



**CDC® CYBER 170
COMPUTER SYSTEMS
MODELS 835 AND 855**

HARDWARE REFERENCE MANUAL

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
Front Cover	-	4-24	A	5-44	A				
Inside Front Cover	A	4-25	A	5-45	A				
Title Page	-	4-26	A	A-1	A				
2	A	4-27	A	Index-1	A				
3/4	A	4-28	A	Index-2	A				
5	A	4-29	A	Index-3	A				
6	A	4-30	A	Index-4	A				
7	A	4-31	A	Index-5	A				
8	A	4-32	A	Index-6	A				
9	A	4-33	A	Index-7	A				
1-1	A	4-34	A	Comment Sheet	A				
1-2	A	4-35	A	Back Cover	-				
1-3	A	4-36	A						
1-4	A	4-37	A						
1-5	A	4-38	A						
1-6	A	4-39	A						
2-1	A	5-1	A						
2-2	A	5-2	A						
2-3	A	5-3	A						
2-4	A	5-4	A						
2-5	A	5-5	A						
2-6	A	5-6	A						
2-7	A	5-7	A						
2-8	A	5-8	A						
2-9	A	5-9	A						
2-10	A	5-10	A						
2-11	A	5-11	A						
2-12	A	5-12	A						
2-13	A	5-13	A						
3-1	A	5-14	A						
3-2	A	5-15	A						
3-3	A	5-16	A						
3-4	A	5-17	A						
3-5	A	5-18	A						
3-6	A	5-19	A						
4-1	A	5-20	A						
4-2	A	5-21	A						
4-3	A	5-22	A						
4-4	A	5-23	A						
4-5	A	5-24	A						
4-6	A	5-25	A						
4-7	A	5-26	A						
4-8	A	5-27	A						
4-9	A	5-28	A						
4-10	A	5-29	A						
4-11	A	5-30	A						
4-12	A	5-31	A						
4-13	A	5-32	A						
4-14	A	5-33	A						
4-15	A	5-34	A						
4-16	A	5-35	A						
4-17	A	5-36	A						
4-18	A	5-37	A						
4-19	A	5-38	A						
4-20	A	5-39	A						
4-21	A	5-40	A						
4-22	A	5-41	A						
4-23	A	5-42	A						
		5-43	A						

MEMORANDUM FOR THE RECORD

DATE: 10/15/54

TO: SAC, NEW YORK

FROM: SA [Name], NEW YORK

RE: [Name]

[Faint, mostly illegible typed text in the main body of the memorandum, consisting of several paragraphs.]



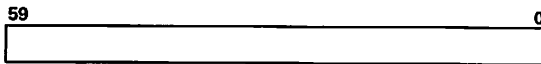
PREFACE

This manual contains hardware reference information for the CDC® CYBER 170 Models 835 and 855 Computer Systems.

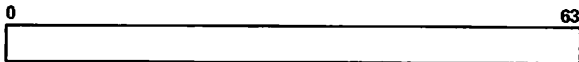
The manual describes the functional, operational, and programming characteristics of the computer system hardware. Additional system hardware information is available in the publications listed in the system publication index on the following page.

This manual is for use by customer, marketing, training, programming, and Engineering Services personnel who operate, program, and maintain the computer systems.

There are two methods used within this manual to designate bit numbers. In the majority of the manual, bits are numbered 59 through 0 reading from left to right.



However, in the context of the two-port multiplexer and maintenance registers, bits are numbered 0 through 63 from left to right.



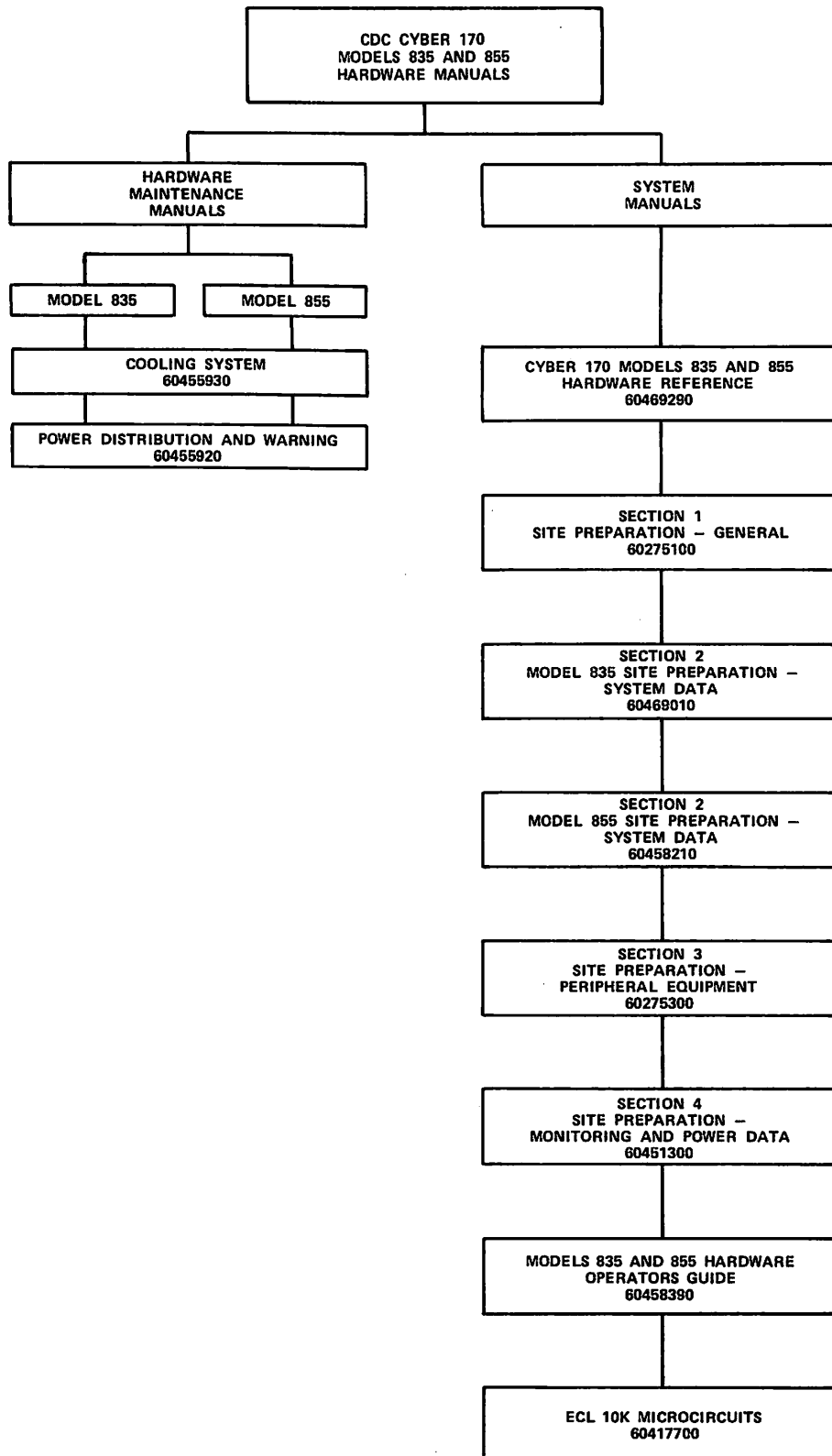
Other manuals that are applicable to the CYBER 170 computer systems but not listed in the following index are:

<u>Control Data Publication</u>	<u>Publication Number</u>
NOS Version 2 Operator/Analyst Handbook	60459310
NOS Version 2 Systems Programmer's Instant	60459370
NOS Version 1 Operator's Guide	60457700
NOS Version 1 Systems Programmer's Instant	60457790
NOS/BE Version 1 Operator's Guide	60457380
NOS/BE Version 1 System Programmer's Reference Manual, Volume 1	60458480
NOS/BE Version 1 System Programmer's Reference Manual, Volume 2	60458490

Publication ordering information and latest revision levels are available from the Literature Distribution Services catalog, publication number 90310500.

WARNING

This equipment generates, uses and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. As temporarily permitted by regulation, it has not been tested for compliance with the limits for Class A computing devices pursuant to Subpart J of Part 15 of the FCC Rules which are designed to provide reasonable protection against such interference. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.



CONTENTS

1. SYSTEM DESCRIPTIONS	1-1	3. OPERATING INSTRUCTIONS	3-1
Introduction	1-1	Controls and Indicators	3-1
Physical Characteristics	1-1	Deadstart Panel Controls/Indicators	3-1
Functional Characteristics	1-2	Central Memory Controls	3-1
Model 835 Characteristics	1-2	Power-On and Power-Off Procedures	3-4
Central Processor	1-2	Operating Procedures	3-4
Central Memory	1-2	Control Checks	3-4
Input/Output Unit	1-3	Deadstart Sequences	3-4
Model 855 Characteristics	1-3	IOU Reconfiguration	3-6
Central Processor	1-3		
Central Memory	1-3		
Input/Output Unit	1-3		
Major System Component Descriptions	1-4	4. INSTRUCTION DESCRIPTIONS	4-1
Central Processor	1-4	CP Instructions	4-1
Control Section	1-4	CP Instruction Formats	4-1
Registers	1-4	CP Operating Modes	4-2
Execution Section	1-4	CP Instruction Descriptions	4-2
Cache Memory	1-4	Instruction Execution Timing	4-20
Addressing Section	1-4	PP Instructions	4-27
Central Memory Control	1-4	PP Instruction Formats	4-27
(Model 855 Only)	1-4	PP Data Format	4-27
Central Memory	1-4	PP Relocation Register Format	4-27
Input/Output Unit	1-5	PP Instruction Descriptions	4-28
Display Station	1-5	Instruction Execution Timing	4-36
2. FUNCTIONAL DESCRIPTIONS	2-1	5. PROGRAMMING INFORMATION	5-1
Central Processor	2-1	CP Programming	5-1
Control Section	2-1	CYBER 170 Exchange Jump	5-1
Model 835 Instruction Lookahead	2-1	Executive State	5-2
Model 855 Instruction Lookahead	2-1	Floating-Point Arithmetic	5-2
Maintenance Access Control	2-1	Format	5-2
Instruction Control Sequences	2-1	Packing	5-2
Registers	2-4	Overflow	5-3
Operating Registers	2-4	Underflow	5-3
Support Registers	2-5	Indefinite	5-3
Execution Section	2-7	Nonstandard Operands	5-4
Cache Memory	2-7	Normalized Number	5-4
Addressing Section	2-7	Rounding	5-4
Central Memory Control (Model 855 Only)	2-7	Double-Precision Results	5-4
Central Memory	2-7	Fixed-Point Arithmetic	5-4
Address Format	2-7	Integer Arithmetic	5-6
CM Access and Cycle Times	2-7	Compare/Move Arithmetic	5-6
Model 835	2-7	Instruction Lookahead Purge Control	5-6
Model 855	2-7	Model 835 Purge Control	5-6
CM Ports and Priorities	2-8	Model 855 Purge Control	5-7
SECDED	2-8	Error Response	5-7
CM Layout	2-8	Illegal Instructions	5-7
CM Bounds Register	2-8	Hardware Errors	5-7
Central Memory Reconfiguration	2-8	Conditional Software Errors	5-12
Input/Output Unit	2-11	Memory Programming	5-12
Peripheral Processor	2-11	Addressing Modes	5-12
Deadstart	2-11	Direct Read/Write Instructions (014, 015, 660, 670)	5-12
Barrel and Slot	2-11	Block Copy Instructions (011, 012)	5-12
PP Registers	2-11	PP Programming	5-14
PP Numbering	2-13	Central Memory Addressing by PPs	5-14
PP Memory	2-13	PP Memory Addressing by PPs	5-14
I/O Channels	2-13	Direct 6-Bit Operand	5-14
Real-Time Clock	2-13	Direct 18-Bit Operand	5-14
Two-Port Multiplexer	2-13		
Maintenance Channel	2-13		
Central Memory Access	2-13		

Direct 6-Bit Address	5-14	Real-Time Clock Programming	5-23
Direct 12-Bit Address	5-14	Two-Port Multiplexer Programming	5-23
Indexed 12-Bit Address	5-14	Two-Port Multiplexer Operation	5-23
Indirect 6-Bit Address	5-14	Terminal Select (7XXX)	5-23
Central Memory Read/Write Instructions	5-14	Terminal Deselect (6XXX)	5-23
PP Central Memory Read		Read Status Summary (00XX)	5-23
Instructions (60, 61)	5-14	PP Read Terminal Data (01XX)	5-23
PP Central Memory Write		PP Write Output Buffer (02XX)	5-24
Instructions (62, 63)	5-15	Set Operation Mode to the	
Input/Output Channel Communications	5-15	Terminal (03XX)	5-24
Inter-PP Communications	5-15	Set/Clear Data Terminal Ready (04XX)	5-24
PP Program Timing Considerations	5-16	Set/Clear Request to Send (05XX)	5-24
Channel Operation	5-16	Master Clear (07XX)	5-24
Channel Control Flags	5-16	Programming Considerations	5-24
Channel Active/Inactive Flag	5-16	Output Data	5-25
Register Full/Empty Flag	5-16	Input Data	5-25
Channel (Marker) Flag Instructions		Request to Send and Data Terminal	
(641, 651)	5-16	Ready	5-25
Error Flag Instructions (661, 671)	5-16	Maintenance Channel Programming	5-25
Channel Transfer Timing	5-16	Maintenance Channel	5-25
Input/Output Transfers	5-18	MCH Function Words	5-25
Data Input Sequence	5-18	MCH Control Words	5-27
Data Output Sequence	5-18	MCH Programming for Halt/Start	
Display Station Programming	5-21	(Opcod 0/1)	5-27
Keyboard	5-21	MCH Programming for Read/Write	
Data Display	5-21	(Opcod 4/5)	5-27
Character Mode	5-21	MCH Programming for Master Clear/	
Dot Mode	5-21	Clear Errors (Opcod 6/7)	5-28
Codes	5-22	MCH Programming for Read IOU Status	
Programming Example	5-22	Summary (Opcod C, IOU only)	5-28
Programming Timing Considerations	5-22		

APPENDIX

A. GLOSSARY

A-1

INDEX

FIGURES

1-1 Models 835 and 855 Computer Systems	1-1	4-4 PP Relocation (R) Register Format	4-28
1-2 Model 835 Chassis Configuration		5-1 CYBER 170 Exchange Package	5-1
(Top Cutaway View)	1-2	5-2 Floating-Point Format	5-2
1-3 Model 855 Chassis Configuration		5-3 Floating-Add Result Format	5-4
(Top Cutaway View)	1-2	5-4 Multiply Result Format	5-4
1-4 Model 835 Computer System	1-5	5-5 Format of Exit Condition Register	
1-5 Model 855 Computer System	1-6	at (RAC)	5-7
2-1 CYBER 170 Exchange Package	2-5	5-6 Memory Map	5-13
2-2 Address Format	2-7	5-7 Channel Transfer Timing	5-17
2-3 CM Layout	2-8	5-8 Data Input Sequence Timing	5-19
2-4 Barrel and Slot	2-12	5-9 Data Output Sequence Timing	5-20
2-5 Formation of Absolute CM Address	2-12	5-10 Display Station Output Function	
3-1 Deadstart Panel	3-1	Code	5-22
3-2 CM Configuration Switches	3-3	5-11 Coordinate Data Word	5-22
4-1 CP Instruction Parcel Arrangement	4-1	5-12 Character Data Word	5-22
4-2 PP Instruction Formats	4-27	5-13 Receive and Display Program Flowchart	5-22
4-3 PP Data Format	4-27		

TABLES

2-1	Port Priority	2-8	5-7	Error Exits in CYBER 170 Monitor Mode (MF=1)	5-8
2-2	Model 835 SECDED Syndrome Codes/Corrected Bits	2-9	5-8	Error Exits in CYBER 170 Job Mode (MF=0)	5-10
2-3	Model 855 SECDED Syndrome Codes/Corrected Bits	2-10	5-9	Keyboard Character Codes	5-21
3-1	Deadstart Panel Controls/Indicators	3-2	5-10	Display Character Codes	5-21
3-2	Central Memory Reconfiguration	3-4	5-11	MCH Function Word Bit Assignments	5-26
3-3	Barrel Numbering Table	3-6	5-12	Model 835 CP Internal Address Assignments	5-28
4-1	Central Processor Instruction Designators	4-2	5-13	Model 835 CM Internal Address Assignments	5-29
4-2	Collate Table	4-16	5-14	Model 855 CP Internal Address Assignments	5-29
4-3	Model 835 CP Instruction Timing	4-20	5-15	Model 855 CM Internal Address Assignments	5-30
4-4	Model 855 CP Instruction Timing	4-24	5-16	IOU Internal Address Assignments	5-30
4-5	PP Nomenclature	4-27	5-17	Model 835 CP Register Bit Assignments	5-31
4-6	PP Instruction Timing	4-37	5-18	Model 835 CM Register Bit Assignments	5-34
5-1	Bits 58 and 59 Configurations	5-2	5-19	Model 855 CP Register Bit Assignments	5-36
5-2	Xj Plus Xk (30, 32, 34 Instructions)	5-5	5-20	Model 855 CM Register Bit Assignments	5-40
5-3	Xj Minus Xk (31, 33, 35 Instructions)	5-5	5-21	IOU Register Bit Assignments	5-42
5-4	Xj Multiplied by Xk (40, 41, 42 Instructions)	5-5			
5-5	Xj Divided by Xk (44, 45 Instructions)	5-7			
5-6	Contents of Exit Condition Register at (RAC)				

The first part of the document discusses the importance of maintaining accurate records. It emphasizes that proper record-keeping is essential for ensuring the integrity and reliability of the data collected. This section also outlines the various methods used to collect and analyze the data, highlighting the challenges faced during the process.

The second part of the document focuses on the results of the study. It presents a detailed analysis of the data, showing the trends and patterns observed. The findings indicate that there is a significant correlation between the variables studied, which supports the hypothesis of the research.

Finally, the document concludes with a summary of the key findings and a discussion of the implications of the study. It suggests that the results have important implications for the field and provides recommendations for further research.

The second part of the document discusses the methodology used in the study. It describes the experimental design, the data collection procedures, and the statistical methods employed to analyze the data. This section provides a clear and concise overview of the research process, allowing readers to understand the validity and reliability of the findings.

The third part of the document presents the results of the study. It includes a series of tables and graphs that illustrate the data collected. The results show that the data is consistent with the expected outcomes, and there are no significant differences between the groups studied.

The final part of the document discusses the conclusions drawn from the study. It summarizes the main findings and discusses their implications for the field. The authors conclude that the study has provided valuable insights into the relationship between the variables studied and offers suggestions for future research.



This section introduces the CYBER 170 Models 835 and 855 Computer Systems, gives physical and functional characteristics, and provides descriptions of major system components.

INTRODUCTION

Models 835 and 855 (figure 1-1) are medium scale, high-speed computer systems for both business and scientific applications. Both systems include the following components.

- Central processor (CP).
- Central memory (CM).
- Input/output unit (IOU).
- Display station.

PHYSICAL CHARACTERISTICS

The mainframe configurations for models 835 (figure 1-2) and 855 (figure 1-3) include a display station and a three-section cabinet for the CP, CM, and IOU. Each cabinet section contains a logic chassis with plug-in circuit boards. The logic chassis in the IOU also contains a deadstart panel with initialization and maintenance controls and displays. Each cabinet section also contains a cooling unit to cool the logic chassis, an ac/dc control section with voltage margin testing facilities, and dc power supplies. For additional cooling or power information, refer to the cooling system and power system manuals listed in the system publication index.

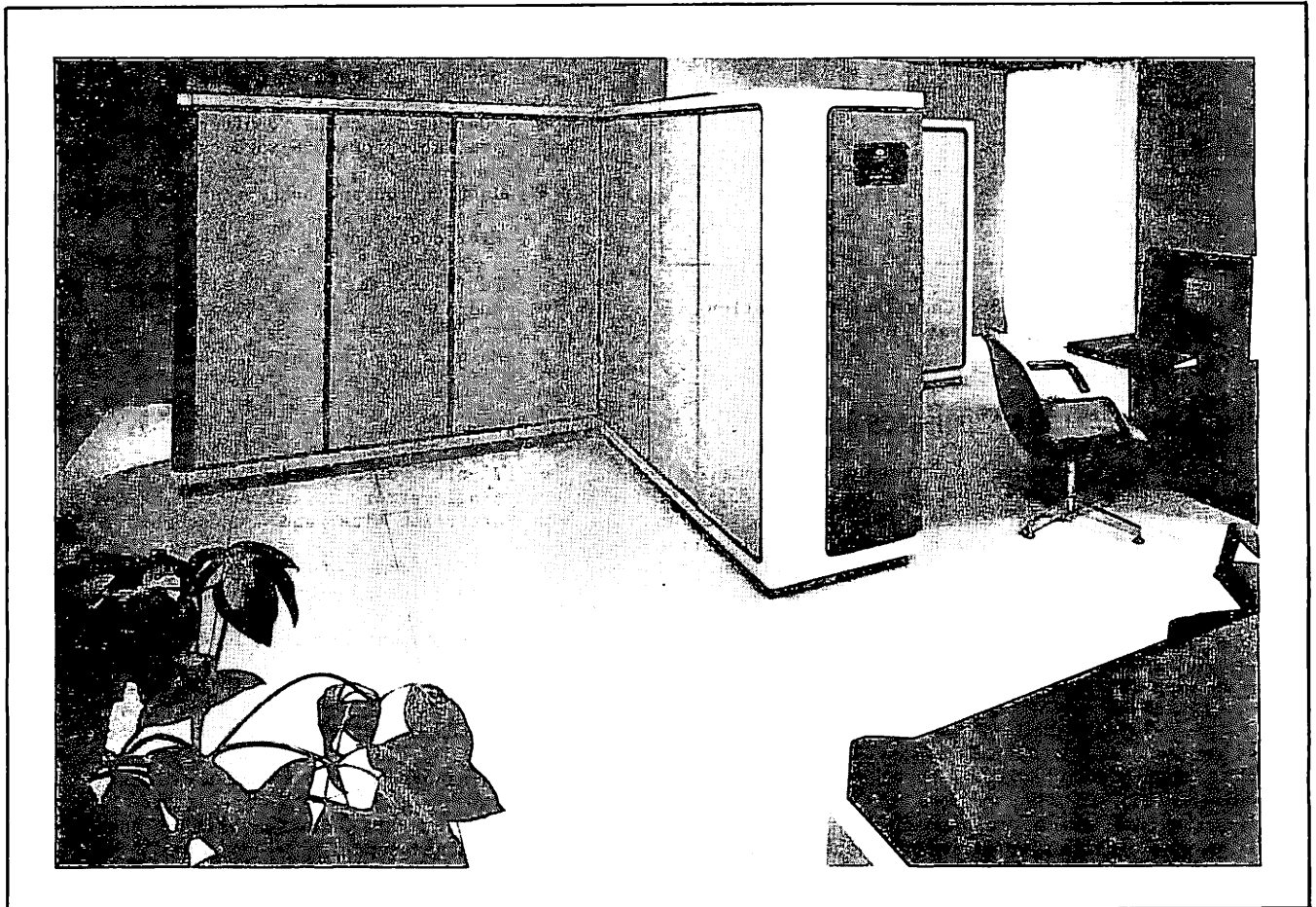


Figure 1-1. Models 835 and 855 Computer Systems

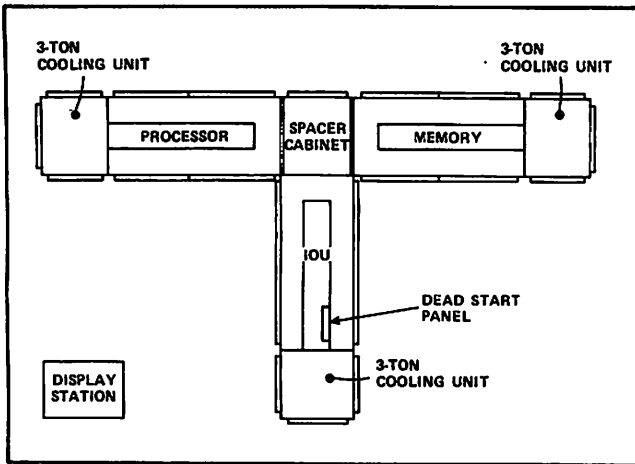


Figure 1-2. Model 835 Chassis Configuration
(Top Cutaway View)

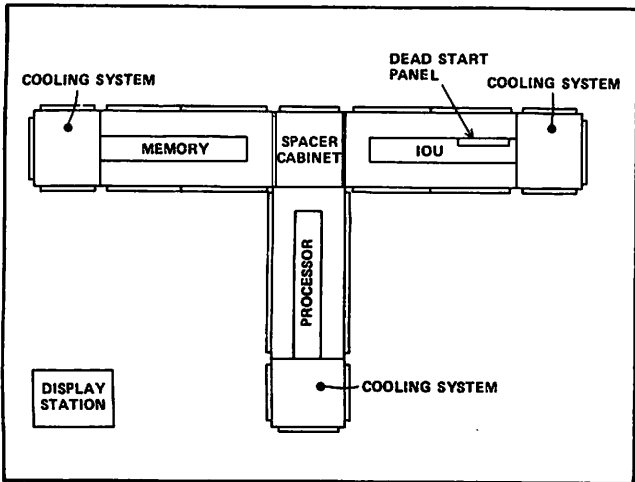


Figure 1-3. Model 855 Chassis Configuration
(Top Cutaway View)

FUNCTIONAL CHARACTERISTICS

To achieve high computation speeds, the model 835 uses emitter-coupled logic (ECL); the model 855 uses ECL and large scale integration (LSI) logic. Design architecture is also oriented towards the objective of high speed. The CP design is based on the assumption that both data and instructions are, in most cases, accessed from successive memory locations. Accordingly, the CP prefetches both instructions and data expected to be used next while the current instruction is being processed.

The semiconductor central memory is divided into eight independent banks. These banks may all be simultaneously in the process of completing read/write requests which are queued and distributed at ECL speeds. System input/output speeds are determined by the capabilities of existing external devices.

MODEL 835 CHARACTERISTICS

Central Processor

- 60-bit internal word.
- Eight 60-bit operand (X) registers.
- Eight 18-bit address (A) registers.
- Eight 18-bit index (B) registers.
- Two registers that isolate each user's central memory space (RAC, FLC).
- Two registers that isolate each user's extended memory space (RAE, FLE).
- Register exchange instructions (exchange jumps) for interrupting programs.
- Floating-point arithmetic (11-bit exponent, 48/96-bit coefficient).
- Integer arithmetic (60/18-bit operands).
- Character string compare/move facilities (6-bit characters).
- Packed instructions (15/30/60-bit instructions in 60-bit words).
- Synchronous internal logic.
- 56-nanosecond clock period.
- 2048-word cache buffer memory, option available for 4096-word cache.
- Instruction and branch instruction lookahead.
- Microcode control.
- Parity checking of all major data and address paths.
- Maintenance channel to IOU.

Central Memory

- 72-bit data word (60 data bits, 8 single-error correction double-error detection bits, and 4 unused bits).
- 524K words of refresh-type semiconductor memory, options available to 2097K words.
- Organization of eight independent banks.
- Two memory ports.
- Bounds register to limit write access.
- 56-nanosecond clock period.
- Maximum data transfer rate of one word every 56 nanoseconds.
- 672-nanosecond read access time.

- 448-nanosecond read/write cycle time.
- 896-nanosecond partial write cycle time.
- Read and write data queuing capability.
- Single-error correction double-error detection (SECDED) on stored data.
- Parity checking of all major data, address and control paths.
- Unified extended memory (UEM) which serves as extended memory within CM.

Input/Output Unit

- Ten peripheral processors (PPs), 15-PP/20-PP options available. Each PP has 4K independent memory (PPM) comprised of 16-bit words with the upper 4 bits zero.
- Port to central memory.
- Bounds register to limit writes to central memory.
- Twelve 12-bit CYBER 170 channels to external devices, 24 channel option available.
- Real-time clock (channel 14_g).
- Display controller (CYBER 170 channel 10_g).
- Two-port multiplexer (channel 15_g).
- Maintenance channel (channel 17_g).
- Parity checking on all major data and address paths.
- Operating speed of 250 nanoseconds and a minor cycle of 50 nanoseconds.

