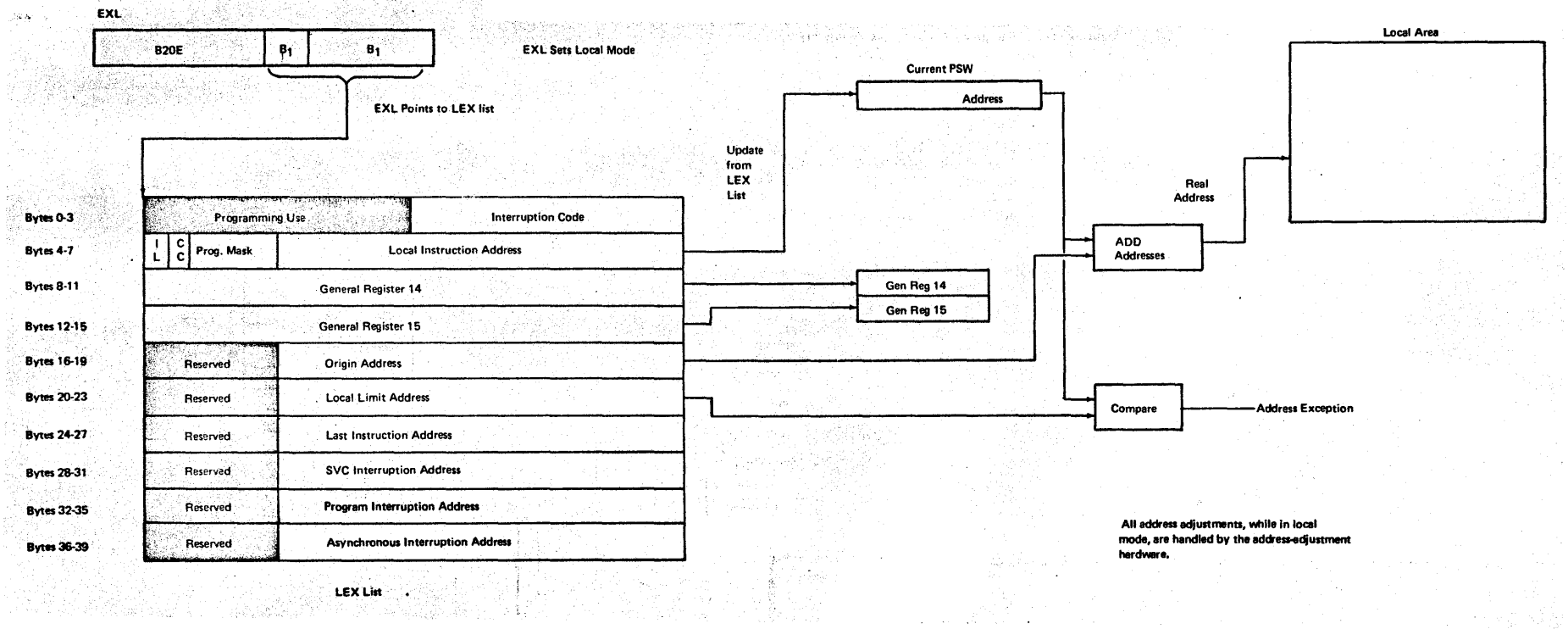
Bytes 24-27 When local mode is terminated by a program or supervisor call interruption, this address points to the instruction causing the interruption. This address is within the boundary of the emulator program. If the instruction causing the interruption was the object of an Execute instruction, the address placed in this field would be the address of the Execute instruction.

The contents of the address field are unpredictable when local mode is terminated by an asynchronous interruption. The high-order byte of this field is set to zero.

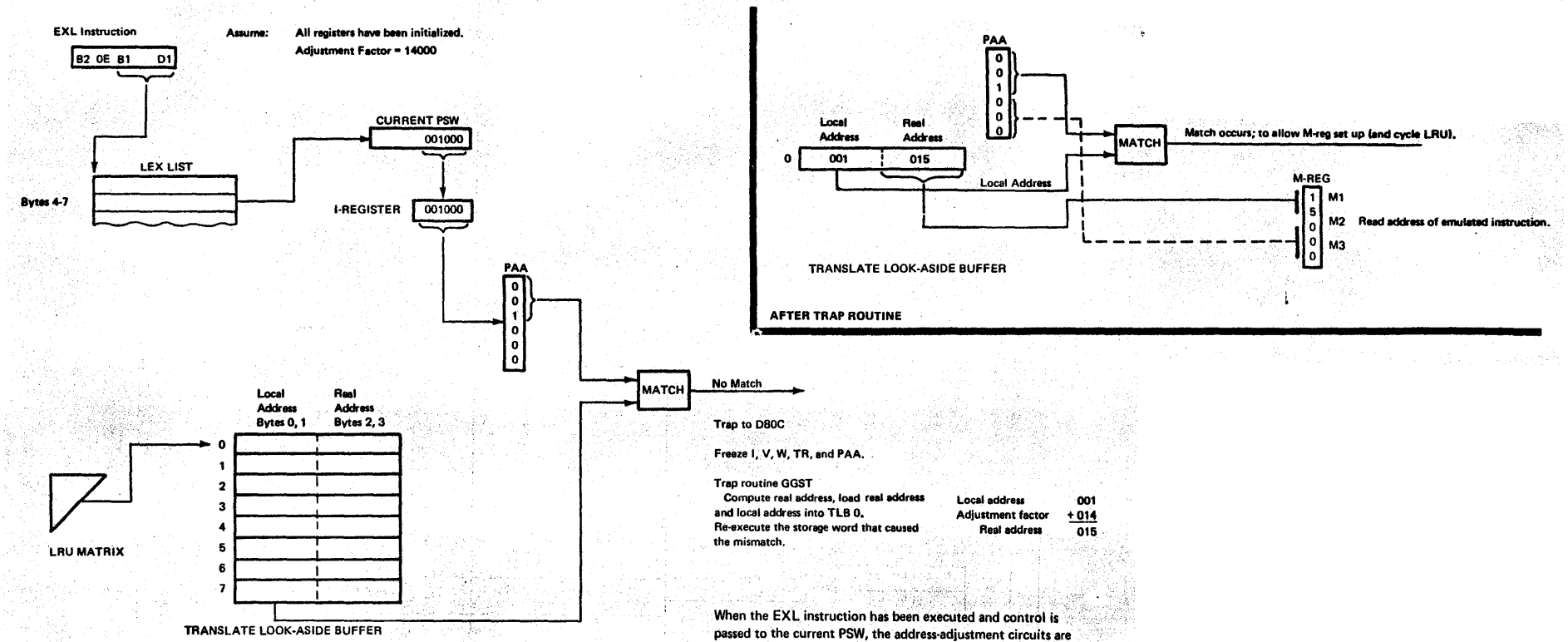
Bytes 28-31 When local mode is terminated by a supervisor call interrupt, the address located in bytes 29-31 is placed in the PSW. This address is within the boundary of the emulator program. The high-order byte is reserved and should be set to zero.

Bytes 32-35 When local mode is terminated by a program interrupt, the address located in bytes 33-35 is placed in the PSW. This address is within the boundary of the emulator program. The high-order byte is reserved and should be set to zero.

EXL EXAMPLE: EXECUTION and LOCAL MODE OPERATION START



Real Address Computation Example



Trap to D80C

Freeze I, V, W, TR, and PAA.

Trap routine GGST
Compute real address, load real address and local address into TLB 0.
Re-execute the storage word that caused the mismatch.

Local address	001
Adjustment factor	+014
Real address	015

When the EXL instruction has been executed and control is passed to the current PSW, the address-adjustment circuits are activated.

In local mode, each time an access is made to main storage, with an ADJ storage word, the local address portion of the PAA is checked against the local address area of all eight TLBs. When a match is detected, the real address from the TLB causing the match is gated to the M-register. The displacement value from PAA is also gated to the M-Reg to make up the complete real address of the location to be accessed.

When a no-match condition occurs, a trap is forced and the GGST microroutine is executed. The real address is formed, and the real address and local address are loaded into the least recently used TLB. The storage word causing the mismatch is re-executed. The resulting match condition gates the real-address to the M-register for a main-storage access and updates the LRU matrix.

Adjust CCW String

	0F	B1	D1

- The operand address designates the Adjust CCW (ACCW) list.
- With the information from the ACCW list, this instruction addresses CCWs and performs adjustment on the data addresses of the CCWs.

The ACCW instruction interprets successive doublewords as CCWs and adjusts their data addresses by algebraically adding the adjustment factor to them. This process continues until:

The last CCW adjusted did not specify chaining, or

A CCW whose command code specifies TIC has been adjusted, or

A CCW whose data address points outside the emulated environment is encountered, or

The address of the next CCW is outside the limits of the emulated environment or does not specify a doubleword boundary.

Any of these conditions terminates the instruction and sets the proper condition code to specify the reason for termination.

When the ACCW instruction is completed, the address of the last CCW adjusted +8 is stored in bytes 17-19 of the ACCW list for condition codes 0, 1, or 2. For condition code 3, the address stored is CCW + 0. If data chaining was in progress, the command code and the address of the CCW containing the command code are set in the operation byte and operation pointer fields, respectively.

If the last CCW adjusted specified transfer in channel, bytes 21-23 of the ACCW list contain the unadjusted data address from the TIC CCW. If the TIC CCW is encountered in a data-chaining sequence, the operation byte and operation pointer of the ACCW list contain the values set from the first CCW of the chain. When the TIC is not data-chained, the operation byte in the ACCW list is set to zero. The CCW address field in the ACCW list is set to the address +8 of the TIC CCW.

Condition Code

0 End of the CCW string. The last CCW adjusted specified neither data chaining nor command chaining.
1 A TIC CCW was the last CCW adjusted.
2 An adjusted data address was encountered that fell outside the area of the emulated environment.
3 The address of the next CCW to be adjusted did not specify a doubleword boundary or fell outside the area of the emulated environment.

Program Interruptions

Addressing: The address of the ACCW list is outside available storage. The operation is suppressed.
The address of a CCW is outside available storage. The operation is terminated..

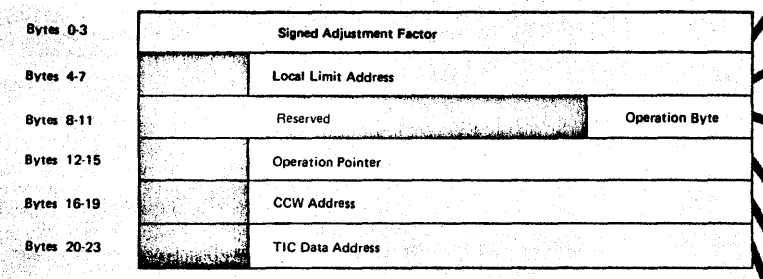
Operation: The instruction is not installed. The operation is suppressed.

Protection: The ACCW list is protected for storing or fetching. The operation is suppressed.
A CCW is protected for fetching or storing. The operation is terminated.

Specification: The first operand address does not specify a 64-byte boundary; the signed adjustment factor is not a multiple of 4096; the local limit address is not one less than a multiple of 4096. The operation is suppressed.

Special Operation: The ACCW instruction was encountered while in local mode. The operation is suppressed. The interruption is reflected to the program that placed the CPU in local mode by an address in the local mode or by an address in the local list.

Adjust CCW List Format and Definition



Bytes 0-3 — **The signed binary number located in this field is added to the data address of the CCW addressed by bytes 16-19 of the ACCW list. The 24 low-order bits of the result are set into the data address field of the CCW. The CCW data address, which is local to the emulated environment, is compared against the local limit address. If the comparison indicates that the local address is above the local limit address, a program interruption occurs.**

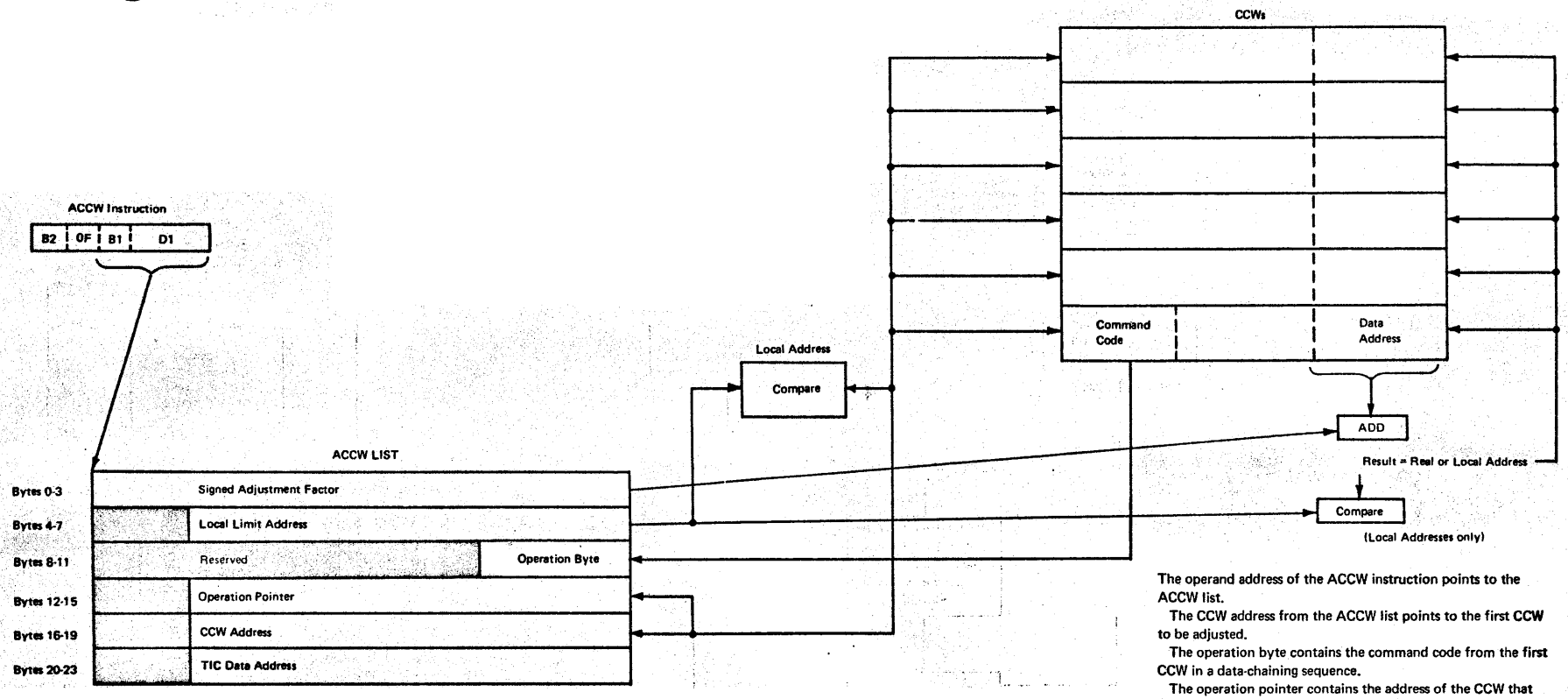
Bytes 4-7 — The address contained in bytes 5-7 specifies the upper address of the emulated environment. The address must specify one less than a 4096 boundary, or a specification interrupt occurs. The local limit address is compared with the local CCW address, and the extreme local address of the storage area defined for each CCW by the data address, command code, and unit, to assure that the local address is within the limits of the storage area assigned to the emulated environment.

Bytes 8-11 — The operation byte (byte 11) carries the command code for CCWs that are data-chained. The operation byte is set to zero if the CCW being processed is not data-chained. When the operation byte is fetched from the ACCW list, the high-order bytes of this field are ignored. When the operation byte is stored in the ACCW list, the high-order bytes are set to zero. A nonzero operation byte encountered upon initiation of the instruction indicates that the first CCW to be adjusted is part of a data-chained sequence.

Bytes 12-15 — This field contains the address of the CCW that originated the operation byte for the last non-TIC CCW adjusted. When this address is fetched from the ACCW list, the high-order byte is ignored. When the address is stored in the ACCW list, the high-order byte is set to zero.

Bytes 16-19 — The address contained in this field is the address of the first CCW of a string when the ACCW instruction is encountered. When the ACCW is completed, this address points to the doubleword above the last CCW adjusted when terminated with condition 0, 1, or 2. When terminated with condition code 3, the address points to the CCW causing termination. When this address is fetched from the ACCW list, the high-order byte is ignored. When the address is stored in the ACCW list, the high-order byte is set to zero.

Bytes 20-23 — Contains the unadjusted data address from the CCW whose command specifies transfer in channel. The high-order byte is set to zero.



The operand address of the ACCW instruction points to the ACCW list.

The CCW address from the ACCW list points to the first CCW to be adjusted.

The operation byte contains the command code from the first CCW in a data-chaining sequence.

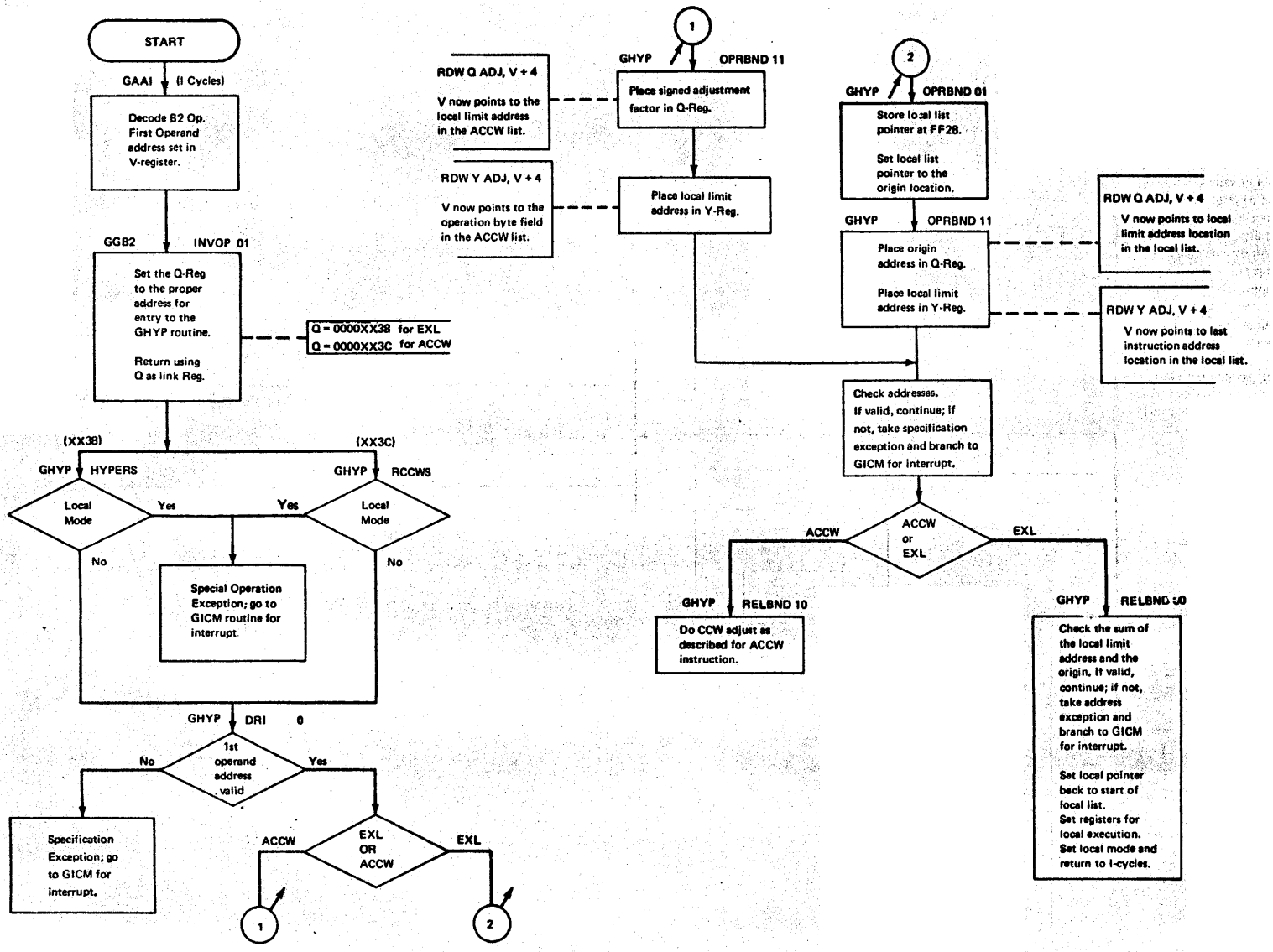
The operation pointer contains the address of the CCW that provided the operation byte.

Each CCW in the chain is addressed from the ACCW list.

The data address from the CCW is added with the signed adjustment factor from the ACCW list. The local CCW address is compared with the local limit address. If the comparison indicates a valid local address, the operation continues. If the comparison indicates an invalid local address, the operation is terminated and a condition code of 3 is set. The extreme local address of the storage area defined for each CCW by the data address, command code, and count are compared with the local limit address and zero. If the comparisons indicate a valid local storage area, the adjusted data address is placed in the data address field of the CCW. If the address compare is invalid, the operation is terminated and a condition code of 2 is set.

The CCW address in the ACCW list is updated +8 to point to the next CCW to be adjusted. When the last CCW in the string has been adjusted, the CCW address points to the next sequential doubleword.

EXL and ACCW Instruction Execution



Any interrupt removes the CPU from local mode.

All synchronous interrupts that occur while in local mode are handled by the emulator program.

All asynchronous interrupts that occur while in local mode are first handled by the OS supervisor.

If a supervisor call, program, External, I/O, or recoverable machine-check interruption occurs while the CPU is in local mode, the following actions take place:

The 16-bit interruption code associated with the supervisor call or program interruption is stored in the interruption code field of the LEX list. The contents of this field after an external, I/O, or machine-check interruption are unpredictable.

The ILC, CC, program mask, and instruction address of the current PSW are stored in the bytes 4-7 of the LEX list. The value of the ILC after an asynchronous interruption is unpredictable.

The current contents of general registers 14 and 15 are stored into bytes 8-15 of the LEX list.

If the interrupt is a supervisor call or program interrupt, the local address of the instruction causing the interruption is stored into bytes 25-27 of the LEX list. If the instruction causing the interrupt was the object of an Execute instruction, the local address of the Execute instruction is stored.

The address of the corresponding interrupt (SVC, program, or asynchronous) is loaded into the current PSW from the LEX list.

If the interrupt is an asynchronous interrupt, (I/O, external, or machine-check) the adjusted current PSW is stored in the corresponding low storage old PSW and interrupt is then sent to the OS supervisor.

The CPU is removed from local mode.

LEX LIST AFTER SVC INTERRUPT

Bytes 0-3	Programming Use	SVC Interrupt Code
Bytes 4-7	ILC, CC, Program Mask	Local Instruction Address
Bytes 8-11	General Register 14	
Bytes 12-15	General Register 15	
Bytes 16-19	Reserved	Origin Address
Bytes 20-23	Reserved	Local Limit Address
Bytes 24-27	Reserved	SVC or Execute Address
Bytes 28-31	Reserved	SVC Interruption Address
Bytes 32-35	Reserved	Program Interruption Address
Bytes 36-39	Reserved	Asynchronous Interruption Address

When the interruption is a specification exception due to an odd address, the ILC is unpredictable. The Last instruction field contains the odd address.

LEX LIST AFTER PROGRAM INTERRUPT

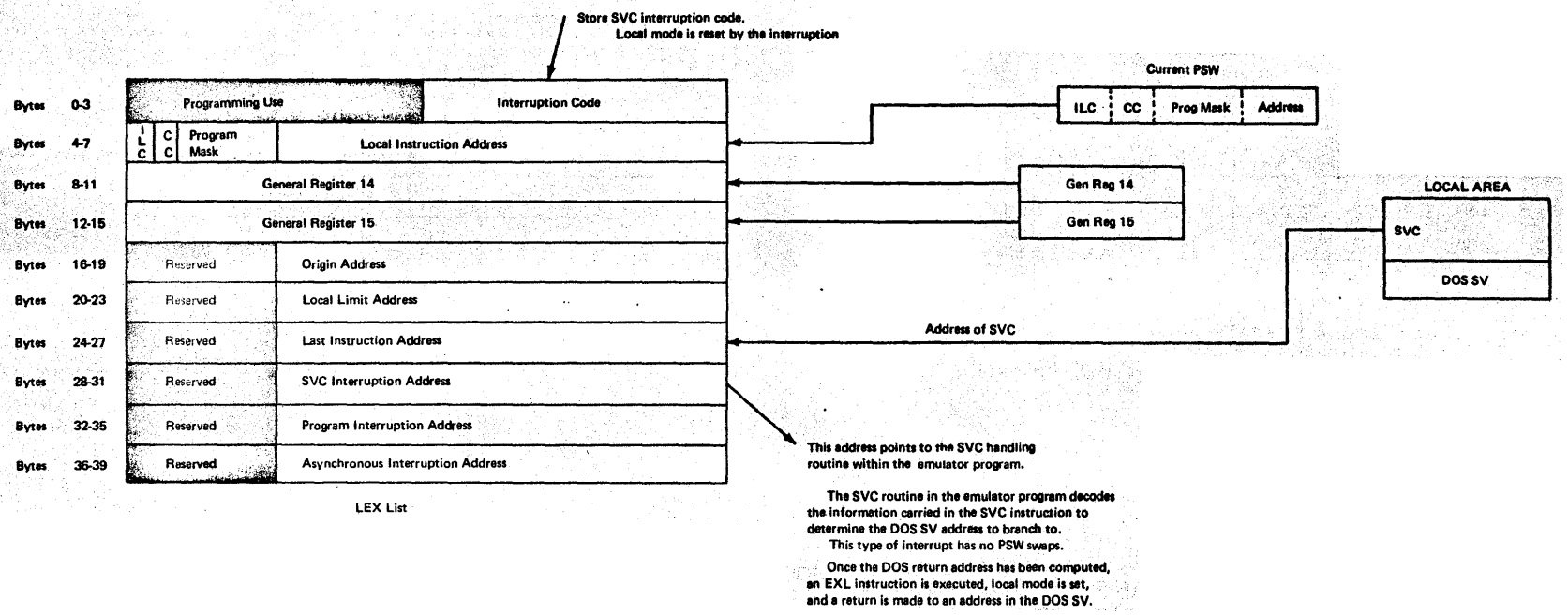
Bytes 0-3	Programming Use	Program Interrupt Code
Bytes 4-7	ILC, CC, Program Mask	Local Instruction Address
Bytes 8-11	General Register 14	
Bytes 12-15	General Register 15	
Bytes 16-19	Reserved	Origin Address
Bytes 20-23	Reserved	Local Limit Address
Bytes 24-27	Reserved	Addr of instruction causing interrupt or execute address
Bytes 28-31	Reserved	SVC Interruption Address
Bytes 32-35	Reserved	Program Interruption Address
Bytes 36-39	Reserved	Asynchronous Interruption Address

LEX LIST AFTER AN ASYNCHRONOUS INTERRUPT

Bytes 0-3	Programming Use	UNPREDICTABLE
Bytes 4-7	*ILC, CC, Program Mask	Local Instruction Address
Bytes 8-11	General Register 14	
Bytes 12-15	General Register 15	
Bytes 16-19	Reserved	Origin Address
Bytes 20-23	Reserved	Local Limit Address
Bytes 24-27	Reserved	UNPREDICTABLE
Bytes 28-31	Reserved	SVC Interruption Address
Bytes 32-35	Reserved	Program Interruption Address
Bytes 36-39	Reserved	Asynchronous Interruption Address

*ILC is unpredictable.