MODEL 855 CHARACTERISTICS

Central Processor

- 60-bit internal word.
- Eight 60-bit operand (X) registers.
- Eight 18-bit address (A) registers.
- Eight 18-bit index (B) registers.
- Two registers that isolate each user's central memory space (RAC, FLC).
- Two registers that isolate each user's extended memory space (RAE, FLE).
- Register exchange instructions (exchange jumps) for interrupting programs.
- Floating-point arithmetic (11-bit exponent, 48/96-bit coefficient).

- Integer arithmetic (60/18-bit operands).
- Character string compare/move facilities (6-bit characters).
- Packed instructions (15/30/60-bit instructions in 60-bit words).
- Synchronous internal logic.
- 64-nanosecond clock period.
- 2048-word cache buffer memory, option available for 4096-word cache.
- Instruction and branch instruction lookahead.
- Microcode control.
- Parity checking of all major data and address paths.
- Maintenance channel to IOU.

Central Memory

- 72-bit data word (60 data bits, 8 single-error correction double-error detection bits, and 4 unused bits).
- 524K words of refresh-type semiconductor memory, options available to 2097K words.
- Organization of eight independent banks.
- Two memory ports (located in the central processor cabinet).
- Bounds register to limit write access.
- 64-nanosecond clock period.
- Maximum data transfer rate of one word every 64 nanoseconds.
- 528-nanosecond read access time.
- 448-nanosecond read/write cycle time.
- 896-nanosecond partial write cycle time.
- Read and write data queuing capability.
- Single-error correction double-error detection (SECDED) on stored data.
- Parity checking of all major data, address and control paths.
- Unified extended memory (UEM) which serves as extended memory within CM.

Input/Output Unit

The model 855 input/output unit (IOU) is the same as that of the model 835. Refer to the description of the IOU under Model 835 Characteristics.

MAJOR SYSTEM COMPONENT DESCRIPTIONS

CENTRAL PROCESSOR

The CP hardware (figures 1-4 and 1-5) consists of the following:

- Control section.
- Registers.
- Execution section.
- Cache memory.
- Addressing section.
- On the model 855, central memory control.

The CP is isolated from the IOU and is thus able to carry on computation or character manipulation unencumbered by I/O requirements.

Control Section

The control section directs the arithmetic and manipulative functions for instruction execution. The control section prefetches instruction words from memory and disassembles them into instructions.

Registers

Operating registers reduce storage accesses for operands used during the execution of an instruction. These registers are:

- Eight 60-bit X registers (X0 through X7) which hold operands used for computation.
- Eight 18-bit A registers (A0 through A7) which use A0 primarily for indexing and A1 through A7 for CM operand addressing.
- Eight 18-bit B registers (B0 through B7) which are primarily indexing registers to control program execution. The B0 register always contains all zeros.

Eight support registers support the operating registers during program execution. These registers are:

- 18-bit program address (P) register.
- 21-bit reference address for CM (RAC) register. This is a program's lower bound.
- 21-bit field length for CM (FLC) register. This is a program's upper bound.
- 6-bit exit mode (EM) register.
- 6-bit flag register.
- 21-bit reference address for UEM (RAE) register.
- 24-bit field length for UEM (FLE) register.
- 18-bit monitor address (MA) register.

The registers store data and control information, present operands to the execution section, and store results.

Execution Section

The execution section combines the operands to achieve the result.

Cache Memory

The cache memory consists of two sets of fast bipolar memory, capable of storing 2048 60-bit words. It can be expanded to four sets with a capacity of 4096 words. The memory addressing sections determine whether a requested word is in the cache memory. If it is not, they read four consecutive words from central memory into the cache memory.

Addressing Section

The addressing section checks memory addresses against the CP registers RAC, FLC, RAE, and FLE to ensure isolation of user memory space.

Central Memory Control (Model 855 Only)

On the model 855, central memory control (CMC) is integrated within the CP. CMC controls the flow of data between CM and requesting system components.

CENTRAL MEMORY

The CM (figures 1-4 and 1-5) consists of the following:

- Eight memory banks.
- Two memory ports.
- Distributor.

The CM is a refresh-type metal oxide semiconductor (MOS) memory, which is organized into eight independent banks. Memory read/write requests are stored and distributed at ECL speeds, after which each bank completes the requests presented to it at MOS speeds.

A portion of CM can be reserved for use as extended memory. It is called unified extended memory (UEM), and is referenced by the RAE and FLE registers. On the model 835, UEM operates in 24-bit format standard addressing mode. On the model 855, it can operate in either 24-bit format standard addressing mode or 30-bit format expanded addressing mode.

On the model 835, each memory port has queuing buffers. On the model 855, one port has a queuing buffer. Both model 855 ports are located in the central processor cabinet.

The distributor resolves port conflicts and multiplexes data from ports to the storage unit. It includes the error correction code (ECC) generator, SECDED, and partial write logic. On the model 855, the distributor is located in the central processor cabinet.

INPUT/OUTPUT UNIT

The IOU (figures 1-4 and 1-5) consists of the following:

- Ten logically independent peripheral processors (PPs). Options are available to increase total to 15 or 20 PPs.
- Internal interface to 12 I/O channels. 24-channel option is available.
- External interfaces to I/O channels
 - 11 or 23 CYBER 170 channel interfaces.
 - Display controller interface (CYBER 170 channel 10g).
 - Real-time clock interface (channel 14g).
 - Two-port multiplexer interface (channel 15g).
 - Maintenance channel interface (channel 17g).
- Interface to central memory.
- Bounds register to limit writes to CM.

- On the model 835, cache invalidation bus interface to CP.

The PPs are organized in groups of five, called barrels. The PPs in a barrel time-share common hardware. Each PP has its own independent memory, and communicates with all I/O channels and with central memory.

DISPLAY STATION

The display station provides a visual, alphanumeric readout for the computer. The receipt of symbol and position information from the computer enables displaying program information on a 21-inch cathode-ray tube (CRT). The station also contains an alphanumeric keyboard which enables an operator to send data to the computer. The keyboard and CRT combination permits the computer operator to modify computer programs and view the result on the screen. The computer outputs two alternate, nonrelated data streams. The display station keyboard has a switch which enables the operator to select either of the data streams or to select both for presentation on the CRT. Except for programming information in section 5, refer to the display station manual listed in the system publication index in the preface of this manual for further display station information.

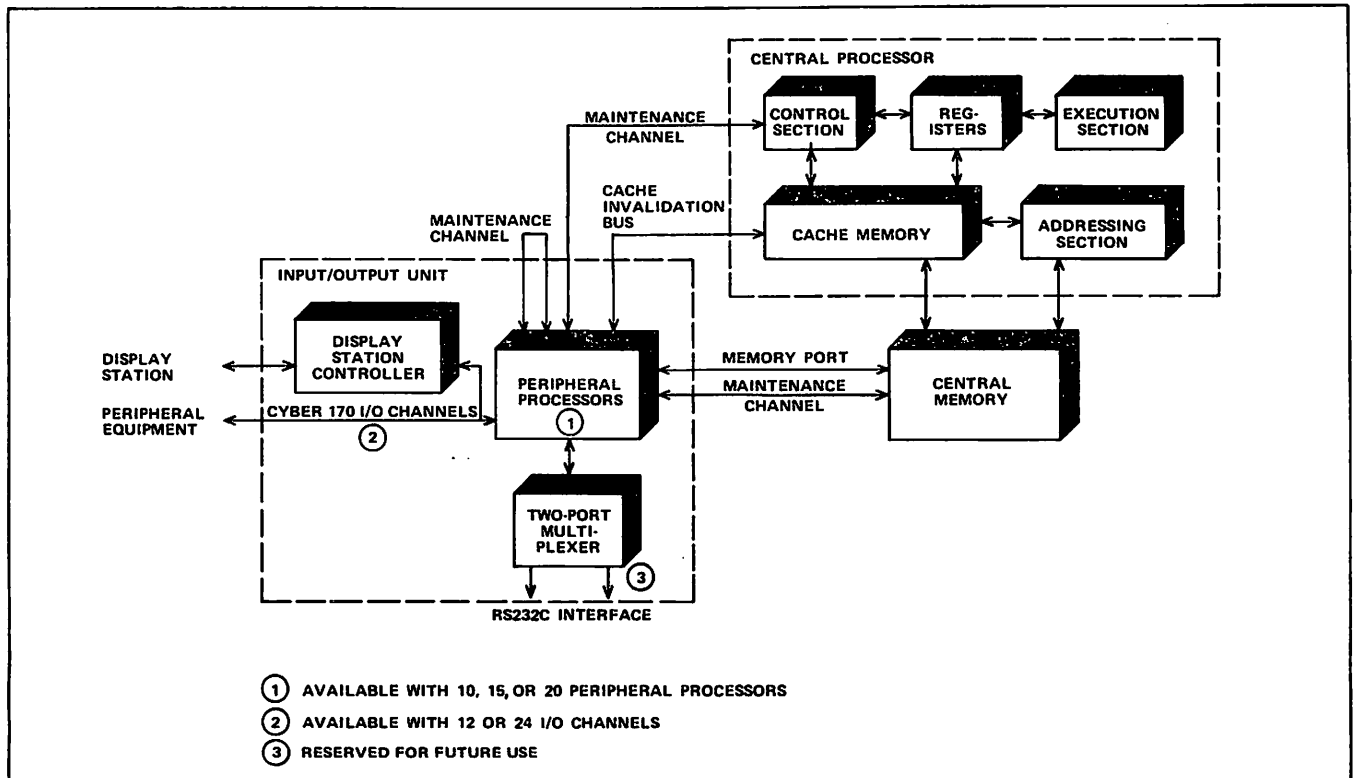
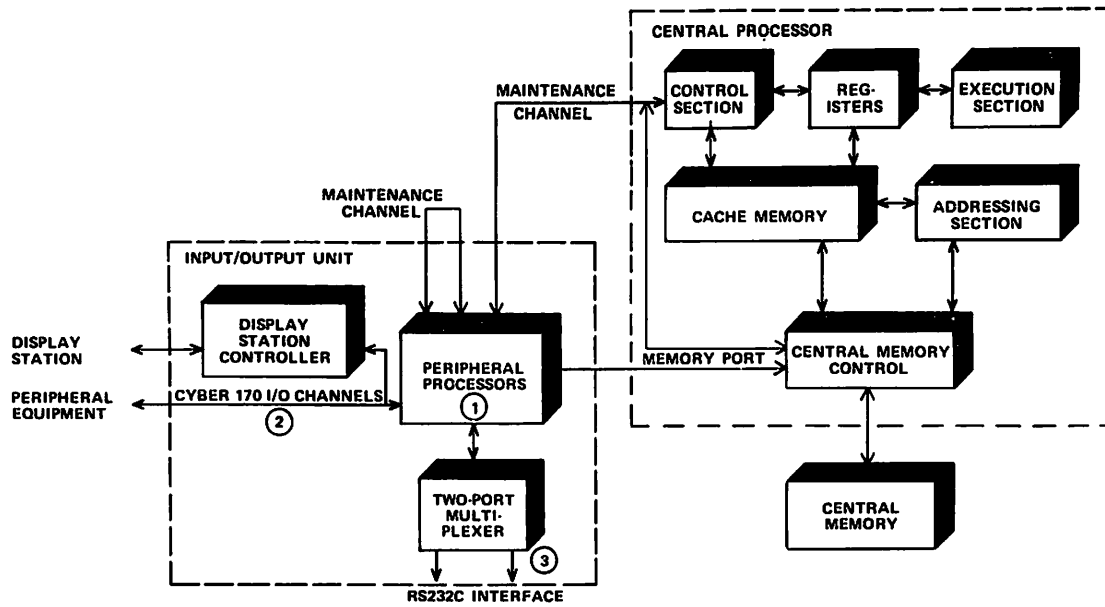


Figure 1-4. Model 835 Computer System



- ① AVAILABLE WITH 10, 15, OR 20 PERIPHERAL PROCESSORS
- ② AVAILABLE WITH 12 OR 24 I/O CHANNELS
- ③ RESERVED FOR FUTURE USE

Figure 1-5. Model 855 Computer System

This section provides functional descriptions of the central processor (CP), central memory (CM), and input/output unit (IOU) as shown in the block diagrams in section 1. Functional descriptions for the system display station and the cooling system are in their respective manuals listed in the system publication index in the preface of this manual.

CENTRAL PROCESSOR

The CP consists of the control section, registers, the execution section, cache memory, and the addressing section. The model 855 CP also includes central memory control.

CONTROL SECTION

The control section consists of logic for instruction control, 24 operating registers, and 8 support registers.

Model 835 Instruction Lookahead

The model 835 instruction lookahead hardware (ILH) prefetches instruction words to make the next instruction immediately available when the execution of the previous instruction is complete; for example, during conditional branch instructions. To accomplish this, ILH reads instructions from cache/CM into a three-word, first-in, first-out buffer.

When ILH detects a conditional branch, it reads two instruction words from cache/CM, starting at the target address, into a branch buffer, and holds them until the branch is resolved. If the branch takes place, the branch buffer contains the next two executable instruction words; if not, ILH purges the branch buffer and processing continues with the next instruction in the three-word buffer.

Model 855 Instruction Lookahead

The model 855 instruction lookahead hardware (ILH) prefetches a maximum of 12 instructions to make the next instruction immediately available when the execution of the previous instruction is complete. This is accomplished by reading instructions from cache/CM into a series of buffer ranks.

When ILH detects a conditional branch, it assumes that the branch condition will be met. ILH computes the branch target address and reads instructions from cache/CM starting at the target address. If the branch is taken, the buffer ranks contain the

next executable instruction words. If the branch is not taken, the hardware purges the buffer ranks and resumes prefetching at the instruction word following the unsatisfied branch instruction.

Maintenance Access Control

The maintenance access control performs initialization and maintenance operations in the CP.

Instruction Control Sequences

The instruction control section performs instruction translation and control sequences. Each control sequence obtains the necessary instruction operands from the operating registers and provides the control signals for execution. Instructions read from CM are 60-bit instruction words that are in four 15-bit groups, two 30-bit groups, or a combination of 15-bit and 30-bit groups. The 15-bit groups are termed parcels with the first parcel (parcel 0) being the highest-order 15 bits of a 60-bit CM word. Second, third, and fourth parcels (parcels 1, 2, and 3) follow in order. The 30-bit groups contain two 15-bit parcels.

The instruction control sequences control the execution of one or more instructions of a common type. These sequences and associated instructions are briefly described in this section. For further information, refer to CP Instruction Descriptions in section 4.

Boolean Sequence

The Boolean sequence controls instructions that require bit-by-bit data manipulation. This includes both the logical and transmissive operations. The instructions requiring logical operations are:

- 11 Logical product (Xj) and (Xk) to Xi $BX_i X_j * X_k$
- 12 Logical sum of (Xj) and (Xk) to Xi $BX_i X_j + X_k$
- 13 Logical difference of (Xj) and (Xk) to Xi $BX_i X_j - X_k$
- 15 Logical product of (Xj) with complement of (Xk) to Xi $BX_i -X_k * X_j$
- 16 Logical sum of (Xj) with complement of (Xk) to Xi $BX_i -X_k + X_j$
- 17 Logical difference of (Xj) with complement of (Xk) to Xi $BX_i -X_k - X_j$

The instructions requiring transmissive operations are:

10	Transmit (Xj) to Xi	BXi Xj
14	Transmit complement of (Xk) to Xi	BXi -Xk

Shift Sequence

The shift sequence controls instructions that require shifting the 60-bit field of data within the operand word. The shift instructions are:

20	Left shift (Xi) by jk	LXi jk
21	Right shift (Xi) by jk	AXi jk
22	Left shift (Xk) nominally (Bj) places to Xi	LXi Bj Xk
23	Right shift (Xk) nominally (Bj) places to Xi	AXi Bj Xk
43	Form mask of jk bits to Xi	MXi jk

The shift sequence also controls the pack and unpack instructions. In the packed floating format, the coefficient is contained in the lower 48 bits. The sign and biased exponents are contained in the upper 12 bits. The unpack instruction obtains the packed word from the Xk register, delivers the coefficient to the Xi register, and delivers the exponent to the Bj register. The unpack and pack instructions are:

26	Unpack (Xk) to Xi and Bj	UXi Bj Xk
27	Pack (Xk) and (Bj) to Xi	PXi Bj Xk

The shift sequence also controls the normalize operations. The coefficient portion of the operand is repositioned, and the exponent is adjusted so that the most significant bit of the coefficient is in the highest-order bit position of the coefficient, and the exponent is decreased by the number of bit positions shifted. The normalize instructions are:

24	Normalize (Xk) to Xi and Bj	NXi Bj Xk
25	Round normalize (Xk) to Xi and Bj	ZXi Bj Xk

Floating-Add Sequence

The floating-add sequence controls the operations necessary to form the 48-bit floating sum with a 12-bit exponent of the floating-point sum or difference of two floating-point operands. The floating-add instructions are:

30	Floating sum of (Xj) and (Xk) to Xi	FXi Xj + Xk
31	Floating difference of (Xj) and (Xk) to Xi	FXi Xj - Xk
32	Floating double-precision sum of (Xj) and (Xk) to Xi	DXi Xj + Xk

33	Floating double-precision difference of (Xj) and (Xk) to Xi	DXi Xj - Xk
34	Round floating sum of (Xj) and (Xk) to Xi	RXi Xj + Xk
35	Round floating difference of (Xj) and (Xk) to Xi	RXi Xj - Xk

Floating-Multiply and Floating-Divide Sequence

The floating-multiply and floating-divide sequence controls the operation of floating-multiply, floating-divide, and population-count instructions.

The multiply instructions are:

40	Floating product of (Xj) and (Xk) to Xi	FXi Xj * Xk
41	Round floating product of (Xj) and (Xk) to Xi	RXi Xj * Xk
42	Floating double-precision product of (Xj) and (Xk) to Xi	DXi Xj * Xk

The divide instructions are:

44	Floating divide (Xj) by (Xk) to Xi	FXi Xj/Xk
45	Round floating divide (Xj) by (Xk) to Xi	RXi Xj/Xk

The population-count instruction counts the number of one bits in a 60-bit operand. The instruction is:

47	Population count of (Xk) to Xi	CXi Xk
----	--------------------------------	--------

Increment Sequence

The increment sequence controls the one's complement addition and subtraction of 18-bit fixed-point operands for increment instructions 50 through 77. The sequence also controls the 60-bit one's complement sum and difference values for long-add instructions 36 and 37.

The increment instructions are:

50	Set Ai to (Aj) + K	SAi Aj K
51	Set Ai to (Bj) + K	SAi Bj K
52	Set Ai to (Xj) + K	SAi Xj K
53	Set Ai to (Xj) + (Bk)	SAi Xj + Bk
54	Set Ai to (Aj) + (Bk)	SAi Aj + Bk
55	Set Ai to (Aj) - (Bk)	SAi Aj - Bk
56	Set Ai to (Bj) + (Bk)	SAi Bj + Bk
57	Set Ai to (Bj) - (Bk)	SAi Bj - Bk
60	Set Bi to (Aj) + K	SBi Aj K

61	Set Bi to (Bj) + K	SBi Bj K
62	Set Bi to (Xj) + K	SBi Xj K
63	Set Bi to (Xj) + (Bk)	SBi Xj + Bk
64	Set Bi to (Aj) + (Bk)	SBi Aj + Bk
65	Set Bi to (Aj) - (Bk)	SBi Aj - Bk
66	Set Bi to (Bj) + (Bk)	SBi Bj + Bk
67	Set Bi to (Bj) - (Bk)	SBi Bj - Bk
70	Set Xi to (Aj) + K	SXi Aj K
71	Set Xi to (Bj) + K	SXi Bj K
72	Set Xi to (Xj) + K	SXi Xj K
73	Set Xi to (Xj) + (Bk)	SXi Xj + Bk
74	Set Xi to (Aj) + (Bk)	SXi Aj + Bk
75	Set Xi to (Aj) - (Bk)	SXi Aj - Bk
76	Set Xi to (Bj) + (Bk)	SXi Bj + Bk
77	Set Xi to (Bj) - (Bk)	SXi Bj - Bk

The long-add instructions are:

36	Integer sum of (Xj) and (Xk) to Xi	IXi Xj + Xk
37	Integer difference of (Xj) and (Xk) to Xi	IXi Xj - Xk

Compare/Move Sequence

The compare/move sequence controls data manipulation on a character basis. The compare/move instructions (also referred to as CMU instructions) are 60-bit instructions that use six support registers for source and result field CM addresses and character position offsets. The support registers load from the 60-bit instruction word. The compare/move instructions are:

464	Move indirect (Bj) + K	IM
465	Move direct	DM
466	Compare collated	CC
467	Compare uncollated	CU

The support registers are:

- An 18-bit K1 register that specifies which relative CM address word contains the first character of the source data field.
- An 18-bit K2 register that specifies which relative CM address word contains the first character of the result field.
- A 4-bit C1 register that specifies the character position or offset of the first CM word of the source field.

- A 4-bit C2 register that specifies the character position or offset of the first CM word of the result field.
- Two 16-bit L registers (LA and LC) that specify the number of characters in the data field. The LA register is associated with K1, and the LC register is associated with K2. Instruction 464 uses 14 register bits. Instructions 465, 466, and 467 use only the lower eight register bits.

NOTE

CMU instructions are provided for compatibility with previous systems. For better performance, recompile jobs to avoid use of CMU instructions.

CYBER 170 Exchange Sequence

The CYBER 170 exchange sequence is the method used to swap jobs in and out of execution. When a CYBER 170 exchange jump instruction occurs, the CYBER 170 exchange sequence writes the contents of the current job's CP registers (described later in this section) into an area of central memory called a CYBER 170 exchange package. A CYBER 170 exchange package is associated with each job. It contains sufficient information to restart a job if the job is interrupted during execution and swapped out by a CYBER 170 exchange jump. To complete the sequence, CP registers for another job are read from its CYBER 170 exchange package and that job begins or resumes execution. For further information, refer to CYBER 170 Exchange Jump in section 5.

Block Copy Sequence

The block copy sequence controls the transfer of data between CM and UEM. The number of words to be transferred is determined by the addition of K to the contents of Bj. The starting address for CM is formed by adding either the A0 register or certain bits of the X0 register to the RAC reference address. The starting address for UEM is formed by adding certain bits of the X0 register to the RAE reference address. The block copy instructions are:

011	Block copy Bj + K words from UEM to CM	REC Bj + K
012	Block copy Bj + K words from CM to UEM	WEC Bj + K

Direct Read/Write Sequence

Instructions 014 and 015 perform single word direct read and write operations for UEM, and instructions 660 and 670 perform single word direct read and write operations for central memory.

014	Read one word from UEM at (Xk + RAE) to Xj	RXj Xk
-----	--	--------

015	Write one word from Xj to UEM at (Xk + RAE)	WXj Xk
660	Read central memory at (Xk) to Xj	CRXj Xk
670	Write Xj into central memory at (Xk)	CWXj Xk

Normal Jump Sequence

The normal jump sequence controls the execution of branch instructions 02 through 07. The 02 instruction performs an unconditional jump to the Bi register address plus K. The branch address is K when i equals 0. The 02 instruction is:

02	Jump to (Bi) + K	JP
----	------------------	----

The conditional jump instructions 03 through 07 branch to address K if the jump condition is met. These instructions are:

030	Branch to K if (Xj) = 0	ZR
031	Branch to K if (Xj) ≠ 0	NZ
032	Branch to K if (Xj) is positive	PL
033	Branch to K if (Xj) is negative	NG
034	Branch to K if (Xj) is in range	IR
035	Branch to K if (Xj) is out of range	OR
036	Branch to K if (Xj) is definite	DF
037	Branch to K if (Xj) is indefinite	ID
04	Branch to K if (Bi) = (Bj)	EQ
05	Branch to K if (Bi) ≠ (Bj)	NE
06	Branch to K if (Bi) ≥ (Bj)	GE
07	Branch to K if (Bi) < (Bj)	LT

Return Jump Sequence

The return jump sequence controls the execution of three instructions.

00	Error exit to MA or program stop	PS
010	Return jump to K	RJ
013	Central exchange jump to (Bj) + K or monitor exchange jump to MA	XJ Bj + K

REGISTERS

The CP contains the operating and support registers described in the following paragraphs. The contents of these registers can be written into memory and reloaded from memory as a CYBER 170 exchange package by a single CP instruction (CYBER 170 exchange jump). Figure 2-1 shows the CYBER 170 exchange package.

The time a CYBER 170 exchange package resides in CP hardware is called an execution interval. During this interval, the contents of X, A, B, and P registers can be changed by CP instructions. The contents of other support registers change only as a result of a CYBER 170 exchange jump. For further information, refer to CYBER 170 Exchange Jump in section 5.

Operating Registers

The operating registers consist of operand (X), address (A), and index (B) registers. These registers minimize memory references for arithmetic operands and results.

X Registers

The CP contains eight 60-bit X registers, X0 through X7. The X0 register is used in the compare instructions to indicate if two fields of characters are equal. Also, the X0 register provides the relative UEM starting address in a block copy operation.

The X1 through X7 registers are primarily data handling registers for computation. X1 through X5 are used to input data from CM and X6 and X7 are used to transmit data to CM.

Operands and results transfer between CM and the X registers as a result of placing CM addresses into corresponding A registers.

A Registers

The CP contains eight 18-bit A registers, A0 through A7. The A0 register serves as an intermediate register for the user's discretion. The A0 register is used in the compare collate instruction for the collate table address. Also, the A0 register provides the relative CM starting address in a block copy operation.

The A1 through A7 registers are essentially CM operand address registers associated one-for-one with the X registers. Placing a quantity into an address register (A1 through A5) causes a CM read reference to that address and transmits the CM word to the corresponding X register (X1 through X5). Similarly, placing a quantity into the A6 or A7 register causes the word in the corresponding X6 or X7 register to be written into that relative address of CM.

B Registers

The CP contains eight 18-bit B registers, B0 through B7. These registers are primarily indexing registers to control program execution. Program

loop counts may also be incremented or decremented in these registers.

Program addresses may be modified on the way to an A register by adding or subtracting B register quantities. The B registers also hold shift counts for the nominal B_j shifts, the resultant exponent for the unpack, the operand exponent for the pack, and the resultant shift count from a normalize. The B0 register always contains positive zero which can be used as an operand. This register cannot hold results from instructions.

Support Registers

Eight support registers assist the operating registers during the execution of programs. The contents of the support registers are stored in CM, and their new contents are loaded from CM during a CYBER 170 exchange sequence. With the exception of the P register, the contents of the support registers cannot be altered during the execution interval of a CYBER 170 exchange package. When the execution interval completes, the data in the support registers is sent back to CM through a CYBER 170 exchange jump.