SVC INTERRUPTION EXAMPLE



The DOS supervisor operates in problem state. When a Start I/O instruction (privileged instruction) is executed in the DOS supervisor, the DOS emulator intercepts the program interrupt causing the following to be performed by the 3145 microprogram (Emulator Interrupts GHYI).

A
1. The privileged operation interruption code is set into bytes 2 and 3 of the LEX list.
2. The selected current PSW instruction address is loaded into bytes 4-7 of the LEX list.
3. The address of the Start I/O instruction is stored into bytes 25-27 of the LEX list.
4. The current values of general registers 14 and 15 are stored into bytes 8-15 of the LEX list.
5. The program interruption address, bytes 33-35 of the LEX list, is placed into the current PSW.
6. The CPU is removed from LEX Mode.

B The DOS emulator commences instruction execution at the the address specified by the program interruption address and performs the following:

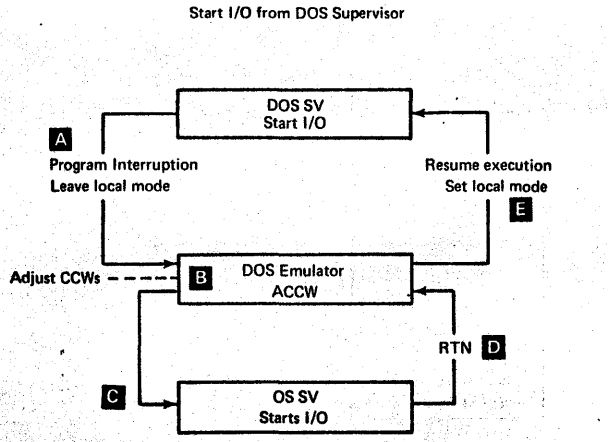
1. The ACCW instruction is executed to adjust the CCW data addresses to their values in the emulator program.

C 2. Control is transferred to the OS SV to handle the Start I/O.

D When the OS SV has handled the Start I/O, a return is made to the DOS Emulator. The EXL instruction is then executed to restore information prior to restarting the DOS SV at the point of interruption. Refer to "EXL Example: Execution."

E Upon completion of the I/O operation, another ACCW instruction is executed to restore the CCW data addresses to their local values.

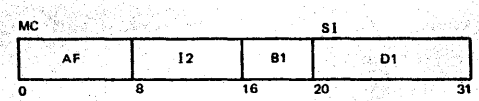
START I/O INTERRUPTION EXAMPLE



MONITOR CALL

The monitor call feature allows the programmer to place a trace on his program operation. The monitor call instruction is inserted as desired, throughout the program to cause program interrupt when that point is reached. The conditions can be made selective by segregating them into up to sixteen classes. These classes can be masked to allow interrupts only for specific conditions. A code number placed in the operand field of the instruction is reported along with the monitor class number during each interruption.

Monitor Call Instruction



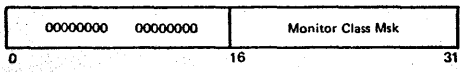
I2 Bits 8-11 must be zero.
Bits 12-15 contains one of sixteen monitor class numbers in binary notation for the test.

B1 General register to used in determining the monitor code along with D1.

D1 This field, along with the register defined by B1, is added to produce the monitor code. (D1 can identify 256 codes without the use of the register.)

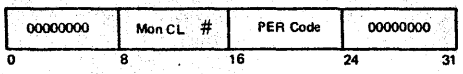
The monitor class number specified in the I2 field of the instruction is tested against the monitor class mask in control register 8. The sixteen bit mask identifies the monitor classes with bit 16 for class 0 and bit 31 for class 16. If the mask allows the class, a program interrupt is initiated. When a class is inhibited, the monitor call instruction passes as a No-Op.

Control Register 8

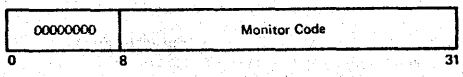


When the instruction initiates an interrupt, the monitor class number specified by the instruction is posted in permanent storage byte location 95. The class is posted as a binary number in the low-order with the high-order set to zero. The computed monitor code is set into permanent storage word location 156 with byte 0 set to zero.

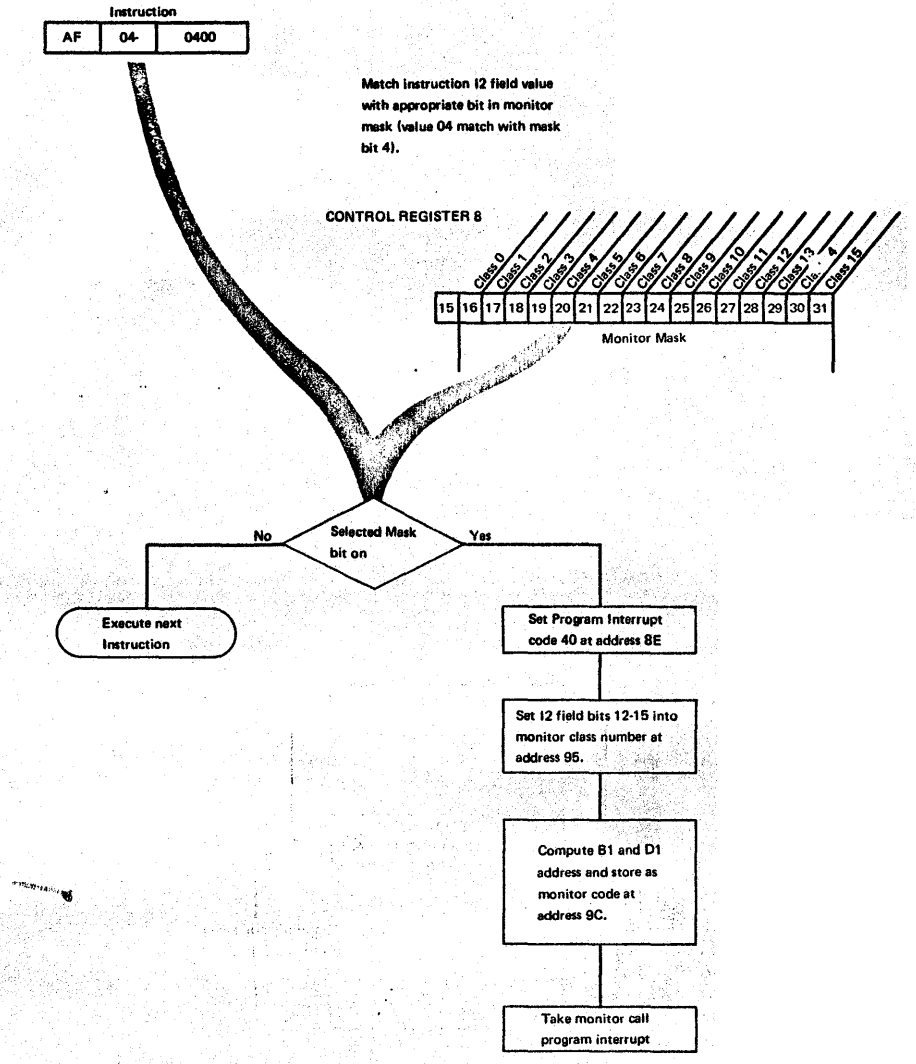
Permanent Storage 94



Permanent Storage 9C



Example:



EXTENDED CONTROL MODE

Introduction

The extended control (EC) feature of the 3145 initiates and reports a number of System/370 functions. The CPU can still operate in basic control (BC) mode to accommodate programs written for System/360. A new PSW format is used along with control registers and extensions to the permanent assigned storage to implement EC mode.

Bit 12 of both the old and new formats of the PSW has a common notation for identification. When the bit is set to 0, the CPU reads the PSW as BC mode. If the bit is set to 1, the PSW is read as EC mode. The change from one mode to the other can be made with any PSW interchange. An I/O operation can be started in one mode and continued to an ending in the other mode. EC mode is required in order to perform most of the System/370 features.

Feature Mask

The changes to the basic PSW are made to satisfy requirements of the EC feature. An expanded system mask controls additional features.

Mask bits for the new features replace the system (I/O) mask of the BC mode PSW. The system mask is moved to the control registers. The condition code and program mask fields have been moved to bits 18 - 23 of the EC mode PSW. The ILC and the interruption codes have been moved to an area of the permanently assigned storage.

EC mode activates the following features on the 3145. System Mask bits on 1 allow the indicated features to function.

R Program Event Recording (PER) – This feature allows the programmer to debug his programs by identifying instructions that could cause trouble in operation. These include branch conditions, instruction fetch, storage alterations, and general register alterations. The recognition of one of these conditions causes a program interruption with the instruction address and code posted in a permanent storage location.

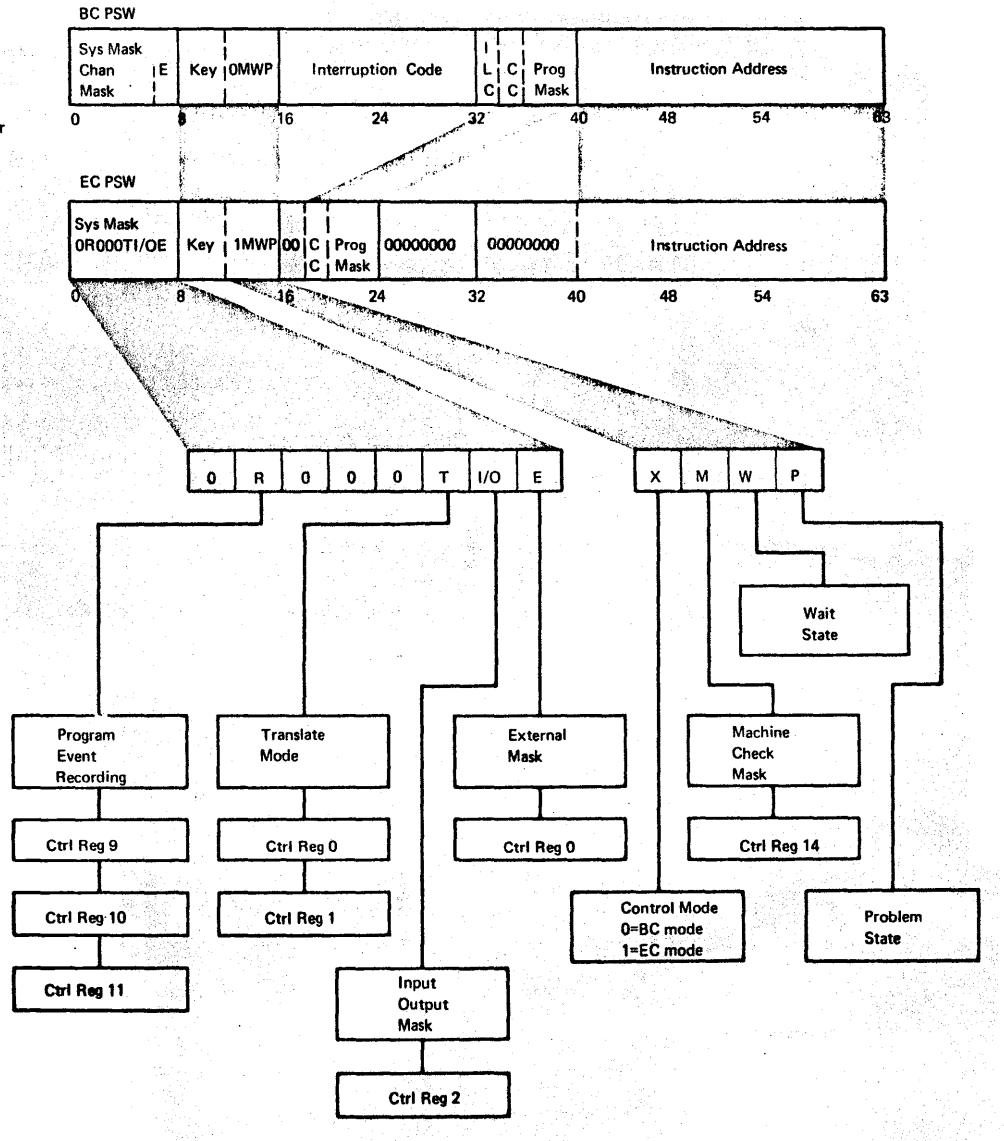
T Dynamic Address Translation (DAT) – This feature allows conversion of programs expressed in virtual address to real address in main storage. The translation does not occur for I/O addresses or for permanently assigned addresses used by CPU. The I/O counterpart is the Indirect Data Addressing (IDA) feature that is not masked and is allowed in BC mode.

I/O Input/Output Mask – The I/O mask that allows interruptions for the I/O channels selectively is located in control register 2. The mask bit in the EC mode PSW represents a master mask.

E External Mask – The External mask that allows interruptions for the external devices selectively is located in control register 0. The mask bit in the EC mode PSW represents a master mask.

The diagram shows the related mask bit, feature and control registers. Greater detail of the feature is defined under the feature name.

BC and EC PSW formats



Control Registers

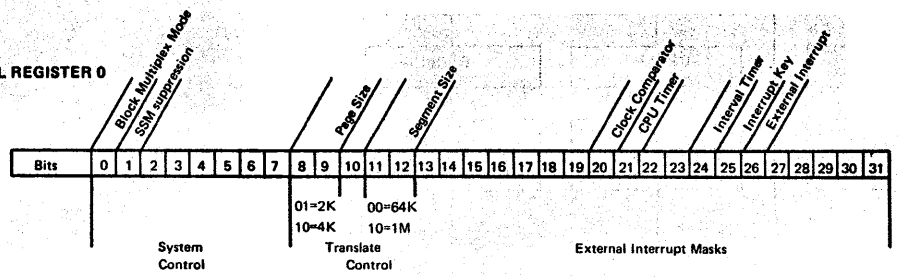
The 3145 has sixteen one-word control registers that provide an extension to the EC mode PSW. The control registers are located in control storage at addresses F480 through F4BF. Three of these control registers also function with BC mode for external mask and machine check controls. Details of the register assignments are contained in the sections of this manual that discuss the features.

The registers are loaded to standard masking and address values during system clear. The registers can be loaded from main storage with the load control instruction (LCTL). The information in the registers can be transferred to main storage with the store control instruction (STCTL). When using the load and store instructions, the unused registers must be considered in the main storage area specified.

CONTROL REGISTER ASSIGNMENTS

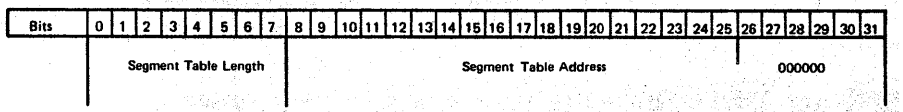
	0	8	16	24	31
0	System Control	Translate Control	External Interrupt Masks		
1	Segment Table Length	Segment Table Origin Address			
2	Channel Masks				
3	Reserved				
4	Reserved				
5	Reserved				
6	Unassigned				
7	Unassigned				
8			Monitor Mask		
9	PER Event Masks	00000000	PER General Register Alteration Mask		
10	00000000	PER Starting Address			
11	00000000	PER Ending Address			
12	Unassigned				
13	Unassigned				
14	Error Recovery Control and Masks				
15	00000000	MCEL Address			

CONTROL REGISTER 0



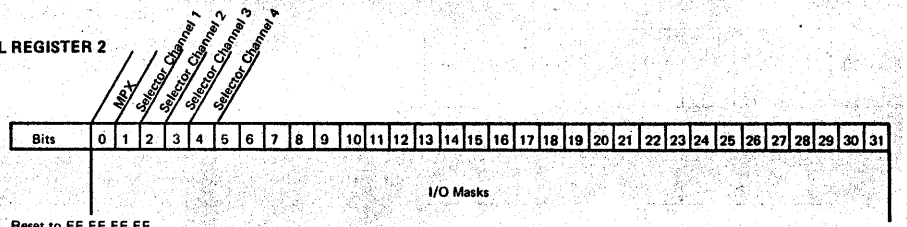
Reset to 00 00 00 E0

CONTROL REGISTER 1



Reset to 00 00 00 00

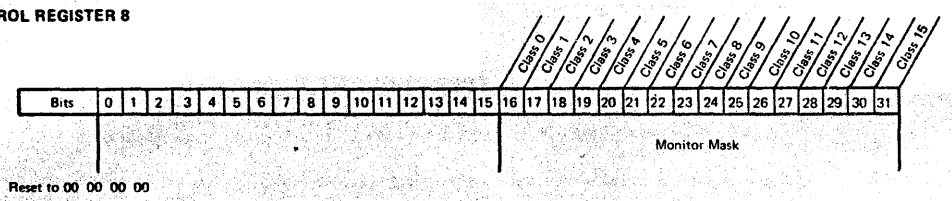
CONTROL REGISTER 2



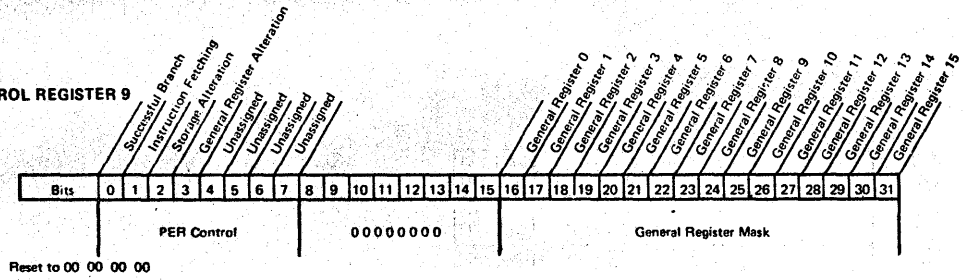
Reset to FF FF FF FF

Note: Initial Value of unassigned positions in all registers is unpredictable but is assumed to be 0.

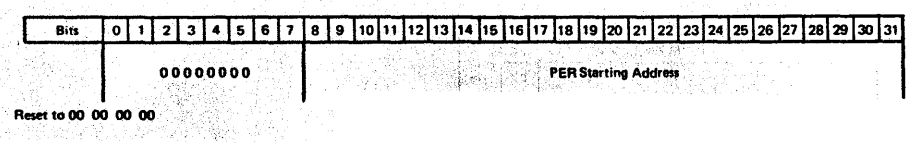
CONTROL REGISTER 8



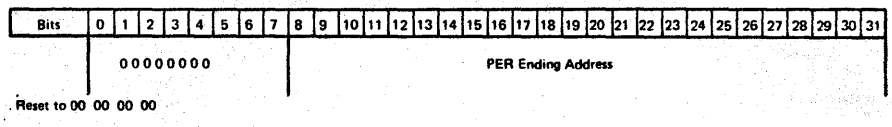
CONTROL REGISTER 9



CONTROL REGISTER 10



CONTROL REGISTER 11



Permanent Storage Assignments

System/370 assigns the first 128 bytes of main storage as fixed operation-addresses and adds assignments to much of the area up to 256 bytes. It also reserves the main storage addresses between 256 and 512 for feature use and an option for logout starting at address 512.

Most of the new area between address 80 and 100 is used only by EC mode operation and features. BC mode operations do not develop the interruption codes and logging information that is stored here. The features added with EC mode also post their result codes and addresses in this new area.

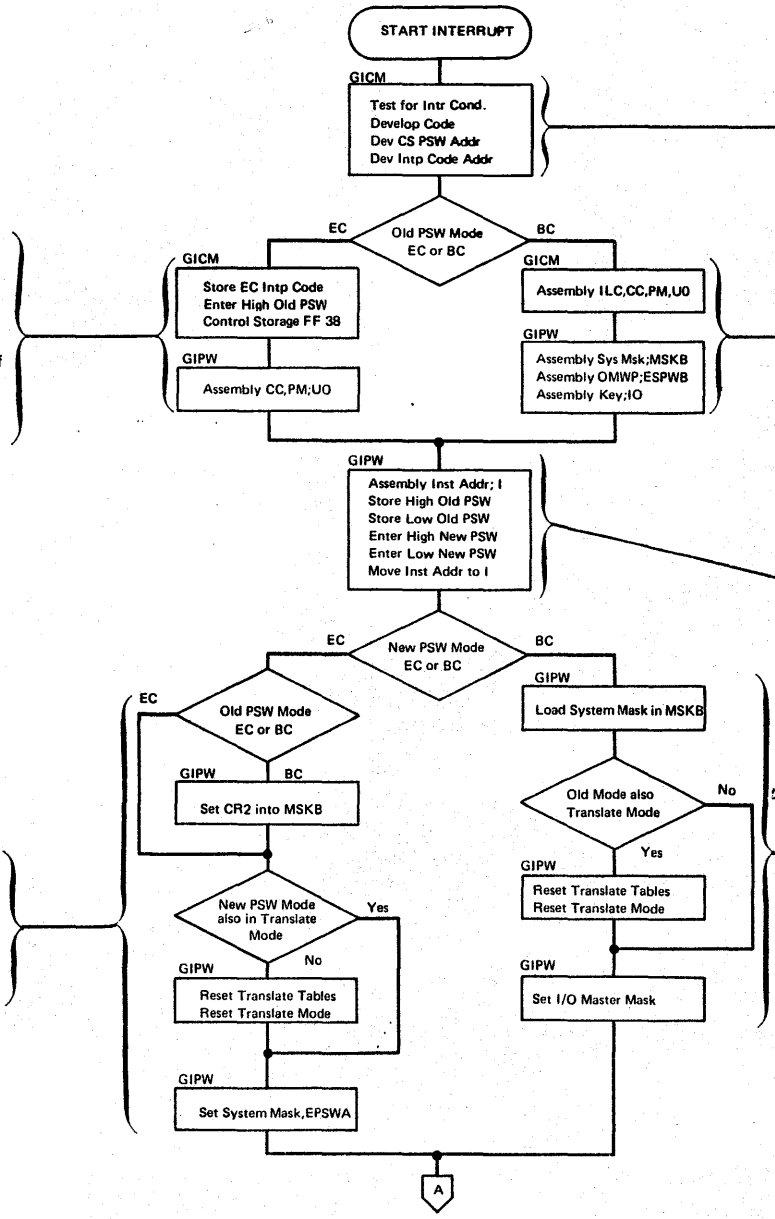
An I/O interruption posting area located between addresses AC and C0 allows greater error definition in EC mode, than in BC mode. A portion of this posting area is also posted by BC mode operations to allow for combinations of EC/BC mode. The BC mode program cannot always handle the information automatically. Details of the permanent storage assignments are in the manual sections that discuss the features.

DEC	HEX	PERMANENT STORAGE ASSIGNMENT
0	00	Initial Prog Load PSW or
0	00	
4	04	PSW Restart New PSW
8	08	Initial Prog Load CCW1 or
12	0C	PSW Restart Old PSW
16	10	Initial Prog Load CCW2
20	14	
24	18	External Old PSW
28	1C	
32	20	Supervisor Call Old PSW
36	24	
40	28	Program Old PSW
44	2C	
48	30	Machine Check Old PSW
52	34	
56	38	Input/Output Old PSW
60	3C	
64	40	Channel Status Word
68	44	
72	48	Channel Address Word
76	4C	Unassigned
80	50	Timer
84	54	Unassigned
88	58	External New PSW
92	5C	
96	60	Supervisor Call New PSW
100	64	
104	68	Program New PSW
108	6C	
112	70	Machine Check New PSW
116	74	
120	78	Input/Output New PSW
124	7C	

DEC	HEX	PERMANENT STORAGE ASSIGNMENT			
128	80	Unassigned			
132	84	0000000000000000	Ext Intp Code		
136	88	0000000000000 ILC	SCV Intp Code		
140	8C	0000000000000 ILC	Prog Intp Code		
144	90	00000000	Translation Excpt Addr		
148	94	00000000	Monitor Class No.	PER Code	00000000
152	98	00000000	PER Prog Event Addr		
156	9C	00000000	Monitor Code		
160	A0	Unassigned			
164	A4				
168	A8	Channel ID			
172	AC	00000000	IOEL Pointer		
176	B0	Limited Channel Logout (ECSW)			
180	B4	Unit St	Chan St	Count	
184	B8	Key | Flag	I/O Address		
188	BC	CCW Address			
192	C0	Unassigned			
196	C4				
200	C8				
204	CC				
208	D0				
212	D4				
216	D8	CPU Timer			
220	DC				
224	E0	Clock Comparator	Reserved		
228	E4				
232	E8	Machine Check Intp Code			
236	EC				
240	F0	Unassigned			
244	F4				
248	F8	00000000	Failing Storage Addr		
252	FC	Region Code			
256	100	CPU Independent Log			

PSW Interchange Sequence

When starting an interruption sequence, the current PSW is stored as the old PSW for that class of interruption. The new PSW for the interruption class is then read in and set as the current PSW for the interruption routine. The sequence is similar in both BC and EC mode. The differences result from the allowable features. A similar sequence is taken at the end of the interruption when the old PSW is returned from storage to fill the current PSW in order to continue the problem program.



- When a pending interruption is recognized following the completion of an instruction, the PSW interchange is started in the GICM routine.
- The interruption code is developed, and the addresses required for EC mode operation are developed before testing for the control mode.

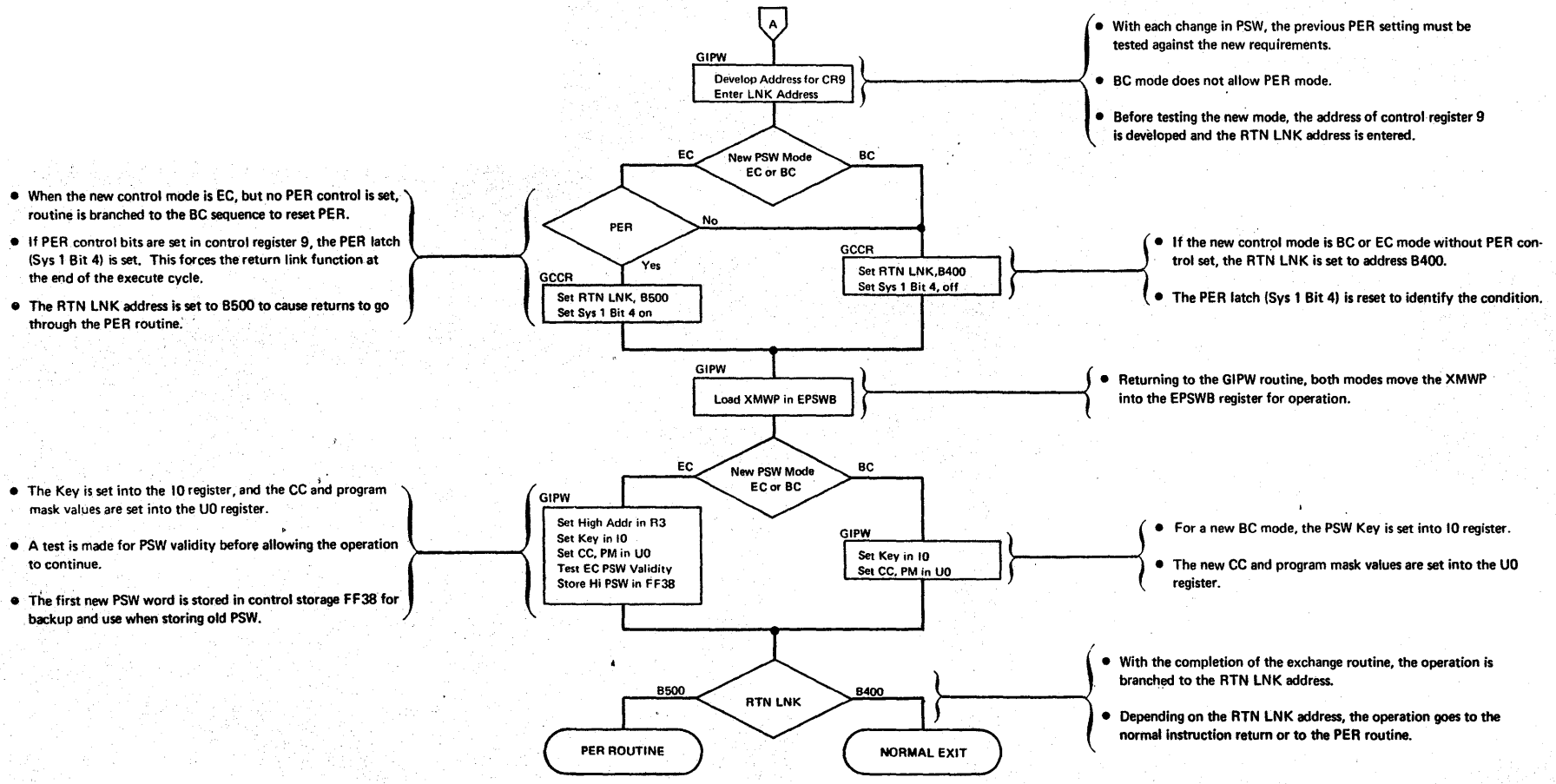
- Before leaving the GICM routine, the EC interruption code is stored in the assigned permanent storage location, and the first word of the entered PSW is returned from control storage FF38.
- After branching to the GIPW routine, the CC and the program mask from the UO register are set in byte-2 of the first word of the PSW.
- The ILC is not stored in the old PSW for EC mode but is stored with the interruption code when required.