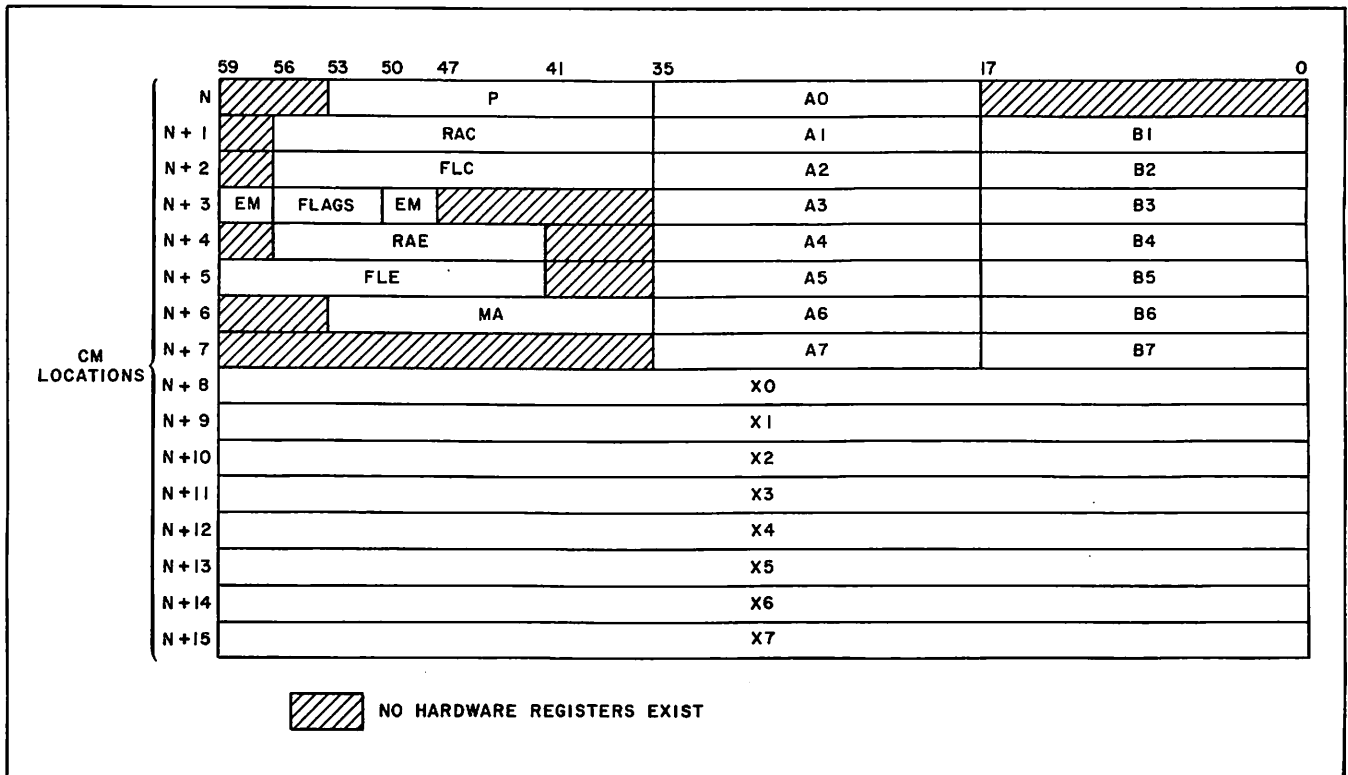


Figure 2-1. CYBER 170 Exchange Package

P Register

The 18-bit program address (P) register loads from CM during the first word of a CYBER 170 exchange sequence and contains the current program execution address. The register serves as a program address counter and holds the relative CM address for each program step.

RAC Register

The 21-bit CM reference address (RAC) register loads from CM during the second word of a CYBER 170 exchange sequence. An absolute CM address forms by adding RAC to a relative address determined by the instruction. The content of the P register is added to RAC to form the program address in CM. A P-equal-to-zero condition specifies relative address zero and, therefore, (RAC). This CM location is reserved for recording error exit conditions and should not be used to store data or instructions.

FLC Register

The 21-bit CM field length (FLC) register loads from CM during the third word of a CYBER 170 exchange sequence. The FLC register defines the size of the field of the program in execution. Relative CM addresses are compared with FLC to check that the program is not going out of its allocated memory range.

EM Register

The 6-bit exit mode (EM) register loads from CM during the fourth word of a CYBER 170 exchange sequence. The EM register holds 6 exit mode selection bits that control individual error conditions for a program. Selected EM register bits cause the CP to error exit when the corresponding conditions occur. Any or all of the 6 bits can be set at one time. Clear EM register bits allow the CP to continue, without error processing, when most of the corresponding conditions occur. Refer to the error exit tables under Error Response in section 5 for specific cases. The exit mode selection bits appear in the exchange package as bits 48 through 50, and 57 through 59. The bits and their corresponding conditions are:

<u>Mode Selection Bit</u>	<u>Significance</u>
48	Address out of range
49	Infinite operand
50	Indefinite operand
57	Hardware error
58	Hardware error
59	Hardware error

Flag Register

The 6-bit flag register loads from CM during the fourth word of a CYBER 170 exchange sequence. The flag register holds 6 bits that function as control flags.

<u>Bits</u>	<u>Condition</u>
51	Hardware error bit.
52	Instruction stack (lookahead) purge flag. If set, extended purging of instruction lookahead registers is enabled. For further information, refer to Instruction Lookahead Purge Control under CP Programming in section 5.
53	CMU interrupted flag. If set, one of instructions 464 through 467 has been interrupted. The information necessary to resume operation has been saved.
54	Block copy flag. If set, block copy instructions (011, 012) use bits 30 through 50 of XO rather than A0 to determine the CM address. For further information, refer to the descriptions of the block copy instructions in section 4.
55	Expanded addressing select flag. If set, UEM is operating in expanded addressing mode; if clear, UEM is operating in 24-bit standard addressing mode. This bit must be clear on the model 835. For further information, refer to Addressing Modes under Memory Programming in section 5.
56	UEM enable flag. If set, UEM is available. This flag must be set to allow 011, 012, 014, and 015 instructions to access UEM.

RAE Register

The 21-bit UEM reference address (RAE) register loads from CM during the fifth word of a CYBER 170 exchange sequence. The lower 6 bits of this register are always zero. An absolute UEM address forms by adding RAE to the relative address which is determined by the instruction.

FLE Register

The 24-bit UEM field length (FLE) register loads from CM during the sixth word of a CYBER 170 exchange sequence. The lower 6 bits of this register are always zero. The FLE register defines the size of the field in UEM for the program in execution. Relative UEM addresses are compared with FLE.

MA Register

The 18-bit monitor address (MA) register loads from CM during the seventh word of a CYBER 170 exchange sequence. The MA register contains the absolute starting address of an exchange package which is used when executing a central exchange jump (013) instruction with the CYBER 170 monitor flag set, or when honoring a monitor exchange jump to MA (262x) instruction with the CYBER 170 monitor flag clear. For further information, refer to CYBER 170 Exchange Jump in section 5.

EXECUTION SECTION

The execution section combines the operands into results, providing additional sequencing control where necessary.

CACHE MEMORY

Cache memory is a high-speed buffer memory which is transparent to the user. It reduces effective CM access time as follows. When the CP first reads CM, a block of four words from CM (containing the requested word) is read rapidly into cache memory. On subsequent reading of any of these words, CM need not be accessed when these words are in cache memory. Often this is the case because the same data is read more than once, or because a loop of instructions is repeatedly executed. Also, when CM words are read sequentially, there is often data lookahead. Cache memory is 2048 words or, optionally, 4096 words.

ADDRESSING SECTION

An address adder calculates memory addresses for data and unconditional jump instructions.

Memory management hardware verifies that memory addresses are to access permitted memory areas. If this is the case, this hardware accesses cache memory and, if necessary, central memory.

CENTRAL MEMORY CONTROL (MODEL 855 ONLY)

Central memory control (CMC) provides an interface to CM for the CP and IOU. On the model 855, it is physically located in the CP cabinet. CMC includes:

- Ports and distributor.
- SECEDED logic.
- Partial write logic.
- Memory control logic.
- Maintenance registers.

CENTRAL MEMORY

The CM performs the following functions.

- The eight memory banks store from 524K to 2097K of 64-bit words (the leftmost 4 bits are undefined) and an 8-bit SECEDED code.

- The two ports make CM accessible to the CP and every PP.
- A bounds register limits access to CM from either or both ports.
- The SECEDED generators generate the SECEDED code bits stored with each word. SECEDED checks circuits, corrects single-bit errors, and detects double-bit errors.
- The maintenance channel interface gives a PP in the IOU access to the CM maintenance registers for system initialization, corrective action, error reporting and diagnostics, and for setting the port bounds register.

ADDRESS FORMAT

Figure 2-2 illustrates the address format.

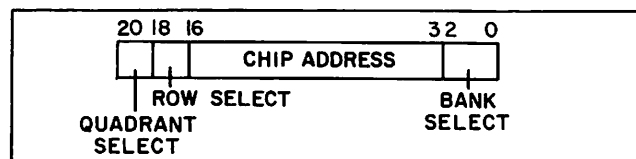


Figure 2-2. Address Format

The following list defines the address fields.

- Bank select specifies one of eight banks. Since the bank address is the lowest order 3 bits of the storage address, sequential addressing results in a phased-bank operation which allows a maximum data transfer rate of one word each clock period.
- Chip address specifies the address of one word in 16K MOS memory chips for the selected bank.
- Row select selects one of four word rows in a quadrant.
- Quadrant select selects one of four quadrants. It is used only for storage units larger than 524K.

CM ACCESS AND CYCLE TIMES

Model 835

The CM access time for a read operation is 672 nanoseconds.

One bank cycle is 8 clock periods (448 nanoseconds). Cycle time for a read or write operation is 448 nanoseconds (8 clock periods). Cycle time for a partial write (read/modify/write) is 896 nanoseconds (16 clock periods).

Model 855

The CM access time for a read operation is 528 nanoseconds.

One bank cycle is 7 clock periods (448 nanoseconds). Cycle time for a read or write operation is 448 nanoseconds (7 clock periods). Cycle time for a partial write (read/modify/write) is 896 nanoseconds (14 clock periods).

7. If multiple bit or other uncorrectable error, send uncorrectable error response code to CP or IOU. A PP in the IOU may then analyze the syndrome bits using the maintenance channel.

CM PORTS AND PRIORITIES

A priority network resolves access conflicts on a rotating basis, preventing long-term lockout of any port. In case of simultaneous requests, the CP has priority. The CM also has a refresh mechanism which may consume a maximum of 6 percent of memory time on a model 835 and 7 percent on a model 855. Refresh requests have priority over port requests. Refer to table 2-1 for maximum request lockout time.

Table 2-1. Port Priority

Maximum Request Lockout Time in Bank Cycles	
Port	Read or Write Requests
Refresh	1
Port 0	4
Port 1	5

NOTE: For a model 835, 1 bank cycle equals 8 clock periods which equals 448 nanoseconds. For a model 855, 1 bank cycle equals 7 clock periods which equals 448 nanoseconds.

CM LAYOUT

Central memory contains an area that is reserved for special software called executive state software. Along with the hardware and microcode, this software handles the operations of executive state as described in section 5. Executive state software is located at the higher end of memory. The remaining memory is available to the CYBER 170 state and may be allocated as central memory (accessible via RAC and FLC) or as unified extended memory (accessible via RAE, FLE, and the 011, 012, 014, and 015 instructions). Refer to figure 2-3.

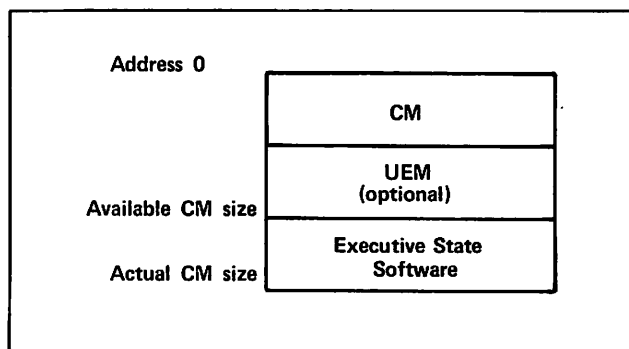


Figure 2-3. CM Layout

SECDED

The SECDED logic corrects single-bit errors during a CM read, permitting unimpeded computer operation. The SECDED logic prepares for the error correction by generating error correction code (ECC) bits for each data word, and by storing these ECC bits in CM with the data word during the CM write. Tables 2-2 (model 835) and 2-3 (model 855) list the hexadecimal codes for all the combinations of syndrome bits with the number of the data bit assigned each code or a note categorizing the code. Then, during a CM read, CM performs the following SECDED sequence.

1. Read one CM word and generate new ECC bits for data portion of CM word.
2. Compare new ECC bits with CM word ECC bits.
3. If old and new ECC bits match, no error exists. Send data to requesting unit.
4. If bits do not match, generate syndrome bits from result of ECC compare.
5. Decode syndrome bits to determine if single or multiple bit failure.
6. If single bit failure, correct by inverting failing bit in data word. Send corrected word to requesting unit.

CM BOUNDS REGISTER

The CM bounds register limits the write access to CM from specified ports. The ports are limited to the area between an upper and lower bound as specified in the CM bounds register. Bits in byte 0 specify the port(s) from which the write access is limited. The CM bounds register is set through the maintenance channel. For further information, refer to Maintenance Channel Programming in section 5.

CENTRAL MEMORY RECONFIGURATION

Central memory reconfiguration is a manually performed function that permits the computer operator to restructure the CM addresses so that a failing part of CM can be quickly locked out to provide a continuous block of usable CM. CM reconfiguration is accomplished by setting the switches on the memory unit to manipulate the upper address bits.

When a configuration switch is set forcing a CM address bit to a zero/one, the address range corresponding to the original installed memory accesses some parts of the reconfigured memory more than once. Addresses up to the rightmost forced bit, and half the addresses using the rightmost forced bit, cover a contiguous address space from location zero, which is the reconfigured memory. For further information, refer to section 3.

Table 2-2. Model 835 SECDED Syndrome Codes/Corrected Bits

Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit
00	⑦	20	66 ②	40	65 ②	60	③	80	64 ②	A0	③	C0	0/1 ⑥	E0	32 ①
01	71 ②	21	③	41	③	61	④	81	③	A1	④	C1	④	E1	32 ⑤
02	70 ②	22	③	42	③	62	④	82	③	A2	④	C2	④	E2	36 ⑤
03	6/7 ⑥	23	④	43	④	63	③	83	④	A3	③	C3	③	E3	36 ①
04	69 ②	24	③	44	③	64	④	84	③	A4	④	C4	④	E4	34 ⑤
05	③	25	④	45	④	65	③	85	④	A5	③	C5	③	E5	34 ①
06	③	26	④	46	④	66	③	86	④	A6	③	C6	③	E6	38 ①
07	24 ①	27	28 ⑤	47	26 ⑤	67	30 ①	87	25 ⑤	A7	29 ①	C7	27 ①	E7	31/38 ⑤
08	68 ②	28	③	48	③	68	④	88	③	A8	④	C8	④	E8	33 ⑤
09	③	29	④	49	④	69	③	89	④	A9	③	C9	③	E9	33 ①
0A	③	2A	④	4A	④	6A	③	8A	④	AA	③	CA	③	EA	37 ①
0B	16 ①	2B	20 ⑤	4B	18 ⑤	6B	22 ①	8B	17 ⑤	AB	21 ①	CB	19 ①	EB	23/37 ⑤
0C	4/5 ⑥	2C	④	4C	④	6C	③	8C	④	AC	③	CC	③	EC	35 ①
0D	8 ①	2D	12 ⑤	4D	10 ⑤	6D	14 ①	8D	9 ⑤	AD	13 ①	CD	11 ①	ED	15/35 ⑤
0E	0 ①	2E	4 ⑤	4E	2 ⑤	6E	6 ①	8E	1 ⑤	AE	5 ①	CE	3 ①	EE	7/39 ⑤
0F	③	2F	④	4F	④	6F	③	8F	④	AF	③	CF	③	EF	39 ①
10	67 ②	30	2/3 ⑥	50	③	70	56 ①	90	③	B0	48 ①	D0	40 ①	F0	③
11	③	31	④	51	④	71	56 ⑤	91	④	B1	48 ⑤	D1	40 ⑤	F1	④
12	③	32	④	52	④	72	60 ⑤	92	④	B2	52 ⑤	D2	44 ⑤	F2	④
13	④	33	③	53	③	73	60 ①	93	③	B3	52 ①	D3	44 ①	F3	③
14	③	34	④	54	④	74	58 ⑤	94	④	B4	50 ⑤	D4	42 ⑤	F4	④
15	④	35	③	55	③	75	58 ①	95	③	B5	50 ①	D5	42 ①	F5	③
16	④	36	③	56	③	76	62 ①	96	③	B6	54 ①	D6	46 ①	F6	③
17	24 ⑤	37	28 ①	57	26 ①	77	30/62 ⑤	97	25 ①	B7	29/54 ⑤	D7	27/46 ⑤	F7	③
18	③	38	④	58	④	78	57 ⑤	98	④	B8	49 ⑤	D8	41 ⑤	F8	④
19	④	39	③	59	③	79	57 ①	99	③	B9	49 ①	D9	41 ①	F9	③
1A	④	3A	③	5A	③	7A	61 ①	9A	③	BA	53 ①	DA	45 ①	FA	③
1B	16 ⑤	3B	20 ①	5B	18 ①	7B	22/61 ⑤	9B	17 ①	BB	21/53 ④	DB	19/45 ⑤	FB	23 ①
1C	④	3C	③	5C	③	7C	59 ①	9C	③	BC	51 ①	DC	43 ①	FC	③
1D	8 ⑤	3D	12 ①	5D	10 ①	7D	14/59 ⑤	9D	9 ①	BD	13/51 ⑤	DD	11/43 ⑤	FD	15 ①
1E	0 ⑤	3E	4 ①	5E	2 ①	7E	6/63 ⑤	9E	1 ①	BE	5/55 ⑤	DE	3/47 ⑤	FE	7 ①
1F	④	3F	③	5F	③	7F	63 ①	9F	③	BF	55 ①	DF	47 ①	FF	③

Notes:

- ① Corrected single-bit error.
- ② Syndrome code bit failed (single code bit set).
- ③ Double error or multiple error (even number of code bits set).
- ④ Multiple error reported as a single error.
- ⑤ Double error or multiple error with indicated bit(s) inverted.
- ⑥ Double error or multiple error or forced double error due to a partial write parity error on one of the two bytes.
- ⑦ No error detected.

Table 2-3. Model 855 SECCED Syndrome Codes/Corrected Bits

Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit
00	⑥	20	66 ②	40	65 ②	60	③	80	64 ②	A0	③	C0	0/1 ⑤	E0	32 ①
01	71 ②	21	③	41	③	61	④	81	③	A1	④	C1	④	E1	⑤
02	70 ②	22	③	42	③	62	④	82	③	A2	④	C2	④	E2	⑤
03	6/7 ⑤	23	④	43	④	63	③	83	④	A3	③	C3	③	E3	36 ①
04	69 ②	24	③	44	③	64	④	84	③	A4	④	C4	④	E4	⑤
05	③	25	④	45	④	65	③	85	④	A5	③	C5	③	E5	34 ①
06	③	26	④	46	④	66	③	86	④	A6	③	C6	③	E6	38 ①
07	24 ①	27	⑤	47	⑤	67	30 ①	87	⑤	A7	29 ①	C7	27 ①	E7	⑤
08	68 ②	28	③	48	③	68	④	88	③	A8	④	C8	④	E8	⑤
09	③	29	④	49	④	69	③	89	④	A9	③	C9	③	E9	33 ①
0A	③	2A	④	4A	④	6A	③	8A	④	AA	③	CA	③	EA	37 ①
0B	16 ①	2B	⑤	4B	⑤	6B	22 ①	8B	⑤	AB	21 ①	CB	19 ①	EB	⑤
0C	4/5 ⑤	2C	④	4C	④	6C	③	8C	④	AC	③	CC	③	EC	35 ①
0D	8 ①	2D	⑤	4D	10 ⑤	6D	14 ①	8D	⑤	AD	13 ①	CD	11 ①	ED	⑤
0E	0 ①	2E	4 ⑤	4E	⑤	6E	6 ①	8E	⑤	AE	5 ①	CE	3 ①	EE	⑤
0F	③	2F	④	4F	④	6F	③	8F	④	AF	③	CF	③	EF	39 ①
10	67 ②	30	2/3 ⑤	50	③	70	56 ①	90	③	B0	48 ①	D0	40 ①	F0	③
11	③	31	④	51	④	71	⑤	91	④	B1	⑤	D1	⑤	F1	④
12	③	32	④	52	④	72	⑤	92	④	B2	⑤	D2	⑤	F2	④
13	④	33	③	53	③	73	60 ①	93	③	B3	52 ①	D3	44 ①	F3	③
14	③	34	④	54	④	74	⑤	94	④	B4	⑤	D4	⑤	F4	④
15	④	35	③	55	③	75	58 ①	95	③	B5	50 ①	D5	42 ①	F5	③
16	④	36	③	56	③	76	62 ①	96	③	B6	54 ①	D6	46 ①	F6	③
17	⑤	37	28 ①	57	26 ①	77	⑤	97	25 ①	B7	⑤	D7	⑤	F7	31 ①
18	③	38	④	58	④	78	⑤	98	④	B8	⑤	D8	⑤	F8	④
19	④	39	③	59	③	79	57 ①	99	③	B9	49 ①	D9	41 ①	F9	③
1A	④	3A	③	5A	③	7A	61 ①	9A	③	BA	53 ①	DA	45 ①	FA	③
1B	⑤	3B	20 ①	5B	18 ①	7B	⑤	9B	17 ①	BB	⑤	DB	⑤	FB	23 ①
1C	④	3C	③	5C	③	7C	59 ①	9C	③	BC	51 ①	DC	43 ①	FC	③
1D	⑤	3D	12 ①	5D	10 ①	7D	⑤	9D	9 ①	BD	⑤	DD	⑤	FD	15 ①
1E	⑤	3E	4 ①	5E	2 ①	7E	⑤	9E	1 ①	BE	⑤	DE	⑤	FE	7 ①
1F	④	3F	③	5F	③	7F	63 ①	9F	③	BF	55 ①	DF	47 ①	FF	③

Notes:

- ① Corrected single-bit error.
- ② Syndrome code bit failed (single code bit set).
- ③ Double error or multiple error (even number of code bits set).
- ④ Multiple error reported as a single error.
- ⑤ Double error or multiple error or forced double error due to a partial write parity error on one of the two bytes indicated.
- ⑥ No error detected.

INPUT/OUTPUT UNIT

The input/output unit (IOU) performs the functions required to locate, select, and initialize the external devices connected to the system, and controls the transfer of data between a selected device and CM. The IOU also performs system maintenance functions.

The IOU contains the following functional areas.

- Peripheral processor (PP).
- I/O channels.
- Real-time clock.
- Two-port multiplexer.
- Maintenance channel.
- CM access.

PERIPHERAL PROCESSOR

The basic IOU contains 10 PPs and can be expanded to 20 PPs in 5-PP increments. Each PP is a logically independent computer with its own memory. Each 5-PP group is organized into a multiplexing system which allows the PPs to share common hardware for arithmetic, logical, and I/O operations without losing independence. This multiplexing system comprises five ranks of registers termed a barrel. Each rank contains information related to the instruction being executed by one PP.

Each PP can communicate with the CP by issuing a CYBER 170 exchange request to a specific CYBER 170 exchange package associated with the issuing PP. In addition, a PP can also communicate with the CP via CM read and write operations. PPs can communicate with each other over the I/O channels and through CM.

Each PP executes programs alone or with other PPs to control data transfers between external devices and CM. These programs are comprised of instructions from the IOU instruction set and respond to requests issued through CM by the operating system. The programs translate generalized operating system requests into control functions for accessing the external devices and may also perform device scheduling and optimization. The programs use PP memory as a buffer for the data transfer between external devices and CM to isolate IOU data transfer from variations in CM transfer rate.

Deadstart

A deadstart sequence allows the IOU to initialize itself. This deadstart sequence is initiated by the DEAD START switch on the deadstart panel or the DEAD START switch on the display station. The panel includes controls for assigning any PPM to PPO. For further information, refer to section 3.

Barrel and Slot

The barrel consists of the R, A, P, Q, and K registers, each one of which has five ranks 0 through 4 (refer to figure 2-4). Information in these registers moves from one rank to the next at a uniform 20-megahertz rate, providing a multiplexed system of five PPs, each operating at a 4-megahertz rate. The registers are stationary while the PPs rotate. For example, rank 4 registers contain PP0, PP1, PP2, PP3, and PP4 in succession, each consuming 50 nanoseconds of the total cycle time of 250 nanoseconds. Since PP memories operate at a slower rate, independent memory with its own address and data registers is provided for each PP.

Each time data enters the slot, a portion of the instruction for that data is executed. The slot performs tasks such as arithmetic and logic operations and program address manipulation. Complete execution of an instruction may require the R, A, P, Q, and K register quantities to go more than one trip around the barrel and through the slot.

The PPM may be referenced once each time the PP passes around the barrel and through the slot. During its slot time, the PP may also communicate with CM or with any of the I/O channels.

PP Registers

R Register

The 28-bit R register, in conjunction with the A register, forms an absolute CM address for CM read/write instructions. When bit 17 of the A register is set, the absolute CM address is formed by appending six zeros to the lower end of the contents of the R register and adding to the result bits 0 through 16 of the contents of the A register (refer to figure 2-5).

A Register

The 18-bit A register holds one operand for arithmetic, logic, or selected I/O operations. The content of A may be an arithmetic operand, CM address, I/O function, or I/O data word. Various instructions operate on 6, 12, or 18 bits of the A register.

When the A register is used as the CM address, parity is generated for transmission with the address to memory control. At deadstart, the A register is set to 10000 (octal). When bit 17 of the A register is clear, the A register is used as the CM address; however, when bit 17 is set, the R register is added to the A register (as described in the R register description) to obtain the absolute CM address for CM read/write instructions.

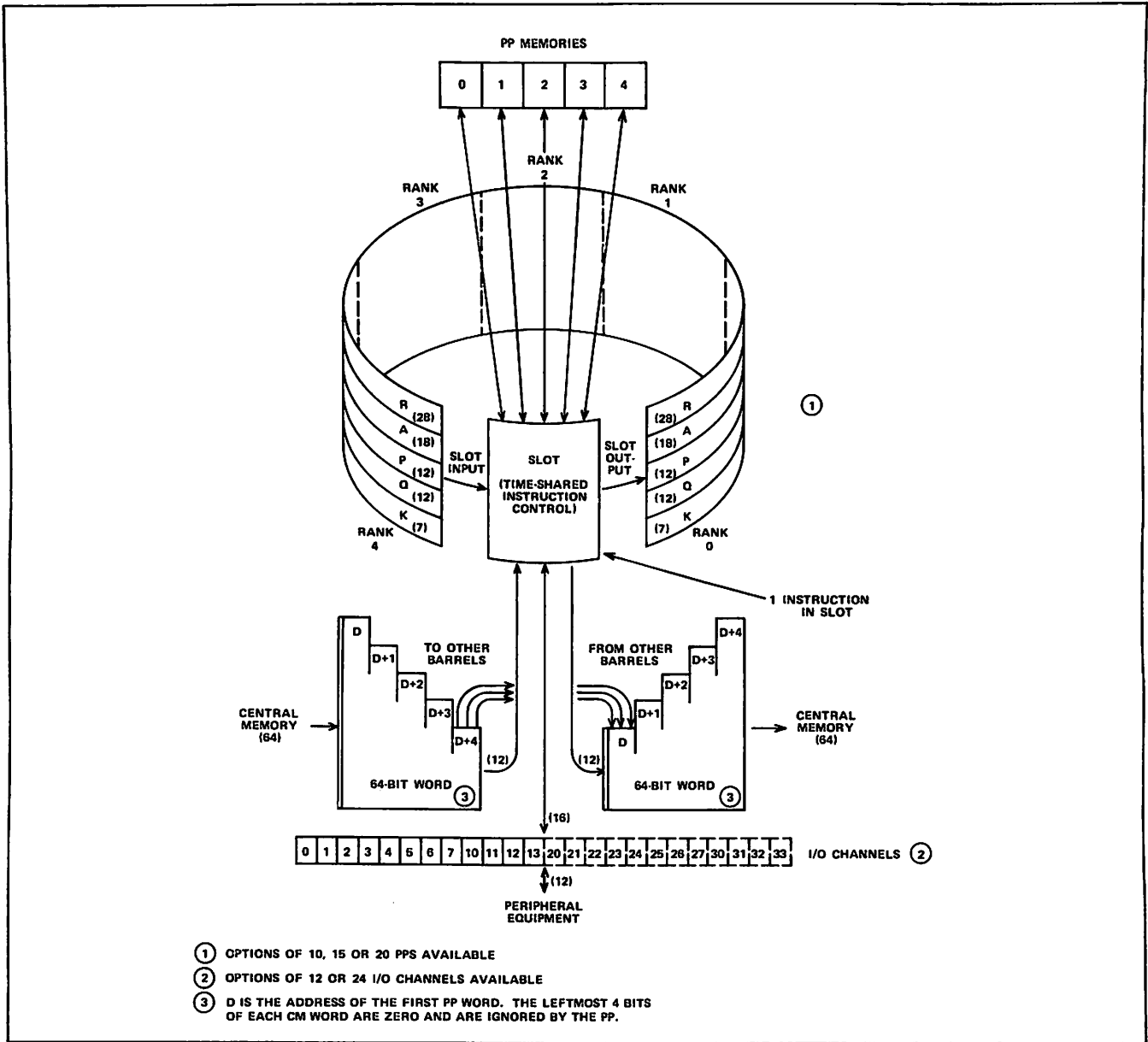


Figure 2-4. Barrel and Slot

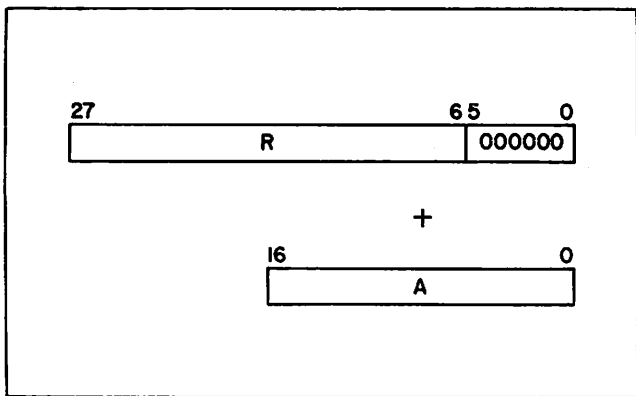


Figure 2-5. Formation of Absolute CM Address

P Register

The 12-bit P register is the program address register, except during the execution of instructions 61, 63, 71, and 73. For these instructions, the P register contains the PPM address of the data transfer. At deadstart, the P register is set to zero.

Q Register

The 12-bit Q register holds data for several functions such as the address of the operand during direct addressing and indirect addressing, peripheral address of data used during one-word central read or write instructions, upper 6 bits during constant mode instructions, channel number on all I/O and channel instructions, shift count, and

relative jump designator. At deadstart, each rank of the Q register is set to a corresponding PP number. Rank 0 is set to PPO, rank 2 is set to PP2, and so on.

K Register

The 7-bit K-register is visible to the programmer through the maintenance channel only. This register holds the operation code field of an instruction for display on the IOU deadstart panel and for deadstart panel interrogation. When a PP is halted (idled), this register contains all ones.

PP Numbering

PPs are numbered as follows:

<u>Barrel</u>	<u>PPs</u>
0	00 to 04
1	05 to 11 (octal)
2	20 to 24 (octal)
3	25 to 31 (octal)

The deadstart sequence decodes deadstart panel switches to determine PP numbering within a barrel. The sequence assigns barrel numbers according to the switch settings and, during the first minor cycle after deadstart, loads a zero into the Q register in barrel 0. This defines all the data in that rank of the barrel as belonging to PPO and since Q is the channel selector, assigns PPO to channel 0. During the next minor cycle, Q loads with a 1. This defines PP1 and assigns it to channel 1. This process occurs in parallel in all barrels until the IOU assigns each rank of each barrel with a PP number and a channel number. Reassignment can be done only during a deadstart.

PP Memory

Each PP has an independent 4K word memory; each word contains 16 data bits with the upper 4 bits set to zero, and 1 parity bit. PPO reads the deadstart program from the deadstart panel during the deadstart operation. Therefore, PP memory 0 must be operational. A PP memory reconfiguration feature allows the user to restore IOU operation if the IOU detects a fault in the PP memory normally assigned to PPO.

To reconfigure, the operator assigns a good PP memory to PPO and the operating system removes the failing PP memory. Computer operation can continue without the failing PP memory, and repairs can be made during scheduled maintenance. The system must be deadstarted to reconfigure PPMs.

I/O CHANNELS

The I/O channels are comprised of an internal interface that allows common hardware and software to control the external devices, and an external interface that allows the IOU to communicate with the external devices using 12-bit data channels. The internal interface can transfer 16-bit data words between two PPs, or between a PP and an

external device at a maximum rate of one word every 250 nanoseconds. This rate can be sustained for the maximum practical channel transfer (4096 words). During transfers between PPs, if the PPs are in the slot at the same time, the transfer rate is 500 nanoseconds.

Any PP can access any of the CYBER 170 bidirectional I/O channels. All PPs communicate with external devices through the independent I/O channels. Each channel may be connected to one or more pieces of external equipment, but only one piece of equipment can use a channel at one time. All channels can be active simultaneously.

The display station controller (DSC) is attached to CYBER 170 channel 10_g. The DSC is the IOU interface between the PPs and the display station, servicing both the keyboard and the cathode-ray tube (CRT). It transmits function words and digital symbol size/position data to the display station, and receives digital character codes from the keyboard. It also receives digital symbol codes from the PPs and converts these to analog signals to the CRT.

REAL-TIME CLOCK

The real-time clock is a 12-bit free-running counter, incrementing at a 1-megahertz rate. It is permanently attached to channel 14_g. This channel may be read at any time as its active and full flags are always set.

TWO-PORT MULTIPLEXER

The two-port multiplexer provides communication capability between a PP and two attached terminals. One port is reserved for maintenance purposes and the other port is reserved for future use. The two-port multiplexer is permanently attached to channel 15_g.

MAINTENANCE CHANNEL

The maintenance channel is used for initialization of the CP and CM maintenance registers and monitoring of error status.

The maintenance channel consists of the maintenance channel interface on channel 17_g, a maintenance access control in each system element, and a set of interconnecting cables.

CENTRAL MEMORY ACCESS

Any PP can access CM. During a write from the IOU to CM, the IOU assembles five successive 12-bit PP words into a 64-bit CM word with the leftmost four bits undefined. During a CM read, the IOU disassembles the rightmost 60 bits of the 64-bit CM word into five PP words. To find the CM address, a PP reads the A register. If bit 17 of the A register is clear, the PP uses the contents of the A register for the CM address. If bit 17 of the A register is set, the PP adds the relocation address from the R register to the A register to form the CM address.

A maximum of 20 PPs in various stages of assembly/disassembly can simultaneously read CM words, and five PPs can write CM words.

1. The first part of the document discusses the importance of maintaining accurate records of all transactions. It emphasizes that proper record-keeping is essential for the integrity of the financial system and for the ability to detect and prevent fraud.

2. The second part of the document outlines the various methods used to collect and analyze data. It describes the use of statistical techniques to identify trends and anomalies in the data, and the importance of using reliable sources of information.

3. The third part of the document discusses the role of the auditor in the process. It highlights the need for the auditor to maintain independence and objectivity, and to follow a systematic approach to the audit process.

4. The fourth part of the document describes the various types of audits that can be performed. It includes a discussion of the differences between internal and external audits, and the specific objectives of each type of audit.

5. The fifth part of the document discusses the importance of communication in the audit process. It emphasizes the need for the auditor to communicate clearly and effectively with the client, and to provide a comprehensive report of the findings of the audit.

6. The sixth part of the document discusses the various factors that can affect the quality of the audit. It includes a discussion of the importance of the auditor's qualifications and experience, and the need for the auditor to maintain a high level of professional skepticism.

7. The seventh part of the document discusses the various risks that are associated with the audit process. It includes a discussion of the risk of the auditor failing to detect a material misstatement, and the risk of the auditor being accused of fraud.

8. The eighth part of the document discusses the various ways in which the audit process can be improved. It includes a discussion of the importance of continuing education for auditors, and the need for the audit firm to maintain a strong quality control system.

9. The ninth part of the document discusses the various ways in which the audit process can be made more efficient. It includes a discussion of the use of technology in the audit process, and the importance of streamlining the audit process to reduce costs and improve the quality of the audit.

10. The tenth part of the document discusses the various ways in which the audit process can be made more transparent. It includes a discussion of the importance of providing the client with a clear and concise report of the findings of the audit, and the need for the auditor to be open and honest in the audit process.

11. The eleventh part of the document discusses the various ways in which the audit process can be made more effective. It includes a discussion of the importance of the auditor's communication skills, and the need for the auditor to be able to identify and solve problems during the audit process.

12. The twelfth part of the document discusses the various ways in which the audit process can be made more reliable. It includes a discussion of the importance of the auditor's adherence to the audit standards, and the need for the auditor to maintain a high level of professional competence.

13. The thirteenth part of the document discusses the various ways in which the audit process can be made more consistent. It includes a discussion of the importance of the auditor's use of the same audit procedures for all audits, and the need for the auditor to maintain a high level of consistency in the audit process.

14. The fourteenth part of the document discusses the various ways in which the audit process can be made more comprehensive. It includes a discussion of the importance of the auditor's coverage of all areas of the client's financial statements, and the need for the auditor to maintain a high level of thoroughness in the audit process.

15. The fifteenth part of the document discusses the various ways in which the audit process can be made more effective. It includes a discussion of the importance of the auditor's communication skills, and the need for the auditor to be able to identify and solve problems during the audit process.

16. The sixteenth part of the document discusses the various ways in which the audit process can be made more reliable. It includes a discussion of the importance of the auditor's adherence to the audit standards, and the need for the auditor to maintain a high level of professional competence.

17. The seventeenth part of the document discusses the various ways in which the audit process can be made more consistent. It includes a discussion of the importance of the auditor's use of the same audit procedures for all audits, and the need for the auditor to maintain a high level of consistency in the audit process.

This section describes mainframe controls and indicators and the operating procedures which are hardware-dependent. Software-dependent procedures are in system software reference manuals listed in the preface.

CONTROLS AND INDICATORS

This section describes IOU deadstart panel controls and indicators and CM configuration switches used by the system operator. Other controls used by maintenance personnel are described in the hardware operator's guide and the hardware maintenance manuals of the power distribution and warning system, the cooling system, and the display console listed in the system publication index.

DEADSTART PANEL CONTROLS/INDICATORS

The deadstart panel (figure 3-1) is in the IOU. It contains PP register selection and display facilities, deadstart controls, error indicators, and a switch matrix, which is the source for a short

PP program for initialization or troubleshooting. The switches, indicators, and their functions are listed in table 3-1.

CENTRAL MEMORY CONTROLS

The CM for both models contains four three-position configuration switches (figure 3-2). On the model 835, these switches are located along the edge of a

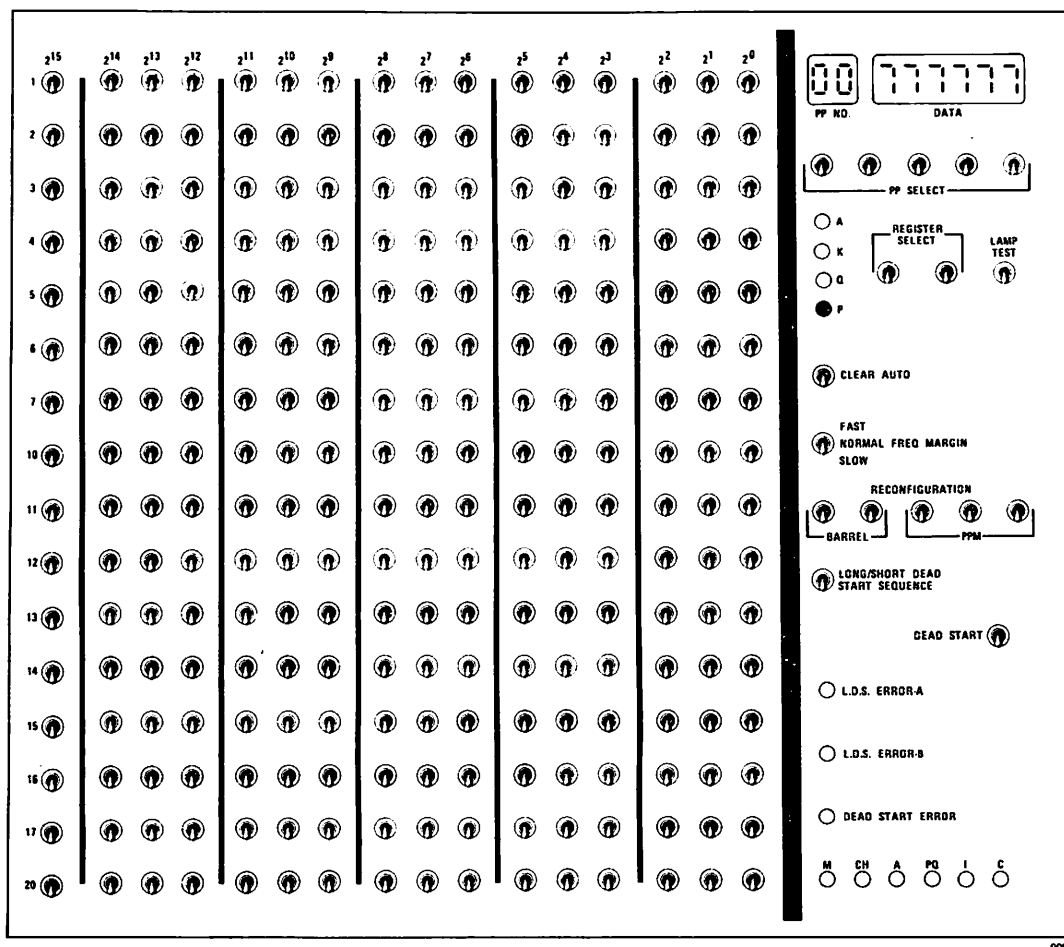


Figure 3-1. Deadstart Panel

printed circuit board located just to the right of the center post in the middle section of the memory cabinet (location D01). On the model 855, these switches are located along the edge of a printed circuit board located just to the right of the center post in the lower section of the memory cabinet (location F04). The switches are used to eliminate CM sections with malfunctions. Each switch, SW3 through SW6, forces one corresponding CM address bit, 23 through 20, either to a zero (switch down) or to a one (switch up). Refer to table 3-2.

In case of CM malfunctions, the remaining good memory can be reconfigured so it is accessible by contiguous addresses from zero to the maximum remaining address. This is accomplished by setting configuration switches (figure 3-2) as listed in table 3-2. Refer to the hardware operator's guide listed in the system publication index.

Table 3-1. Deadstart Panel Controls/Indicators (Sheet 1 of 2)

Panel Nomenclature	Description	Function
2 ⁰ through 2 ¹⁵ by 1 through 20 _g	Toggle switch matrix (two-position switches)	Provides a 16-word deadstart program for PP0. Switches 2 ⁰ through 2 ¹¹ set 12 bits for each of the program words, labeled 1 through 20 (octal). Switches 2 ¹² through 2 ¹⁵ are set to zero. Up position sets bit. Down position clears bit.
PP NO	Octal display	Shows the PP selected by PP SELECT switches.
DATA	Octal display	Shows the content of the register selected by REGISTER SELECT switches.
PP SELECT	Toggle switches (two-position)	Selects the PP whose register is to be displayed.
REGISTER SELECT	Toggle switches (two-position)	Selects the register to be displayed (00 = P, 01 = Q, 10 = K, 11 = A).
A, K, Q, P	Indicators	One of these lights to indicate which register is selected by REGISTER SELECT switches.
LAMP TEST	Toggle switch (two-position)	Lights all indicators and display segments.
CLEAR AUTO	Toggle switch (two-position)	Allows manual clearing of auto mode bit (bit 34 of the environment control register) to override possible auto mode selection. This allows the selection of the PP and register from the deadstart panel if bit 34 is set.
FREQ MARGIN	Toggle switch (three-position)	Determines the frequency margin selected (FAST/NORMAL/SLOW). The setting of this switch is sensed only at deadstart time.
RECONFIGURATION, BARREL	Toggle switches (two-position)	Selects the physical barrel which is logical barrel 0. All the other logical barrels are numbered from the selected physical barrel circularly. (If physical barrel 1 is selected by the switches, physical barrel 2 is logical barrel 1, and so on.)
RECONFIGURATION, PPM	Toggle switches (two-position)	Selects the physical PP memory which is logical PPM0. All the other PPMs in all barrels are numbered from the selected physical PPM circularly. If the switches are set to a value greater than four, no reconfiguration takes place.

Table 3-1. Deadstart Panel Controls/Indicators (Sheet 2 of 2)

Panel Nomenclature	Description	Function
LONG/SHORT DEAD START SEQUENCE	Toggle switch (two-position)	Selects the LONG/SHORT deadstart sequence. The setting of this switch is sensed only at deadstart.
DEAD START	Toggle switch (three-position, center is off)	Selects the fast or slow repetitive deadstart, which generates a master clear pulse every 250 or 4000 microseconds respectively. Up position selects fast deadstart; down position selects slow deadstart. (The single deadstart control pushbutton is on the display console.)
L.D.S. ERROR-A	Indicator	Remains lit when long deadstart branch tests are not completed within 10.25 microseconds.
L.D.S. ERROR-B	Indicator	Remains lit when a long deadstart sequence does not go to completion.
DEAD START ERROR	Indicator	Lights in case of long deadstart ROM address/data parity error.
M, CH, A, PQ, I, C	Indicators	Lights in case of hardware failures as follows: M: PP memory failure CH: I/O channel failure A: A barrel failure PQ: P or Q barrel failure I: Firmware or control failure C: 12/16 conversion failure

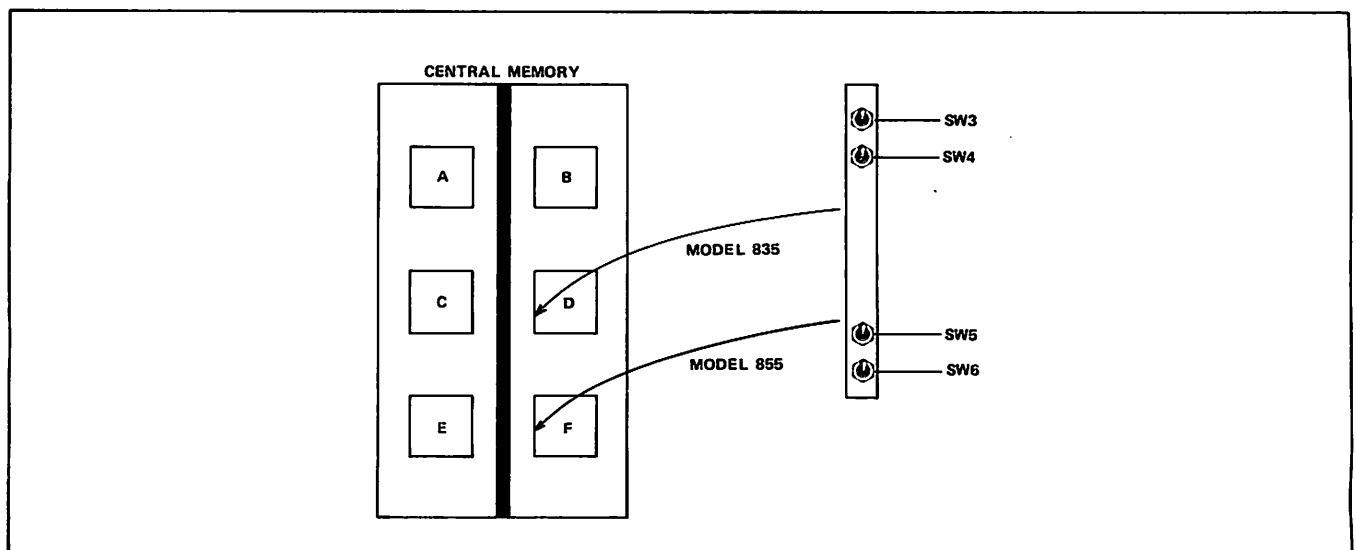


Figure 3-2. CM Configuration Switches

Table 3-2. Central Memory Reconfiguration

Original CM		Reconfigured CM								
Size	Address Range	Size	Location of Failing CM				Reconfiguration Setting			
Words		Words	Bit 23	Bit 22	Bit 21	Bit 20	SW3	SW4	SW5	SW6
524K	0-1 777 777	262K			0	X	-	-	U	-
		262K			1	X	-	-	D	-
1049K	0-3 777 777	524K		0	X	X	-	U	-	-
		524K		1	X	X	-	D	-	-
1573K	0-5 777 777	524K	0	0	X	X	-	U	-	-
		524K	0	1	X	X	-	D	-	-
		1049K	1	0	X	X	D	-	-	-
2097K	0-7 777 777	1049K	0	X	X	X	U	-	-	-
		1049K	1	X	X	X	D	-	-	-
		(1)				(2)				

NOTES:

- (1) CM remaining can be further reconfigured by setting additional configuration switches.
- (2) U equals up, D equals down, and - equals center position.

POWER-ON AND POWER-OFF PROCEDURES

In case of an emergency, use the system EMERGENCY OFF switch. The power-on and power-off procedures are described in the hardware operator's guide listed in the system publication index.

CAUTION

Improper application or removal of power may damage system circuits and/or air conditioning system. Power must be turned on/off by designated personnel only, except for the system EMERGENCY OFF switch. Use only for extreme emergency, not for normal shutdown.

OPERATING PROCEDURES

Refer to the hardware operator's guide listed in the system publication index. The system is initialized by setting its control switches, and then by running either a long or short deadstart sequence (defined later in this section). After initialization, the keyboard is used to instruct the system further, under program control.

CONTROL CHECKS

Before activating a long or short deadstart sequence, check the positions of deadstart panel switches against their intended use. These checks can be made by using table 3-1. The normal settings of these switches is as follows:

<u>Switch</u>	<u>Position</u>
CLEAR AUTO	Down
FREQ MARGIN	Center
RECONFIGURATION	All down
LONG/SHORT DEAD START SEQUENCE	Down for a short deadstart sequence
DEAD START	Center
All error lights	Not lit

Deadstart Sequences

In response to a deadstart signal from either the deadstart pushbutton on the display console, or from

the DEAD START switch on the deadstart panel, circuits in the IOU perform a deadstart sequence. Depending on the setting of the LONG/SHORT DEAD START SEQUENCE switch on the deadstart panel, either the long or the short deadstart sequence is performed. The short deadstart sequence is used when hardware integrity verification is not required. The long deadstart sequence performs all the tasks performed by the short deadstart sequence and some additional tasks. The main additional task is the running of a diagnostic program, from a read-only memory (ROM) in the IOU, on logical PPO. The diagnostic program takes approximately one minute to run.

Both deadstart sequences begin with a master clear which sets up all PPs, except logical PPO, for a 4096-word block input starting at PP location 0. The input into each PP is from the channel with the same number as the logical number of the PP concerned. The master clear also resets all external devices and sets maintenance channel connect code bit 52. The individual channels and registers are set as follows:

<u>Channel</u>	<u>Active/ Inactive Flag</u>	<u>Full/ Empty Flag</u>	<u>Channel Flag</u>	<u>Channel Error Flag</u>
0	Inactive	Empty	Clear	Clear
10 (display controller)	Active	Empty	Clear	Clear
14 (real-time clock)	Active	Full	Set	Set
15 (two-port mux)	Active	Empty	Clear	Clear
17 (maintenance)	Active	Empty	Clear	Clear
Other installed channels	Active	Empty	Clear	Clear
Noninstalled channels	Inactive	Empty	Clear	Clear

The flags of channel 14 and of noninstalled channels are fixed by hardware and cannot be changed.

<u>Register</u>	<u>Initialization</u>	<u>Description</u>
K	007100 _g	Instruction display on deadstart panel
P	007777 _g	Causes block input to start from location 0
A	10,000 _g	Count of 4096 words
Q	0, 1, 2...	I/O channel numbers (PPO: 0, PPI: 1, and so on)

All registers in all barrels are set to these values, except the registers of PPO.

If the long deadstart sequence is being performed, hardware clears location 7777_g in all PP memories and sets the P register of PPO to 6000_g. PPO starts performing a test program from a read-only memory in IOU and lights the deadstart panel L.D.S. ERROR-A and L.D.S. ERROR-B indicators. Indicator A remains lit unless the test program reaches location 6200_g within 10.25 microseconds. Indicator B remains lit until the test program reaches location 7776_g. When this happens, the unique part of the long deadstart sequence ends with a master clear.

Next, both deadstart sequences clear PPO location 0, write the settings of the deadstart panel matrix switches into PPO memory locations 1 to 20_g, and clear PPO location 21_g. PPO then starts executing the program entered from the matrix switches, which is normally a bootstrap program to input more data from an assigned external device.

The short deadstart sequence does not disturb PP memory other than PPO locations 0 to 21_g. Both deadstart sequences leave all PPs, except PPO, waiting for a block input, or for action through the maintenance channel. After the block input is complete, each PP starts executing the program entered from whatever address was entered into location 0 of that PP.

IOU Reconfiguration

The logical PP numbers and hardware are assigned to physical PPs circularly from the settings of IOU deadstart panel RECONFIGURATION switches, which specify which physical barrel and PPM is PPO. If the PPM section of these switches is set to a value greater than four, the value zero is substituted. If the BARREL section of these switches is set to a

value greater than the number of installed barrels, the value zero is substituted. Thus, possible barrel numbering is as described in table 3-3.

NOTE

The minimum system option is 10 PPs.

Table 3-3. Barrel Numbering Table

Barrels Installed	Physical Barrel	Logical PPs in Physical Barrel with BARREL RECONFIGURATION Switch Values			
		0	1	2	3
4 Barrels (20 PPs)	0	0-4	25-31	20-24	5-11
	1	5-11	0-4	25-31	20-24
	2	20-24	5-11	0-4	25-31
	3	25-31	20-24	5-11	0-4
3 Barrels (15 PPs)	0	0-4	20-24	5-11	(0-4)
	1	5-11	0-4	20-24	(5-11)
	2	20-24	5-11	0-4	(20-24)
2 Barrels (10 PPs)	0	0-4	5-11	(0-4)	(0-4)
	1	5-11	0-4	(5-11)	(5-11)
1 Barrel (5 PPs)	0	0-4	(0-4)	(0-4)	(0-4)

CP INSTRUCTIONS

CP INSTRUCTION FORMATS

NOTE

CYBER 170 CP instructions use the rightmost 60 bits in the 64-bit word. The leftmost 4 bits are undefined. For these instructions, the most significant bit is bit 59 and the least significant bit is bit 0.

Program instruction words are divided into 15-bit fields called parcels. The first parcel (parcel 0) is the highest-order 15 bits of the 60-bit word. The second, third, and fourth parcels (parcels 1, 2, and 3) follow in order. Figure 4-1 shows possible parcel arrangements for instructions within a program instruction word.

An instruction may occupy one, two, or four parcels. This arrangement depends upon the instruction format. When an instruction occupies two parcels, it must occupy two parcels within the same program word. A program word may be filled with a one-parcel pass instruction or an instruction acting as a two-parcel pass instruction. These instructions are used to fill a program word when necessary to place a particular instruction in the first parcel of a program word or to avoid starting a two-parcel instruction in the fourth parcel of a program word. Pass instructions may also be used for branch entry points because a branch instruction destination address must begin with a new word. One-parcel pass instructions are 460xx through 463xx. Instructions 60xxx through 62xxx may be used as two-parcel pass instructions by setting the i instruction designator to zero. Refer to table 4-1 for CP instruction designators.