- Before leaving the GICM routine, the ILC, the CC, and the program mask are entered from the UO register and set as byte-0 of the second word of the PSW.
- After branching to the GIPW routine, the system mask (I/O) from the MSKB register is set as byte-0 of the first PSW word.
- For byte-1 of the first word the OMWP is entered from the EPSWB register and the protection key is entered from the the IO register.

- The low-order three bytes of the I-register are set as bytes 1, 2, and 3 of the second word of the PSW.
- The two assembled PSW words are stored in the appropriate old PSW storage location.
- The two new PSW words are entered from their permanent storage location, and the instruction address is moved to the I-register.

- If the control mode is changing from BC to EC, the I/O mask from control register 2 is moved to the MSKB register.
- If the new PSW does not call for translate mode, the translate tables and the mode bit are reset to ensure that the mode is off.
- The EC system mask is moved to the EPSWA register for operation.

- The BC mode system mask is the I/O mask, and is moved to the MSKB register.
- If the previous operation was in EC mode with translate mode, the translate tables and the mode bit are reset.
- The master I/O and external mask bits in the EPSWA register are set if bits are set in the MSKB register.



- With each change in PSW, the previous PER setting must be tested against the new requirements.
- BC mode does not allow PER mode.
- Before testing the new mode, the address of control register 9 is developed and the RTN LNK address is entered.

- When the new control mode is EC, but no PER control is set, routine is branched to the BC sequence to reset PER.
- If PER control bits are set in control register 9, the PER latch (Sys 1 Bit 4) is set. This forces the return link function at the end of the execute cycle.
- The RTN LNK address is set to B500 to cause returns to go through the PER routine.

- If the new control mode is BC or EC mode without PER control set, the RTN LNK is set to address B400.
- The PER latch (Sys 1 Bit 4) is reset to identify the condition.

- Returning to the GIPW routine, both modes move the XMWP into the EPSWB register for operation.

- The Key is set into the I0 register, and the CC and program mask values are set into the U0 register.
- A test is made for PSW validity before allowing the operation to continue.
- The first new PSW word is stored in control storage FF38 for backup and use when storing old PSW.

- For a new BC mode, the PSW Key is set into I0 register.
- The new CC and program mask values are set into the U0 register.

- With the completion of the exchange routine, the operation is branched to the RTN LNK address.
- Depending on the RTN LNK address, the operation goes to the normal instruction return or to the PER routine.

Interrupt Codes

INTERRUPT					
			ADDRESS		
CLASS		CODE	BC	EC	FUNCTION
External		1005	1A	86	CPU Timer
		1004			Clock Comparator
		0040			External Interrupt Key
		0080			Interval Timer
	(Note 1)	00XX			Direct Control
Supervisor Call	(Note 2)	XXXX	22	8A	
Program		01-0F	2A	8E	(See *System/370 Principles of Operation,* GA22-7000.)
		10			Segment Translation Exception
		11			Page Translation Exception
		12			Translation Specification
		40			Monitor Call
		80			PER (Note 4)
Machine Check			32	E8	See Error Handling (Machine Check Logout)
I/O	(Note 3)	XXXX	3A	BA	

Note 1 Code is bit significant by external interrupt signals.

Note 2 Code is dependent on the I-field of the supervisor call instruction.

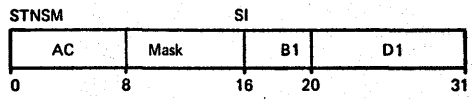
Note 3 Code is dependent on I/O device address.

Note 4 PER interrupt concurrent with another program interrupt ORs the value 80 with another interrupt code to yield a resultant interrupt code.

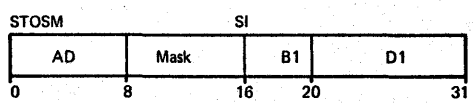
Store Then Mask Instructions

The two store then mask instructions can adjust the system mask during the program sequence. The AND instruction removes mask bits from the mask. The OR instruction adds bits to the mask. In either case the current mask is stored at the address specified for later recall, when required. If control register 0 bit 1 = 1, the instructions are not executed.

Store Then AND System Mask



Store Then OR System Mask



The store then mask instructions are performed in supervisor mode for either BC or EC mode. The first eight bits of the current PSW are stored at the specified first operand address in main storage. The current system mask bits are then logically ANDed or ORed with the immediate operand of the instruction and the results returned to the operating register. In BC mode, the I/O mask stored in the MSKB external register is stored and adjusted. For EC mode the new system mask stored in the EPSWA external register is stored and adjusted. No other changes are made to the operation.

DYNAMIC ADDRESS TRANSLATION

- DAT (dynamic address translation) is a combination of software, hardware, and microprogramming.
- No special programming conventions are required by the problem programs to utilize the DAT feature.
- The Supervisor program handles the problem programs by dividing them into unique blocks, and moving these blocks in and out of real storage as needed.

Introduction

The DAT feature enables the storage capabilities of the 3145 to be extended beyond the real storage size. This is accomplished by locating programs and data on an external direct access storage device (disk, drum, etc.). This is referred to as virtual storage. The information resides in virtual storage until the controlling program requires it. At this time, control is transferred to a supervisor program which uses I/O routines to move the needed information into main storage. Control is returned to the program requiring the information to complete the interrupted operation. A maximum of sixteen million bytes (16M) may be used as virtual storage by the 3145.

The supervisor program controls data movements from or to virtual storage by dividing it into increments designated as segments and pages. A segment is the largest division of virtual storage and may be either 1M or 64k bytes in size. The page is a subdivision of the segment and may be either 4k or 2k bytes in size. Segment and page size are parameters set up in the supervisor program according to the customer operational needs. The movement of data between real storage and virtual storage is always in pages of 4k- or 2k-byte increments depending on which size the supervisor is using. To control this data movement, the supervisor program initially builds segment and page tables when the real and virtual storages are loaded. These tables are continuously being updated to reflect the current location of data as it is being used by problem programs and moved between storages by the supervisor.

DAT is a hardware addressing function. The software supervisor manipulates data between storages and sets up and maintains the tables and control used by the hardware. DAT uses the hardware assigned for the OS DOS Compatability feature. Both the DAT and OS DOS features can be active at the same time. Handling of the address adjustment factor for both is accomplished by the GGST micro routine.

Addresses Subject to Translation

When the System/370 microprogram handles addresses that are subject to translation, the storage control word using that address, specifies the ADJ function. The ADJ decode enables the address adjustment hardware. If address adjustment is designated by the EPSW, the translation process takes place.

The following addresses are maintained, interpreted, or stored as virtual addresses, and are subject to translation:

1. Instruction address in PSW
2. Branch addresses
3. Operand addresses when that instruction uses the address to refer to a main storage location. This excludes Set Storage Key, Insert Storage Key, and Reset Reference Bit Addresses.
4. Address stored in register 1 by Translate and Test and Edit and Mark.
5. Address stored at location Hex 90 on a translation exception interruption.
6. Address stored at location Hex 98 on a program event recording (PER) interruption.
7. PER starting address in control register 10.
8. PER ending address in control register 11.

Examples of storage words designating address translation:

```
RDW   Q   ADJ, W+4
STW   X   ADJ, V
```

Addresses Not To Be Translated

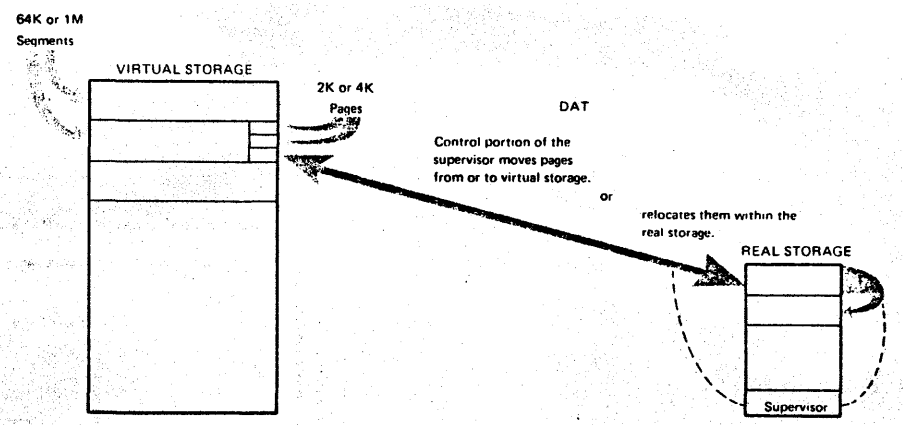
Address that are not to be translated are handled by storage control words that do not specify the ADJ function. When a storage control word without ADJ is executed, the ADR ADJ trap latch cannot be set (no mis-match trap can occur), and no addressing of the translation lookaside buffers takes place.

The following addresses are not translated by the CPU or channel.

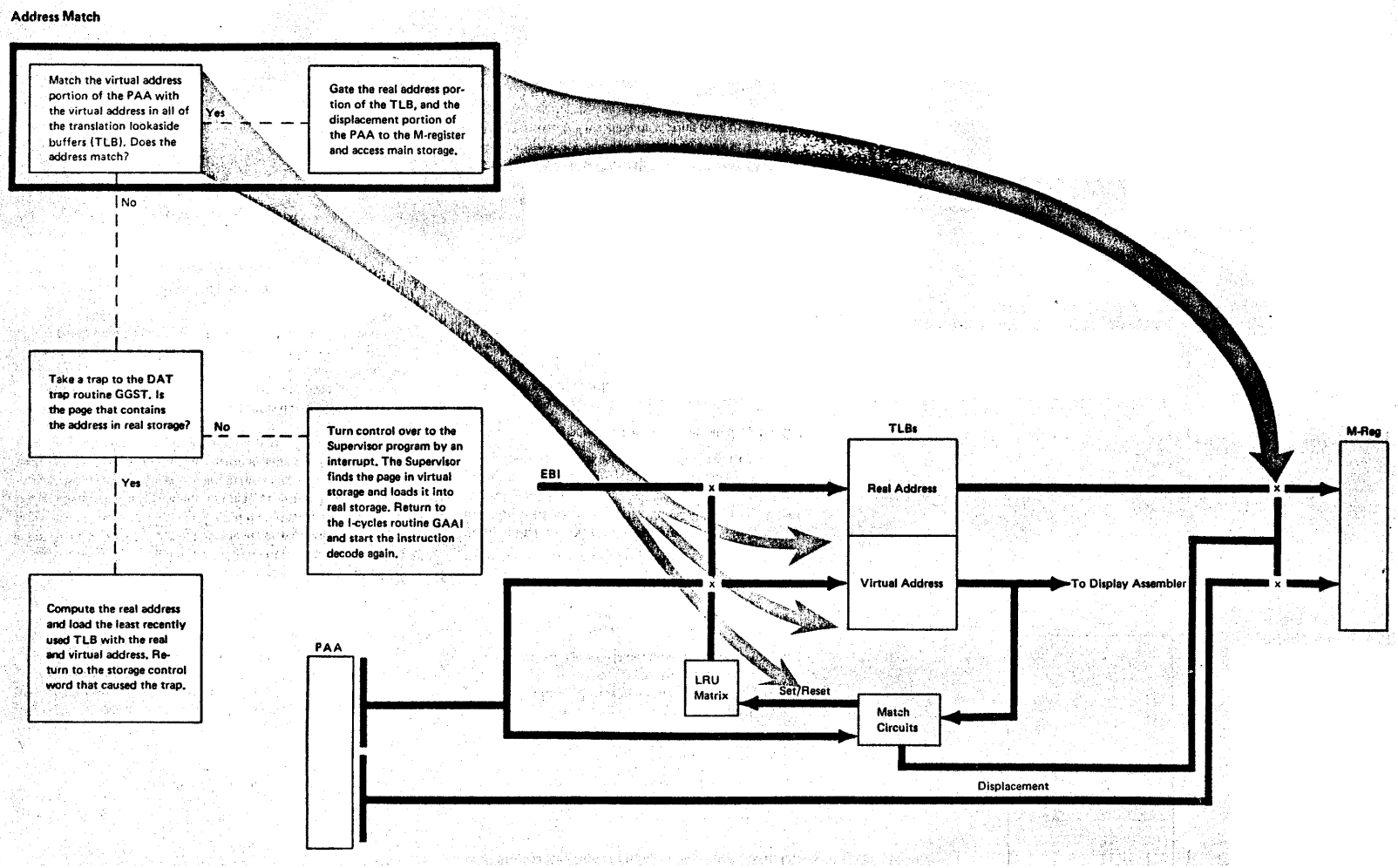
1. Segment table origin address.
2. Page table origin address.
3. Machine check extended log pointer in control register 15.
4. I/O extended log pointer at location Hex AC.
5. Address stored at location Hex F8 (failing storage address) on machine check interruption.
6. Region code stored at location Hex FC on machine check interruption.
7. PSW addresses.
8. Address used by hardware to update location Hex 50 timer.
9. Address for command address word for Start I/O or Start I/O Fast Release Hex 48.
10. Addresses for channel command words.
11. Addresses for fetching or storing data by the channel.
12. Address for channel status word during execution of an I/O instruction or during an interruption.
13. Address of PSW used during an IPL.

Examples of storage words not designating address translation:

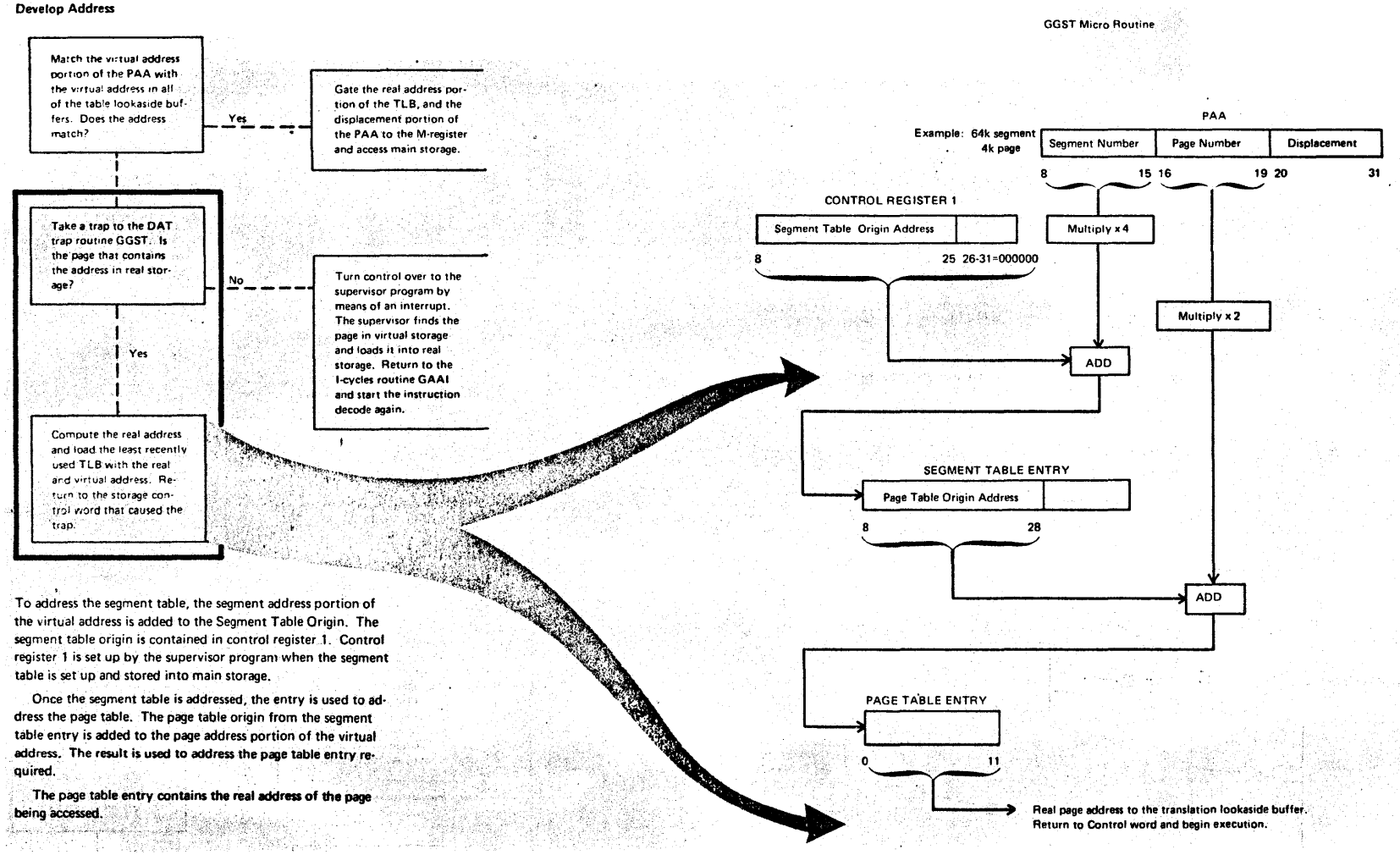
```
RDW   R   W+4
STW   Q   V-4
```



Basic Operation of Dynamic Address Translation



Develop Address

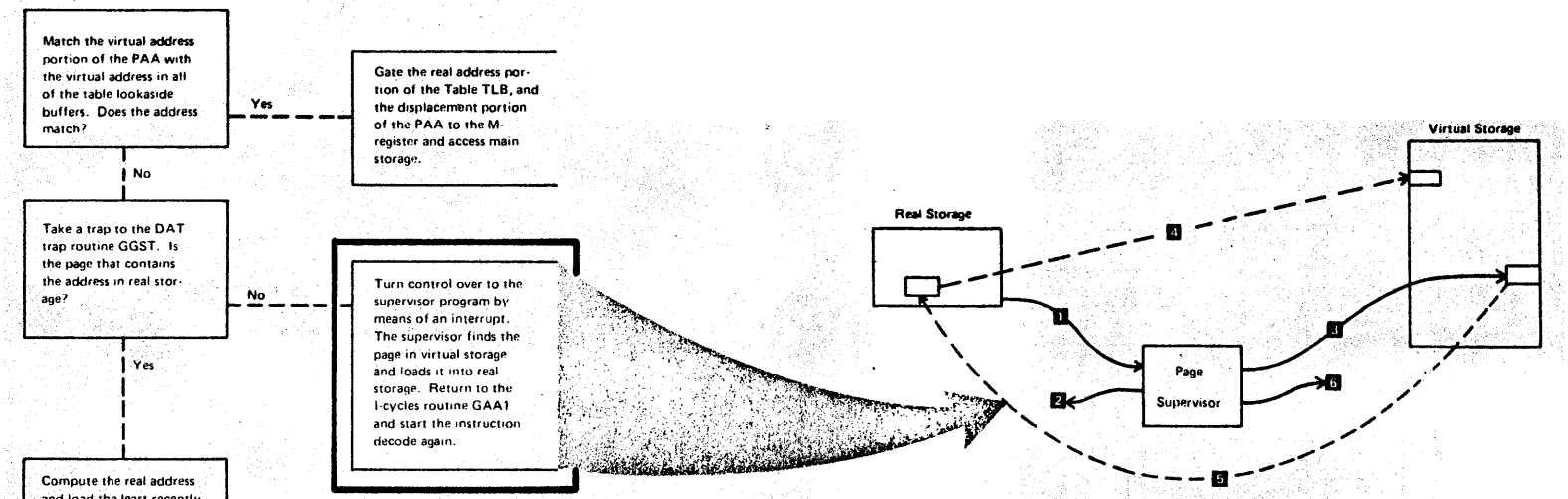


To address the segment table, the segment address portion of the virtual address is added to the Segment Table Origin. The segment table origin is contained in control register 1. Control register 1 is set up by the supervisor program when the segment table is set up and stored into main storage.

Once the segment table is addressed, the entry is used to address the page table. The page table origin from the segment table entry is added to the page address portion of the virtual address. The result is used to address the page table entry required.

The page table entry contains the real address of the page being accessed.

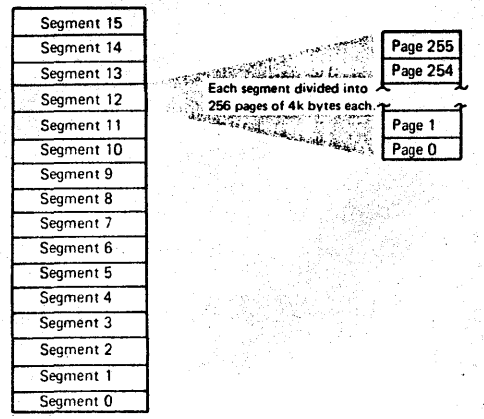
Address Paging



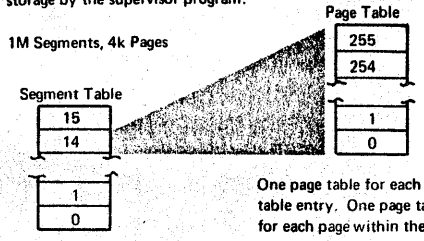
1. An attempt is made to use an address that is not in real storage. A program interrupt transfers control to the paging supervisor.
2. Paging supervisor interrogates page status by examining storage protect keys (reference and change bits), to find a page that can be replaced.
3. Supervisor finds the needed page in virtual storage.
4. Page out the real storage page if required.
5. Page in the virtual page to the real storage location.
6. Return to I-cycles routine GAAI to execute the instruction the trap was initiated from.

Segment and Page Sizes

VIRTUAL STORAGE = 16 MILLION BYTES



Segment and page tables are located in main storage by the supervisor program.



One page table for each segment table entry. One page table entry for each page within the segment.

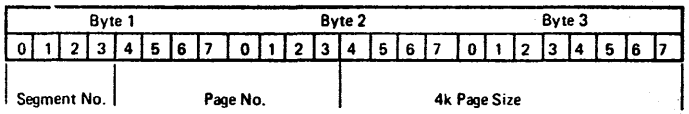
One Segment table entry for each segment of virtual storage.

Control register 0 bits 8, 9, 11, and 12 are set in the following manner to indicate the segment and page sizes:

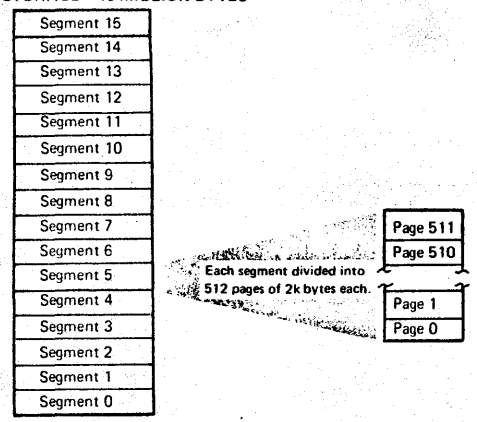
8=1 ------ 4k Page size
9=0
11=1 ------ 1M Segment size
12=0

NP2 bit 3=1 --------- 1M Segment
NP2 bit 4=1 --------- 4k Page size

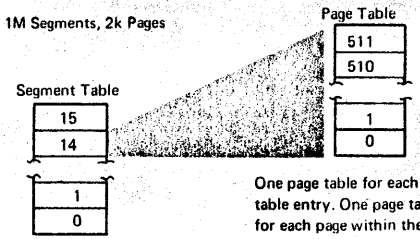
PAA



VIRTUAL STORAGE = 16 MILLION BYTES



Segment and page tables are located in main storage by the supervisor program.



One page table for each segment table entry. One page table entry for each page within the segment.

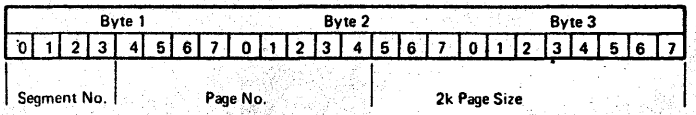
One Segment table entry for each segment of virtual storage.

Control register 0 bits 8, 9, 11, and 12 are set in the following manner to indicate the segment and page sizes:

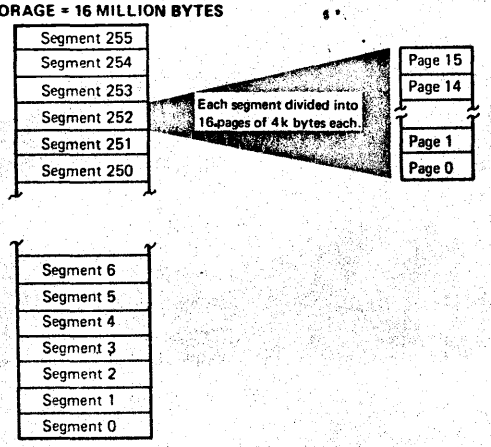
8=0
9=1 ------ 2k Page size
11=1 ------ 1M Segment Size
12=0

NP1 bit 3=1 --------- 1M Segment
NP2 bit 4=0 --------- 2k Page size

PAA

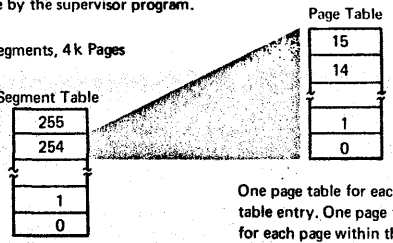


VIRTUAL STORAGE = 16 MILLION BYTES



Segment and page Tables are loaded in main storage by the supervisor program.

64k Segments, 4k Pages



One page table for each segment table entry. One page table entry for each page within the segment.

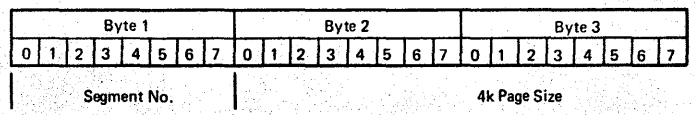
One segment table entry for each segment of virtual storage.