CP instructions 011 and 012 have special properties. They are 60-bit double instructions which must start at parcel 0. The programmer has the option of providing a branch instruction at parcels

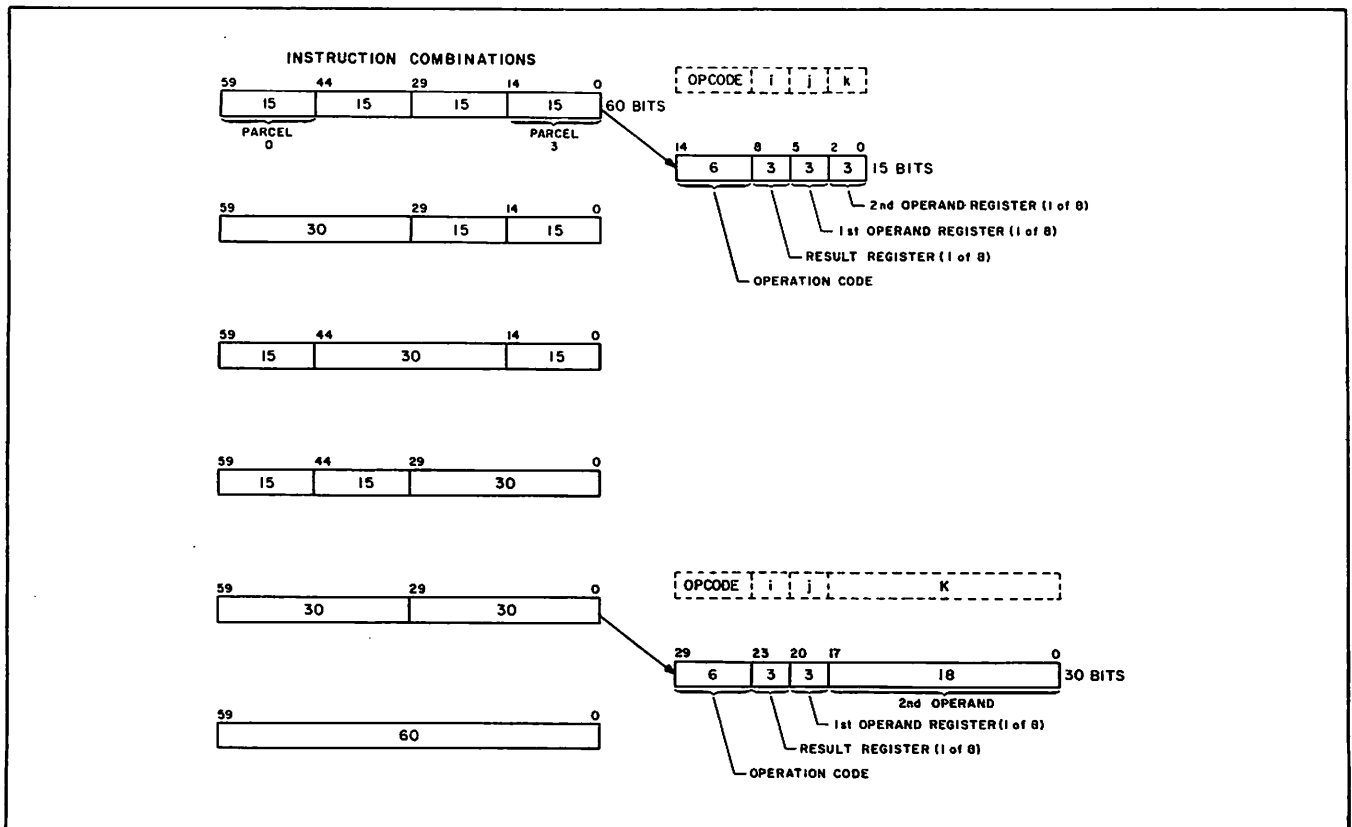


Figure 4-1. CP Instruction Parcel Arrangement

2 and 3 in the same instruction word (to an error handling software routine), or filling this space with pass instructions. Refer to instructions 011 and 012.

Instructions 013 and 464 through 467 are 60-bit instructions which must start at parcel 0. They ignore any information in parcels 2 and 3; however, these parcels are normally set to all zeros.

Table 4-1. Central Processor Instruction Designators

Designator	Use
Opcode	6-bit/9-bit field specifying instruction operation code.
i	3-bit code specifying one of eight registers.
j	3-bit code specifying one of eight registers.
jk	6-bit code specifying amount of shift or mask.
k	3-bit code specifying one of eight registers.
K	18-bit operand or address.
x	Unused designator.
A	One of eight 18-bit address registers.
B	One of eight 18-bit index registers; B0 is fixed and equal to zero.
X	One of eight 60-bit operand registers.
()	Content of the word at a CM address.
C1 †	Offset (character address) of the first character in the first word of the source field.
C2 †	Character address of the first character in the first word of the result field.
K1 †	18-bit address indicating the CM location of the first (leftmost) character of the source field.
K2 †	18-bit address indicating the CM location of the first (leftmost) character of the result field.
LL †	Lower 4 bits of the field length (character count) for a move or compare instruction; used with LU to specify field length.
LU †	Upper 9 bits of the field length (character count) for indirect move instruction or the upper 3 bits for direct instructions; used with LL to specify field length.

†Applicable to compare/move instructions only.

CP OPERATING MODES

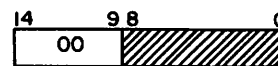
The CP executes instructions in CYBER 170 job mode, CYBER 170 monitor mode, and executive state. Changes between CYBER 170 job mode and CYBER 170 monitor mode are caused by CYBER 170 exchange jumps (CP instruction 013 and PP instructions 2600, 2610, and 2620). A hardware flag called the CYBER 170 monitor flag (MF) indicates whether the CP is in CYBER 170 job mode (flag is clear) or in CYBER 170 monitor mode (flag is set).

Executive state is invisible to the applications programmer. It sets up the CYBER 170 environment during initialization, executes certain instructions, and handles hardware-detected error conditions. Hardware-caused exchanges are called error exits; most of these can be enabled or disabled by setting or clearing bits in the CYBER 170 exchange package. For further information on CP operating modes, refer to CYBER 170 Exchange Jump, Executive State, and Error Response in section 5.

CP INSTRUCTION DESCRIPTIONS

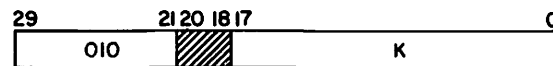
The instruction descriptions are in numerical order. The shaded areas, like those in the following 00xxx and 010xK instruction formats, indicate unused bits. The unused bits are ignored by the CP.

00xxx Error Exit to MA when CYBER 17 MF Clear PS
Interrupt to Executive
Mode when CYBER 170 MF Set



This instruction causes an illegal instruction error exit. CYBER 170 MF is the hardware monitor flag. Refer to Illegal Instructions, section 5.

010xK Return Jump to K RJ



This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction writes a special word into CM at relative address K. The current program sequence then terminates by a jump to address K plus 1. The word stored in memory contains a jump instruction which causes an unconditional jump to the address of this return jump instruction plus 1.

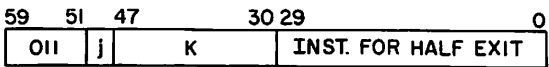
This instruction calls a subroutine and inserts execution of the subroutine between execution of this instruction word and the following instruction word. Instructions appearing after the return jump instruction in the instruction word are not executed. The called subroutine exit must be at address K. The called subroutine entrance address must be K plus 1.

This instruction stores a 60-bit word at address K in memory. The upper half of this word contains an unconditional jump (0400) instruction with an address which is equal to the current program address plus 1. The lower half of the stored word is all zeros. The octal digits in the stored word then appear as illustrated with the x field indicating the location of the current program address plus 1.

```

K      0400x xxxxx 00000 00000 Subroutine
                               exit
K + 1  yyyyy yyyyy yyyyy yyyyy Subroutine
                               entrance
  
```

011jK Block Copy Bj + K Words from UEM to CM REC Bj + K



This instruction copies a block of Bj plus K consecutive words from unified extended memory (UEM) to CM. The source UEM address is XO plus RAE where the bits used depend on the setting of the expanded addressing select flag in the CYBER 170 exchange package. If the flag is clear (UEM is in standard addressing mode), the UEM address is calculated using bits 0 through 22 of XO; bits 24 through 59 are ignored. If the flag is set (UEM is in expanded addressing mode), the UEM address is calculated using bits 0 through 28 of XO; bits 30 through 59 are ignored.

The destination CM address is either AO plus RAC or XO plus RAC depending on the setting of the block copy flag in the CYBER 170 exchange package. When the block copy flag is clear, the CM address is AO plus RAC. When the block copy flag is set, the CM address is calculated using bits 30 through 50 of XO. Bits 51 through 59 must be set to zero; results are undefined if these bits are not zero.

The operation leaves Bj, XO, and AO unchanged. Bj and K are both signed 18-bit one's complement numbers, making it possible to transfer a maximum of 131,071 60-bit words. If Bj plus K is zero, the instruction acts as a 60-bit pass instruction.

If bit 21 or 22 of the result of XO plus RAE is a one, zeros are transferred and the next instruction is taken from parcel 2 of the same instruction word. If this is not the case, the next instruction is taken from parcel 0 of the next instruction word. If execution of the 011jK instruction is interrupted, it is restarted from the beginning.

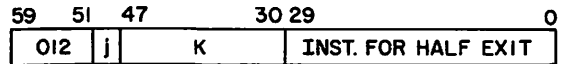
This instruction is illegal if it does not start in parcel 0 or the UEM enable flag in the CYBER 170 exchange package is clear.

In standard addressing mode, 24 bits of XO are checked against 23 bits of FLE with bit 23 of FLE equal to zero. In expanded addressing mode, 30 bits of XO are checked against 29 bits of FLE with bit 29 equal to zero. If the XO bits are greater than or equal to FLE, an address out of range is detected.

If Bj plus K is negative, an address range error exit takes place. If the source field and the destination field overlap in physical memory, the final contents of the destination field are undefined.

For further information, refer to Block Copy Instructions in section 5.

012jK Block Copy Bj + K Words from CM to UEM WEC Bj + K



This instruction copies a block of Bj plus K consecutive words from CM to UEM. The source CM address is either AO plus RAC or XO plus RAC depending on the setting of the block copy flag in the CYBER 170 exchange package. When the block copy flag is clear, the CM address is AO plus RAC. When the block copy flag is set, the CM address is calculated using bits 30 through 50 of XO. Bits 51 through 59 must be set to zero; results are undefined if these bits are not zero.

The destination UEM address is XO plus RAE where the bits used depend on the setting of the expanded addressing select flag in the CYBER 170 exchange package. If the flag is clear (UEM is in standard addressing mode), the UEM address is calculated using bits 0 through 22 of XO; bits 24 through 59 are ignored. If the flag is set (UEM is in expanded addressing mode), the UEM address is calculated using bits 0 through 28 of XO; bits 30 through 59 are ignored.

The operation leaves Bj, XO, and AO unchanged. Bj and K are both signed 18-bit one's complement numbers, making it possible to transfer a maximum of 131,071 60-bit words. If Bj plus K is zero, the instruction acts as a 60-bit pass instruction.

If bit 21 or 22 of the result of XO plus RAE is a one, zeros are transferred and the next instruction is taken from parcel 2 of the same instruction word. If this is not the case, the next instruction is taken from parcel 0 of the next instruction word. If execution of the 012jK instruction is interrupted, it is restarted from the beginning.

This instruction is illegal if it does not start in parcel 0 or the UEM enable flag in the CYBER 170 exchange package is clear.

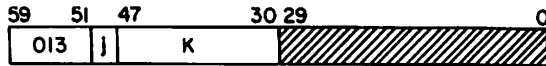
In standard addressing mode, 24 bits of XO are checked against 23 bits of FLE with bit 23 of FLE equal to zero. In expanded addressing mode, 30 bits of XO are checked against 29 bits of FLE with bit 29 equal to zero. If the XO bits are greater than or equal to FLE, an address out of range is detected.

If Bj plus K is negative, an address range error exit takes place. If the source field and the destination field overlap in physical memory, the final contents of the destination field are undefined.

For further information, refer to Block Copy Instructions in section 5.

013jK Central Exchange Jump to XJ Bj + K
Bj + K when CYBER 170 MF Set

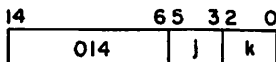
013xx Monitor Exchange Jump to MA XJ
when CYBER 170 MF Clear



This instruction must start at parcel 0. Also, a CYBER 170 exchange package must be ready at address Bj plus K or at address MA.

This instruction stores P plus 1 into the outgoing CYBER 170 exchange package in hardware and then exchanges this CYBER 170 exchange package with the CYBER 170 exchange package stored in memory. If, at the beginning of the instruction, the CYBER 170 MF is set, then the incoming CYBER 170 exchange package starts at absolute address Bj plus K. If, at the beginning, the CYBER 170 MF is clear, then the j and K fields of the instruction are ignored, and the incoming CYBER 170 exchange package starts at absolute address MA which is obtained from the outgoing CYBER 170 exchange package. In either case, the CYBER 170 MF is toggled and the outgoing CYBER 170 exchange package is stored beginning at the same CM address from where the incoming CYBER 170 exchange package is obtained. Also, the jump is always to relative address P, parcel 0, from the new CYBER 170 exchange package. Refer to CYBER 170 Exchange Jump, section 5.

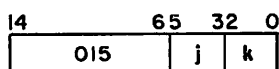
014jk Read One Word from (Xk + RAE) RXj Xk
to Xj



This instruction is illegal if the UEM enable flag in the CYBER 170 exchange package is clear. This instruction reads the 60-bit word from UEM location Xk plus RAE into Xj. Xk is less than FLE.

The number of bits checked for an address out of range condition varies depending on the addressing mode of UEM. In standard addressing mode, 24 bits of Xk are checked against 23 bits of FLE with bit 23 of FLE equal to zero. In expanded addressing mode, 30 bits of Xk are checked against 29 bits of FLE with bit 29 of FLE equal to zero. If Xk is greater than or equal to FLE, an address out of range is detected.

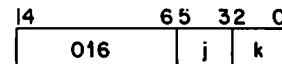
015jk Write One Word from Xj to WXj Xk
(Xk + RAE)



This instruction is illegal if the UEM enable flag in the CYBER 170 exchange package is clear. This instruction writes the 60-bit word from Xj into the UEM location Xk plus RAE. Xk is less than FLE.

The number of bits checked for an address out of range condition varies depending on the addressing mode of UEM. In standard addressing mode, 24 bits of Xk are checked against 23 bits of FLE with bit 23 of FLE equal to zero. In expanded addressing mode, 30 bits of Xk are checked against 29 bits of FLE with bit 29 of FLE equal to zero. If Xk is greater than or equal to FLE, an address out of range is detected.

016jk Read Free Running Counter RC



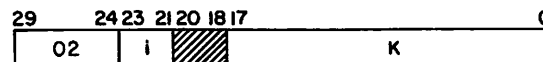
This instruction transfers the current contents of the 48-bit free running counter to the Xj register. The leftmost twelve bits of Xj are set to zero. The k field is ignored.

This instruction is a single parcel instruction that can be located in any parcel.

017jk Illegal Instruction

Refer to Illegal Instructions, section 5.

02ixK Jump to (Bi) + K JP

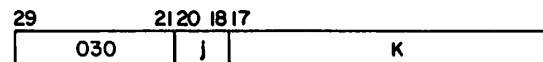


This two-parcel instruction uses the lower-order 18 bits as operand K. The instruction causes the current program sequence to terminate with a jump to address Bi plus K in CM.

This instruction allows computed branch point destinations. This is the only instruction in which a computed parameter can specify a program branch destination address. All other jump instructions have preassigned destination addresses.

The quantities in Bi and operand K are added in an 18-bit one's complement mode. The result is treated as an 18-bit positive integer which specifies the beginning address in CM for the new program sequence. The remaining instructions, if any, in the instruction word do not execute.

030jK Branch to K if (Xj) = 0 ZR

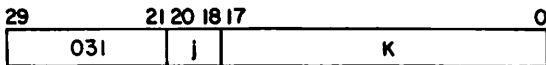


This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The branch to address K occurs only on the following conditions. The current program sequence continues for all other cases.

Jump to K if: (Xj) = 0000 0000 0000 0000 0000
(positive zero)
(Xj) = 7777 7777 7777 7777 7777
(negative zero)

This instruction branches on a zero result from either a fixed-point or a floating-point operation.

031jK Branch to K if (Xj) ≠ 0 NZ

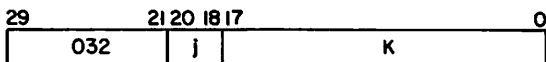


This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The program sequence continues only on the following conditions. The branch to address K occurs for all other cases.

Continue if: (Xj) = 0000 0000 0000 0000 0000
 (positive zero)
 (Xj) = 7777 7777 7777 7777 7777
 (negative zero)

This instruction branches on a nonzero result from either a fixed-point or a floating-point operation.

032jK Branch to K if (Xj) is Positive PL

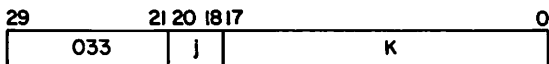


This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The branch decision for this instruction is based on the value of the sign bit in Xj.

Jump to K if: Bit 59 of Xj = 0 (positive)
 Continue if: Bit 59 of Xj = 1 (negative)

This instruction branches on a positive result from either a fixed-point or a floating-point operation.

033jK Branch to K if (Xj) is Negative NG

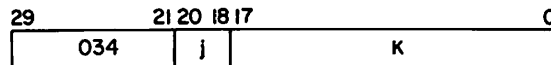


This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The branch decision for this instruction is based on the value of the sign bit in Xj.

Jump to K if: Bit 59 of Xj = 1 (negative)
 Continue if: Bit 59 of Xj = 0 (positive)

This instruction branches on a negative result from either a fixed-point or a floating-point operation.

034jK Branch to K if (Xj) is in Range IR

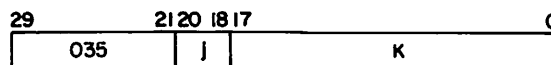


This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The program sequence continues only on the following conditions. The branch to address K occurs for all other cases.

Continue if: (Xj) = 3777 xxxxx xxxxx xxxxx xxxxx
 (positive overflow)
 (Xj) = 4000 xxxxx xxxxx xxxxx xxxxx
 (negative overflow)

This instruction branches on a floating-point quantity within the floating-point range. The value of the coefficient is ignored in making this branch test. An underflow quantity is considered in range for purposes of this test.

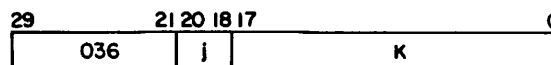
035jK Branch to K if (Xj) is Out of Range OR



This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The branch to address K occurs only on the following conditions. The current program sequence continues for all other cases.

Jump to K if: (Xj) = 3777 xxxxx xxxxx xxxxx xxxxx
 (positive overflow)
 (Xj) = 4000 xxxxx xxxxx xxxxx xxxxx
 (negative overflow)

036jK Branch to K if (Xj) is Definite DF



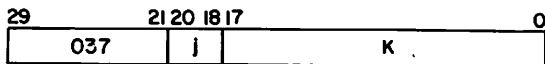
This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj.

The program sequence continues only on the following conditions. The branch to address K occurs for all other cases.

Continue if: (Xj) = 1777 xxxx xxxx xxxx xxxx
 (positive indefinite)
 (Xj) = 6000 xxxx xxxx xxxx xxxx
 (negative indefinite)

This instruction branches on a floating-point quantity which may be out of range but is still defined. The value of the coefficient is ignored in making this branch test. An overflow quantity or an underflow quantity is considered defined for purposes of this test.

037jK Branch to K if (Xj) is Indefinite ID

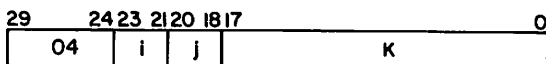


This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of the Xj register. The branch to address K occurs only on the following conditions. The current program sequence continues for all other cases.

Jump to K if: (Xj) = 1777 xxxx xxxx xxxx xxxx
 (positive indefinite)
 (Xj) = 6000 xxxx xxxx xxxx xxxx
 (negative indefinite)

This instruction branches on a floating-point quantity which is not defined. The value of the coefficient is ignored in making this branch test. An overflow quantity or an underflow quantity is considered defined for purposes of this test.

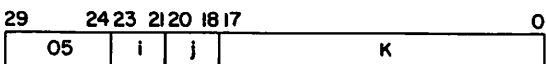
041jK Branch to K if (Bi) = (Bj) EQ



This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon a comparison of the contents of the Bi and Bj registers. The branch to address K occurs only if the two quantities are identical on a bit-by-bit comparison basis. The current program sequence continues for all other cases.

This instruction branches on an index equality test. A quantity consisting of all zeros and a quantity consisting of all ones are not equal for this test.

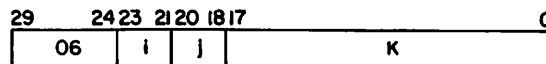
051jK Branch to K if (Bi) ≠ (Bj) NE



This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon a comparison of the contents of the Bi and Bj registers. The program sequence continues only if the two quantities are identical on a bit-by-bit comparison basis. The branch to address K occurs for all other cases.

This instruction branches on an index inequality test. A quantity consisting of all zeros and a quantity consisting of all ones are not equal for this test.

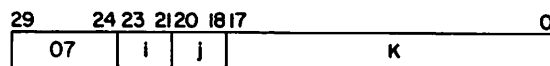
061jK Branch to K if (Bi) ≥ (Bj) GE



This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon a comparison of the contents of Bi and Bj. Both quantities are treated as signed integers. The branch to address K occurs if the content of Bi is greater than or equal to the content of Bj. The current program sequence continues if the content of Bi is less than Bj.

This instruction branches on an index threshold test. A positive zero quantity is considered greater than a negative zero quantity.

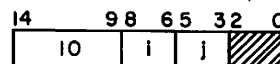
071jK Branch to K if (Bi) < (Bj) LT



This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon a comparison of the contents of Bi and Bj. Both quantities are treated as signed integers. The branch to address K occurs if the content of Bi is less than the content of Bj. The current program sequence continues if the content of Bi is greater than or equal to the content of Bj.

This instruction branches on an index threshold test. A positive zero quantity is considered greater than a negative zero quantity.

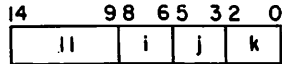
101jx Transmit (Xj) to Xi BXi Xj



This instruction transfers a 60-bit word from Xj into Xi.

This instruction moves data from one X register to another X register. No logical function is performed on the data.

11ijk Logical Product of (Xj) and BXi Xj * Xk
(Xk) to Xi



This instruction reads operands from two X registers, operates upon them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical product of the two operands. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

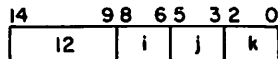
(Xj) = 7777 7000 0123 4567 1010

(Xk) = 0123 4567 0077 7700 1100

(Xi) = 0123 4000 0023 4500 1000

This instruction extracts portions of a 60-bit word during data processing.

12ijk Logical Sum of (Xj) and BXi Xj + Xk
(Xk) to Xi



This instruction reads operands from two X registers, operates upon them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical sum of the two operands. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

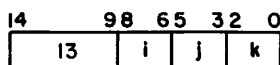
(Xj) = 0000 7777 0123 4567 1010

(Xk) = 0123 4567 7777 0000 1100

(Xi) = 0123 7777 7777 4567 1110

This instruction merges portions of a 60-bit word into a composite word during data processing.

13ijk Logical Difference of (Xj) and BXi Xj -Xk
(Xk) to Xi



This instruction reads operands from two X registers, operates upon them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The

result delivered to Xi is the bit-by-bit logical difference of the two operands. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

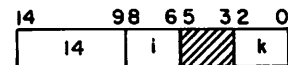
(Xj) = 0123 7777 0123 4567 1010

(Xk) = 0123 4567 7777 3210 1100

(Xi) = 0000 3210 7654 7777 0110

This instruction compares bit patterns or complements bit patterns during data processing.

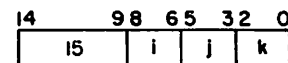
14ixk Transmit Complement of (Xk) BXi -Xk
to Xi



This instruction reads a 60-bit word from Xk, complements the word, and writes the result into Xi.

This instruction changes the sign of a fixed-point or floating-point quantity. The instruction also inverts an entire 60-bit field during data processing.

15ijk Logical Product of (Xj) with BXi -Xk * Xj
Complement of (Xk) to Xi



This instruction reads operands from two X registers, operates upon them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical product of the value in Xj and the complement of the value in Xk. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

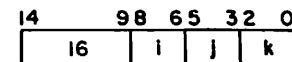
(Xj) = 7777 7000 0123 4567 1010

(Xk) = 0123 4567 0007 7700 1100

(Xi) = 7654 3000 0120 0067 0010

This instruction extracts portions of a 60-bit word during data processing.

16ijk Logical Sum of (Xj) with BXi -Xk + Xj
Complement of (Xk) to Xi



This instruction reads operands from two X registers, operates upon them to form a result, and delivers this result to a third X register. The

operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical sum of the value in Xj and the complement of the value in Xk. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

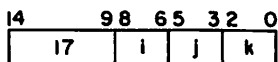
(Xj) = 0000 7777 0123 4567 1010

(Xk) = 0123 4567 7777 0000 1100

(Xi) = 7654 7777 0123 7777 7677

This instruction merges portions of a 60-bit word into a composite word during data processing.

17ijk Logical Difference of (Xj) BXi -Xk - Xj
with Complement of (Xk)
to Xi



This instruction reads operands from two X registers, operates upon them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical difference of the value in Xj and the complement of the value in Xk. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible combinations that may occur.

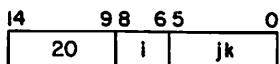
(Xj) = 0123 7777 0123 4567 1010

(Xk) = 0123 4567 7777 3210 1100

(Xi) = 7777 4567 0123 0000 7667

This instruction compares bit patterns or complements bit patterns during data processing.

20ijk Left Shift (Xi) by jk LXi jk



This instruction reads one operand from Xi, shifts the 60-bit word left circularly by jk bit positions, and writes the resulting 60-bit word back into the same Xi register. The j and k designators are treated as a single 6-bit positive integer operand in this instruction.

A left-circular shift implies that the bit pattern in the 60-bit word is displaced towards the highest-order bit positions. The bits shifted off the upper end of the 60-bit word are inserted in the lowest-order bit positions in the same sequence. The resulting 60-bit word has the same quantity of bits with values of one and zero as in the original operand.

A sample computation is listed in octal notation to illustrate the operation performed.

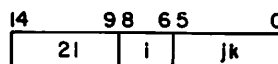
Initial (Xi) = 2323 6600 0000 0000 0111

jk = 12 (octal)

Final (Xi) = 7540 0000 0000 0022 2464

This instruction, together with instruction 21, may be used whenever a data word is to be shifted by a predetermined amount. If the amount of shift is derived in the execution of the program, instruction 22 or 23 should be used.

21ijk Right Shift (Xi) by jk AXi jk



This instruction reads one operand from Xi, shifts the 60-bit word right with sign extension by jk bit positions, and writes the resulting 60-bit word back into the same Xi register. The j and k designators are treated as a single 6-bit positive integer operand in this instruction.

A right shift with sign extension implies that the bit pattern in the 60-bit word is displaced toward the lowest-order bit positions. The bits shifted off the lower end of the word are discarded. The highest-order bit positions are filled with copies of the original sign bit.

Two sample computations are listed in octal notation to illustrate the operation performed. The first example contains a positive operand, and the second example contains a negative operand.

Initial (Xi) = 2004 7655 0002 3400 0004

jk = 30 (octal)

Final (Xi) = 0000 0000 2004 7655 0002

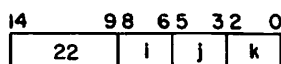
Initial (Xi) = 6000 4420 2222 0000 5643

jk = 10 (octal)

Final (Xi) = 7774 0011 0404 4440 0013

This instruction, together with instruction 20, may be used whenever a data word is to be shifted by a predetermined amount. If the amount of shift is derived in the execution of the program, instruction 22 or 23 should be used.

22ijk Left Shift (Xk) Nominally LXi Bj Xk
(Bj) Places to Xi



This instruction reads a 60-bit operand from Xk, shifts the data either left or right as specified by Bj, and writes the resulting 60-bit word into Xi. If the value in Bj is positive, the data is left

shifted circularly the number of bit positions designated by the value in Bj. If the value in Bj is negative, the data is right shifted with sign extension the number of bit positions designated by the value in Bj. The sign of Bj is determined by Bj bit 17.

A left circular shift implies that the bit pattern in the 60-bit word is displaced towards the highest-order bit positions. The bits shifted off the upper end are inserted in the lowest-order bit positions in the same sequence. The resulting 60-bit word has the same quantity of bits with values of one and zero as in the original operand.

A right shift with sign extension implies that the bit pattern in the 60-bit word is displaced towards the lowest-order positions. The bits shifted off the lower end are discarded. The highest-order bit positions are filled with copies of the original sign bit.

Two sample computations are listed in octal notation to illustrate the operation performed. The first example contains a positive shift count resulting in a left circular shift, and the second example illustrates the right shift with sign extension.

(Xk) = 2323 6600 0000 0000 0111

(Bj) = 00 0012

(Xi) = 7540 0000 0000 0022 2464

(Xk) = 1327 6000 0000 3333 2422

(Bj) = 77 7771

(Xi) = 0013 2760 0000 0033 3324

If Bj bits 6 through 10 are different from Bj bit 17 and Bj bit 17 is set, the shift count is greater than 63 (decimal) places right, and a result of positive zero is returned to Xi. Bj bits 11 through 16 are not tested by this instruction.

This instruction is used when the amount of shift is derived in the computation. The instruction is also used for correcting the coefficient of a floating-point number when the exponent has been unpacked into a B register.

23ijk Right Shift (Xk) Nominally AXi Bj Xk
(Bj) Places to Xi

14	98	65	32	0
23	i	j	k	

This instruction reads a 60-bit operand from Xk, shifts the data either left or right as specified by the content of Bj, and writes the resulting 60-bit word into Xi. If the value in Bj is positive, the data is right shifted with sign extension the number of bit positions designated by the value in Bj. If the value in Bj is negative, the data is left shifted circularly the number of bit positions designated by the value in Bj. The sign of Bj is determined by Bj bit 17.

A left circular shift implies that the bit pattern in the 60-bit word is displaced towards the

highest-order bit positions. The bits shifted off the upper end are inserted in the lowest-order bit positions in the same sequence. The resulting 60-bit word has the same quantity of bits with values of one and zero as in the original operand.

A right shift with sign extension implies that the bit pattern in the 60-bit words is displaced towards the lowest-order bit positions. The bits shifted off the lower end of the word are discarded. The highest-order bit positions are filled with copies of the original sign bit.

Two sample computations are listed in octal notation to illustrate the operation performed. The first example contains a positive shift count resulting in a right shift with sign extension, and the second example contains a negative shift count resulting in a left circular shift.

(Xk) = 1327 6000 0000 3333 2422

(Bj) = 00 0006

(Xi) = 0013 2760 0000 0033 3324

(Xk) = 2323 6600 0000 0000 0111

(Bj) = 77 7765

(Xi) = 7540 0000 0000 0022 2464

If Bj bits 6 through 10 are different from Bj bit 17 and Bj bit 17 is clear, the shift count is greater than 63 (decimal) places right, and a result of positive zero is returned to Xi. Bj bits 11 through 16 are not tested by this instruction.

This instruction is used when the amount of shift is derived in the computation. The instruction is also used for correcting the coefficient of a floating-point number when the exponent has been unpacked into a B register.

24ijk Normalize (Xk) to Xi and Bj NXi Bj Xk

14	98	65	32	0
24	i	j	k	

This instruction reads one operand from Xk, performs a normalizing operation on this word in floating-point format, and delivers the normalized result to Xi. In addition, a positive integer shift count is sent to Bj. This shift count is the number of bit positions of shift required to normalize the original operand coefficient.

The normalizing operation consists of repositioning the coefficient portion of the operand and then adjusting the exponent portion of the operand to leave the value of the result unaltered. The coefficient is shifted towards the higher-order bit positions of the word. The coefficient is shifted the minimum number of bit positions required to make bit 47 different from sign bit 59. This places the most significant bit of the coefficient in the highest-order position. The exponent is then decreased by the number of bit positions shifted.

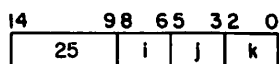
Two sample computations are listed in octal notation to illustrate the operation performed. The first example involves a positive floating-point number, and the second example involves a negative number.

(Xk) = 2034 0047 6500 0000 2262
 (Xi) = 2026 4765 0000 0022 6200
 (Bj) = 00 0006

 (Xk) = 5743 7730 1277 7777 5515
 (Xi) = 5751 3012 7777 7755 1577
 (Bj) = 00 0006

Normalizing a number with either a positive or negative zero coefficient sets a shift count in Bj to 48 (decimal) and enters Xi with positive zero. If Xk contains an infinite quantity (3777xxx...x or 4000xxx...x) or an indefinite quantity (1777xxx...x or 6000xxx...x), no shift takes place. The content of Xk is copied to Xi, and Bj is set to zero. Corresponding infinite and indefinite exit conditions are also set in the CP for exit mode action. If the exponent is less than negative 1777 with a zero coefficient, the contents of Xi and Bj are set to zero. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

25ijk Round Normalize (Xk) to Xi and Bj ZXi Bj Xk



This instruction reads one operand from Xk, performs a rounding and then a normalizing operation in floating-point format, and delivers the round normalized result to Xi. In addition, a positive integer shift count is sent to Bj. This shift count is the number of bit positions of shift required to normalize the original operand coefficient.

The rounding operation consists of adding a bit to the coefficient portion of the operand in a bit position immediately below the least significant bit position. This round bit has a value equal to the complement of the operand sign bit. The result increases the magnitude of the coefficient by one-half the value of the least significant bit.

The normalizing operation consists of repositioning the coefficient and adjusting the exponent to leave the value of the resulting floating-point quantity unaltered. The coefficient is shifted towards the higher-order bit positions. The round bit is shifted along with the coefficient. The displacement is the minimum number of bit positions required to make bit 47 different from sign bit 59. This places the most significant bit of the coefficient in the highest-order bit position. The exponent is decreased by the number of bit positions shifted.

Two sample computations are listed in octal notation to illustrate the normalizing operation performed.

The first example involves a positive floating-point number, and the second example involves a negative number.

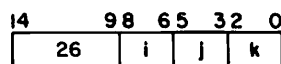
(Xk) = 2034 0047 6500 0000 2262
 (Xi) = 2026 4765 0000 0022 6420
 (Bj) = 00 0006

 (Xk) = 5743 7730 1277 7777 5515
 (Xi) = 5751 3012 7777 7755 1537
 (Bj) = 00 0006

If Xk contains either an infinite quantity (3777xxx...x or 4000xxx...x) or an indefinite quantity (1777xxx...x or 6000xxx...x), no shift takes place. The content of Xk is copied to Xi, and Bj is set to zero. Corresponding infinite and indefinite exit conditions are also set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

26ijk Unpack (Xk) to Xi and Bj UXi Bj Xk



This instruction reads one operand from Xk, unpacks this word from floating-point format, and delivers the coefficient and exponents to Xi and Bj, respectively. The 60-bit word delivered to Xi consists of the lowest 48 bits unaltered from the original operand plus the upper 12 bits, each equal to the original sign bit. This is a signed integer equal to the value of the coefficient in the original operand. The 18-bit quantity delivered to Bj is a signed integer equal to the value of the exponent in the original operand. The 11-bit exponent field in the operand is altered to remove the bias and then sign extended to fill out the 18-bit quantity. The sign of the coefficient is removed in this process.

Four sample sets of operands and unpacked results are listed in octal notation to illustrate the operation performed. These examples contain the four combinations of coefficient sign and exponent sign.

(Xk) = 2034 4500 3333 2000 0077
 (Xi) = 0000 4500 3333 2000 0077
 (Bj) = 00 0034

 (Xk) = 1743 4500 3333 2000 0077
 (Xi) = 0000 4500 3333 2000 0077
 (Bj) = 77 7743

 (Xk) = 5743 3277 4444 5777 7700
 (Xi) = 7777 3277 4444 5777 7700

(Bj) = 00 0034

30ijk Floating Sum of (Xj) and (Xk) FXi Xj + Xk to Xi

(Xk) = 6034 3277 4444 5777 7700

14	98	65	32	0
30	i	j	k	

(Xi) = 7777 3277 4444 5777 7700

(Bj) = 77 7743

This instruction converts a number from floating-point format to fixed-point format. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

This instruction reads operands from two X registers, operates upon them to form a floating-point sum, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The sum of the quantities in Xj and Xk is delivered to Xi in floating-point format and is not necessarily normalized.

27ijk Pack (Xk) and (Bj) to Xi PXi Bj Xk

14	98	65	32	0
27	i	j	k	

This instruction reads the contents of Xk and Bj, packs them into a single word in floating-point format, and delivers this result to Xi. The coefficient for the value in Xi is obtained from the content of Xk, which is treated as a signed integer. The exponent for the value in Xi is obtained from the content of Bj, which is treated as a signed integer.

The two operands are unpacked from floating-point format, and the exponents are compared. The coefficient with the smaller exponent is right shifted by the difference of the two exponents such that both coefficients are the same significance. The two coefficients are then added to form a 96-bit result. The upper half of the result is then selected as a coefficient and packed along with the larger exponent to form the result sent to Xi. If coefficient overflow occurs, the sum is right shifted one place, and the exponent is increased by one.

The lowest-order 48 bits in Xi are copied directly from the lowest-order 48 bits in Xk. The sign bit in Xi is copied directly from the sign bit in Xk. The exponent field in Xi is derived from the value in Bj by extracting the lowest-order 11 bits in Bj and modifying this quantity for exponent bias and coefficient sign.

If the two operands have unlike signs, the result coefficient may have leading zeros. No normalize operation is built into this instruction to correct this situation. A separate normalize instruction must be programmed if the result is to be kept in a normalized form.

Four sample sets of operands and packed results are listed in octal notation to illustrate the operation performed. These examples contain the four combinations of coefficient sign and exponent sign.

When the difference between the exponents is greater than 128 (decimal), the shifted sign bit is extended to the entire shifted operand. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

(Xk) = 0000 4500 3333 2000 0077

For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

(Bj) = 00 0034

31ijk Floating Difference of FXi Xj - Xk (Xj) and (Xk) to Xi

(Xi) = 2034 4500 3333 2000 0077

14	98	65	32	0
31	i	j	k	

(Xk) = 0000 4500 3333 2000 0077

(Bj) = 77 7743

This instruction reads operands from two X registers, operates upon them to form a floating-point difference, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi in floating-point format and is not necessarily normalized.

(Xi) = 1743 4500 3333 2000 0077

(Xk) = 7777 3277 4444 5777 7700

(Bj) = 00 0034

(Xi) = 5743 3277 4444 5777 7700

(Xk) = 7777 3277 4444 5777 7700

(Bj) = 77 7743

The two operands are unpacked from floating-point format, and the exponents are compared. The coefficient with the smaller exponent is right shifted by the difference of the two exponents such that both coefficients are the same significance. The Xk coefficient is then subtracted from the Xj coefficient to form a 96-bit result. The upper half of the result is then selected and packed along with the larger exponent to form the result sent to Xi. If coefficient overflow occurs, the result is right

(Xi) = 6034 3277 4444 5777 7700

This instruction converts a number in fixed-point format to floating-point format. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

shifted one place, and the exponent is increased by one.

If the two operands have like signs, the result coefficient may have leading zeros. No normalize operation is built into this instruction to correct this situation. A separate normalize instruction must be programmed if the result is to be kept in a normalized form. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

32ijk Floating Double-Precision Sum DXi Xj + Xk
of (Xj) and (Xk) to Xi

14	98	65	32	0
32	i	j	k	

This instruction reads operands from two X registers, operates upon them to form a double-precision floating-point sum, and delivers the lower half of this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The sum of the quantities in Xj and Xk is delivered to Xi in floating-point format and is not necessarily normalized.

The two operands are unpacked from floating-point format, and the exponents are compared. The coefficient with the smaller exponent is right shifted by the difference of the two exponents such that both coefficients are the same significance. The two coefficients are then added to form a 96-bit result. The lower half of the result is then selected and packed along with the larger exponent minus 48 (decimal) to form the result sent to Xi. If coefficient overflow occurs, the result is right shifted by one place, and the exponent is increased by one. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

33ijk Floating Double-Precision DXi Xj - Xk
Difference of (Xj) and (Xk)
to Xi

14	98	65	32	0
33	i	j	k	

This instruction reads operands from two X registers, operates upon them to form a double-precision floating-point difference, and delivers the lower half of this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi in floating-point format and is not necessarily normalized.

The two operands are unpacked from floating-point format, and the exponents are compared. The coefficient with the smaller exponent is right shifted by the difference of the two exponents such that both coefficients are the same significance. The Xk coefficient is then subtracted from the Xj coefficient to form a 96-bit result. The lower half of the result is then selected and packed along with the larger exponent minus 48 (decimal) to form the result sent to Xi. If coefficient overflow occurs, the result is right shifted one place, and the exponent is increased by one.

Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

34ijk Round Floating Sum of (Xj) RXi Xj + Xk
and (Xk) to Xi

14	98	65	32	0
34	i	j	k	

This instruction reads operands from two X registers, operates upon them to form a rounded floating-point sum, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result is delivered to Xi in floating-point format and is not necessarily normalized.

The round floating-point sum is a single-precision floating-point sum with a round bit (or bits) inserted before the add operation takes place. A round bit is always inserted in the coefficient with the larger exponent. If the two exponents are equal, the round bit is inserted in the coefficient for Xk. The round bit is equal to the complement of the sign bit and is inserted immediately to the right of the lowest-order bit in the coefficient. This has the effect of increasing the magnitude of the coefficient by one-half of the least significant bit. A second round bit is inserted in a corresponding manner to the other coefficient if both operands are normalized or have unlike signs. The second round bit is inserted before the coefficient has been shifted by the difference of the exponents. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

35ijk Round Floating Difference RXi Xj - Xk
of (Xj) and (Xk) to Xi

14	98	65	32	0
35	i	j	k	

This instruction reads operands from two X registers, operates upon them to form a rounded floating-point difference, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi in floating-point format and is not necessarily normalized.

The round floating-point difference is a single-precision floating-point difference with a round bit (or bits) inserted before the subtract operation takes place. A round bit is always inserted in the coefficient with the larger exponent. If the two exponents are equal, the round bit is added to the coefficient for Xk. The round bit is equal to the complement of the sign bit and is inserted immediately to the right of the lowest-order bit in the coefficient. This has the effect of increasing the magnitude of the coefficient by one-half of the least significant bit. A second round bit is inserted in a corresponding manner to the other coefficient if both operands are normalized or have like signs. The second round bit is inserted before the coefficient has been shifted by the difference of the exponents. Infinite (377xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

36ijk Integer Sum of (Xj) and (Xk) to Xi $IXi Xj + Xk$

14	98	65	32	0
36	i	j	k	

This instruction reads operands from two X registers, operates upon them to form a 60-bit integer sum, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are signed integers. The resulting integer sum is delivered to Xi. Overflow is not detected.

This instruction adds integers too large for handling by 50 through 77 instructions. The instruction also merges and compares data fields during data processing.

For further information, refer to Integer Arithmetic under CP Programming in section 5.

37ijk Integer Difference of (Xj) and (Xk) to Xi $IXi Xj - Xk$

14	98	65	32	0
37	i	j	k	

This instruction reads operands from two X registers, operates upon them to form a 60-bit integer difference, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are signed integers. The

result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi. Overflow is not detected.

This instruction subtracts integers too large for handling by 50 through 77 instructions. The instruction also compares data fields during data processing.

For further information, refer to Integer Arithmetic under CP Programming in section 5.

40ijk Floating Product of (Xj) and (Xk) to Xi $FXi Xj * Xk$

14	98	65	32	0
40	i	j	k	

This instruction reads operands from two X registers, operates upon them to form a floating-point product, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result is delivered to Xi in floating-point format. If both operands are normalized, the result is also normalized. If both operands are not normalized, the result is not normalized.

The two operands are unpacked from floating-point format. The exponents are added with a correction factor to determine the exponent for the result. The coefficients are multiplied as signed integers to form a 96-bit integer product. The upper half of this product is extracted to form the coefficient for the result. If the original operands are normalized and the product has only 95 significant bits, a 1-bit left shift to normalize the result coefficient is done. The resulting exponent is reduced by one count in this case.

If both operands are not normalized, the resulting double-precision product has less than 96 significant bits. No test is made for the position of the most significant bit. The upper 48 bits are read from the double-precision product register. Leading zeros occur in this result coefficient.

This instruction is used in floating-point calculations where rounding of operands is not desired, such as in multiple-precision arithmetic and in calculations involving error analysis. Infinite (377xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

41ijk Round Floating Product of (Xj) and (Xk) to Xi $RXi Xj * Xk$

14	98	65	32	0
41	i	j	k	

This instruction reads operands from two X registers, operates upon them to form a rounded floating-point product, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point

format and are not necessarily normalized. The result is delivered to Xi in floating-point format. If both operands are normalized, the result is also normalized. If both operands are not normalized, the result is not normalized.

The two operands are unpacked from floating-point format. The exponents are added with a correction factor to determine the exponent for the result. The coefficients are multiplied as signed integers to form a 96-bit integer product. A rounding bit is added to bit position 46 of this product. The upper half of this product is extracted to form the coefficient for the result. If the original operands are normalized and the product has only 95 significant bits, a 1-bit left shift to normalize the result coefficient is done. The resulting exponent is reduced by one count in this case.

If both operands are not normalized, the resulting double-precision product has less than 96 significant bits. No test is made for the position of the most significant bit. The upper 48 bits are read from the double-precision product register. Leading zeros occur in this result coefficient.

This instruction is used in single-precision floating-point calculations. For multiple-precision calculations, the 40 and 42 instructions must be used. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

42ijk Floating Double-Precision DXi Xj * Xk
Product of (Xj) and (Xk) to Xi

14	98	65	32	0
42	i	j	k	

This instruction reads operands from two X registers, operates upon them to form a double-precision floating-point product, and delivers the lower half of this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The lower half of the double-precision product is delivered to Xi in floating-point format and is not necessarily normalized.

The operands are not rounded in this operation. The two operands are unpacked from floating-point format. The exponents are added to determine the exponent for the result. The result exponent is exactly 48 less than the exponent for a 40 instruction. The coefficients are multiplied as signed integers to form a 96-bit integer product. The lower half of this product is extracted to form the coefficient for the result. If the original operands are normalized and the double-precision product has only 95 significant bits, a 1-bit left shift to normalize the result coefficient is done. The resulting exponent is reduced by one count in this case.

If both operands are not normalized, the resulting double-precision product has less than 96 significant bits. No test is made for the position of the most significant bit. The lower 48 bits are always read from the 96-bit product register.

This instruction is used in multiple-precision floating-point calculations. This instruction also provides for integer multiplication capabilities where both operands have an exponent value of plus or minus zero, and neither coefficient has been normalized. The integer result sent to Xi is 48 bits with 60-bit sign extension. If the result exceeds 48 bits, the hardware does not detect an overflow. An overflow check can be made by executing a 40 instruction using the same two operands. If the result is nonzero, overflow is then indicated. An integer multiply operation is not intended to be used with normalized operands. Infinite (3777xxx...x and 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

43ijk Form Mask of jk Bits to Xi MXi jk

14	98	65	32	0
43	i	j	k	

This instruction generates a masking word using the j and k designators as parameters. No operands are read from operating registers. The j and k designators are treated as a single 6-bit octal quantity to designate the width of the masking field. A field of ones, beginning at the highest-order end of the word, is extended downward on a background of zeros. The completed masking word consists of one bit in the highest-order jk bit positions and zero bits in the remainder of the word. This masking word is then delivered to Xi. The following are sample parameters.

j = 2

k = 4

Xi = 7777 7760 0000 0000 0000

This instruction generates variable width masks for logical operations. This instruction, together with a shift instruction, generally creates an arbitrary field mask faster than reading a pregenerated mask from CM.

44ijk Floating Divide (Xj)
by (Xk) to Xi FXi Xj/Xk

14	98	65	32	0
44	i	j	k	

This instruction reads operands from two X registers, operates upon them to form a floating-point quotient, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format. The result of dividing the content of Xj by the content of Xk is delivered to Xi. If both operands are normalized, the quotient is also normalized. The remainder from the division process is discarded.

The two operands are unpacked from floating-point format. The exponents are subtracted with a correction factor to determine the exponent for the

result. The coefficient from Xj is positioned in a dividend register. The coefficient from Xk is trial-subtracted repeatedly from the dividend. The quotient bits are assembled in a quotient register. When 48 bits of the quotient are assembled, they are packed with the result exponent into floating-point format and delivered to Xi.

If the exponent subtraction causes an underflow or overflow, an underflow or overflow result is returned even with the occurrence of a divide fault.

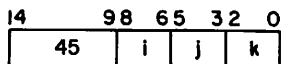
If the dividend is not normalized, the quotient cannot be normalized. However, the quotient is correct even though there may be leading zeros in the coefficient. If the divisor is not normalized, the quotient may be incorrect. If the coefficient for the content of Xj is larger than the coefficient for the content of Xk by a factor of two or more, a divide fault causes an indefinite result to be returned to Xi.

This instruction is used in floating-point calculations where rounding of operands is not desired. In multiple-precision division, this instruction must be followed by a multiplication of the quotient by the divisor and subtracted from the dividend to reconstruct the remainder.

If infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands are used, corresponding exit conditions are set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

45ijk Round Floating Divide (Xj) by (Xk) to Xi RXi Xj/Xk



This instruction reads operands from two X registers, operates upon them to form a rounded floating-point quotient, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format. The result of dividing the content of Xj by the content of Xk is delivered to Xi. If both operands are normalized, the quotient is also normalized. The remainder from the division process is discarded.

The two operands are unpacked from floating-point format in this operation. The exponents are subtracted with a correction factor to determine the exponent for the result. The coefficient from Xj is positioned in a dividend register. The Xj quantity is modified by inserting a 2525...25 round pattern below the lowest-order bit of the dividend coefficient. The coefficient from Xk is trial-subtracted repeatedly from the dividend. The quotient bits are assembled in a quotient register. When 48 bits of the quotient are assembled, they are packed with the result exponent into floating-point format and delivered to Xi.

If the dividend is not normalized, the quotient cannot be normalized. However, the quotient is correct even though there may be leading zeros in the coefficient. If the divisor is not normalized, the quotient may be incorrect. If the coefficient for the

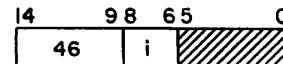
content of Xj is larger than the coefficient for the content of Xk by a factor of two or more, a divide fault occurs. A divide fault causes an indefinite result to be returned to Xi.

This instruction is used in single-precision floating-point calculations where rounding of operands is desired to reduce truncation errors.

If infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands are used, corresponding exit conditions are set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

460xx through 463xx Pass NO



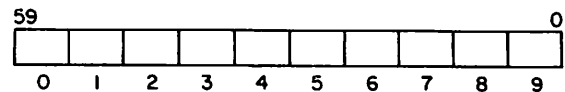
These instructions fill program instruction words where necessary to match jump destinations with word boundaries. The j and k designators are ignored, and a nonzero value has no effect in this instruction.

464 through 467 Compare/Move

The compare/move instructions (also referred to as CMU instructions) are provided for compatibility with previous systems. For better performance, recompile jobs to avoid use of CMU instructions.

CMU instructions must appear in parcel 0 or they are treated as illegal instructions.

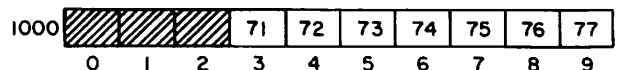
Data fields consisting of 6-bit characters may start or end with any character position (offset) of the 10 6-bit positions in each word. The character positions are designated as follows:



For move instructions, a K1 designator specifies which CM word contains the first character of the source data field, and a C1 designator specifies the character position (offset) of the first character. The K2 designator specifies the CM location in which the first character of the result data field is placed, and the C2 designator specifies the first character position. For compare instructions, both data field addresses specify source fields.

Example:

If the instruction is K1=1000 and C1=3, the first character of the source field is in position 3 of location 1000.



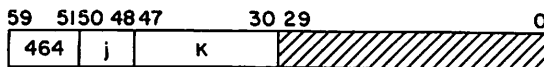
Therefore, the first character of the source field is 71.

An address is out of range if C1 or C2 is greater than 9, K1 plus N1 is greater than the program field length for CM (FLC), or K2 plus N2 is greater than FLC. N1 equals the number of CM references made to the source data field starting at K1, and N2 equals the number of CM references made to the result data field starting at K2. When an address-out-of-range condition occurs, the CMU instruction is not executed.

LL is the lower 4 bits, and LU is the upper 9 bits of the field length designator in numbers of characters. The maximum length of the data fields for the move direct and the compare instructions is 127 (177₈) characters. The maximum data field length for the move indirect instruction is 8191 (17777₈) characters. If L (LU and LL combined) is zero, the instruction becomes a pass.

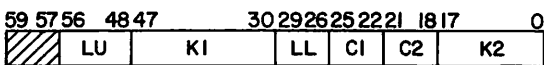
For overlapping move instructions, the address of the source field (specified by K1) must be greater than the address of the result field (specified by K2) to provide proper field overlap. If K1 is less than K2, part of the source field is changed during execution, with the amount of change determined by the number of CM conflicts encountered. Overlapping fields should not contain more than 377 (octal) characters, because an exchange jump interrupts any compare/move operation having a decremented field length greater than 377 (octal).

464jK Move Indirect IM



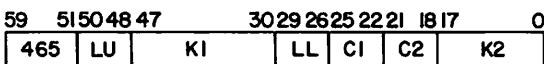
Any instructions located in the lower two parcels of the instruction word do not execute.

Bj plus K specifies a relative address in CM for the following descriptor word.



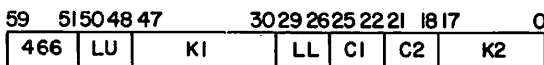
The descriptor word specifies the movement of the source field to the result field. The movement is from left to right through the field. Register X0 clears at the end of the execution.

465 Move Direct DM



This instruction moves the source field to the result field as specified by the 60-bit instruction word. The field length is limited to a 7-bit count.

466 Compare Collated CC



This instruction compares the field designated by K1,C1 with the field designated by K2,C2 as specified by the 60-bit instruction word.

The compare is from left to right through the fields until two unequal characters are found. These two characters are then collated and referenced in the collate table beginning at address A0 (table 4-2). If the table values found for the two unequal characters are equal, the compare continues until another pair of characters is unequal or until the field length is exhausted. If the table values found for the two unequal characters are unequal, X0 is set prior to instruction termination as follows:

If field K1 is greater than field K2, set X0 to 0000 0000 0000 0000 0xxx.

If field K1 is equal to field K2, set X0 to 0000 0000 0000 0000.

If field K1 is less than field K2, set X0 to 7777 7777 7777 7yyy where yyy is the complement of xxx.

The value of the three octal numbers xxx, stored in X0, is determined by the equation L minus N equals xxx (L is the length of the field, and N is the number of pairs of characters that were collated equal prior to instruction termination). In other words, xxx is the number of pairs of characters not yet compared plus one.

The A0 register contains the starting word address of an 8-word, 64-character collate table (table 4-2). This table must have been previously stored in consecutive CM locations.

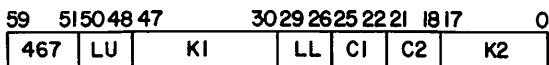
The collated value of a character is found by examining the collate table. The upper 3 bits of the character to be collated are added to A0 to obtain the relative address of the word containing the collated value. The lower 3 bits of the character to be collated specify the character address of the collated value.

Example:

Suppose the character under examination is an octal 63. The 6 is added to the A0 to form the word address. The 3 is used to pick the correct character from that word. The value of 63 is 63 in the collate table.

Table 4-2. Collate Table

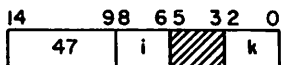
Address	Collating Character Locations									
	00	01	02	03	04	05	06	07	xx	xx
A0	00	01	02	03	04	05	06	07	xx	xx
A0+1	10	11	12	13	14	15	16	17	xx	xx
A0+2	20	21	22	23	24	25	26	27	xx	xx
A0+3	30	31	32	33	34	35	36	37	xx	xx
A0+4	40	41	42	43	44	45	46	47	xx	xx
A0+5	50	51	52	53	54	55	56	57	xx	xx
A0+6	60	61	62	63	64	65	66	67	xx	xx
A0+7	70	71	72	73	74	75	76	77	xx	xx



This instruction is similar to the 466 instruction except that the collate table is not used. The X0 register is set when the first pair of unequal characters is encountered or when the field length is exhausted.

47ixk Population Count of (Xk) to Xi

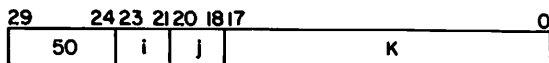
CXi Xk



This instruction reads one operand from Xk, counts the number of one bits in the operand, and stores the count in Xi. The count delivered to Xi is a positive integer. If the operand is all ones, a count of 60 (decimal) is delivered to Xi. If operand is all zeros, a zero word is delivered to Xi.