Control register 0 bits 8, 9, 11, and 12 are set in the following manner to indicate the segment and page sizes:

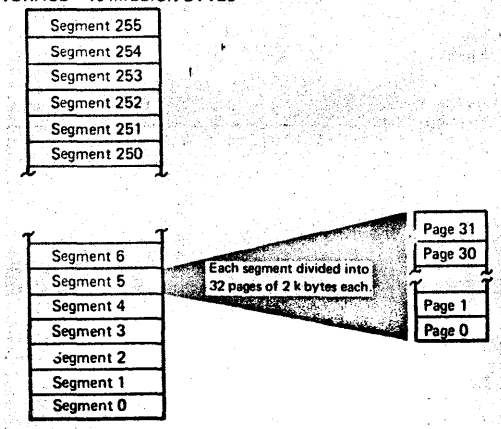
8=1-------4k Page size
9-0
11=0
12= 0--- 64k Segment size

NP2 bit 3=0-------- 64k Segment size
NP2 bit 4=1-------- 4k Page size

PAA

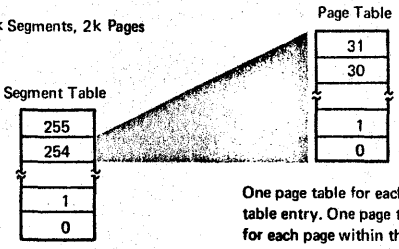


VIRTUAL STORAGE = 16 MILLION BYTES



Segment and page Tables are located in main storage by the supervisor program.

64 k Segments, 2k Pages



One page table for each segment table entry. One page table entry for each page within the segment.

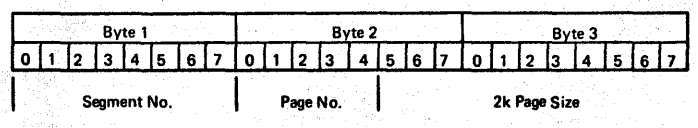
One segment table entry for each segment of virtual storage.

Control register 0 bits 8, 9, 11, and 12 are set in the following manner to indicate the segment and page sizes:

8=0
9=1-------2k Page size
11=0
12=0-----64k Segment size

NP2 bit 3=0-------- 64k Segment size
NP2 bit 4=0-------- 2k Page size

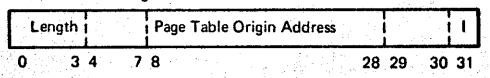
PAA



Segment Table Entries

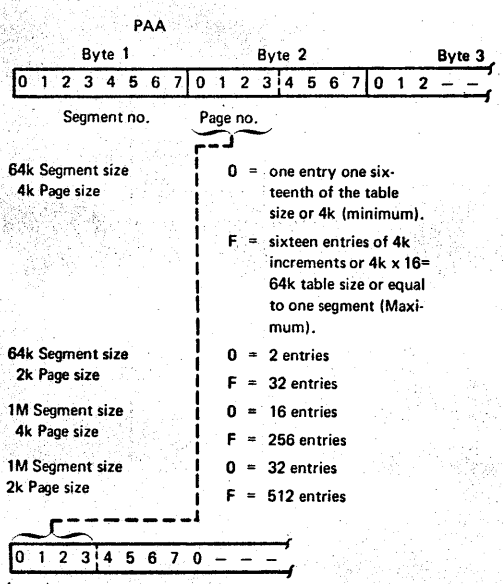
- A segment size of 1M has 16 segment table entries.
- A segment size of 64k has 256 segment table entries.
- Each entry in the segment table designates the length, availability, and the origin of the corresponding page table.

Entries in the segment table have the format:



Length: Bits 0-3 specify the length of the page table in increments that are equal to a sixteenth of the maximum size of the table. This code is compared against the 4 high-order bits of the page number to determine whether the page address designates an entry within the page table.

EXAMPLE



If the address is not within the page table, a page translation error is recognized (Prog Intr code 11).

Page Table Origin: Bits 8-28 with three zeros, gated as bits 29-31, form a 24-bit real address that designates the origin of the page table.

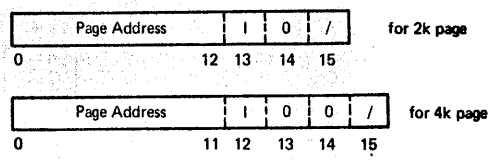
(Invalid):

Bit 31 controls the availability of the segment. When bit 31=0, address translation proceeds. When bit 31=1, a segment translation exception is recognized, and control is passed to the supervisor (Prog Intr code 10).

PAGE TABLE ENTRIES

- Each entry in the page table designates the availability of the page and contains the high-order bits of the real page address.

The page table entries are 2 bytes in length and are divided into fields:



Page Address:

Bits 0 to 11 or 0 to 12, depending on the page size, provide the high-order bits of the real storage address.

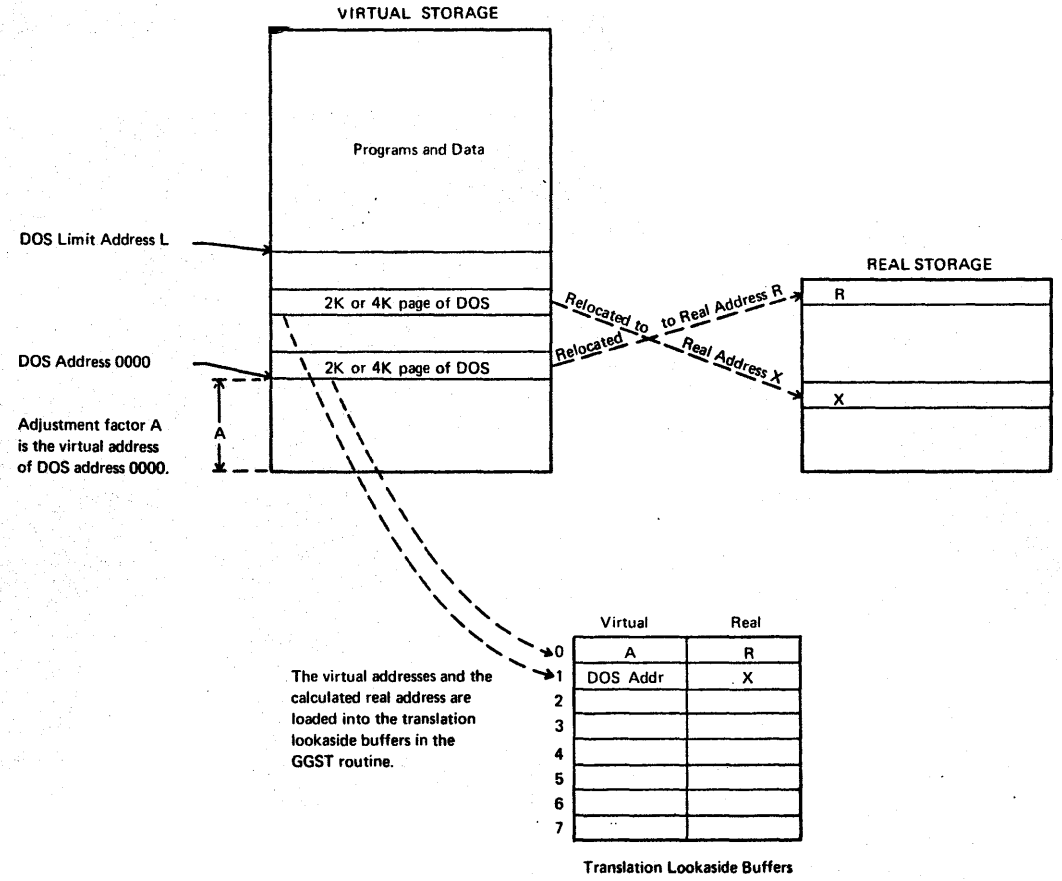
(Invalid):

Bit 12 or 13, depending on the page size, specifies the availability of the page associated with the page entry. When the bit is zero, address translation proceeds. When the bit is one, a page translation exception is recognized (Prog Intr code 11).

Bit 14 (for 2k page) and bits 13 and 14 (for 4k page) are set to zero when the page entries are formed by the supervisor program. If these bits are not 00 whenever the page entries are accessed, a translation specification exception is recognized (Prog Intr code 12).

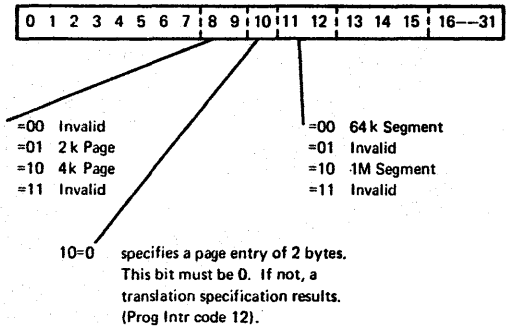
OS DOS Compatability and DAT Active

When both of these features are active and the local area of the DOS emulated environment is being accessed, the adjustment factor is added to the DOS address. The result is the virtual address that is handled by the DAT mechanism. A normal DAT translation is performed with this virtual address. Byte 0 and 1 of the TLBs is the virtual address without adjustment. Bytes 2 and 3 contain the real address entry of the page accessed by the adjusted address.

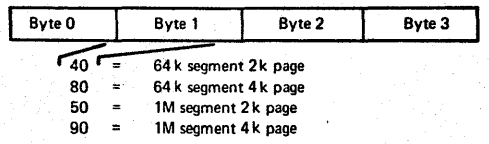


Associated Hardware

Control Register 0

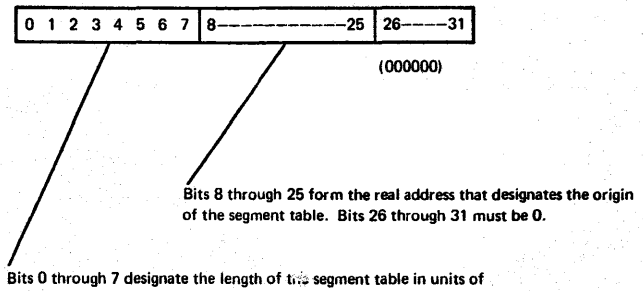


Examples:

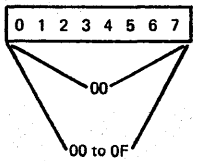


Note: These are the only valid combinations for byte 1. Remaining combinations result in a translation specification. (Prog Intr code 12).

Control Register 1



Segment Table Lengths



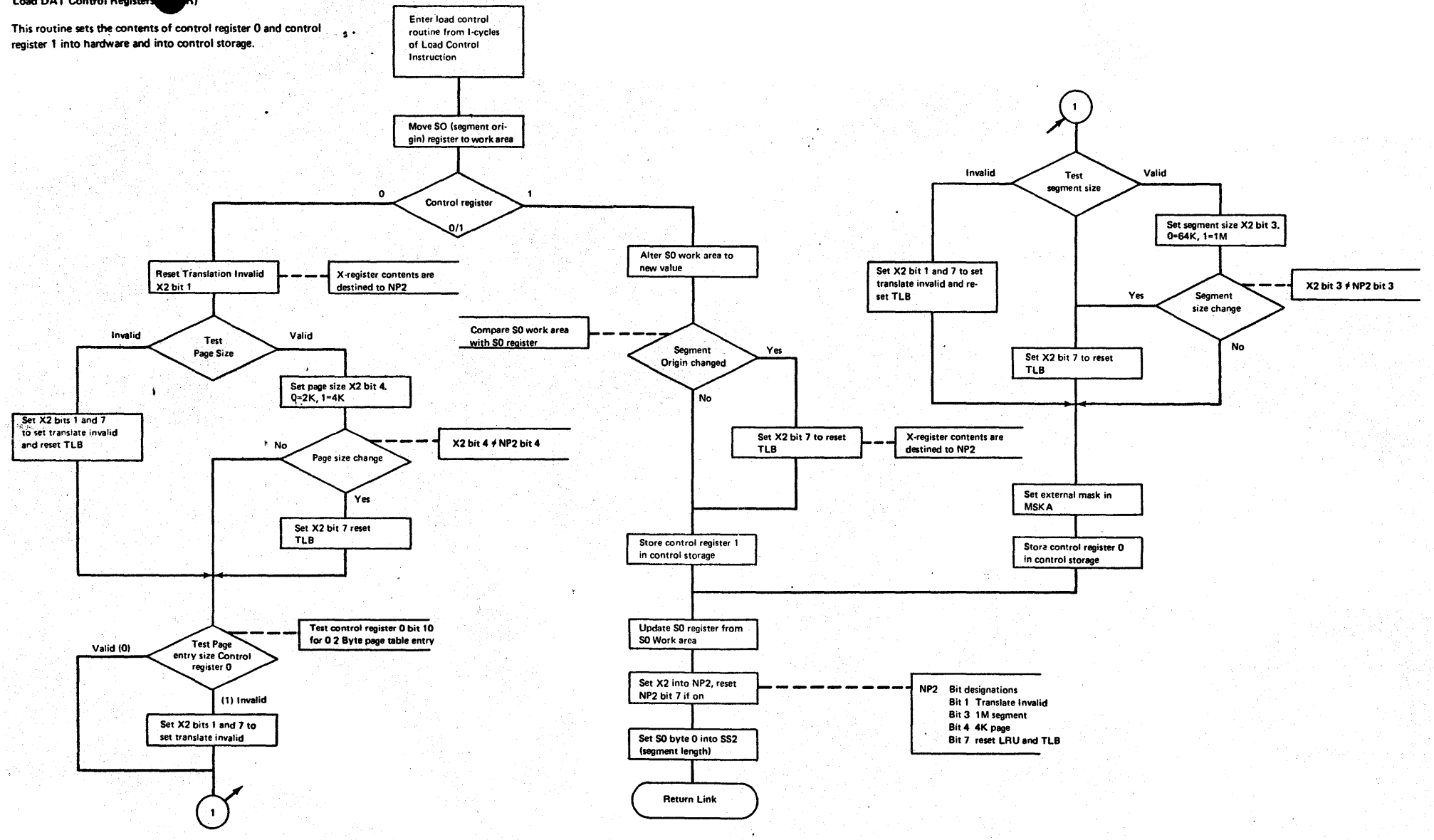
1M = 00 in the segment table length specifies 16 entries in the segment table. This is the only valid entry for 1M segments.

64k = 00 specifies 16 entries in the segment table.

64k = 0F specifies 256 entries in the segment table.

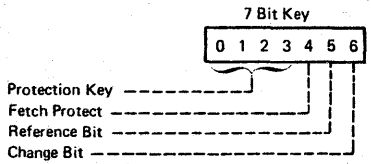
Load DAT Control Registers (LCR)

This routine sets the contents of control register 0 and control register 1 into hardware and into control storage.



Reference and Change Bit Recording

- With the DAT feature installed, 2 additional bits in the storage protect key are activated.
- Bits 0-4 of the storage protect key are used in the standard manner.
- Bit 5 becomes the reference bit and is set each time the associated 2k block is accessed from real storage for a read or store operation.
- Bit 6 becomes the change bit and is set each time a 2k block is accessed for a store operation. This indicates to the supervisor that the page has been altered and must be written back or updated in virtual storage to reflect the change.
- Reference and change recording is active in either EC or BC mode, and whether DAT is active or inactive.



Note: Reference and change recording operates on 2048-byte blocks (storage protect blocks) regardless of page size.

The paging supervisor uses the status of the reference and change bits to keep track of the areas in real storage that are avialable for use.

Whenever a page is referenced that is not located in the real storage, the paging supervisor must fetch this page from virtual storage and place it in an available page location in real storage. By interrogating the storage protect keys, the supervisor determines which page in real storage may be replaced with the page from virtual storage. If possible, a page is selected that has not been accessed. If all pages have been accessed, the paging supervisor selects a page that has been referenced only. If all pages have been modified, the paging supervisor selects a page in real storage, pages it out to the virtual storage device, and pages in the required new page to the vacated area.

Translation Lookaside Buff[redacted]LB)

TLB REGISTERS 0-7, BYTES 0 AND 1

Bytes 0 and 1 of the TLB registers contain the virtual address portion of the page addresses that have been accessed by the current program. The virtual addresses are loaded into the TLBs during the execution of the DAT trap routine GGST.

The virtual address is gated from byte 1 and byte 2 (Bits 0-3 for 4K page; bits 0-4 for 2K page) of the PAA by the destine table line and the active LRU line.

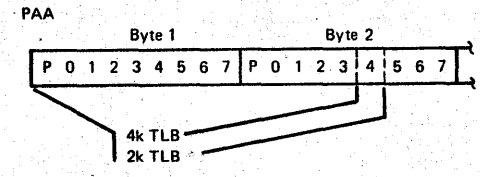
The destine table line is activated by the decode of the DK expanded local storage address. After the segment and page table addresses have been formed in the GGST routine, the control word RDH DK WK, NOP is executed. The WK register contains the page table entry address which in turn contains the REAL address that is associated with the virtual address causing the DAT trap. This real address is gated to bytes 2 and 3 of the TLB selected. At the same time, the virtual address from the PAA is gated to bytes 0 and 1 of the TLB selected.

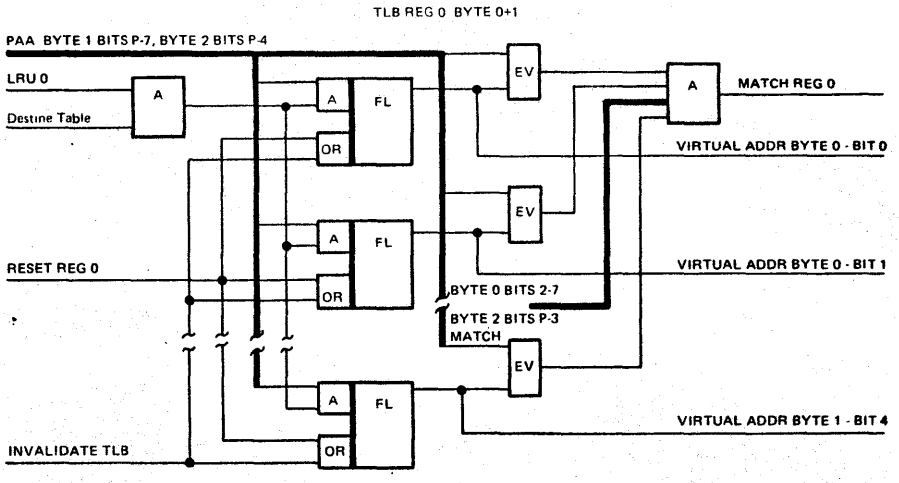
Whenever bit 7 of NP2 is set, the reset table line brings up Reset Reg 0. This is done when the Purge Table instruction is executed, and when a bad page or invalid page is detected in the execution of the GGST routine. All bits including parity are reset.

The output of the TLBs is gated to the MATCH circuits whenever a translatable address is gated to the PAA. When a match occurs between the virtual address from the TLBs and the PAA, a match line is activated. This match line gates the set and reset function for the LRU matrix.

The virtual address is also gated to the display assembler whenever the external address 2E is used in a display operation. Switch H determines the register to be displayed. The match register can be displayed through external 08.

Note: TLB registers may be valid with either odd or even parity.





MT311 - 344

TLB REGISTERS 0-7, BYTES 2 AND 3

Bytes 2 and 3 are set with the real address from the page table entry that is accessed in the translation trap routine GGST. After the segment and page table addresses are formed in the GGST routine, the control word RDH DK WK, NOP is executed. The WK register contains the page table entry address which contains the REAL address associated with the virtual address that caused the translation trap. The REAL address is gated on the EBI to bytes 2 and 3 of the selected TLB.

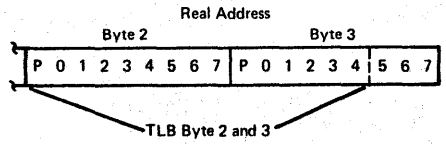
The DK function brings up the destine table line. This, when ANDed with the LRU line results in Set Reset Address Adj Reg 0 which gates the real address to the proper TLB.

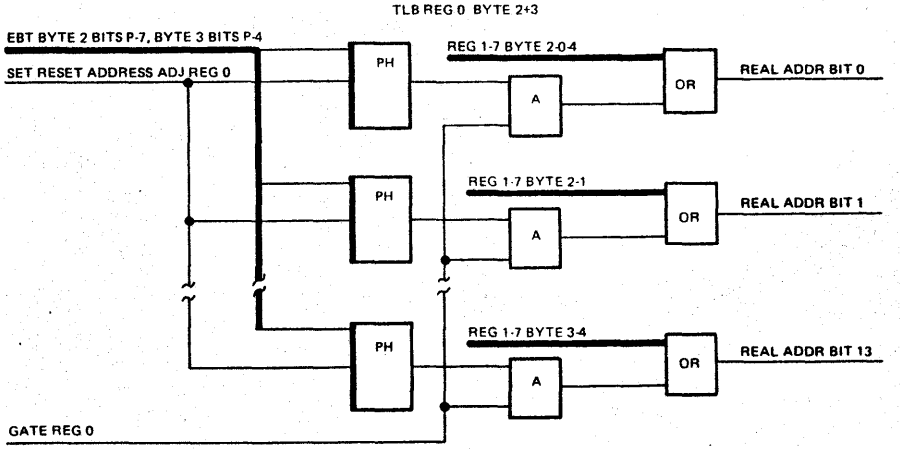
The output of the selected TLB is gated to the real address assembler to the M-register. The displacement value from the PAA is gated to the M-register and forms the low-order portion of the full real address.

Bytes 2 and 3 of the TLBs are displayed in the byte 2 and 3 position of the external display when the external address 2E is used in the display operation.

Switch H determines which register (0-7) is to be displayed.

Example





MT111 - 132

LRU Matrix

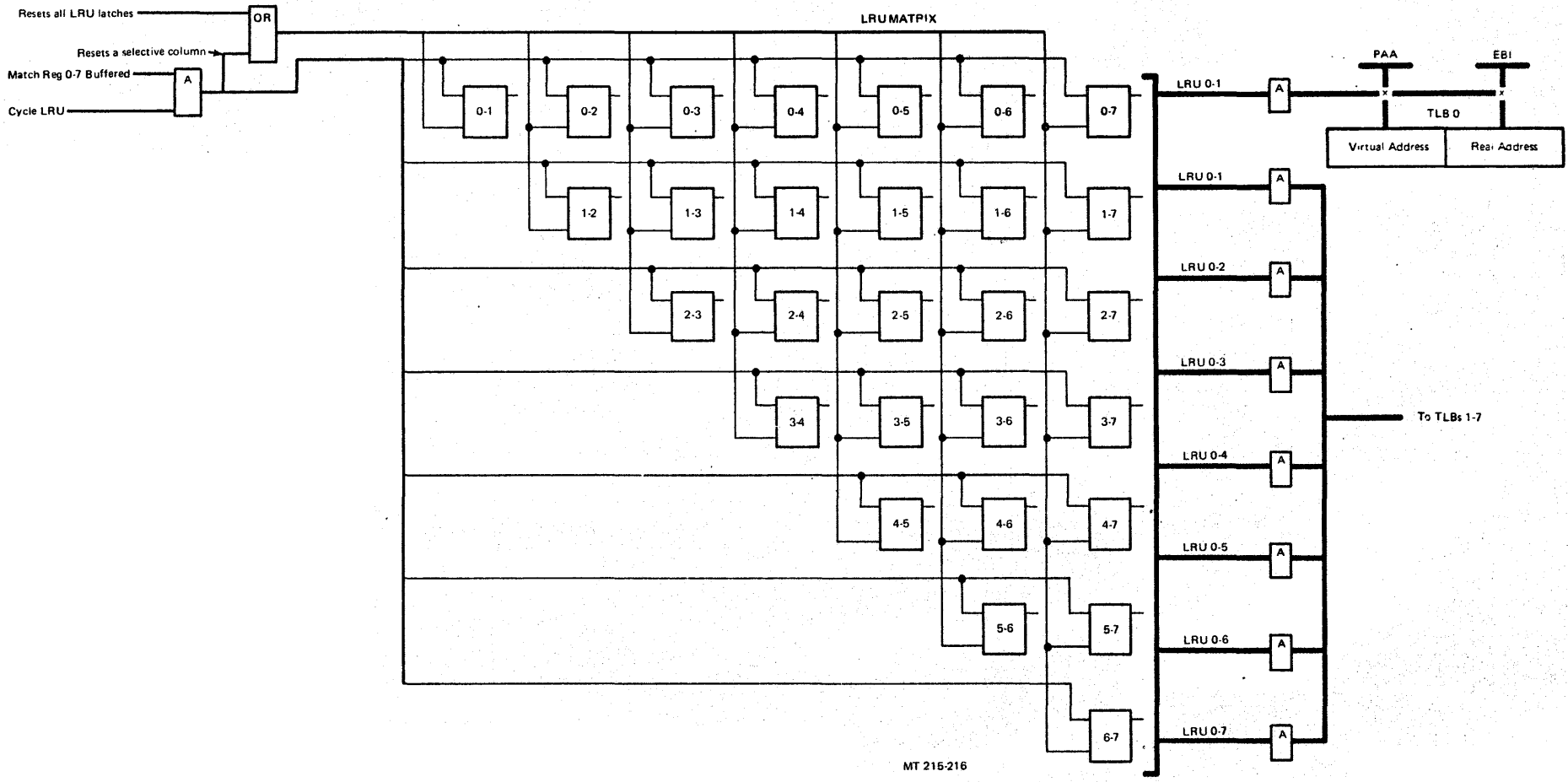
- Used to address the least recently used TLB when an address is to be loaded into the TLBs.
- Set and reset by the match circuit or by the microprogram.

Whenever a storage word with address adjust specified is executed, a match of the virtual address from the PAA and the TLBs is attempted. When a match is made, the real address from the TLB causing the match is gated to the M-register along with the displacement from the PAA.

The column in the LRU corresponding to the TLB causing the match is reset. The row in the LRU corresponding to the TLB causing the match is set. This causes the LRU output to point to the next TLB that has been least recently used. The output of the LRU matrix is only used when the TLBs are being used as a destination.

Whenever the TLBs are reset, the LRU is reset. The line (reset tables) is brought up by bit 7 of NP2. This bit is set by the microprogram when the reset is necessary. The control word, normally used to set bit 7 of NP2, is a branch word—NP2, OR, K01. This word is generally followed by NP2, A-, K01. The second branch word resets bit 7 of NP2 enabling the TLBs and LRU to be used for address translation. The LRU is displayed in external 08 byte 2.

The LRU output is used to gate the virtual address from the PAA and the real address from EBI into the selected TLB when the DK function is used as a destination in a control word. The X and Y lines for the DK function bring up the destine table line, which is ANDed with the output of the LRU matrix to access the proper TLB.