50ijK Set Ai to (Aj) + K

SAi Aj K



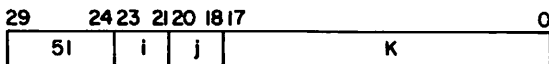
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Aj, forms the sum of the operand plus K, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

- i = 0 No CM reference
- i = 1,2,3,4,5 Read from CM to Xi
- i = 6,7 Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

51ijK Set Ai to (Bj) + K

SAi Bj K



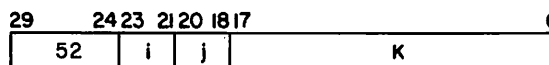
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Bj, forms the sum of the operand plus K, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

- i = 0 No CM reference
- i = 1,2,3,4,5 Read from CM to Xi
- i = 6,7 Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

52ijK Set Ai to (Xj) + K

SAi Xj K



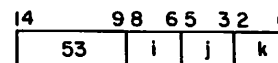
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Xj, forms the sum of the operand plus K, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

- i = 0 No CM reference
- i = 1,2,3,4,5 Read from CM to Xi
- i = 6,7 Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

53ijk Set Ai to (Xj) + (Bk)

SAi Xj + Bk



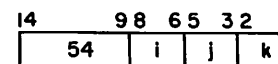
This instruction reads operands from Xj and Bk, forms the sum of the operands, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

- i = 0 No CM reference
- i = 1,2,3,4,5 Read from CM to Xi
- i = 6,7 Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

54ijk Set Ai to (Aj) + (Bk)

SAi Aj + Bk



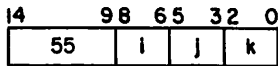
This instruction reads operands from Aj and Bk, forms the sum of the operands, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

- i = 0 No CM reference
- i = 1,2,3,4,5 Read from CM to Xi
- i = 6,7 Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

55ijk Set Ai to (Aj) - (Bk)

SAi Aj - Bk



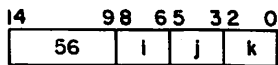
This instruction reads operands from Aj and Bk, subtracts the Bk operand from the Aj operand, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

- i = 0 No CM reference
- i = 1,2,3,4,5 Read from CM to Xi
- i = 6,7 Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the results back into CM.

56ijk Set Ai to (Bj) + (Bk)

SAi Bj + Bk



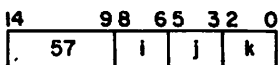
This instruction reads operands from Bj and Bk, forms the sum of the operands, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

- i = 0 No CM reference
- i = 1,2,3,4,5 Read from CM to Xi
- i = 6,7 Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the results back into CM.

57ijk Set Ai to (Bj) - (Bk)

SAi Bj - Bk



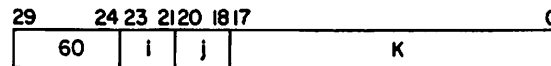
This instruction reads operands from Bj and Bk, subtracts the Bk operand from the Bj operand, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

- i = 0 No CM reference
- i = 1,2,3,4,5 Read from CM to Xi
- i = 6,7 Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

60ijk Set Bi to (Aj) + K

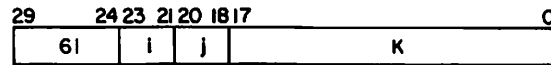
SBi Aj K



This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Aj, forms the sum of the operand plus K and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode. This instruction is for address modification in the increment registers.

61ijk Set Bi to (Bj) + K

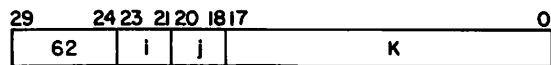
SBi Bj K



This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Bj, forms the sum of the operand plus K, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

62ijk Set Bi to (Xj) + K

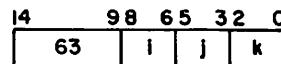
SBi Xj K



This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Xj, forms the sum of the operand plus K, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

63ijk Set Bi to (Xj) + (Bk)

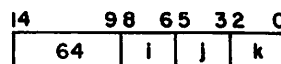
SBi Xj + Bk



This instruction reads operands from Xj and Bk, adds the operands, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

64ijk Set Bi to (Aj) + (Bk)

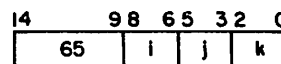
SBi Aj + Bk



This instruction reads operands from Aj and Bk, adds the operands, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

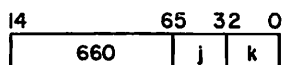
65ijk Set Bi to (Aj) - (Bk)

SBi Aj - Bk



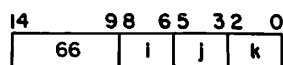
This instruction reads operands from Aj and Bk, subtracts the Bk operand from the Aj operand, and delivers the result to Bi. The difference is formed in an 18-bit one's complement mode. If the i designator is zero, this becomes a pass instruction.

660jk Read Central Memory at (Xk) to Xj CRXj Xk



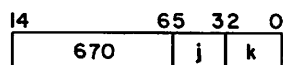
This instruction loads into Xj the word at location (Xk), where Xk is a right-justified 21-bit relative word address. Bits 21 through 59 of Xk are ignored. If the 21 bits of Xk are greater than or equal to FLC, an address out of range is detected.

661jk Set Bi to (Bj) + (Bk) SBi Bj + Bk



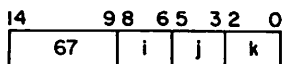
This instruction reads operands from Bj and Bk, adds the operands, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode. If the i designator is zero, this becomes a read central memory instruction.

670jk Write Xj into Central Memory at (Xk) CWXj Xk



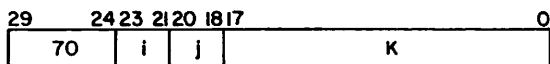
This instruction stores Xj in location (Xk) where Xk is a 21-bit relative word address. Bits 21 through 59 of Xk are ignored. If the 21 bits of Xk are greater than or equal to FLC, an address out of range is detected.

671jk Set Bi to (Bj) - (Bk) SBi Bj - Bk



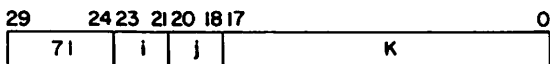
This instruction reads operands from Bj and Bk, subtracts the Bk operand from the Bj operand, and delivers the result to Bi. The difference is formed in an 18-bit one's complement mode. If the i designator is zero, this becomes a write central memory instruction.

701jK Set Xi to (Aj) + K SXi Aj K



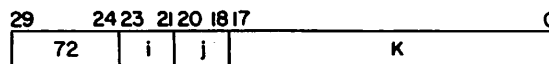
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Aj, forms the sum of the operand plus K, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

711jK Set Xi to (Bj) + K SXi Bj K



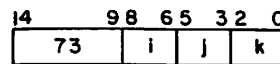
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Bj, forms the sum of the operand plus K, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

721jK Set Xi to (Xj) + K SXi Xj K



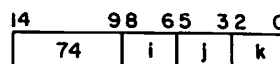
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Xj, forms the sum of the operand plus K, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

731jk Set Xi to (Xj) + (Bk) SXi Xj + Bk



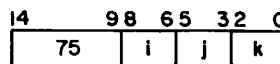
This instruction reads operands from Xj and Bk, adds the operands, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

741jk Set Xi to (Aj) + (Bk) SXi Aj + Bk



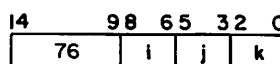
This instruction reads operands from Aj and Bk, adds the operands, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

751jk Set Xi to (Aj) - (Bk) SXi Aj - Bk



This instruction reads operands from Aj and Bk, subtracts the Bk operand from the Aj operand, and delivers the result to Xi. The difference is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

761jk Set Xi to (Bj) + (Bk) SXi Bj + Bk

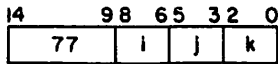


This instruction reads operands from Bj and Bk, adds the operands, and delivers the result to Xi. The sum

is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

77ijk Set Xi to (Bj) - (Bk)

SXi Bj - Bk



This instruction reads operands from Bj and Bk, subtracts the Bk operand from the Bj operand, and delivers the result to Xi. The difference is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

INSTRUCTION EXECUTION TIMING

Approximate execution times for models 835 and 855 CP instructions are listed in tables 4-3 and 4-4. These times are listed with the assumption that no conflicts occur. Execution delays result unless all the conditions listed in the timing notes column exist for the particular instruction. The numbers in the timing notes column refer to notes listed at the end of the table. Execution times are given in 56-nanosecond major cycles.

NOTE

These execution times are approximations only and subject to change without notice. Accurate timings can come only from benchmark tests. Control Data Corporation is not responsible for assumptions made based on the times listed here.

Table 4-3. Model 835 CP Instruction Timing (Sheet 1 of 4)

Instruction Code	Description	Execution Time in 56-ns Cycles	Timing Notes
00xxx	Error exit to MA or interrupt to executive mode	-	-
010xK	Return jump to K	11-14	-
011jK	Block copy Bj + K words from UEM to CM	2	-
012jK	Block copy Bj + K words from CM to UEM	2	-
013jK	Central exchange jump to Bj + K (CYBER 170 monitor flag set)	175-185	-
013xx	Monitor exchange jump to MA (CYBER 170 monitor flag clear)	175-185	-
014jk	Read one word from UEM to Xj	15-19	-
015jk	Write one word from Xj to UEM	15-19	-
016jk	Read free running counter	-	-
017jk	Illegal instruction	80-120	-
021xK	Jump to (Bi) + K	11-14	-
030jK	Branch to K if (Xj) = 0	4 or 11-14	1
031jK	Branch to K if (Xj) ≠ 0	4 or 11-14	1
032jK	Branch to K if (Xj) is positive	2 or 7-10	1
033jK	Branch to K if (Xj) is negative	2 or 7-10	1
034jK	Branch to K if (Xj) is in range	4 or 11-14	1
035jK	Branch to K if (Xj) is out of range	4 or 11-14	1
036jK	Branch to K if (Xj) is definite	4 or 11-14	1

Timing Notes:

1. First time shown if branch was not taken; second time shown if branch was taken.

Table 4-3. Model 835 CP Instruction Timing (Sheet 2 of 4)

Instruction Code	Description	Execution Time in 56-ns Cycles	Timing Notes
037jK	Branch to K if (Xj) is indefinite	4 or 11-14	1
041jK	Branch to K if (Bi) = (Bj)	6 or 11-14	1
051jK	Branch to K if (Bi) ≠ (Bj)	6 or 11-14	1
061jK	Branch to K if (Bi) ≥ (Bj)	6 or 11-14	1
071jK	Branch to K if (Bi) < (Bj)	6 or 11-14	1
101jx	Transmit (Xj) to Xi	2	-
111jk	Logical product of (Xj) and (Xk) to Xi	2	-
121jk	Logical sum of (Xj) and (Xk) to Xi	2	-
131jk	Logical difference of (Xj) and (Xk) to Xi	2	-
141xx	Transmit complement of (Xk) to Xi	2	-
151jk	Logical product of (Xj) with complement of (Xk) to Xi	4	-
161jk	Logical sum of (Xj) with complement of (Xk) to Xi	4	-
171jk	Logical difference of (Xj) with complement of (Xk) to Xi	4	-
201jk	Left shift (Xi) by jk	2	-
211jk	Right shift (Xi) by jk	2	-
221jk	Left shift (Xk) nominally (Bj) places to Xi	6 or 7-10	2
231jk	Right shift (Xk) nominally (Bj) places to Xi	6 or 7-10	2
241jk	Normalize (Xk) to Xi and Bj	6	-
251jk	Round normalize (Xk) to Xi and Bj	11-14	-
261jk	Unpack (Xk) to Xi and Bj	7-10	-
271jk	Pack (Xk) and (Bj) to Xi	5	-
301jk	Floating sum of (Xj) and (Xk) to Xi	4	-
311jk	Floating difference of (Xj) and (Xk) to Xi	4	-
321jk	Floating double-precision sum of (Xj) and (Xk) to Xi	40-60	-
331jk	Floating double-precision difference of (Xj) and (Xk) to Xi	40-60	-
341jk	Round floating sum of (Xj) and (Xk) to Xi	7-10	-
351jk	Round floating difference of (Xj) and (Xk) to Xi	7-10	-
361jk	Integer sum of (Xj) and (Xk) to Xi	2	-
371jk	Integer difference of (Xj) and (Xk) to Xi	2	-
401jk	Floating product of (Xj) and (Xk) to Xi	11-14	-

Timing Notes:

1. First time shown if branch was not taken; second time shown if branch was taken.
2. First time shown if left shift; second time shown if right shift. Type of shift depends on the sign.

Table 4-3. Model 835 CP Instruction Timing (Sheet 3 of 4)

Instruction Code	Description	Execution Time in 56-ns Cycles	Timing Notes
41ijk	Round floating product of (Xj) and (Xk) to Xi	11-14	-
42ijk	Floating double-precision product of (Xj) and (Xk) to Xi	11-14	-
43ijk	Form mask of jk bits to Xi	4	-
44ijk	Floating divide (Xj) by (Xk) to Xi	40-60	-
45ijk	Round floating divide (Xj) by (Xk) to Xi	40-60	-
460xxx- 463xxx	Pass	2	-
464jK	Move indirect	-	3
465	Move direct	-	3
466	Compare collated	-	3
467	Compare uncollated	-	3
47ixk	Population count of (Xk) to Xi	20-25	4
50ijk	Set Ai to (Aj) + K	3, 7-10, or 20-25	5
51ijk	Set Ai to (Bj) + K	3, 7-10, or 20-25	5
52ijk	Set Ai to (Xj) + K	3,7-10, or 20-25	5
53ijk	Set Ai to (Xj) + (Bk)	3, 7-10, or 20-25	5
54ijk	Set Ai to (Aj) + (Bk)	3, 7-10, or 20-25	5
55ijk	Set Ai to (Aj) - (Bk)	3, 7-10, or 20-25	5
56ijk	Set Ai to (Bj) + (Bk)	3, 7-10, or 20-25	5
57ijk	Set Ai to (Bj) - (Bk)	3, 7-10, or 20-25	5
60ijk	Set Bi to (Aj) + K	3	-
61ijk	Set Bi to (Bj) + K	3	-
62ijk	Set Bi to (Xj) + K	3	-
63ijk	Set Bi to (Xj) + (Bk)	3	-
64ijk	Set Bi to (Aj) + (Bk)	3	-
65ijk	Set Bi to (Aj) - (Bk)	3	-
660jk	Read CM at (Xk) to Xj	4	-

Timing Notes:

3. CMU instructions are simulated. For best results, recompile to avoid use of these instructions.
4. P equals the number of bits.
5. 3 cycles when i equals 0; 7-10 cycles when i equals 6 or 7; 20-25 cycles when i equals 1 through 5.

Table 4-3. Model 835 CP Instruction Timing (Sheet 4 of 4)

Instruction Code	Description	Execution Time in 56-ns Cycles	Timing Notes
66ijk	Set Bi to (Bj) + (Bk)	3	-
670jk	Write Xj into CM at (Xk)	4	-
671jk	Set Bi to (Bj) - (Bk)	3	-
701jK	Set Xi to (Aj) + K	5	-
711jK	Set Xi to (Bj) + K	5	-
721jK	Set Xi to (Xj) + K	5	-
731jk	Set Xi to (Xj) + (Bk)	5	-
741jk	Set Xi to (Aj) + (Bk)	5	-
751jk	Set Xi to (Aj) - (Bk)	5	-
761jk	Set Xi to (Bj) + (Bk)	5	-
771jk	Set Xi to (Bj) - (Bk)	5	-

Table 4-4. Model 855 CP Instruction Timing (Sheet 1 of 3)

Instruction Code	Description	Execution Time in 64-ns Cycles	Timing Notes
00xxx	Error exit to MA or interrupt to executive mode	-	-
010xK	Return jump to K	10-20	-
011jK	Block copy Bj + K words from UEM to CM	-	1
012jK	Block copy Bj + K words from CM to UEM	-	1
013jK	Central exchange jump to Bj + K (CYBER 170 monitor flag set)	125	-
013xx	Monitor exchange jump to MA (CYBER 170 monitor flag clear)	125	-
014jk	Read one word from UEM to Xj	10-20	-
015jk	Write one word from Xj to UEM	10-20	-
016jk	Read free running counter	10-20	-
017jk	Illegal instruction	66-75	-
021xK	Jump to (Bi) + K	6	-
030jK	Branch to K if (Xj) = 0	1 or 6	2
031jK	Branch to K if (Xj) ≠ 0	1 or 6	2
032jK	Branch to K if (Xj) is positive	1 or 6	2
033jK	Branch to K if (Xj) is negative	1 or 6	2
034jK	Branch to K if (Xj) is in range	1 or 6	2
035jK	Branch to K if (Xj) is out of range	1 or 6	2
036jK	Branch to K if (Xj) is definite	1 or 6	2
037jK	Branch to K if (Xj) is indefinite	1 or 6	2
041jK	Branch to K if (Bi) = (Bj)	1 or 6	2
051jK	Branch to K if (Bi) ≠ (Bj)	1 or 6	2
061jK	Branch to K if (Bi) ≥ (Bj)	1 or 6	2
071jK	Branch to K if (Bi) < (Bj)	1 or 6	2
101jx	Transmit (Xj) to Xi	1	-
111jK	Logical product of (Xj) and (Xk) to Xi	1	-
121jk	Logical sum of (Xj) and (Xk) to Xi	1	-
131jk	Logical difference of (Xj) and (Xk) to Xi	1	-
141xk	Transmit complement of (Xk) to Xi	1	-
151jk	Logical product of (Xj) with complement of (Xk) to Xi	1	-

Timing Notes:

1. Execution time varies depending on number of words and number of 16-word blocks. Execution time is: $36 + 24 * (\text{number of words} + 5 * (\text{number of 16-word blocks}))$. Map hit rate of 100% and cache hit rate of 75% is assumed.
2. First time shown if branch was taken; second time shown if branch was not taken.

Table 4-4. Model 855 CP Instruction Timing (Sheet 2 of 3)

Instruction Code	Description	Execution Time in 64-ns Cycles	Timing Notes
16ijk	Logical sum of (Xj) with complement of (Xk) to Xi	1	-
17ijk	Logical difference of (Xj) with complement of (Xk) to Xi	1	-
20ijk	Left shift (Xi) by jk	4-5	-
21ijk	Right shift (Xi) by jk	1	-
22ijk	Left shift (Xk) nominally (Bj) places to Xi	4-5	-
23ijk	Right shift (Xk) nominally (Bj) places to Xi	4-5	-
24ijk	Normalize (Xk) to Xi and Bj	2	-
25ijk	Round normalize (Xk) to Xi and Bj	2	-
26ijk	Unpack (Xk) to Xi and Bj	2	-
27ijk	Pack (Xk) and (Bj) to Xi	1	-
30ijk	Floating sum of (Xj) and (Xk) to Xi	2	-
31ijk	Floating difference of (Xj) and (Xk) to Xi	2	-
32ijk	Floating double-precision sum of (Xj) and (Xk) to Xi	2	-
33ijk	Floating double-precision difference of (Xj) and (Xk) to Xi	2	-
34ijk	Round floating sum of (Xj) and (Xk) to Xi	2	-
35ijk	Round floating difference of (Xj) and (Xk) to Xi	2	-
36ijk	Integer sum of (Xj) and (Xk) to Xi	1	-
37ijk	Integer difference of (Xj) and (Xk) to Xi	1	-
40ijk	Floating product of (Xj) and (Xk) to Xi	3	-
41ijk	Round floating product of (Xj) and (Xk) to Xi	3	-
42ijk	Floating double-precision product of (Xj) and (Xk) to Xi	3	-
43ijk	Form mask of jk bits to Xi	2	-
44ijk	Floating divide (Xj) by (Xk) to Xi	10-20	-
45ijk	Round floating divide (Xj) by (Xk) to Xi	10-20	-
460xx- 463xx	Pass	1	-
464jK	Move indirect	-	3
465	Move direct	-	3
466	Compare collated	-	3
467	Compare uncollated	-	3
47ixk	Population count of (Xk) to Xi	10-20	-

Timing Notes:

3. CMU instructions are simulated. For best results, recompile to avoid use of these instructions.

Table 4-4. Model 855 CP Instruction Timing (Sheet 3 of 3)

Instruction Code	Description	Execution Time in 64-ns Cycles	Timing Notes
501jK	Set A _i to (A _j) + K	1	-
511jK	Set A _i to (B _j) + K	4-5	4
521jK	Set A _i to (X _j) + K	4-5	4
531jk	Set A _i to (X _j) + (B _k)	4-5	4
541jk	Set A _i to (A _j) + (B _k)	4-5	4
551jk	Set A _i to (A _j) - (B _k)	4-5	4
561jk	Set A _i to (B _j) + (B _k)	2	5
571jk	Set A _i to (B _j) - (B _k)	2	5
601jK	Set B _i to (A _j) + K	1	-
611jK	Set B _i to (B _j) + K	1	-
621jK	Set B _i to (X _j) + K	1	-
631jk	Set B _i to (X _j) + (B _k)	1	-
641jk	Set B _i to (A _j) + (B _k)	1	-
651jk	Set B _i to (A _j) - (B _k)	1	-
660jk	Read CM at (X _k) to X _j	4-5	4
661jk	Set B _i to (B _j) + (B _k)	1	6
670jk	Write X _j into CM at (X _k)	1	-
671jk	Set B _i to (B _j) - (B _k)	1	-
701jK	Set X _i to (A _j) + K	1	-
711jK	Set X _i to (B _j) + K	1	-
721jK	Set X _i to (X _j) + K	1	-
731jk	Set X _i to (X _j) + (B _k)	1	-
741jk	Set X _i to (A _j) + (B _k)	1	-
751jk	Set X _i to (A _j) - (B _k)	1	-
761jk	Set X _i to (B _j) + (B _k)	1	-
771jk	Set X _i to (B _j) - (B _k)	1	-

Timing Notes:

4. No map miss and cache hit rate of 75% is assumed.
5. No map miss is assumed.
6. $i \neq 0$ is assumed.

PP INSTRUCTIONS

PP INSTRUCTION FORMATS

Figure 4-2 shows PP instruction formats. PP instructions are 16 or 32 bits long. In instruction descriptions, the operation code is given either by two or three octal digits. The third digit, when used, indicates the state of the s-bit (zero or one) in I/O instructions (refer to table 4-5).

The upper 4 bits of the PP instructions must be zero to ensure that the instructions operate as defined in this section.

Table 4-5. PP Nomenclature

Term	Description
Opcode	Specifies instruction operation code.
s	Specifies I/O instruction subcode.
c	Specifies channel number.
A	Refers to the A register (arithmetic register) or the content of the A register.
(A)	Refers to the content of the word at the CM address specified by the A register.
P	Refers to the P register or to the content of the P register (program address register).
R	Refers to the R register or to the content of the R register (relocation register).
(d)	Refers to the content of the word at the PP memory address specified by the d field (direct mode).
((d))	Refers to the content of the word at the PP memory address specified by the content of the word at the PP memory address specified by the d field (indirect mode).
m + (d)	Refers to the PP memory address specified by the m field indexed by the content of the word at the PP memory addressed specified by the d field.
(m + (d))	Refers to the content of the word at the PP memory address specified by the m field indexed by the content of the word at the PP memory address specified by the d field (memory mode).

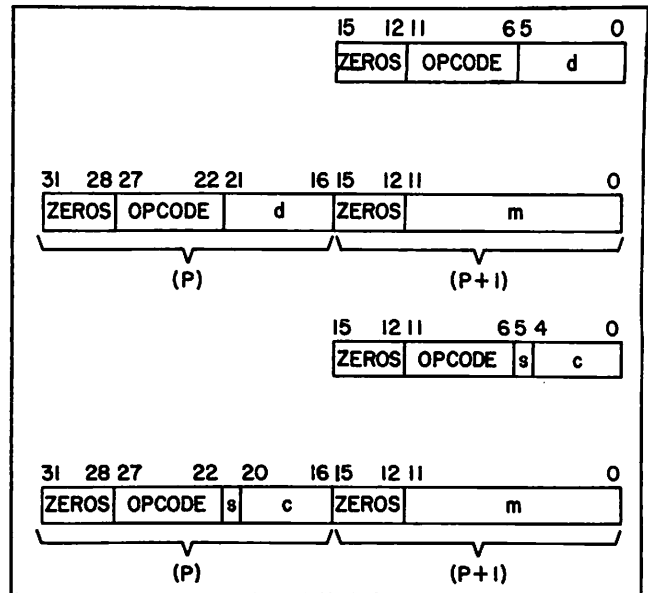


Figure 4-2. PP Instruction Formats

PP DATA FORMAT

Figure 4-3 shows PP data format and how 12-bit data is packed into 64-bit CM words or unpacked from 64-bit CM words.

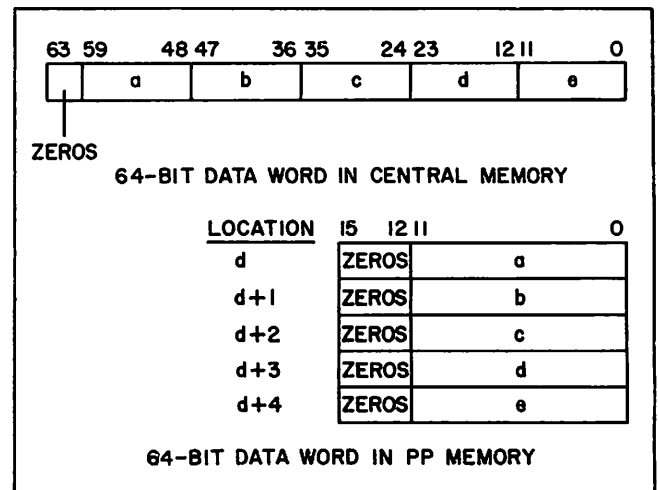


Figure 4-3. PP Data Format

PP RELOCATION REGISTER FORMAT

Figure 4-4 shows PP relocation (R) register format. This register is loaded-from/stored-into PP memory by instructions 24 and 25 (load/store R register).

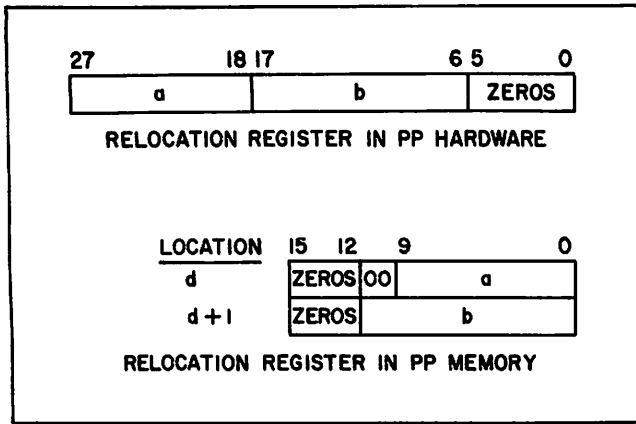
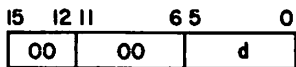


Figure 4-4. PP Relocation (R) Register Format

PP INSTRUCTION DESCRIPTIONS

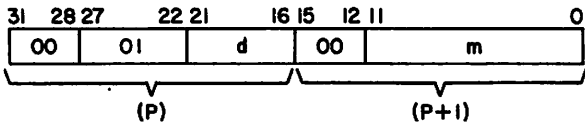
PP instruction descriptions are in numerical order. Refer to section 5, Programming Information.

00xx Pass PSN



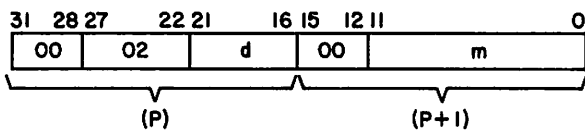
This instruction specifies that no operation is to be performed. The instruction provides a means of padding out a program.

01dm Long jump to m + (d) LJM m,d



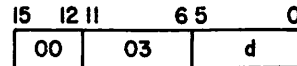
This instruction jumps to the address given by m plus the content of location d. If d equals zero, m is not modified.