MT 215-216

Working Register (WK)

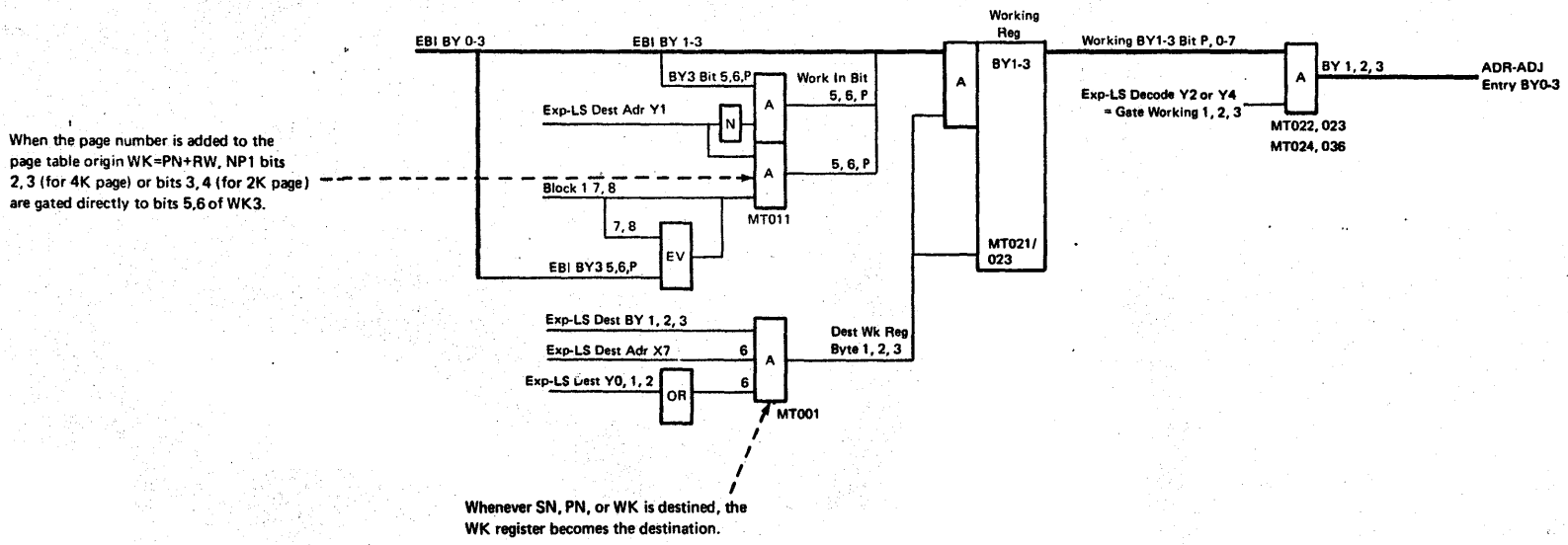
The working register is used as the destination of the computed segment table entry address or the computed page table entry address.

The segment table entry address is formed by the control word: WK=SN+SO.

The WK register is then used as the address source register in the storage control word that accesses the segment table entry: RDW RW WK, NOP. This word places the page table origin address into the RW register.

The page table entry address is formed by the control word: WK=PN+RW;

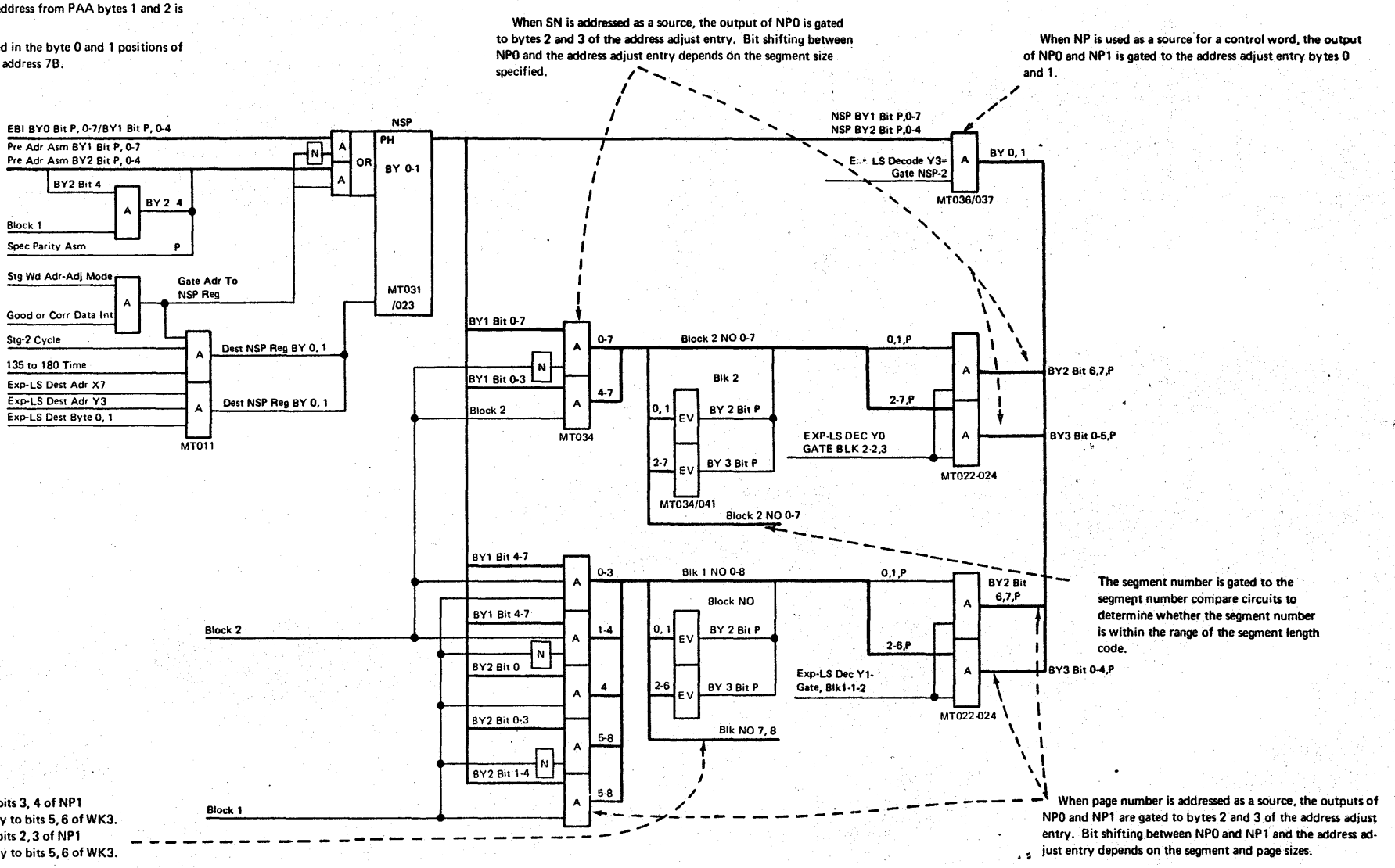
The WK register is then used as the address source register in the storage control word that accesses the page table entry: RDH DK WK, NOP. This word gates the real address from the page table entry to the TLB, and also gates the virtual address from the PAA to the TLB.



NP0 and NP1 Registers

NP0 and NP1 registers may be set from either the EBI or the PAA. When NP0 or NP1 are specified as the destination by any control word except a storage word, the data setting these registers is gated from the EBI. When a storage word with ADR ADJ active is executed, the address from PAA bytes 1 and 2 is gated to NP0 and NP1.

NP0 and NP1 are displayed in the byte 0 and 1 positions of expanded local storage using address 7B.



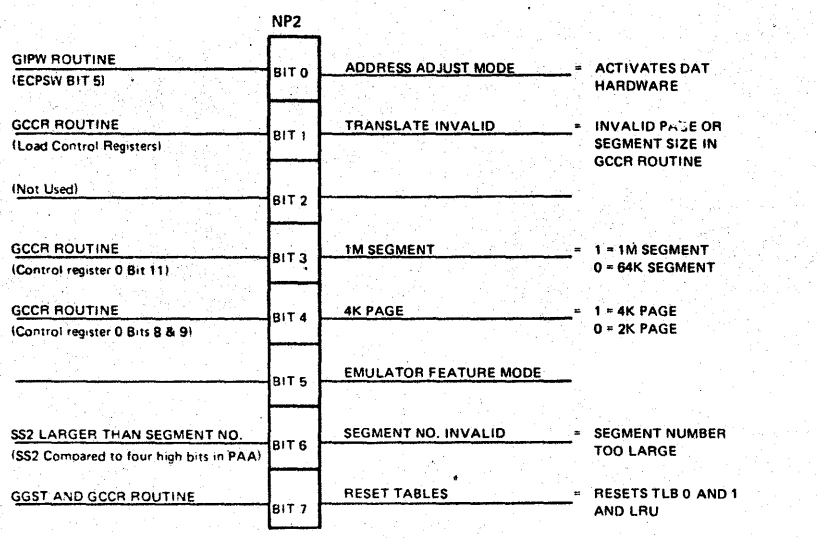
NP2 and NP3 Registers

NP2 is loaded from the EBI whenever NP2 is addressed as a data destination. The outputs of the NP2 register are used for various controls throughout the DAT hardware.

When NP2 is addressed as a source, it is gated out to byte 2 of the address adjust entry. Bit 6 (Segment Number Invalid) is sampled directly from the segment number compare circuit to bit 6 of byte 2 address adjust entry.

NP3 bits 0, 1, and 3 are set from the EBI by the gating shown in the diagram. NP3 bit 2 is set directly from the page number compare circuits. The outputs of these latches are gated to byte 3 of the address adjust entry when NP3 is addressed as a source.

NP2 and NP3 are displayed in the byte 2 and 3 positions of the external display using expanded local storage address 7B.

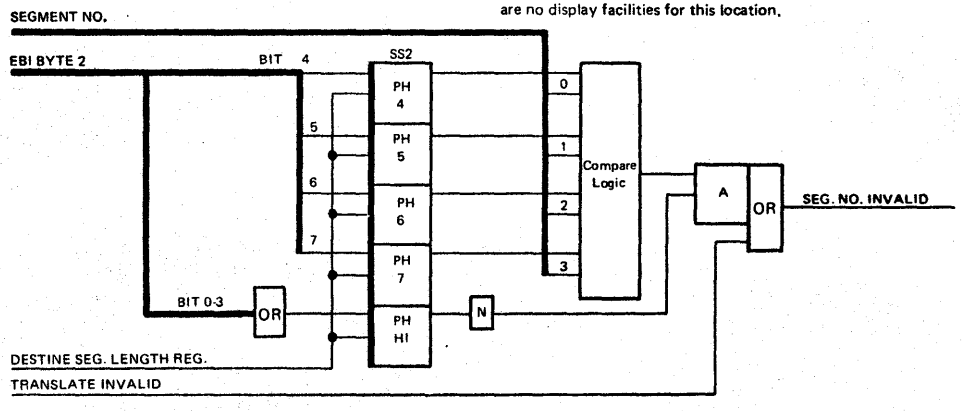


Note 1: Results in a interrupt if the bit is not on.

Segment Number Compare

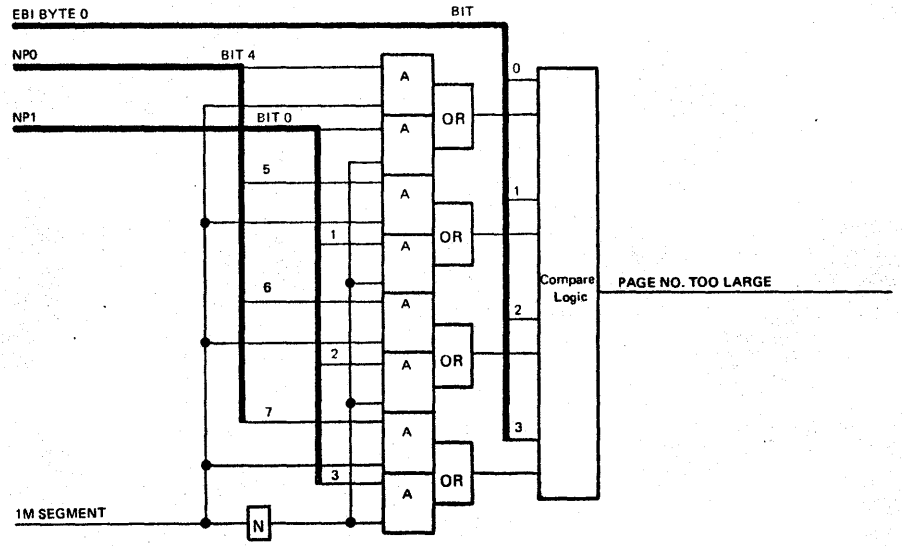
The SS2 latches 4-7 are set in the GCCR routine by the arithmetic word SS2=SOO, OE, 0. The SS2 latches now contain the four low-order bits of the segment length code. The high bit latch is set if any of the high-order bits of the segment length code are on.

A comparison is made between the SS2 latches and the segment number from NP0. For a 64K segment, bits 4-7 of NP0 are used in the compare operation. For a 1M segment, bits 0-3 of NP0 are used in the comparison. SS2 is in Exp LS 7D but there are no display facilities for this location.



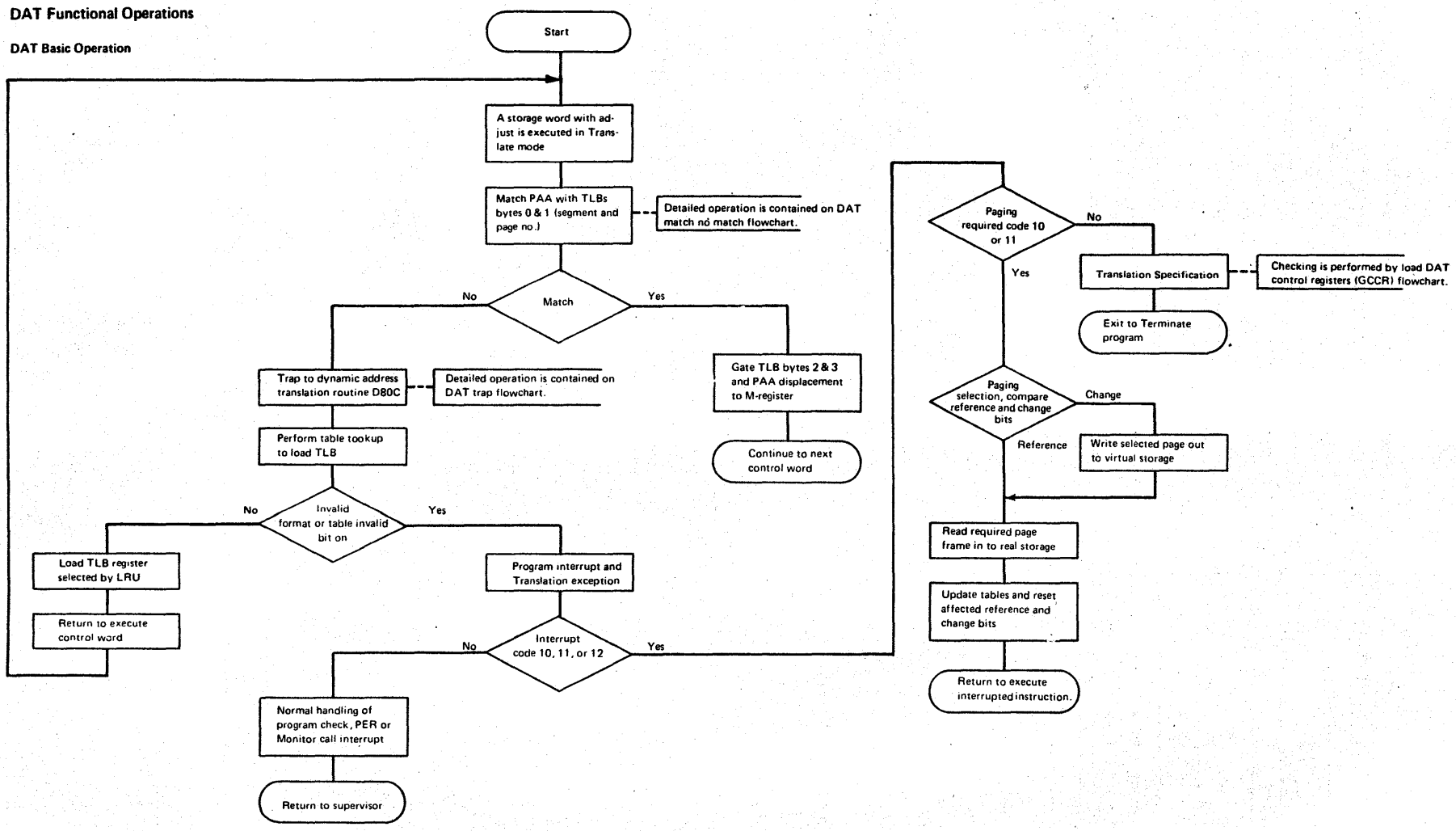
Page Number Compare

The page number from NP0 or NP1 is compared with the page table length code from the segment table entry. For a 1M segment, bits 4-7 of NP0 are compared with the page table length code. For a 64K segment, bits 0-3 of NP1 are compared with the page table length code.

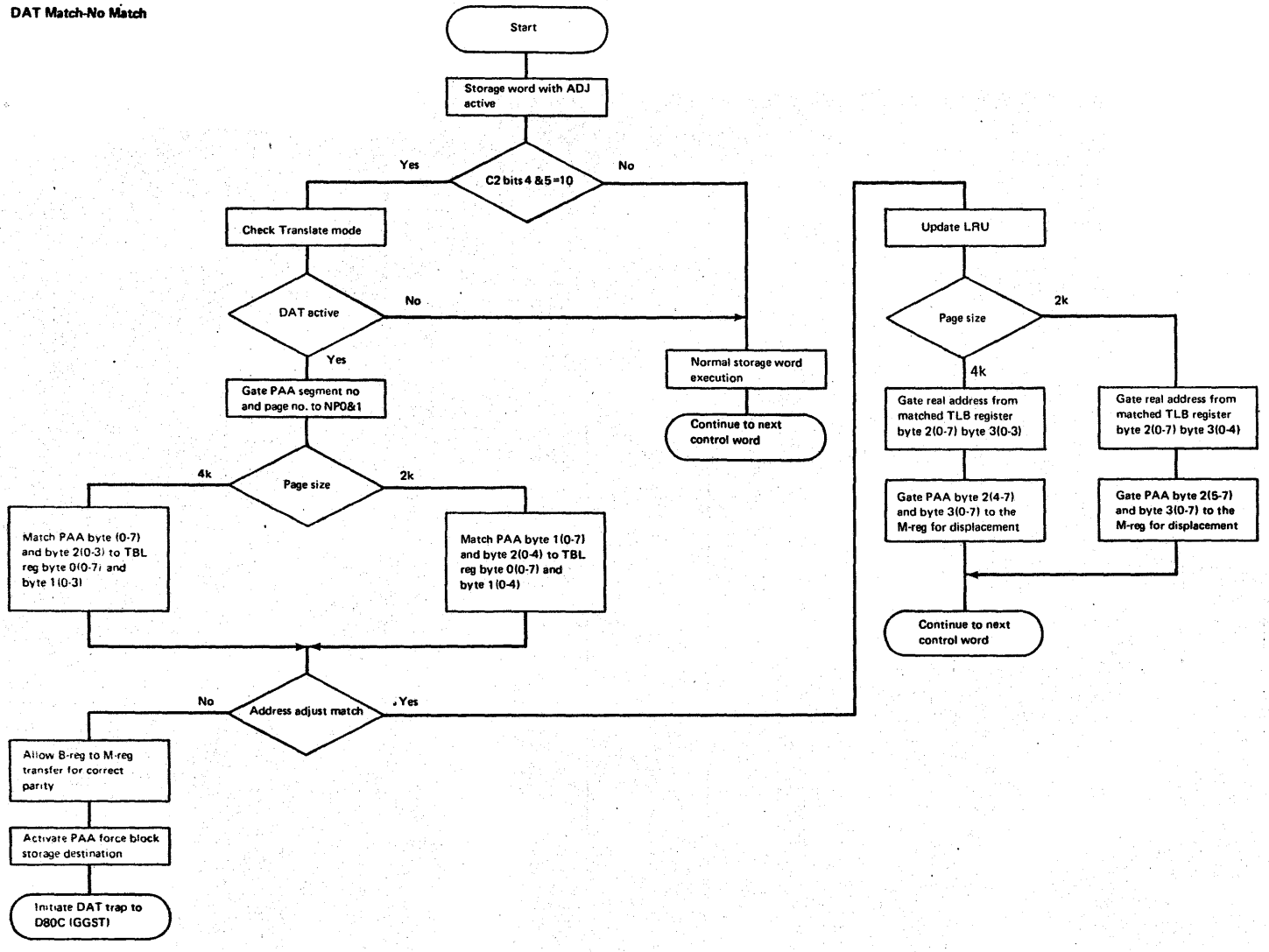


DAT Functional Operations

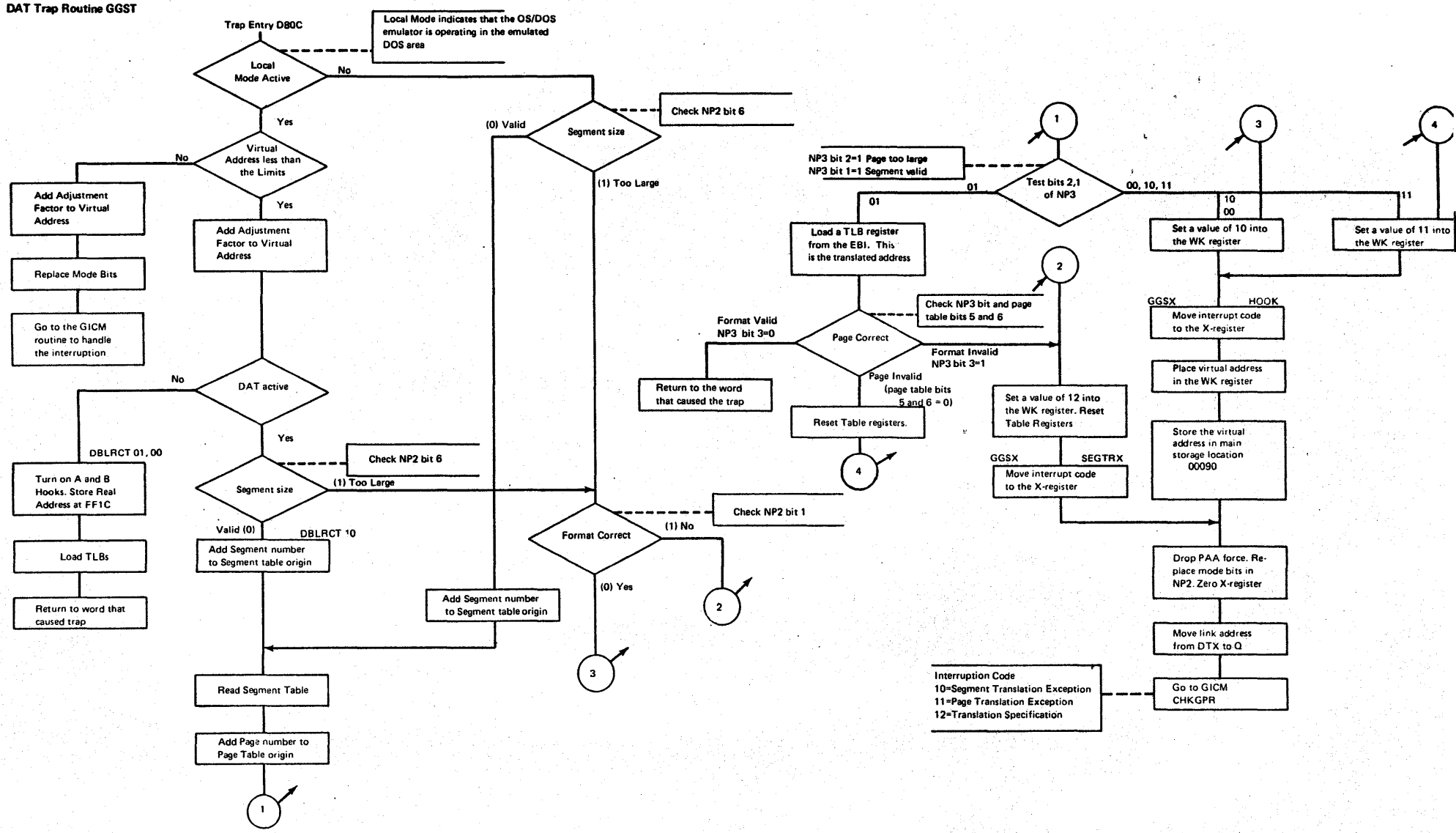
DAT Basic Operation

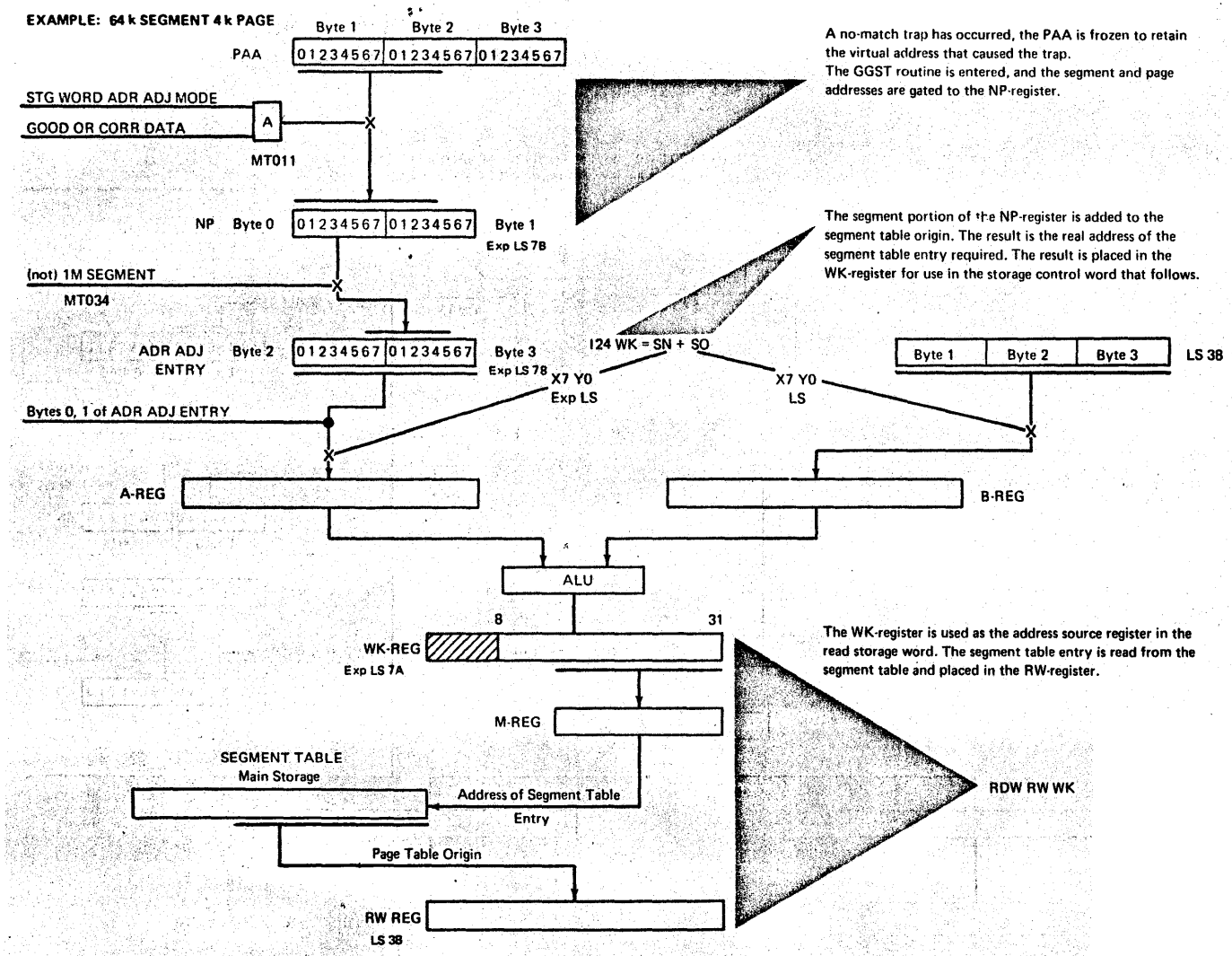


DAT Match-No Match



DAT Trap Routine GGST



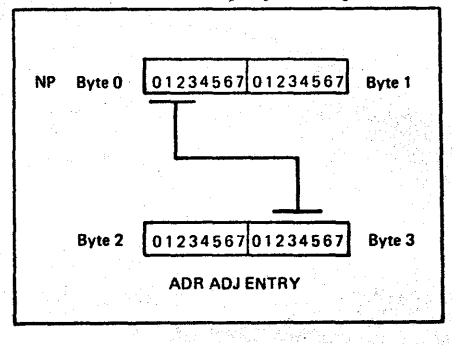


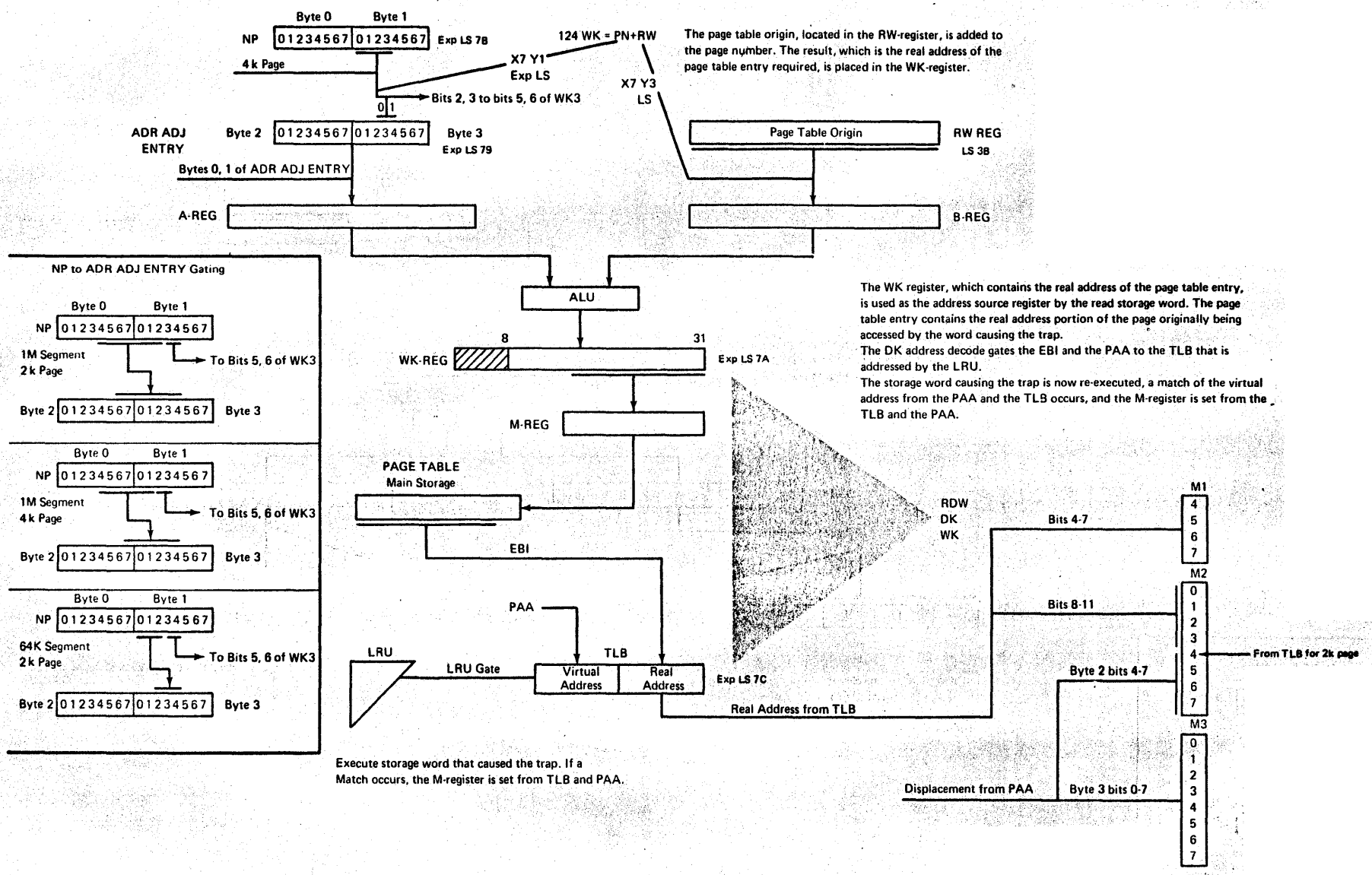
A no-match trap has occurred, the PAA is frozen to retain the virtual address that caused the trap.
The GGST routine is entered, and the segment and page addresses are gated to the NP-register.

The segment portion of the NP-register is added to the segment table origin. The result is the real address of the segment table entry required. The result is placed in the WK-register for use in the storage control word that follows.

The WK-register is used as the address source register in the read storage word. The segment table entry is read from the segment table and placed in the RW-register.

NP to ADR ENTRY gating for 1M Segment





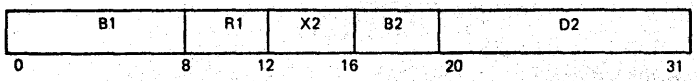
The page table origin, located in the RW-register, is added to the page number. The result, which is the real address of the page table entry required, is placed in the WK-register.

The WK register, which contains the real address of the page table entry, is used as the address source register by the read storage word. The page table entry contains the real address portion of the page originally being accessed by the word causing the trap.
The DK address decode gates the EBI and the PAA to the TLB that is addressed by the LRU.
The storage word causing the trap is now re-executed, a match of the virtual address from the PAA and the TLB occurs, and the M-register is set from the TLB and the PAA.

Execute storage word that caused the trap. If a Match occurs, the M-register is set from TLB and PAA.

Load Real Address (RX)

LRA



The real address of the second operand is inserted in the general register specified by the R1 field. The remaining high-order bits of the general register are set to zero.

The virtual address specified by the X2, B2, and D2 fields is translated by the address translation facility regardless of the setting of the translation bit in the PSW. The 24-bit real address is inserted in bit positions 8-31 of the general register specified by the R1 field, and bits 0-7 are set to zero. The translated address is not inspected for resolution, protection, or validity.

The condition code is set to 0 when translation can be completed; that is, the entry in each table is within the specified limits and the invalid bits are zero.

When the invlalid bit in the segment table entry is on, the condition code is set to 1 and the real address of the segment table entry is placed in the general register specified by the R1 field.

When the invalid bit in the page table entry is on, the condition code is set to 2 and the real address of the page table entry is placed in the general register specified by the R1 field.

When either segment table entry or the page table entry is outside the table, the condition code is set to 3 and the register designated by the R1 field contains the address of the entry that would have been referred to if the length violation had not ocurred.

Resulting Condition Code

0 ---- Translation available
1 ---- Segment table entry invalid
2 ---- Page table entry invalid
3 ---- Segment or page table length violation

Program Interruptions

Operation: The DAT feature is not installed. The operation is suppressed.

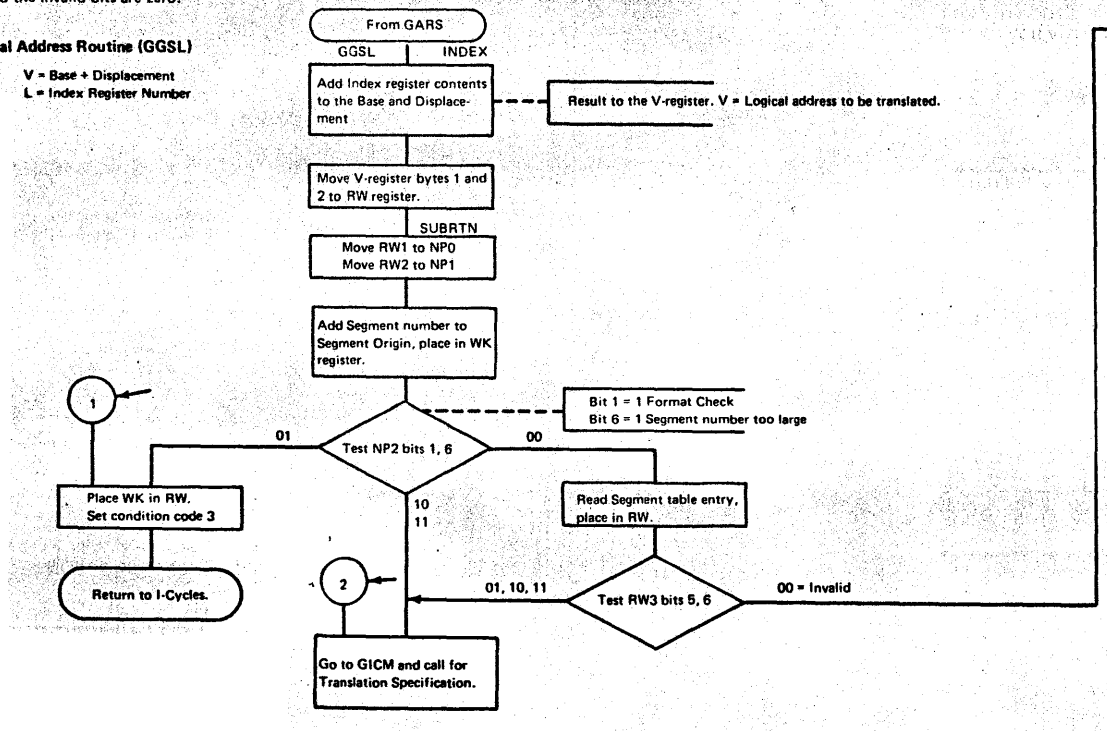
Privileged Operation: The CPU is in the problem state. The operation is suppressed.

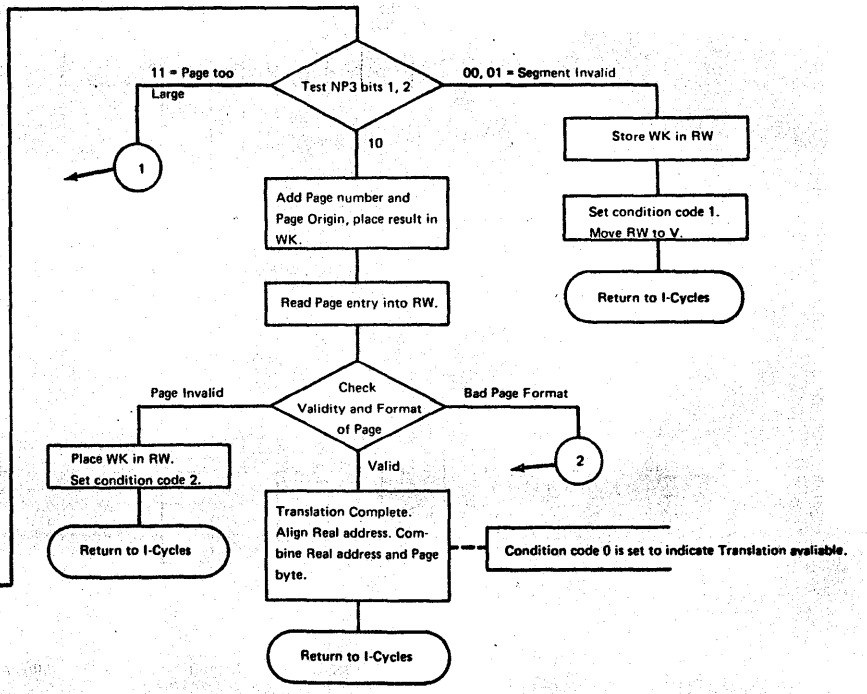
Addressing: The address of the segment table entry designates a location outside the available main storage. The operation is suppressed.

Translation Specification: Bits 8-12 of control register 0 contain an invalid code, or the page table entry has a format error. The operation is suppressed.

Load Real Address Routine (GGSL)

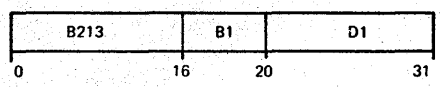
On Entry: V = Base + Displacement
L = Index Register Number





Reset Reference Bit

RRB



The reference bit in the storage key associated with the operand address is set to zero.

No other access to the key is permitted between the moment of fetching and the moment of storing the key. The remaining bits of the key are not affected by this instruction.

The operand address designates a location in real storage and is not subject to address translation. Protection does not apply to this reference.

The condition code is set to reflect the status of the reference and change bits prior to the setting of the reference bit to zero.

Resulting Condition Code

0 ---- Reference bit 0, change bit 0
1 ---- Reference bit 0, change bit 1
2 ---- Reference bit 1, change bit 0
3 ---- Reference bit 1, change bit 1

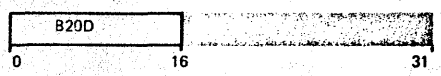
Program Interruptions

Operation: The DAT feature is not installed. The operation is suppressed.

Privileged Operation: The CPU is in the problem state. The operation is suppressed.

Purge TLB

PTLB



All information in the translation lookaside buffers is made invalid. No changes are made to the contents of addressable storage or other registers. The contents of bits 16-31 of this instruction are ignored.

Condition Code

The code remains unchanged.

Program Interruptions

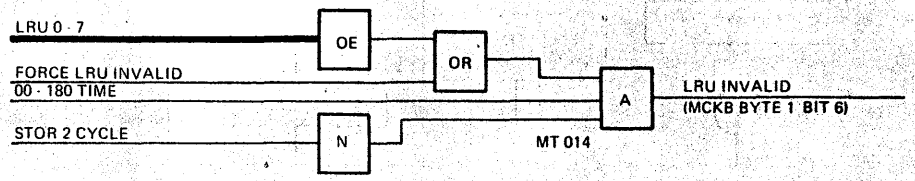
Operation: The DAT feature is not installed. The operation is suppressed.

Privileged Operation: The CPU is in problem state. The operation is suppressed.

Hardware Error Checking

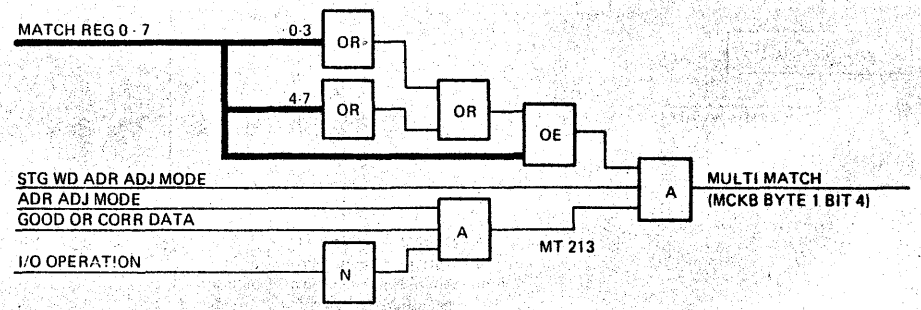
Adr X-Late LRU Inval (Address Translate LRU Invalid)

The LRU should have only one active line at any time. The ADR X-Late LRU Invalid error indicates that an even number of lines are active which differs from the normal odd number condition. This error indicates a failure of the LRU hardware.



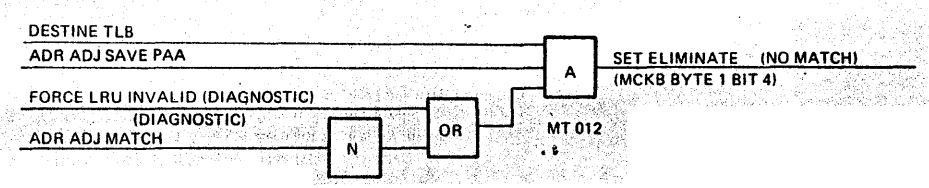
Adr X-Late Mult Match (Address Translate Multiple Match)

During DAT translation operations, the ADJ storage words cause a comparison of the PAA (virtual) address to the eight TLB registers (bytes 0 & 1). Either a no-match condition results and a trap to GGST routing is performed to load the TLB, or a successful match results in gating the selected TLB (bytes 2 & 3) to the M-register. This error indicates a failure of the LRU, TLB, or match circuit resulting in the PAA matching more than one TLB register.



Adr X-Late No Match (Address Translate No Match)

When a TLB register is loaded (with a storage word of the form "RDH DK WK"), a match is made via the normal match hardware. The absence of this match signal during the execution of this storage word indicates a hardware failure in the LRU, TLB registers, or match circuits and allows MCKB1 bit 4 to be set.

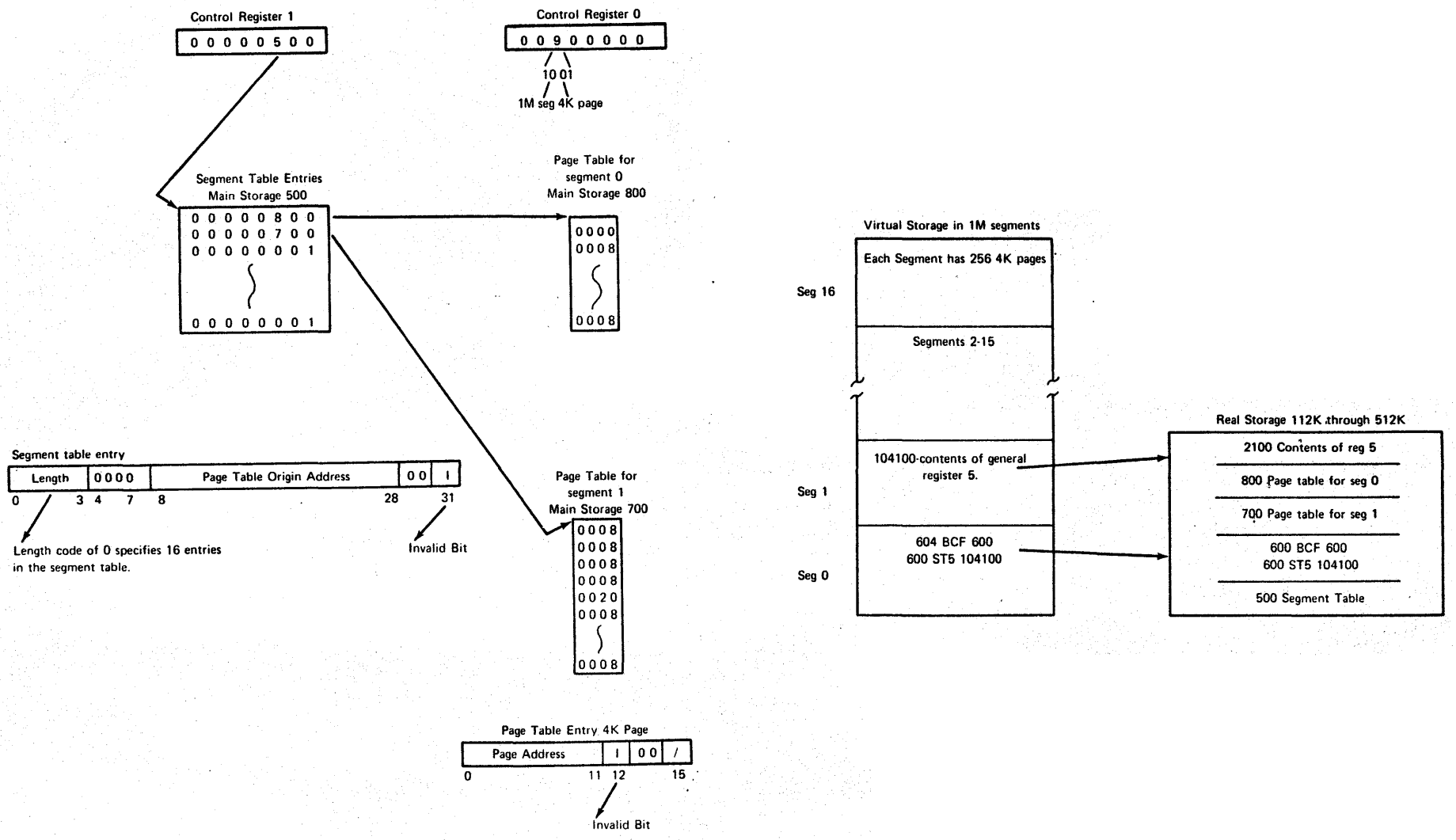


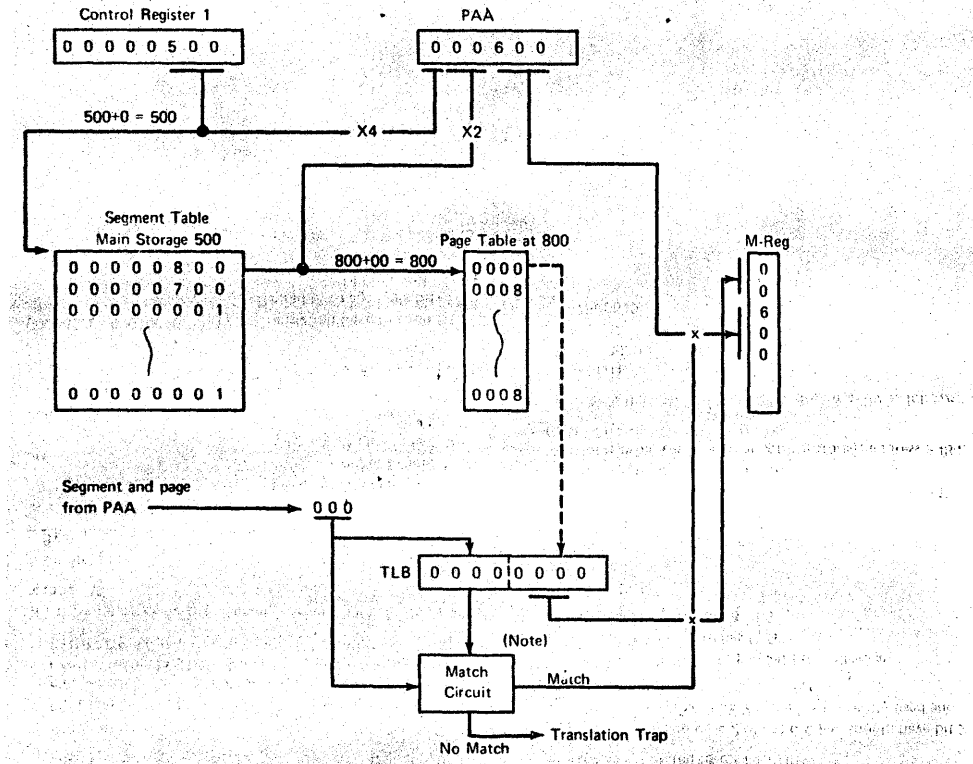
DAT Exercise This exercise may be used to verify the operation of the address translation hardware and microprograms. By altering the loop that is entered and the microprogram, scoping procedures may be formed for detailed checks of hardware operations.

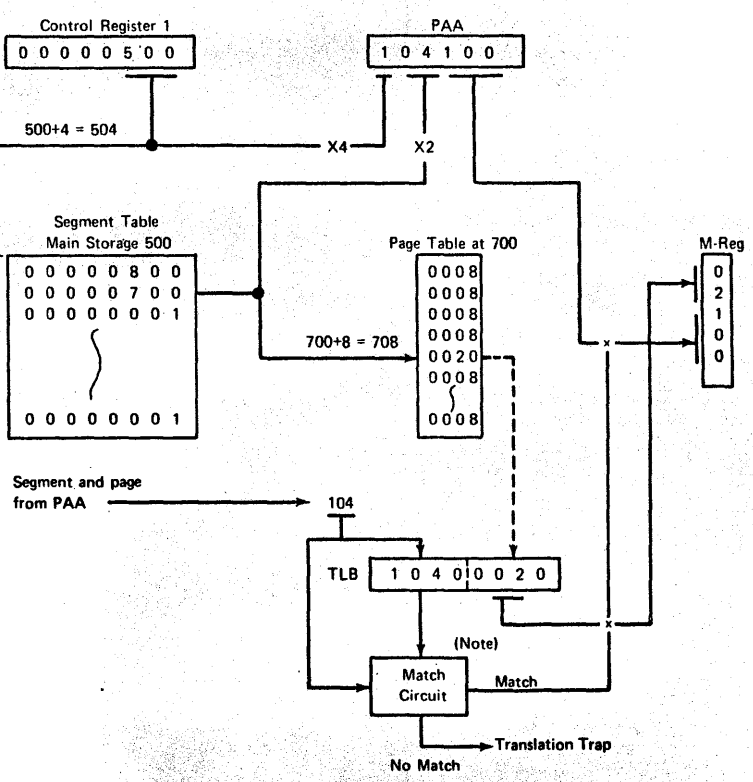
Assume that the DAT feature is installed and that the System/370 microprogram is loaded.

PR-KB ENTRY	DESCRIPTION
Step 1 AM 600 50605100	Load an RX Store instruction with an R1 field of 6, B2 field of 5, and a displacement of 100, at location 600. RX Store — OP: 5 0; R1: 6; X2: 0; B2: 5; Displacement: 1 0 0
Step 2 AG 5 00104000	Load general register 5 with the base number to be used in calculating the effective address of the second operand for the Store instruction. During I-cycles, the second operand effective address is formed by adding the contents of general register 5 with the displacement value. The result is the virtual address 104100. This address is beyond the real storage range of any Model 145 system. Therefore, this address must be translated to form a real address. The real address is loaded into a TLB by the address translation trap routine GGST. For the purpose of this exercise, the real address is a known value and is loaded into the page table later in this procedure.
Step 3 AG 6 C0C0C0C0	Load general register 6 with the value to be stored at the second operand real address.
Step 4 AM 604 47F00600	Load an RX Branch on condition instruction, with a mask field of F (branch unconditionally), and a displacement of 600, at location 604. This instruction branches to the Store instruction to form a two-instruction loop. RX Branch on Condition — OP: 4 7; M1: F; X2: 0; B2: 0; Displacement: 6 0 0
Step 5 AC 0 00900000	Load control register 0 with bits 11, 12 = 10 to specify a 1M segment, and bits 8, 9 = 10 to specify a 4K page.
Step 6 AC 1 00000500	Load control register 1 with a segment length code of 0 (byte 0 = 00), and a segment table origin address of 500.
Step 7 AM 500 00000800 00000700 00000001 00000001 00000001 00000001 00000001 00000001 00000001 00000001 00000001 00000001 00000001 00000001 00000001 00000001	Load the segment table, located at address 500, with 16 entries as specified by the segment length code of 0. The first enty contains the page table origin address 800. The page table located at 800 is associated with the instruction addresses, which are located in the same 4K page of real storage. The second entry contains the page table origin address 700 which is associated with the second operand address of the store instruction. The remaining 14 entries are loaded with the invalid bits set. The page table length codes in both the first and second entries of the segment table, are 0, indicating a page table length of 16 entries.
Step 8 AM 800 00000008 00080008 00080008 00080008 00080008 00080008 00080008 00080008	Load the first entry of the page table located at address 800 with 0000. This is the high-order portion of the real address for both instruction addresses. Load the remaining 15 entries in this page table with the invalid bits set.
Step 9 AM 700 00080008 00080008 00200008 00080008 00080008 00080008 00080008 00080008	Load the fifth page table entry of the page table located at address 700, with 0020. This is the high-order portion of the real address of the second operand of the store instruction. Load the remaining entries with the invalid bits set.
Step 10 AM 0 04080000 00000600	Load the initial program PSW with bit 5 (translate mode), and bit 12 (EC mode) set. Place the address 600 in the instruction address field.
Step 11 AM 68 040A0000 00001234	Load the Program new PSW with bits 5, 12, and 14 (wait state) on. Place the address 1234 in the instruction address field. This address is used to identify any program interrupts that may occur.
Step 12 DV 104100 = 002100 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xx------	Perform the display virtual storage function. This operation causes the real address (right of the equal sign) to be calculated by the address translation routine GGST. The GGST routine is branched from the alter display routine. 104100 is the effective virtual address of the second operand of the Store instruction.
Step 13 DV 000600 = 000600 5050511 4700600 xxxxxxxx	Displaying the virtual address of the Store instruction, reveals that the virtual and real address for the instruction are equal.
Step 14 Press the restart key. The address 600 from the PSW is gated to the I-register and the I-Cycles routine is entered. EC and translate mode are set and the system should run with the EC mode and ADR X-LATE indicators on. If the WAIT indicator is on and the address 1234 is displayed continually in the A-Register display indicators, interrogate the interruption code in the PSW located at address 8C.	

DAT EXERCISE FUNCTIONAL DIAGRAM







Note: A match condition gates the real address from the TLB, and gates the displacement value from the PAA to the M-register. A no-match condition causes a translation trap to be taken. After the trap routine has loaded the TLB with the real address, the storage word causing the trap is executed again and a match occurs.

Check Execution

The following procedure verifies execution and DAT register loading.

1. Place address 600 in the address switches, Address Compare switch in I-COUNTER LOGICAL, Address Compare Control switch in STOP, Storage Select switch to MAIN STORAGE, System should stop.
2. Place address 104100 in the address switches, Address Compare switch to ANY LOGICAL, Storage Select switch to MAIN STORAGE, Address Compare Control switch in STOP. System should stop.
3. Place address 600 in the address switches, Address Compare switch in I-COUNTER REAL, Storage Select switch to MAIN STORAGE, Address Compare Control switch in STOP. System should stop.
4. Place address 2100 in the address switches, Address Compare switch in ANY or DATA STORE, Address Compare Control switch in STOP, Storage Se lect Switch to MAIN STORAGE. System should stop.

If a program check occurs, display Program Interrupt Codes at location 8C.

Main Storage
Byte 3

00000000 00010000 — 10 — Segment Translation Exception
Caused by: Segment invalid bit on, or Segment length code error.

00000000 00010001 — 11 — Page Translation Exception
Caused by: Page invalid bit on, or Page length code error.

00000000 00010010 — 12 — Translation Specification
Caused by: Page table bits 13, 14 not equal to zero, or Segment table bits 29, 30 not equal to zero.
Control reg 0 Bit 10 not=0. Invalid combination of page and segment bits in control reg 0 bits 8, 9 and 11, 12.

Display DAT Registers

1. Display TLB 0 (External)
   Word address 2E in switches F, G
   Switch H to position 0
   Storage Select switch to EXT REGS

   The virtual address (bytes 0, 1) and the real address (bytes 2,3) are zero with the parity bits on. Parity bits on indicate that this TLB was loaded by the microprogram.

2. Display TLB 1 (External)
   Word address 2E in switches F, G
   Switch H to position 1
   Storage Select switch to EXT REGS

   Byte 0 = 10 Virtual address
   Byte 1 = 40
   Byte 2 = 00 Real address
   Byte 3 = 20

   Place switch H in any position 2-7, Bytes 0 and 1 have not been accessed and are blank with no parity bits. TLBs 2-7 could contain residual data.

3. Display the LRU (External)
   Word address 08 in switches F, G
   Storage Select switch to EXT REGS

   The LRU, located in byte 2 of the display, should have bit 2 on. This indicates that TLB 2 is the least recently used and is to be loaded next.

4. Display the NP register (Expanded Local Storage)
   Word address 7B in switches F, G
   Storage Select switch to EXP LS

   Byte 0 = 10
   Byte 1 = 40
   Byte 2 = 98
   Byte 3 = 40

5. Control register 0 is located at control storage address F480.
   Control register 0 = 00 90 00 00
   Control register 1 is located at control storage address F484.
   Control register 1 = 00 00 05 00

6. To modify the exercise for a scope loop, modify as follows:

   AM 604
   B20D0000
   AM 60A
   47F00600

CHANNEL INDIRECT DATA ADDRESS (CIDA)

Introduction

The CIDA feature can extend the address adjustment of the dynamic address translation (DAT) feature to the I/O channels. A contiguous set of virtual addresses can be mapped into a non-contiguous set of pages in real storage. Because only a single data address need be in effect at a time for a channel, using the DAT hardware to handle the adjustment with each access is not necessary. The adjustment factor is applied by the program, and the real address is stored for the CIDA controls. These adjusted addresses are stored as an IDA list (IDAL) for each CCW. Each word of the list is referred to as an IDA list word (IDALW).

The operation of the CIDA feature is selective and can be used in either BC or EC mode. An CIDA flag in the flag byte of the channel CCW functions is the indirect addressing switch. When the CIDA flag = 1, the normal data address of the CCW is the address of an IDAL in main storage. This list contains one or more addresses of main storage on page boundaries that can be used in sequence for the CCW operation. The first address is not required to be on a page (2K) boundary to permit filling a partial page. All remaining addresses in the list must be for the starting address of a page in main storage. The operation need not fill the last page used.

In operation the hardware must recognize the end of a page and enter the next address (IDALW) from the IDAL. For all operations except read backward, the end of the page is recognized by the change of the low-order eleven address bits from ones to zeros. For the read-backward operation, the change is from zeros to ones. The low-order eleven bits of an address must be zeros (2K boundry) for a forward operation.

The change is detected by testing for an inversion of byte 2 bit 4 of the data address. Any address in the IDAL other than the first that is not on page boundary, or that contains information in byte 0, causes a program check indication and the operation is terminated.

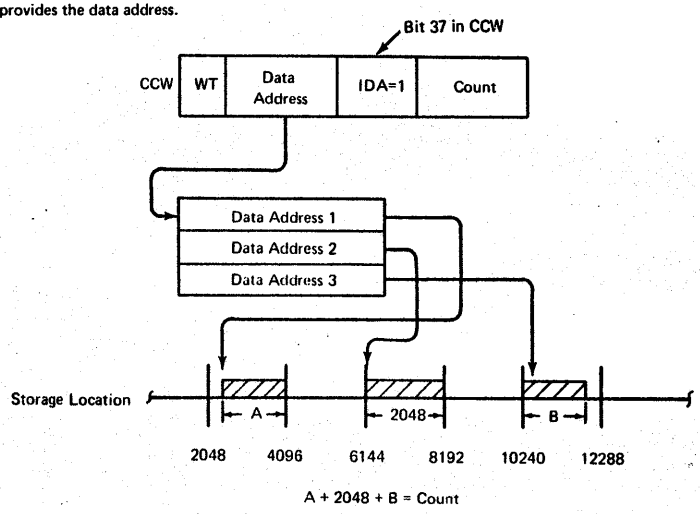
Byte Multiplexer Channel

For the byte-multiplexer channel, the CIDA feature requires a fourth word in each assigned UCW. This word holds the IDAL address for the assigned device. During the entry of the CCW, the CIDA flag bit is tested to determine the use of the data address field. A CIDA flag causes the address to be stored in the fourth word of the UCW, and then uses that address to enter the first IDALW to store as the data address in the UCW.

With each use of the data address for data transfer, a test is made for the updated address crossing a 2K boundary. This test is made by comparing byte 2 bit 4 of the address before and after the update. Any change represents the cross of a 2K boundary. When a change is detected, the IDAL address is entered from the UCW to fetch the next IDALW for the new data address. In CIDA command chaining, each CCW is tested for the IDA flag: either the normal data address or the CIDA addressing routine.

COMMAND CODE = WRITE
IDA = 1

The data address in the CCW provides the address of which in turn provides the data address.



A + 2048 + B = Count

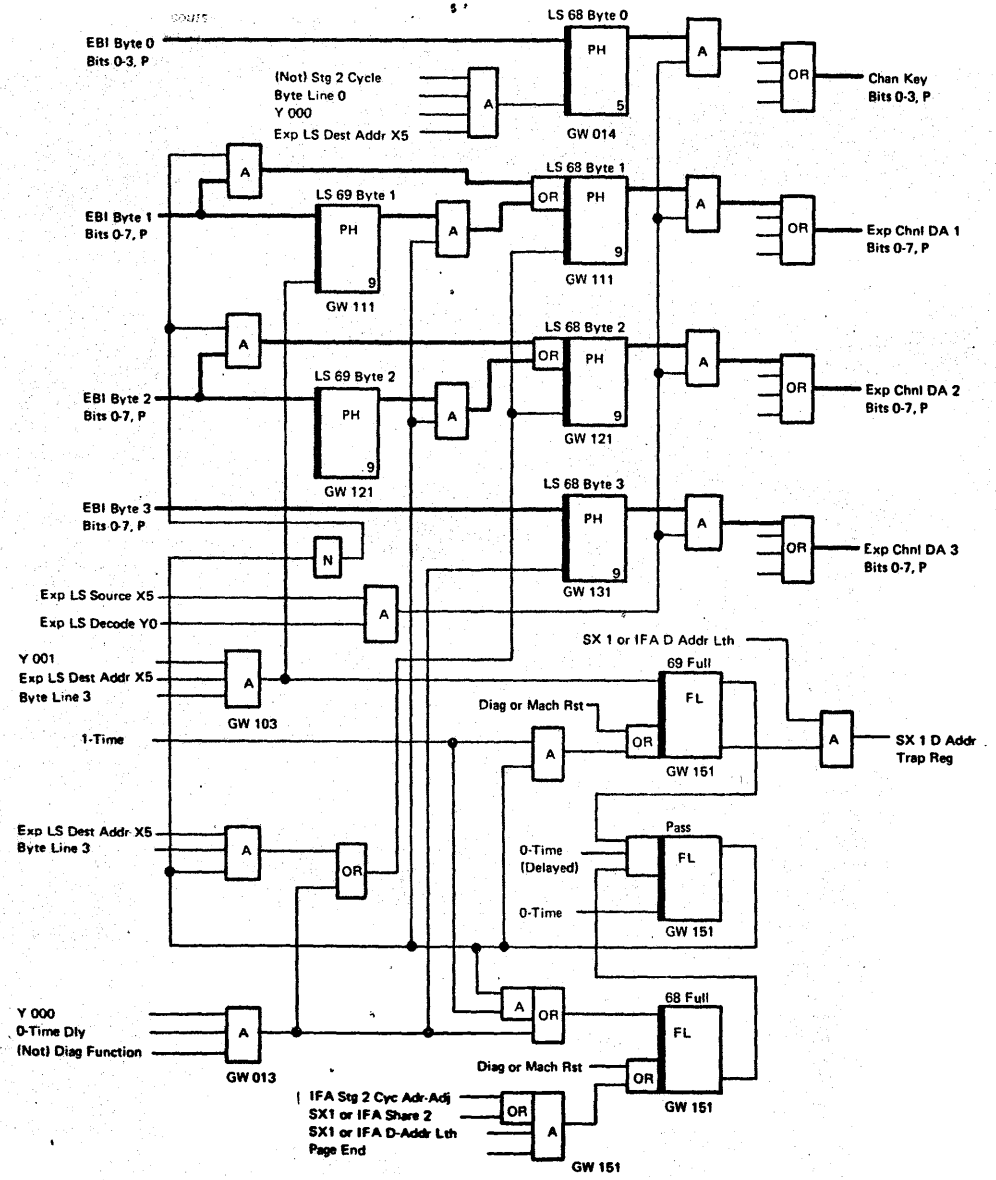
The data addressing for console printers is the same as for the multiplexer channel. The CIDA feature applies only to operations initiated by the Start I/O instruction. Alter and display operation do not translate addresses.

Selector Block-Multiplexer Channels and IFA

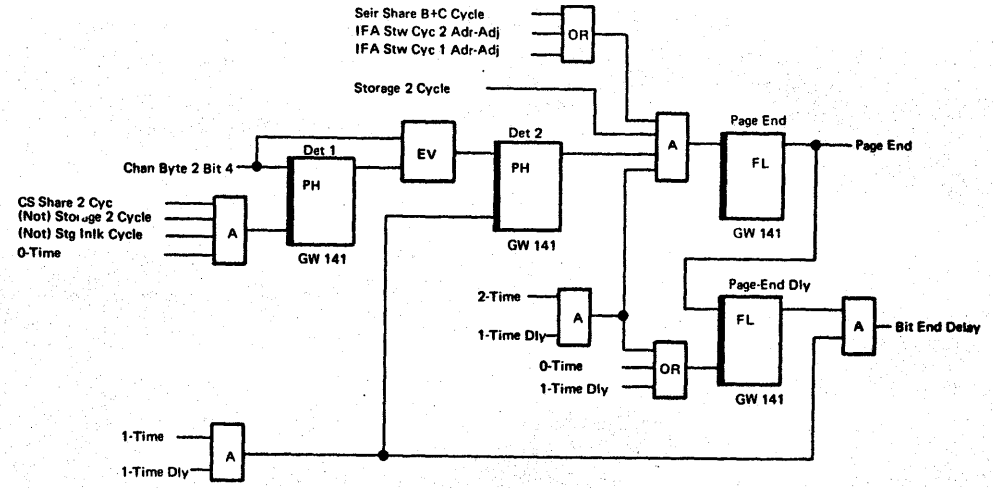
With the CIDA feature, the selector and block-multiplexer channels operate in the same manner. A block-multiplexer channel can only disconnect at the completion of a command and, therefore, does not require storing the CIDA information in the UCW. Upon entry of a CCW either on initial selection or subsequent command chaining, the CIDA flag is tested. When the flag is present, the address in the CCW is not entered into the GDRL register. Instead, the address is that of the IDAL to be used for entering an IDALW into the GDRL register for the data address. The next IDALW is immediately read into the GDBRL register to back up the GDRL register. The updated IDAL address is retained in the GD register.

When the data address in the GDRL register has been either incremented or decremented until it crosses a 2K boundary, the hardware transfers the contents of the GDBRL register into the GDRL register and requests a CIDA data trap (D10C). When the trap is honored, the IDAL address is used to enter the next IDALW into the GDBRL register. The addresses entering the GDBRL register contain only bytes 1 and 2 because they contain no key and because the low-order byte must be either 00 or FF. The data transfer stops in the normal manner at the end of the record or in the case of an error.

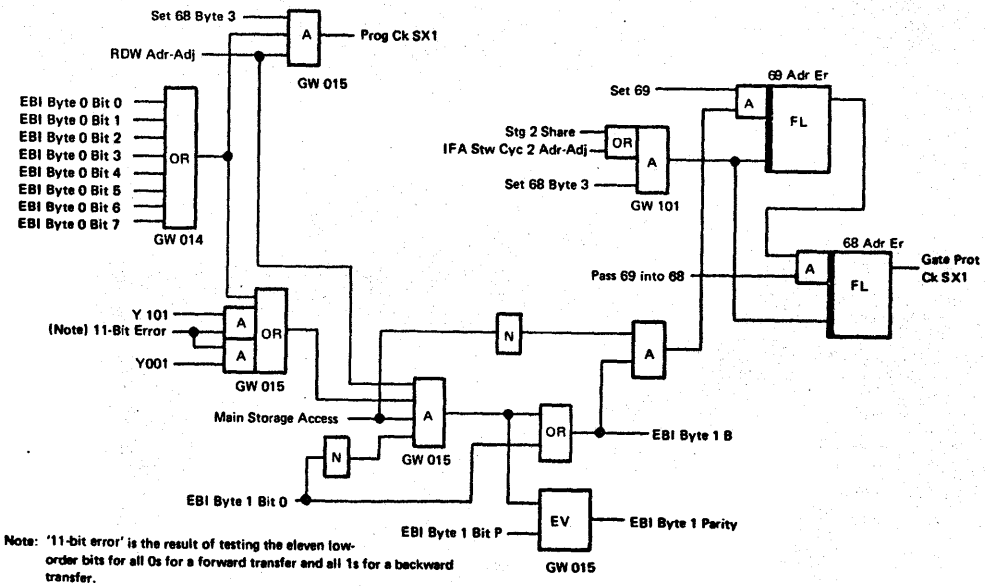
The IFA functions in the same manner as the selector channel for CIDA operations that use share cycles to transfer data to or from main storage. For control, sense, and operations that use microprogram to transfer data, the hardware forces the addressing to use the FDRL register when the FD register is addressed. The CIDA data trap request uses the IFA gated-attention trap (D120) and branches to the IDALW entry routine when no gated-attention condition is found.



Page-End Detection



Error Detection



Note: '11-bit error' is the result of testing the eleven low-order bits for all 0s for a forward transfer and all 1s for a backward transfer.

Basic Selector Channel Addressing

The selector channel data addresses supplied by the CCW define the starting address in main storage to be used for the operation. The data address is initially set into the GD register in local storage. This same data address is also set into the expanded local storage GDRL register for use in addressing by the share cycle. The share cycle hardware actually addresses the GD register but raises the B-register addressing gate to enter the GDRL register. During the second cycle count update, the address gate is dropped so that the GD address 1 enters the GC register. The address in the GD register is not used for the data movement.

Basic IFA Addressing

The IFA addressing is similar to the selector channel addressing but involves some exceptions because of the hardware integration of the control unit. The share cycle transfers between the file and main storage are identical but use the FD and FDRL registers. Share cycle transfers between the file and control storage involve the FA register for data address and the B-register address gate is not raised. The transfers of control, sense, and data information between control or local storage and main storage do not use the share cycle controls. The main storage address is set into the FDRL register and the B-register addressing gate is forced by the hardware condition.

The data address from the CCW is entered into both the FD local storage register and the GDRL expanded local storage register. The address for any control storage area involved is entered into the local storage FA register by the microprogram. The share cycle controls address either the FD or the FA register for the data address depending on whether main storage or control storage is affected. When the FD register is used, the B-register addressing gate is raised to enter the FDRL expanded local storage register.

The IFA data transfers that do not use the share cycle controls force a three-step clocking sequence to gate the two-cycle storage word. During the second cycle count update, the gate is dropped and the count is entered from the data register and incremented by 1 for either the FC or the FB register. When transferring information between main and control storage, both of these addressing systems are used but without the share cycle controls. The microprogram controls the addressing with the gate being forced through hardware.

CIDA Backup Control

When the CIDA flag is set, byte 2 bit 4 of the address sets a polarity-hold latch. After the address has been updated as the result of the transfer, the same bit is compared with the polarity-hold latch to determine whether the level has been changed. A change represents the cross of a 2K boundary. The address backup register (GDBRL) bytes 1 and 2 are transferred into the GDRL register to continue the operation. Byte 3 of the GDRL register now stands at either all zeros or all ones that represent the 2K boundary for forward or backward transfers. Byte 0 of the GDRL register contains the protect key and does not change for the new address.

When the backup register is transferred, a request is made for a CIDA data trap to enter the next IDALW. When this trap is honored, the IDAL address in the GD register is used to enter the next IDALW into the GDBRL register. All IDAL words except the first are tested for 2K boundary. A part of the zero-count detection circuits is used to test the 11 low-order bits. These are all zeros for a forward transfer and all ones for a backward transfer. The read backward line reverses the bit levels through OE logic circuits to test for ones in the zero test circuit.

If this hardware detects a program violation in the CIDA list, the hardware forces on the highest-order address bit when the bad list entry is moved into the backup data address register (GDBRL). If data transfer continues until this entry of the list is needed, an address check occurs which results in a channel program check through the normal address check mechanism. If the data transfer is concluded before the bad entry is needed, no check is indicated.

If other activity on the system prevents honoring the CIDA trap before the data transfer exhausts the contents of both GDRL and GDBRL, a line is sent to the channel hardware. This prevents further share cycles from that channel until the CIDA trap has been honored. Note that this is a highly unlikely event because it requires that a one megabyte device has its trap request locked out for two milliseconds.

PROGRAM EVENT RECORDING (PER)

The program event recording feature provides a means for debugging new programs or revisions. PER can alert the programmer when these events occur.

- Successful execution of a branch instruction.
- Alterations of the contents of designated main-storage locations.
- Alteration of the contents of a specified general register.
- Fetching of an instruction from designated main-storage locations.

Introduction

The program has control over the conditions that are considered events in the program sequence. These events can be selectively monitored to aid in program analysis. When an event occurs, a program interruption is initiated if the masking conditions allow. An interruption for an event does not remain pending: if the interruption is masked, the information is lost. Information concerning the events is reported through the interruption codes. The PER feature operates only in EC mode.

Program Event Operations

PER operations are initiated by setting one or more PER control bits to a 1 in control register 9 and setting the PER mask (ECPSW bit 1). The program to be scanned is performed in the normal manner.

Addresses defined in control registers 10 and 11 apply to both the 'instruction fetch' and the 'storage alteration' events. When the starting address is smaller than the ending address, the event area is from the starting address through the ending address. If the ending address is smaller than the starting address, the event area is from the starting address to the end of storage and from address zero through the ending address. A single storage address is defined when the starting and ending addresses are the same.

With the PER feature allowed (ECPSW bit 1 = 1) and when an event is recognized, the interrupt is taken with the appropriate indications stored in the PER interrupt information in main storage words 94 and 98. If the PER mask bit is not set, the event conditions are lost.

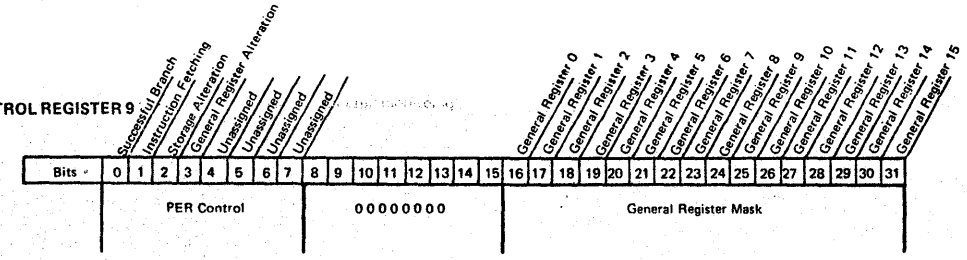
Control Register Allocation

Control registers 9, 10, and 11 are used to define the events to be monitored and the limitations imposed for the PER operation.

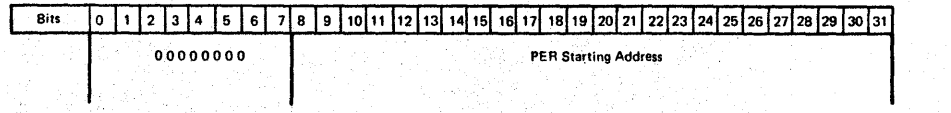
PER Control Bits

Bits 0-7 of control register 9 specify the events to be monitored. When a bit = 1 the event is monitored.

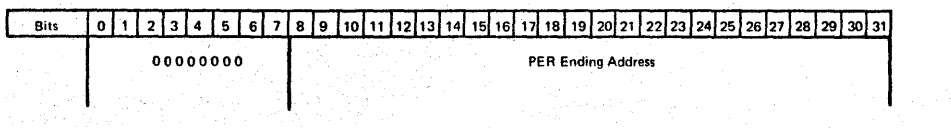
CONTROL REGISTER 9



CONTROL REGISTER 10



CONTROL REGISTER 11



Bit 0 Successful Branch Instruction
Bit 1 Instruction Fetching
Bit 2 Storage Alteration
Bit 3 General Register Alteration
Bit 4 Unassigned
Bit 5 Unassigned
Bit 6 Unassigned
Bit 7 Unassigned

PER GR Alteration Masks

Bits 16-31 of control register 9 specify which general registers are to be monitored for alteration of their contents. The 16 bits are assigned in the order of their ascending bit numbers to the 16 registers. When a bit = 1 the register is monitored.

PER Starting Address

Bits 8-31 of control register 10 define the first address of the main storage area to be monitored.

PER Ending Address

Bits 8-31 of control register 11 defines the last address of the main storage area to be monitored.

Extended Interrupt Code

A detected event condition is reported in the extended interruption code during the effected program interrupt for EC mode. The program interruption code of 80 defines PER as the cause. The PER conditions are reported in main storage words 94 and 98.

PER Code

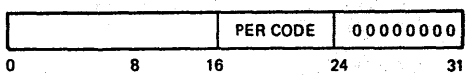
Bits 16-23 of main storage word 94 specify event or events causing the interruption. These bits are arranged in the same relation as the control bits in control register 9. When a bit = 1, the event was detected.

Bit 0 Successful Branch Instruction
Bit 1 Instruction Fetching
Bit 2 Storage Alteration
Bit 3 General Register Alteration
Bit 4 Unassigned
Bit 5 Unassigned
Bit 6 Unassigned
Bit 7 Unassigned

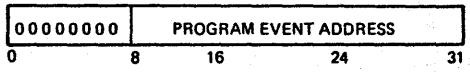
Program Event Address

Bits 8-31 of main storage word 98 specify the address associated with the recognized event. This is normally the address of the instruction causing the event except an execute initiated instruction when the execute instruction address is stored.

Word 94



Word 98



Successful Branch Instruction

When the PER control bit 0 = 1, the operation sequence tests for a successful branch event following the execution of each branch instruction. A successful branch results from branching the instruction flow to the branch address of the instruction. The branch at the end of the load-PSW instruction is not considered a successful branch because that is the normal instruction flow.

When bit 0 of the PER code is set to 1, the program event address is set to the address of the branch instruction unless it was initiated by an 'execute' instruction that stores the instruction address. The address to which the instruction branched is stored as the instruction address in the old PSW.

Instruction Fetching

When the PER control bit = 1, the operation sequence tests that the initial byte of the instruction falls within the monitored area of main storage. When an instruction is performed as the result of the 'execute' instruction, the initial bytes of both instructions are tested. The event is recognized if either byte falls within the monitored area.

When bit 1 of the PER code is set to 1, the program event address is set to that of the fetched instruction. If the fetched instruction is the object of an 'execute' instruction, that instruction address is stored. The next instruction address is stored as the instruction address of the old PSW.

Storage Alteration

When the PER control bit 2 = 1, the operation sequence tests that the information is destined to an address within the monitored area of main storage. This does not include addresses initiated by the CPU for permanent storage and logout assignments. The storage is considered altered when the instruction could change the information, even if the value is modified by zero.

When bit 2 of the PER code is set to 1, the program event address is set to that of the instruction causing the reference. If the 'execute' instruction initiated the reference instruction, the address of the execute instruction is stored. The next instruction address of the old PSW.

General Register Alteration

When the PER control bit 3 = 1, the operation sequence tests that the information is destined to a general purpose register defined by the alter mask in control register 9. The register is considered to be altered, even when the value is unchanged, if the instruction could have changed the value. The register could have been transferred to itself or the value modified by zero. If the instruction involves multiple registers, the event is reported when any one of the group is defined by the alter mask.

When bit 3 of the PER code is set to 1, the program event address is set to the address of the reference instruction except when it was initiated by the execute instruction when that address is stored. The next instruction address is stored as the instruction address of the old PSW.

PER Operations Introduction

At the end of each instruction, if the PER control is active, the operation is branched to the entry of the PER routine (GQER). The normal RTN LNK statement performs the branch because the link address has been altered during the setup to the B5 module. Of interest in this routine is the successful branch ending because this is one of the events being monitored. When the entry is through the branch leg, a test is made for the load-PSW instruction that is not included in the event test. If the PER control bit 0 is set, the routine posts the PER code bit 0 as the indicator in the PM register byte 1. One or more of the remaining PER code bits may have been posted by the routine before the instruction was performed. Any bits in this byte indicated that a PER interrupt is pending and the operation branches to the GICM routine to post the interrupt conditions. If no PER interrupt is pending, a test is made for other interrupts pending. In which case, the information for the one of highest priority is posted.

After posting the interrupt information for an interrupt, the operation branches to the GIPW routine to store the current PSW as the old PSW and enter the appropriate new PSW to process the interrupt. The exchange of PSWs normally results in masking the PER feature (PSW bit 1) for the interrupt sequence. The exchange may require additional setup operations for a change in the control mode. Before returning to the I-cycle hardware to start the first instruction of the interrupt program, the tests for PER events must be made on that addres if PER is masked on in the new PSW.

When no interrupt conditions are involved upon entry into the GQER routine or with the return from the GIPW routine, the three events involving addressing are tested. The address of the next instruction in the I-register is stored in control storage FFD0 for use in posting an interrupt after the instruction operation. The PER starting address and the PER ending address are entered to define the monitored main storage area. The two addresses are compared to determine whether the monitored area wraps storage which is considered acceptable. After determining that the instruction-fetch event is to be monitored, the instruction address in the I-register is compared to the starting and ending addresses to determine whether it falls within the monitored area. When it does, the PER code bit 1 is posted to indicate.

Further event testing requires the entry of the instruction op-code. This op-code is masked to determine whether it is the execute instruction. In which case, the object instruction address must also be tested for possible location within the monitored area. The operation sequence enters the object op-code because it is this operation that involves the operand address. A test is also made for a possible execute-execute sequence that is invalid.

Following the instruction-fetch event test, the op-code digits are used to address the F3 module of control storage. The byte thus entered contains a PER control mask and four control bits that define unique conditions for the op-code. The PER control mask is ANDed with the PER control bits to remove those events that cannot occur. A bit remaining in the modified PER control byte indicates that the operand is considered changed. The four low-order bits are defined as follows:

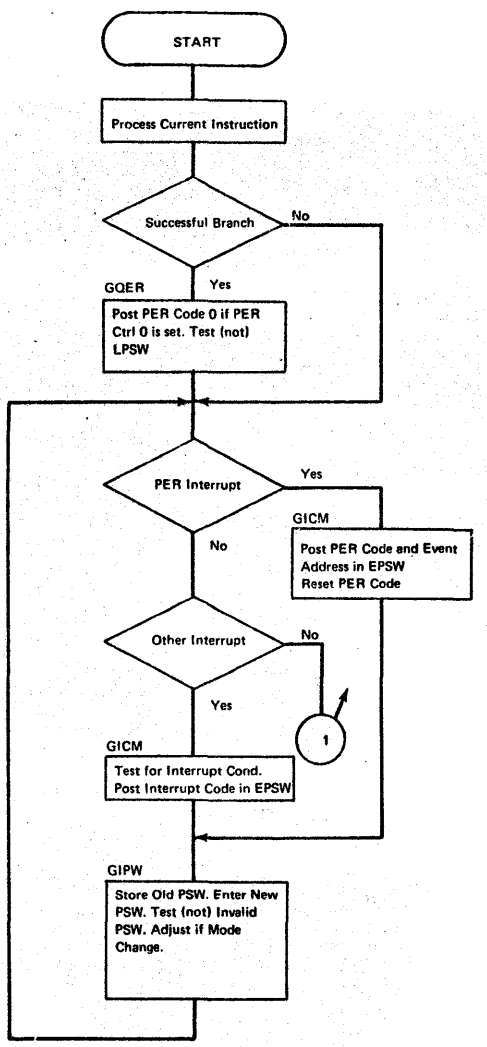
Bit 4 Special Instructions (B2 and EA)
Bit 5 Insert Char under Mask (BF)
Bit 6 Load Multiple (98)
Bit 7 Long Operation (2 GPRs).

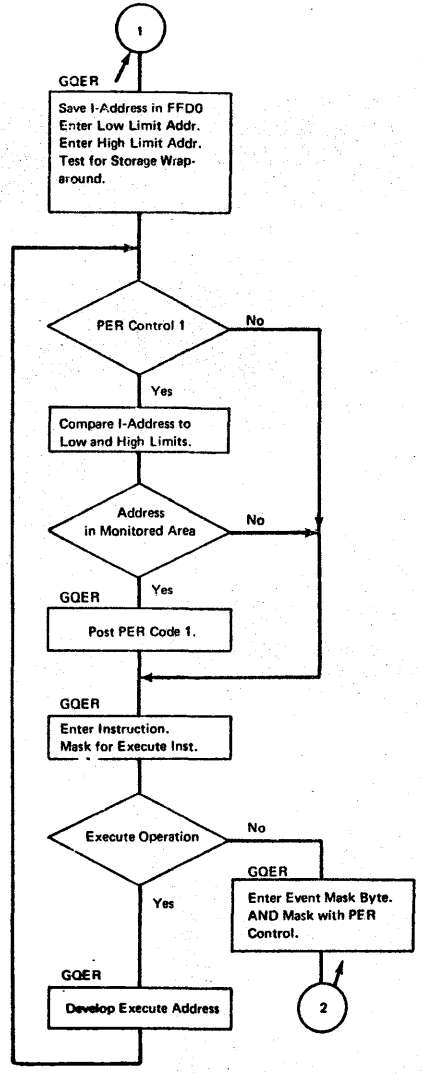
The modified PER control byte is tested for bit 2 to determine whether the storage-alter event is to be considered. To make the test, the operand address information of the instruction must be entered and the address developed. This address is compared with the previously entered PER starting and ending addresses defining the monitored area of main storage. If the operand address falls within this area, the PER code bit 2 is set.

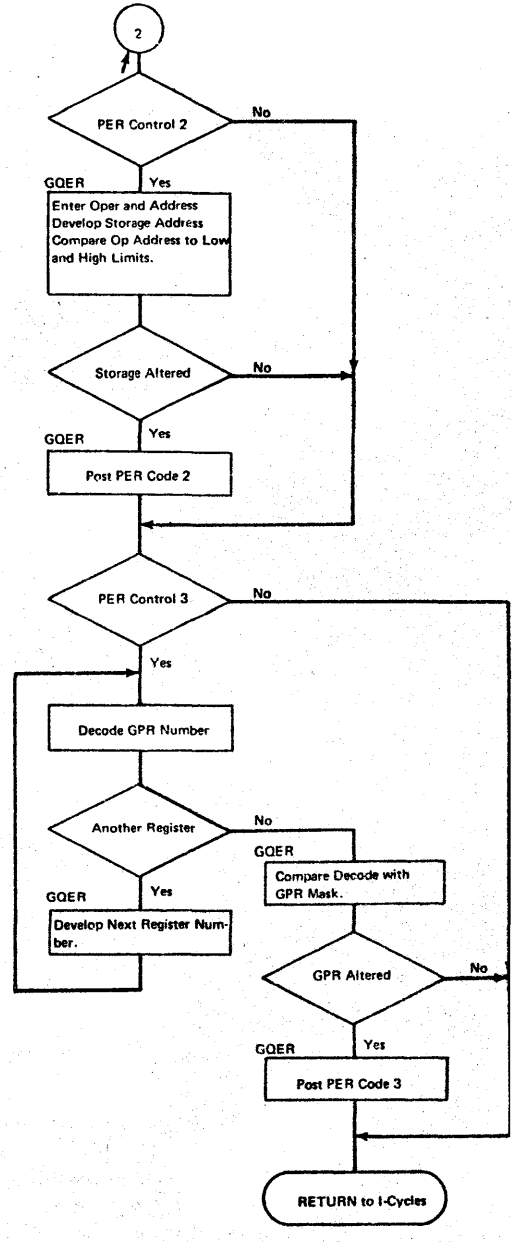
When the modified PER control byte contains bit 3, a test is made with the GPR-alter mask to determine whether the altered register is monitored. Bits 4, 5, 6, and 7 of the modified control byte represent special conditions to be considered in testing. Bit 4 identifies the B2 and EA instructions that are not tested. Bit 5 indicates an insert character under mask instruction that does not represent an alter when the mask is zero. Bit 6 indicates a load-multiple instruction in which a series of GPRs are altered. Bit 7 identifies a long operation instruction in which two GPRs are altered.

The address for a GPR is in binary notation in the instruction and must be encoded into a bit identification to compare with the GPR alter mask in control register 9. For a load-multiple operation, the instruction defines the first and last register involved and the bit identification for each register must be encoded. For a long operation the two consecutive registers are encoded. The identification bits are assembled into bytes 2 and 3 of a working register to be ANDed with bytes 2 and 3 of control register 9. Any matching bit condition results in a non-zero result to indicate that the event has taken place. The operation defines that a monitored register has been altered but does not indicate which one or whether more than one was altered. The PER code bit 3 is set to indicate the GPR-alter event.

Following the tests for PER for the next instruction, the operation returns to hardware I-cycles to process the next instruction. This instruction is the first instruction of the interrupt routine if the PSWs were exchanged.







DOCUMENTATION PLAN

VOL 30

INDEX — Alphabetical subject index of documentation contents, and index of cross references among all volumes.

PLAN — Documentation plan defines the organization of the manual and the contents of major sections. Includes legend pages describing manual symbology.

INTRoduction — Presents high level description of the CPU. Shows data and control flow, plus a brief description of major functional area.

CPU Hardware — Describes the working relationship of the functional units that make up the CPU. Uses unit data flows, operational diagrams, text, and second level diagrams where necessary.

VOL 31

MICroprogram — Contains microprogram information needed to read the microlisting (microprogram source document). Describes the control words that make up the microprograms, and contains microprogram instruction examples.

Console-File Adapter — Presents a high level description of the Console File and a detailed description of the Console-File attachment circuits.

Console Printer-Keyboard — Presents a detailed description of the 3210 and 3215 Console Printer-Keyboards and associated attachment circuitry. Includes microprogram operation and programming information.

VOL 32

PoWeR — Describes motor-generator concepts and CPU power system. Includes sequencing, distribution, service checks, removal and replacement, and preventive maintenance.

STORage — Presents high level description of Phase 2I technology. Describes the storage areas and functions which include: addressing, BSM data flow, and timings. Explains the use of Error Checking and Correction (ECC) in detecting and correcting data errors.

VOL 33

CHaNneLs — Provides a detailed description of the Model 145 I/O Channels: Byte-Multiplexer, Selector, and Block-Multiplexer. Included are operational diagrams and flow diagrams with related text. Defines the standard interface with expansions.

Integrated File Adapter — Covers the details of the adapter data flow, file operations, checking facilities, functional units, and interrupt handling.

Optional FEATures — Covers optional features that are not covered in related sections. Included are descriptions of IBM 1401/1440/1460 and 1410/7010 Compatibility, Direct Control, and Channel-to-Channel Adapter.

VOL 34

SystemCoNSoLe — Provides descriptions of keys, lights, and switches on the system operator console. Included are console operating procedures and applications.

RECovery Features — Describes hardware and programs involved in detecting and handling of machine malfunctions.

DIAGnostic Functions — Describes the diagnostic hardware and programs used in the Model 145.

REFerence — Contains a summary of data that is useful to the service representative when troubleshooting.

ABBREVIATIONS

A

A	and function
AAR	A-address register
ABM	advanced bi-polar monolithic (storage)
ABRTY	A and B retry register
ac	alternating current
ACB	address check boundary
ACBR	address check boundary register
ACR	automatic carrier return
A/D	alter/display
ADDR	address
adj	adjust
Adr-I	address-in
Adr-O	address-out
adv	advance
ALD	automated logic diagram
A-LS	A-local storage
alt	alter
Alt/Disp	Alter/Display
ALU	arithmetic logic unit
AM	address mark
amp	amplifier, ampere
ANUM	add numeric
appndg	appendage
APR	alternate path retry
arith	Arithmetic
ASCII	american standard code information interchange
ASCP	automatic system checkout program
asm	assembler
AT	attention (file)
ATTN	attention
avl	available

B

BAL	branch and link
BAR	B-address register
BBE	branch on bit equal
BC	basic control
BCA	bit count appendage
BCA	basic channel adapter
BCAI	basic channel adapter interface
BCD	binary coded decimal
BCE	branch on character equal
BDIL	branch and do interpretive loop
bfr	buffer
BI FLAG	branch on invalid flag
bin	binary
blk	block
B-LS	B-local storage
BMF	block multiplexer feature
BR	bit ring
BR	branch
brd	board
BS	byte source, bit select
BSM	basic storage module
bwd	backward
BWF	branch if wordmark or zone equal
BYTDST	byte destination

C

C	count
CA	control address
CAR	cylinder address register
CAW	channel address word
CB	circuit breaker
CC	condition code, chain command, cyclic code
CCC	channel control check
CCH	channel check handler
CCW	channel command word
CD	chain data time-of-day clock
CDA	chained data
CDC	channel data check
CE	customer engineer
CF	console file
CFC	console-file checking
CFDA	console-file disk address
CFDR	console-file data register
chan	channel
char	character
chk	check
chng	change
chnl	channel
CIDA	channel indirect data address
ck	check
CKD	count, key, and data
clk	clock
CM	current module
cmd	command
Cmd-O	command-out
cmnd	command
cncl	cancel
cnd	condition
cnsl	console
cnt	count
cntr	counter
coax	coaxial cable
CO	convenience outlet
comp	compare
con-con	contingent connection
cond	condition
cons	console
conv	convenience
corr	correction
CP	circuit protector
CPK	console printer-keyboard
cpmt	complement
CPU	central processing unit
CPURTY	central processing unit retry register
CR	control register
CS	control storage
CSW	channel status word
CTCA	channel-to-channel adapter
CTCAX	channel-to-channel adapter X system
CTCAY	channel-to-channel adapter Y system
ctr	counter
ctrl	control
CU	control unit
CUA	control unit address
CUB	control unit busy
CUE	control unit end
cyc	cycle
cyl	cylinder

D

D	data
DA	data address
DAT	dynamic address translation
DASD	direct access storage device
dbl	double
DC	direct control
dc	direct current
DCBI	direct control bus-in
DCBO	direct control bus-out
DCC	disconnect command chaining
DCHI	direct control hold-in
DCPL	De-coupler
DDR	dynamic device reconfiguration
DE	device-end
dec	decode, decimal, decrement
DED	double error detect
Del	delay
dest	destination
det	detect
DF	disk file
diag	diagnostic
diff	difference
DIL	do interpretive loop
Dir-In	direct control bus-in
Dir-Out	direct control bus-out
Disc-I	disconnect-in
dist	distribution
div	division
DL	data length
dly	delay
D-Mod	D-modifier
Doc	documentary console
DOS	disk operating system
dply	display
dsbld	disabled
dup	duplicate

E

EBCDIC	extended binary coded decimal interchange code
EBI	external bus-in
EBO	external bus-out
EC	external control, engineering change extended control
ECC	error checking and correction
ECCL	error checking and correction logic
ECNT	error count register
ECSW	extended channel status word
ED	external damage
EDBI	external data bus-in
EDBO	external data bus-out
EM	external damage report mask
env	envelope
EOF	end of file
EOL	end of line
EP	emergency pull (switch)
EPO	emergency power off
EPSW	extended PSW
eq	equal
ERDS	environmental recording data set
EREP	environment recording edit and print program
ERP	error recovery procedure
err	error
ev	even
exc	exception
EXCA	external control assembler
EXE CPLT	execute complete
exp	expanded
EXPLS	expanded local storage
ext	external
ext asm	external assembler
ext dst	external register destination
extint	external interrupt

F

f	...e
FBAK	...backup external word
FBO	file bus-out
FCH	file count register high
FCL	file count register low
FCND	file conditions external word
fdbck	feed back
FDR	file data register
FERR	file error external word
FM	file mask
FF	flip flop
FLS	feature local storage
FM	file mask
FOP	file operation register
F STAT	file status external word
FTAG	file tags external word
FTC	flush through checking
fwd	forward

G

gen	generate, generator
GM	group mark
gnd	ground
GR	general registers
grp	group
GSTAT	selector channel status external word

H

HA	home address
HDV	halt device
hdwr	hardware
hex	hexadecimal
hi	high
HIO	halt input/output
HMRTY	H and M retry registers
HS	hard stop
hwd	hardware
Hz	hertz

I

IAR	instruction address register
IB	interrupt buffer
IBU	I-register backup
IC	instruction counter
ICC	interface control check
icplt	incomplete
I-cy	instruction cycle
id	identifier
IDA	indirect data address
IDAL	indirect data address list
IDALW	indirect data address list word
if	interface
IFA	integrated file adapter
IFCC	interface control check
IFCU	integrated file control unit
IL	incorrect length
ILC	instruction length code
IM	input/output extended logout mask
IMPL	initial microprogram program loading
INB	in backward
inc	increment
ind	indicator
INF	in forward
inh	inhibit
inst	instruction, instruct
instr	instruction
intf	interface
intlk	interlock
intr	interrupt
intv	interval
intvn	intervention
invld	invalid
invrt	invert
I/O	input or output
IOCA	input/output communications area
IOEL	input/output extended logout
IPL	initial program load
ISC	integrated storage control
ISK	insert storage key

J

JCL	job control language

K

K	kilo, relay, key
KD	key and data
keybd	keyboard
KL	key length

L

L	length
LC	lower case
LCTL	load control
LD	line driver
LEX	local execute mode
LO	low
Logl	logical
LH	L-register high half
LHM	left-hand margin
LL	L-register low
lnk	link
LR	line receiver
LRA	load real address
LRU	least recently used
LS	local storage
LSCA	local storage control assembler
LSCS	local storage control storage
LSDST	local store destination
lth	latch

M

mach	machine
MB	M-register back up
MBO	multiplexer bus-out
MC	machine check
MCAR	machine-check analysis and recording
MCEL	machine-check extended logout
MCH	machine-check handler
MCK	machine-check register
MCKB	machine-check B-register
MCIC	machine-check interruption code
MCKA	machine-check A-register
MCPU	move data in CPU
MCRR	machine-check recovery recorder
MFE	magnetic feedback emitter
MFT	multiple fixed tasks
MG	motor generator
Mid-Pac	middle power package regulator
MIO	move data for I/O
misc	miscellaneous
MLC	machine level control
mod	module
mono	monolithic
MOP	mini operation register
MP	matrix printer
MPX	multiplexer
MRTY	M-retry register
ms	millisecond
us.	microsecond
MS	main storage
MSF	main-storage frame
MST	monolithic systems technology
MTO	multiplexer tags-out
MTI	multiplexer tags-in
MUA	multiple unit address
MVT	multiple variable task

N

n	inverter
NA	next address
N/L	new line
n/o	normally open
NOP	no operation
norm	normal
NPL	new product language
ns	nanoseconds

O

OBR	outboard recorder
OC	overcurrent
OE	exclusive OR
OP	operation
OP-I	operational-in
OPL	operational
OP-O	operational-out
OS	operating system
osc	oscillator
OV	over voltage

P

PAA	pre-address assembler
P-Bit	parity bit
PCI	program-controlled interrupt
PD	instruction processing damage
PDAR	program damage assessment and repair
PE	print emitter
PER	program event recording
PF	power frame
PG	power gate
PGA	power gate A
PGB	power gate B
PH	polarity hold
PIR	priority interrupt register
PIRM	priority interrupt register mask
POH	power-on hours
POR	power-on reset
pos	position
prev	previous
PR-KB	printer-keyboard
proc	process
prog	program
prot	protect
PSW	program status word
pt	point
PTLB	page table lookaside buffer
ptr	printer
pty	parity
pwr	power

R

r	register
R0	record zero
R1	record 1
RAC	remote analysis center
RAS	reliability availability serviceability
RCNT	retry count register
RCS	reloadable control storage
RD	read
RDK	reset diagnostic key

rdy	ready
rec	recovery
recal	recalibrate
ref	reference
reg	regulator, register
req	request
reqd	required
Req-I	request-in
rev	reverse
RHM	right-hand margin
RM	record mark
RMS	recovery management system
rms	root mean square
RR	record ready recovery report
RRB	reset reference bit
Rst	reset, restart
rtn	return
rty	retry
rty flg	retry flag

S

SAR	storage address register
SCAMPART	storage console approaches, manual procedures, and reference timings
SCF	storage and control frame
sch	search
SCP	system control panel
SCR	silicon controlled rectifier
sct	sector
SD	system damage
SDBI	storage data bus-in
SDBO	storage data bus-out
SDC	suppress data check
SDK	set diagnostic key
SDR	storage data register, statistical data recorder
sec	secondary
sect	sector
sel	select
seld	selected
Sel-I	select-in
Sel-O	select-out
selr	selector
seq	sequence
serdes	serializer/deserializer
SEREP	system error record editing program
Serv-I	service-in
Serv-O	service-out
sht	short
SI	system incidents
SIO	start input/output
SIOF	start I/O fast release
S/L	sense preamplifier and data latch
SLI	suppress length indication
SLT	solid logic technology
SM	synchronous mask
sng	single
SO	segment origin
spec	special
SPTL	S, P, T and L-registers
SPTLB	SPTL back-up word for SPTL registers
SR	system recovery
S/R	set or reset
SSK	set storage key
st	start
Stat-I	status-in
STCTL	store control
std	standard
stg	storage
STI	status-in
STIDC	store identification channel
STIDP	store identification processor
stkd	stacked
stor	storage
stp	stop
SUA	single unit address
supp	suppress
Sup-O	suppress-out
SUT	system unavailable time
SVC	supervisor call
SVI	service-in
sw	switch
SX	selector/block-multiplexer channel
sync	synchronize
synd	syndrome
sys	system

T

TB	terminal block
TCH	test channel
TD	timer damage, time delay
temp	temporary, temperature
term	terminator
tfmr	transformer
TH	T-register high
therm	thermal
thld	threshold
thru	through
TIC	transfer in channel
TIO	test input/output
TLB	translation lookaside buffer
TOD	time of day
TODH	time-of-day high word
TODL	time-of-day low word
TP	timing pulse
TR	transformer-rectifier, trap
T/R	tilt or rotate
tgr	trigger
TSBO	timing signal bus-out

U

UC	upper case
UCW	unit control word
U/L	upper or lower
us.	microsecond

V

V	volts
vac	volts alternating current
val	validate
vdc	volts direct current
VFO	variable frequency oscillator
VMA	virtual machine assist feature
VOM	volt/ohm meter

W

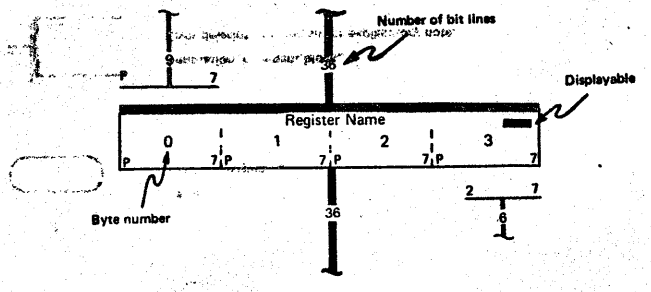
WB	word bottom
wd	word
WK	working register
WLR	wrong length record
WM	word mark
Wr	write
WS	word separator, word source
WT	word-top
WTC	World Trade Corp.

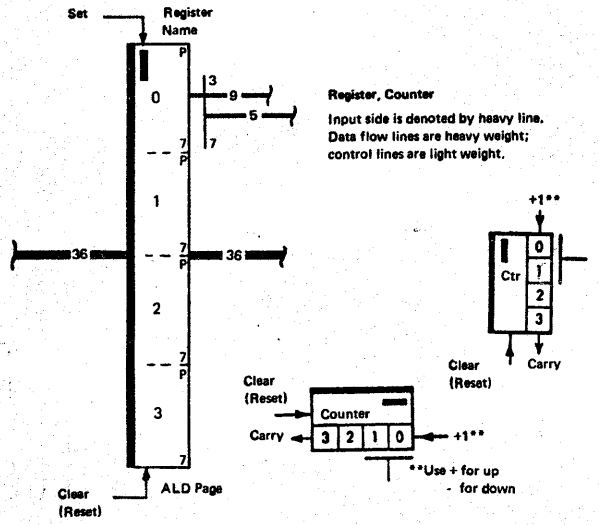
X

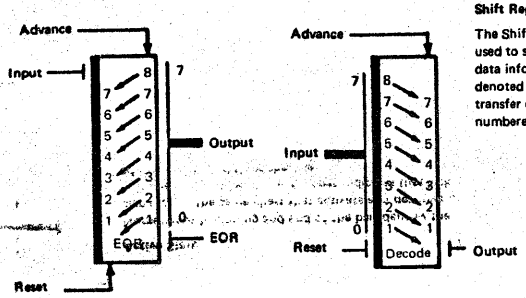
XFer	transfer
X-Late	translate

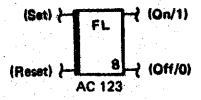
*BAS	basic micro-diagnostic group
μ	micro
μpm	microprogram
μs	microsecond
Φ	phase
ΦCR	phase control regulator

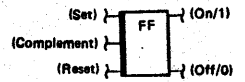
LEGEND

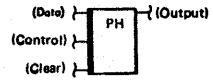


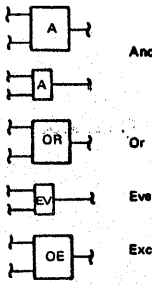


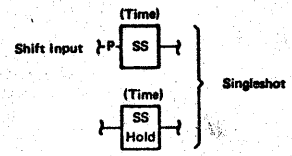






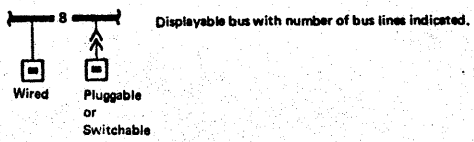






Indicator Lamp

Identifies displayable bus, register, latch, etc. such as:

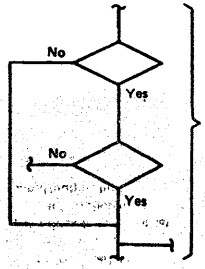


TIMING CHARTS

Active State

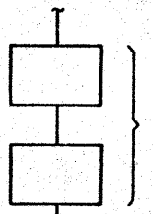
Numerals at beginning and end of the bar identify the signal(s) on the same chart that activate and deactivate this line. "(Not)" with the number indicates that lack of the signal conditions the line.

Flowcharts



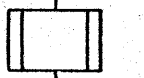
Decision

Indicates a point in a flow chart where a branch to alternate paths is possible.



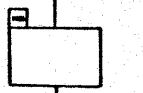
Process

Major Functions or Events

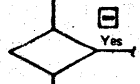


Function or Process Detailed Elsewhere

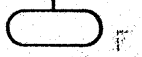
Indicate in blockwhere the detailed, flow chart of the process is located.



Flow chart block where action is indicatable. (Use and placement of indicator is optional)



Flow chart block where condition branched on is indicatable. (Use and placement of indicator is optional.)



Terminal

Indicates beginning or end of flow path.

Annotation, comment block

Gives descriptive comment or explanatory note.

GENERAL

On-Page Connector

Indicates connection between two parts of the same diagram. Arrow leaving symbol points (line-of-sight) to correspondingly numbered symbol.

A3

On-Page Connector

Indicates connection between two parts of the same diagram. Alphameric grid coordinate of complementary connector shown beneath.

Diag 1-2

Off-Page Connector

Indicates connection between diagrams located on separate pages. Location of correspondingly - lettered symbol shown adjacent. Alphameric grid coordinate may be included.

INDEX

D

T

NUMERIC INDEX