02dm Return Jump to m + (d) RJM m,d



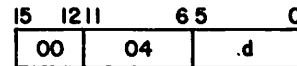
This instruction jumps to the address given by m plus the content of location d. If d equals zero, m is not modified. The current program address (P) plus 2 is stored at the jump address. The next instruction starts at the jump address plus 1. The subprogram exits with a long jump or normal sequencing to the jump address minus 1, which in turn contains a long jump, 0100. This returns the original program address plus 2 to the P register.

03d Unconditional Jump d UJN d



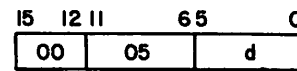
This instruction provides an unconditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. The value of d is added to the current program address. If d is positive (01 through 37), 0001 through 0037 is added, and the jump is forward. If d is negative (40 through 76), 7740 through 7776 is added, and the jump is backward. When d equals 00 or 77, the PP hangs; a deadstart is required to restart the PP.

04d Zero Jump d ZJN d



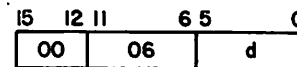
This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the content of the A register is zero, the jump is taken. If the content of A is nonzero, the next instruction executes from P plus 1. Negative zero (777777) is treated as nonzero. For interpretation of d, refer to the 03 instruction.

05d Nonzero Jump d NJN d



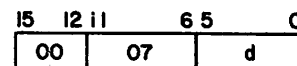
This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the content of the A register is nonzero, the jump is taken. If the content of A is zero, the next instruction executes from P plus 1. Negative zero (777777) is treated as nonzero. For interpretation of d, refer to the 03 instruction.

06d Plus Jump d PJN d



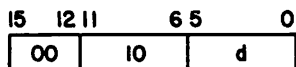
This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the sign of the A register is positive, the jump is taken. If the sign of A is negative, the next instruction executes from P plus 1. Positive zero is treated as a positive quantity. Negative zero is treated as a negative quantity. For interpretation of d, refer to the 03 instruction.

07d Minus Jump d MJN d



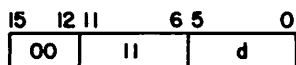
This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the content of the A register is negative, the jump is taken. If the content of A is positive, the next instruction executes from P plus 1. Positive zero is treated as a positive quantity. Negative zero is treated as a negative quantity. For interpretation of d, refer to the 03 instruction.

10d Shift d SHN d



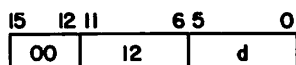
This instruction shifts the content of the A register right or left d places. If d is positive (00 through 37), the shift is left circular. If d is negative (40 through 77), the shift is right (end-off with no sign extension). Thus, d equal to 06 requires a left shift of six places; d equal to 71 requires a right shift of six places.

11d Logical Difference d LMN d



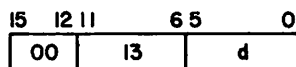
This instruction forms the bit-by-bit logical difference of d and the lower 6 bits of A in the register in A. This is equivalent to complementing individual bits of A that correspond to bits of d that are one. The upper 12 bits of A are not altered.

12d Logical Product d LPN d



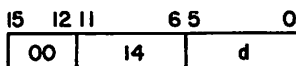
This instruction forms the bit-by-bit logical product of d and the lower 6 bits of the A register and leaves this quantity in the lower 6 bits of A. The upper 12 bits of A are zero.

13d Selective Clear d SCN d



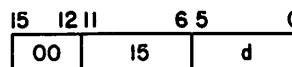
This instruction clears any of the lower 6 bits of the A register where corresponding bits of d are one. The upper 12 bits of A are not altered.

14d Load d LDN d



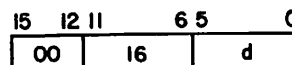
This instruction clears the A register and loads d. The upper 12 bits of A are zero.

15d Load Complement d LCN d



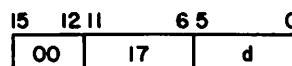
This instruction clears the A register and loads the complement of d. The upper 12 bits of A are one.

16d Add d ADN d



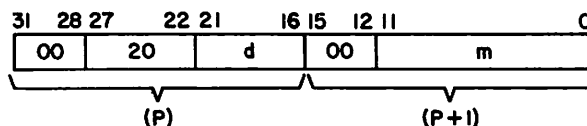
This instruction adds d (treated as a 6-bit positive quantity) to the content of the A register.

17d Subtract d SBN d



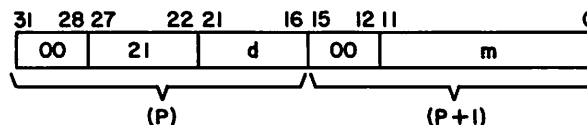
This instruction subtracts d (treated as a 6-bit positive quantity) from the content of the A register.

20dm Load dm LDC dm



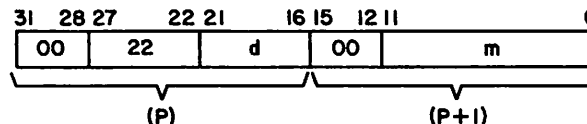
This instruction clears the A register and loads an 18-bit quantity consisting of d as the upper 6 bits and m as the lower 12 bits. The content of the location (P plus 1) which follows the present program address (P) is read to provide m.

21dm Add dm ADC dm



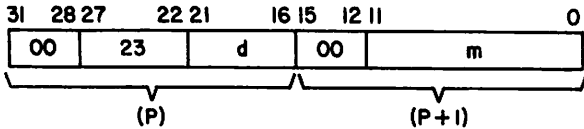
This instruction adds to the A register the 18-bit quantity consisting of d as the upper 6 bits and m as the lower 12 bits. The content of the location (P plus 1) which follows the present program address (P) is read to provide m.

22dm Logical Product dm LPC dm



This instruction forms the bit-by-bit logical product of the content of the A register and the 18-bit quantity *dm* in A. The upper 6 bits of this quantity consist of *d*, and the lower 12 bits are the content of the location (*P* plus 1), which follows the present program address (*P*).

23dm Logical Difference *dm* LMC *dm*



This instruction forms the bit-by-bit logical difference of the content of the A register and the 18-bit quantity *dm* in A. This is equivalent to complementing individual bits of A which correspond to bits of *dm* that are one. The upper 6 bits of the quantity consist of *d*, and the lower 12 bits are the content of the location (*P* plus 1), which follows the present program address (*P*).

24d Load R Register LRD *d*

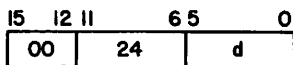


Figure 4-4 shows R register format. If *d* is not equal to 0, this instruction loads the R register from PP memory locations *d* (rightmost 10 bits) and *d* plus 1 (next 12 bits). If *d* equals 0, the instruction is a pass.

25d Store R Register SRD *d*

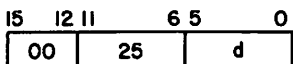
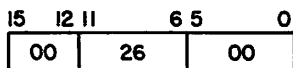


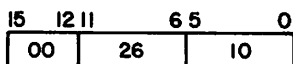
Figure 4-4 shows R register format. If *d* is not equal to 0, this instruction stores the R register into PP locations *d* (rightmost 10 bits) and *d* plus 1 (next 12 bits). If *d* equals 0, the instruction is a pass.

2600 Exchange Jump EXN



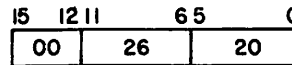
This instruction causes an unconditional exchange jump in the CP, leaving the CP CYBER 170 monitor flag unaltered. The new CYBER 170 exchange package begins at central memory location R plus A when the leftmost bit in A is set. When this bit is clear, A specifies the address. The PP waits until the exchange has been completed before proceeding with the next instruction.

2610 Monitor Exchange Jump MXN



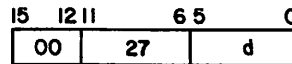
If the CP is in the CYBER 170 monitor mode, this instruction is a pass. If the CP is in the CYBER 170 job mode, it causes a CYBER 170 exchange jump in the CP, switching the CP to the CYBER 170 monitor mode (MF equals 1). The new CYBER 170 exchange package begins at central memory location R plus A when the leftmost bit in A is set. When this bit is clear, A specifies the address. The PP waits until the exchange has been completed before proceeding with the next instruction.

2620 Monitor Exchange Jump to MA MAN



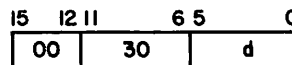
If the CP is in CYBER 170 monitor mode, this instruction is a pass. If the CP is in CYBER 170 job mode, it causes a CYBER 170 exchange jump in the CP, switching the CP to CYBER 170 monitor mode (MF equals 1). The new CYBER 170 exchange package begins at the absolute address given in the MA field of the outgoing CYBER 170 exchange package. The PP waits until the exchange has been completed before proceeding with the next instruction.

27d Pass KPT *d*



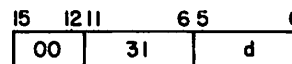
This instruction is no operation. However, it generates a pulse to a testpoint (keypoint) for optional monitoring by external equipment.

30d Load (*d*) LDD *d*



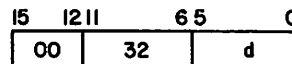
This instruction clears the A register and loads the content at location *d*. The upper 6 bits of A are zero.

31d Add (*d*) ADD *d*



This instruction adds the content at location *d* (treated as a 12-bit positive quantity) to the A register.

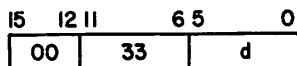
32d Subtract (*d*) SBD *d*



This instruction subtracts the content at location *d* (treated as a 12-bit positive quantity) from the A register.

33d Logical Difference (d)

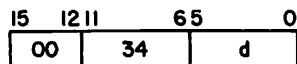
LMD d



This instruction forms in the A register the bit-by-bit logical difference of the lower 12 bits of the A register and the content at location d. This is equivalent to complementing individual bits of A which correspond to bits in location d that are ones. The upper 6 bits are not altered.

34d Store (d)

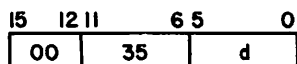
STD d



This instruction stores the lower 12 bits of the A register at location d.

35d Replace Add (d)

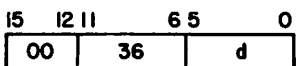
RAD d



This instruction adds the quantity at location d to the content of the A register and stores the lower 12 bits of the result at location d. The result remains in A at the end of the operation and the original content of A is destroyed.

36d Replace Add One (d)

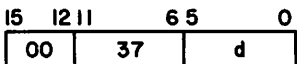
AOD d



This instruction replaces the quantity at location d with its original value plus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

37d Replace Subtract One (d)

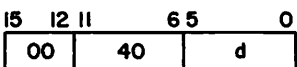
SOD d



This instruction replaces the quantity at location d with its original value minus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

40d Load ((d))

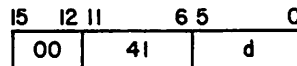
LDI d



This instruction clears the A register and loads a 12-bit quantity that is obtained by indirect addressing. The upper 6 bits of A are zero. Location d is read from PPM, and the word read is used as the operand address.

41d Add ((d))

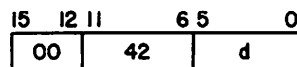
ADI d



This instruction adds to the content of the A register a 12-bit operand (treated as a positive quantity) obtained by indirect addressing. Location d is read from PPM, and the word read is used as the operand address.

42d Subtract ((d))

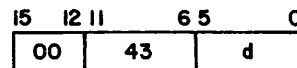
SBI d



This instruction subtracts from the A register a 12-bit operand (treated as a positive quantity) obtained by indirect addressing. Location d is read from PPM, and the word read is used as the operand address.

43d Logical Difference ((d))

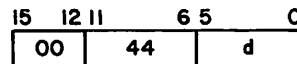
LMI d



This instruction forms in the A register the bit-by-bit logical difference of the lower 12 bits of the A register and the 12-bit operand read by indirect addressing. Location d is read from PPM, and the word read is used as the operand address. The upper 6 bits of A are not altered.

44d Store ((d))

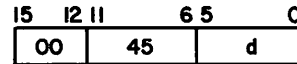
STI d



This instruction stores the lower 12 bits of the A register at the location specified by the content of location d.

45d Replace Add ((d))

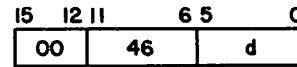
RAI d



This instruction adds the operand, which is obtained from the location specified by the content at location d, to the content of the A register. The lower 12 bits of the sum replace the original operand. The result remains in A at the end of the operation.

46d Replace Add One ((d))

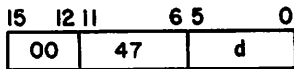
AOI d



This instruction replaces the operand, which is obtained from the location specified by the content at location d, by its original value plus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

47d Replace Subtract One ((d))

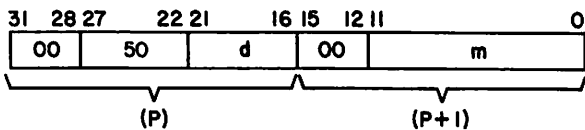
SOI d



This instruction replaces the operand, which is obtained from the location specified by the content at location d, by its original value minus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

50dm Load (m + (d))

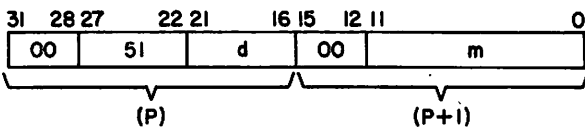
LDM m,d



This instruction clears the A register and loads a 12-bit quantity. The upper 6 bits of A are zeros. The 12-bit operand is obtained by indexed direct addressing. The quantity m, read from PPM location P plus 1, serves as the base operand address to which the content of d is added. If d equals 0, the operand address is m, but if d is not equal to 0, m plus the content in d is the operand address. Thus, location d may be used as an index quantity to modify operand addresses.

51dm Add (m + (d))

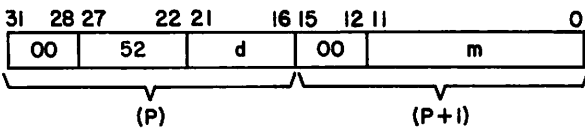
ADM m,d



This instruction adds the 12-bit operand (treated as a positive quantity) read by indexed direct addressing (refer to 50 instruction) to the A register.

52dm Subtract (m + (d))

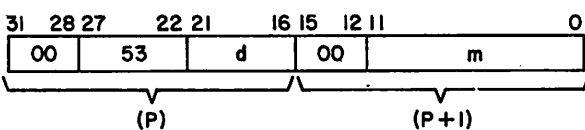
SBM m,d



This instruction subtracts the 12-bit operand (treated as a positive quantity) read by indexed direct addressing (refer to the 50 instruction) from the A register.

53dm Logical Difference (m + (d))

LMM m,d

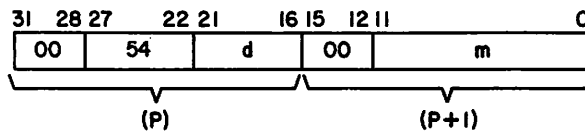


This instruction forms the bit-by-bit logical difference of the lower 12 bits of the A register and

a 12-bit operand obtained by indexed direct addressing (refer to the 50 instruction) in A. The upper 6 bits of A are not altered.

54dm Store (m + (d))

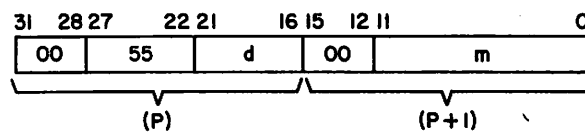
STM m,d



This instruction stores the lower 12 bits of the A register in the location determined by indexed direct addressing (refer to 50 instruction).

55dm Replace Add (m + (d))

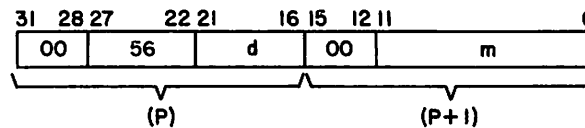
RAM m,d



This instruction adds the operand, which is obtained from the location determined by indexed direct addressing (refer to the 50 instruction), to the A register. The lower 12 bits of the sum replace the original operand in PPM. The result remains in A at the end of the operation, and the original content of A is destroyed.

56dm Replace Add One (m + (d))

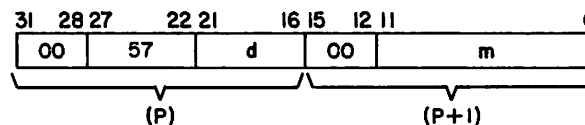
AOM m,d



This instruction replaces the operand, which is obtained from the location determined by indexed direct addressing (refer to the 50 instruction), by its original value plus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

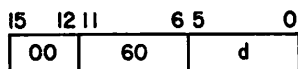
57dm Replace Subtract One (m + (d))

SOM m,d



This instruction replaces the operand, which is obtained from the location determined by indexed direct addressing (refer to the 50 instruction), by its original value minus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

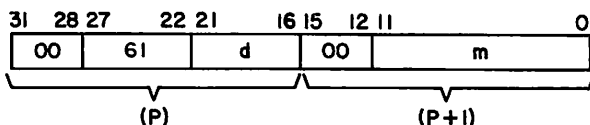
60d Central Read from (A) to d CRD d



This instruction disassembles one 60-bit word from central memory into five 12-bit words and stores these in five consecutive PP memory locations, beginning with the leftmost 12 bits of the 60-bit word. The parameters of the transfer are as follows:

If bit 17 of A is zero, A bits 0 through 16 contain the absolute address of the 60-bit word transferred. If bit 17 of A is one, hardware adds relocation register R to zero-extended A bits 0 through 16 to obtain the absolute address of the 60-bit word transferred. For further information, refer to R Register under Input/Output Unit in section 2, and PP Relocation Register Format at the beginning of this section on PP Instructions. Field d gives the PP location which receives the first 12-bit word transferred. PP memory addressing is cyclic and location 0000 follows location 7777.

61dm Central Read (d) Words from (A) to m CRM d,m



PP location 0000 is used by hardware. This instruction disassembles 60-bit words from central memory into 12-bit words, and places these in consecutive PP memory locations, beginning with the leftmost 12 bits of the first 60-bit word. The parameters of the transfer are as follows:

If bit 17 of A is zero, A bits 0 through 16 contain the absolute address of the first 60-bit word transferred. If bit 17 of A is one, hardware adds relocation register R to zero-extended A bits 0 through 16 to obtain the absolute address of the first 60-bit word transferred. For further information, refer to R Register under Input/Output Unit in section 2, and PP Relocation Register Format at the beginning of this section on PP Instructions. PP location d must contain the number of 60-bit words transferred. Field m gives the PP location into which the first 12-bit word is placed.

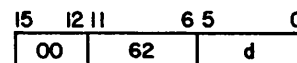
This instruction stores P plus 1 into PP location 0000 before beginning the transfer. After the transfer is completed, the next instruction is taken from one plus whatever address is stored in location 0000. If the transfer overwrites location 0000, execution resumes at the location specified by (0000) plus 1 and results are undefined. (PP memory addressing is cyclic and location 0000 follows location 7777.)

The A register is incremented by one after each 60-bit word is read from central memory. If the incrementing changes A bit 17, the central memory addressing is switched between direct address and relocation address modes. Refer to Central Memory Addressing by PPs, section 5. After the transfer is

completed, the A register contains either the address of the last word transferred plus one (direct addressing) or the same address less the contents of the relocation address register (relocation addressing), except as follows:

If the last word transferred is from a relative address 377776₈ and relocation is in effect, then the A register is cleared, and the value returned in A may not point to the last word transferred plus one.

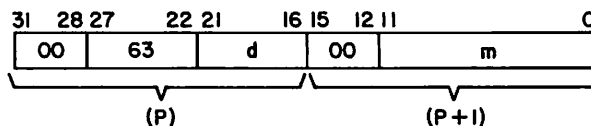
62d Central Write to (A) from d CWD d



This instruction assembles five 12-bit words from consecutive PP memory locations into one 60-bit word and stores the 60-bit word in central memory. The first 12-bit word is stored in the leftmost 12 bits of the 60-bit word. (PP memory addressing is cyclic and location 0000 follows location 7777.) The parameters of the transfer are as follows:

If bit 17 of A is zero, A bits 0 through 16 contain the absolute address of the 60-bit word stored. If bit 17 of A is one, hardware adds relocation register R to zero-extended A bits 0 through 16 to obtain the absolute address of the 60-bit word stored. For further information, refer to R Register under Input/Output Unit in section 2, and PP Relocation Register Format at the beginning of this section on PP Instructions. Field d gives the PP location of the first 12-bit word transferred. The transfer is subject to the CM bounds test.

63dm Central Write (d) Words to (A) from m CWM m,d



PP location 0000 is used by hardware. This instruction assembles 12-bit words from consecutive PP memory locations into 60-bit words and stores these in central memory. The first 12-bit word is stored in the leftmost 12 bits of the 60-bit word. (PP memory addressing is cyclic and location 0000 follows location 7777.) The parameters of the transfer are as follows:

If bit 17 of A is zero, A bits 0 through 16 contain the absolute address of the first 60-bit word transferred. If bit 17 of A is one, hardware adds relocation register R to zero-extended A bits 0 through 16 to obtain the absolute address of the first 60-bit word transferred. For further information, refer to R Register under Input/Output Unit in section 2, and in PP Relocation Register Format at the beginning of this section on PP Instructions. PP location d must contain the number of 60-bit words transferred. Field m gives the PP location from where the first 12-bit word is obtained. The transfer is subject to the CM bounds test.

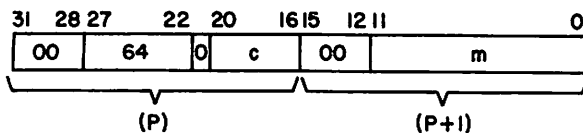
This instruction stores P plus 1 into PP location 0000 before beginning the transfer. After the transfer is completed, the next instruction is taken

from one plus whatever address is stored in location 0000.

The A register is incremented by one after each 60-bit word is written into central memory. If the incrementing changes A bit 17, the central memory addressing is switched between direct address and relocation address modes. Refer to Central Memory Addressing by PPs, section 5. After the transfer is completed, the A register contains either the address of the last word transferred plus one (direct addressing), or the same address less the contents of the relocation address register (relocation addressing), except as follows:

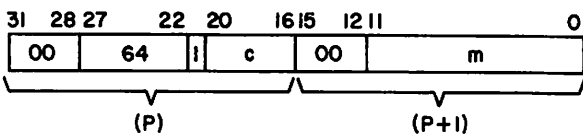
If the last word transferred is from a relative address 377776g and relocation is in effect, then the A register is cleared, and the value returned in A may not point to the last word transferred plus one.

640cm Jump to m if Channel c Active AJM m,c



If channel c is active this instruction causes a jump to m. Otherwise, it is a pass.

641cm Test and Set Channel c Flag SCF m,c

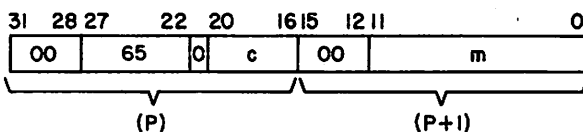


If the channel c flag is set, this instruction causes a jump to m. If the channel c flag is clear, it sets this flag and continues with the next instruction. When m is set to P plus 2, the channel flag is unconditionally set when the program reaches P plus 2.

If two or more PPs simultaneously issue this instruction for the same channel, the conflict is resolved as follows:

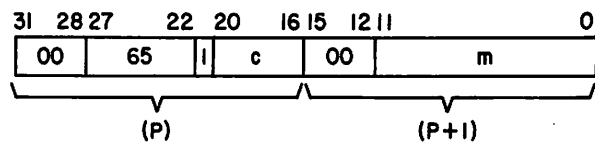
If one of the competing channels is channel 17 (maintenance channel), the PP in the lowest physical level sees the true condition of the flag; the other conflicting PPs see the flag set (and hence take a jump). If the competing channel is any other channel, software must resolve the conflict. Any five consecutively numbered PPs (in the same barrel) issue instructions at different times.

650cm Jump to m if Channel c Inactive IJM m,c



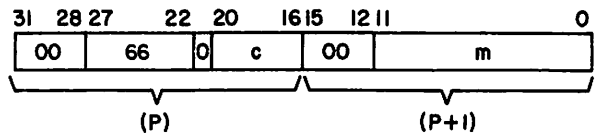
This instruction provides a conditional jump to a new address specified by m. The jump is taken if the channel specified by c is inactive. The next instruction is at P plus 2 if the channel is active.

651cm Clear Channel c Flag CCF m,c



This instruction clears the channel c flag. The m field is required but not used.

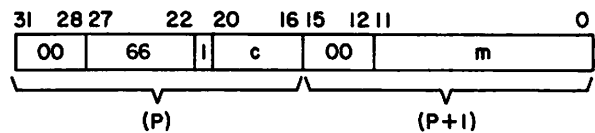
660cm Jump to m if Channel c Full FJM m,c



This instruction provides a conditional jump to a new address specified by m. The jump is taken if the channel designated by c is full. The next instruction is at P plus 2 if the channel is empty.

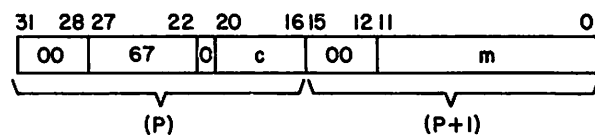
An input channel is full when the input equipment places a word in the channel and that word has not been accepted by a PP. The channel is empty when a word has been accepted. An output channel is full when a PP places a word on the channel. The channel is empty when the output equipment accepts the word.

661cm Jump to m if Channel c Error Flag Set SFM m,c



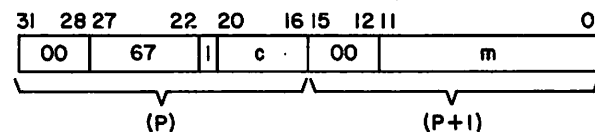
If the channel c error flag is set, this instruction clears the error flag and causes a jump to m. If this error flag is clear, the instruction is a pass. When m is set to P plus 2, the channel error flag is unconditionally cleared when the program reaches P plus 2.

670cm Jump to m if Channel c Empty EJM m,c



This instruction provides a conditional jump to a new address specified by m. The jump is taken if the channel specified by c is empty. The next instruction is at P plus 2 if the channel is full. Refer to 660 instruction for explanation of full and empty.

671cm Jump to m if Channel c Error Flag Clear CFM m,c

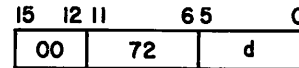


If the channel c error flag is clear, this instruction causes a jump to m. If this error flag is set, the instruction clears the error flag and proceeds with the next instruction. When m is set

to P plus 2, the channel error flag is unconditionally cleared when the program reaches P plus 2.

72d Output from A on Channel d

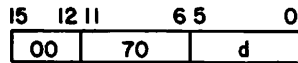
OAN d



This instruction transfers a word from the A register (lower 12 bits) to output channel d.

70d Input to A from Channel d

IAN d



This instruction transfers a word from input channel d to the lower 12 bits of the A register. The upper 6 bits of A are cleared to zero.

NOTE

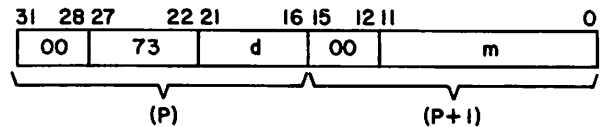
If bit 5 of d is clear and the channel is inactive, this instruction hangs the PP, waiting for the channel to go active and full, if executed. If bit 5 of d is set and the channel is inactive, the program continues at P plus 1. The word is not transferred.

NOTE

If bit 5 of d is clear and the channel is inactive, this instruction hangs the PP, waiting for the channel to go active and full, if executed. If bit 5 of d is set and the channel is inactive or is deactivated before a full is received, the instruction exits. The word is not accepted, and the A register clears.

73dm Output A Words from m on Channel d

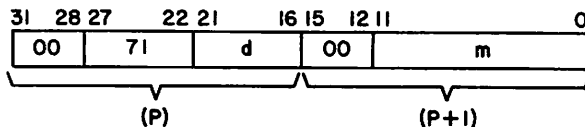
OAM m,d



This instruction transfers a block of words from PPM to channel d. The first word is read from the address specified by m. The A register holds the number of words to be sent. A reduces by one as each word is read. The output operation completes when A equals zero or the channel becomes inactive.

71dm Input A Words to m from Channel d

IAM m,d



This instruction transfers a block of 12-bit words from input channel d to PPM. The first word goes to the PPM address specified by m. The A register holds the block length. A reduces by one as each word is read. The input operation completes when A equals zero or the data channel becomes inactive. If the operation terminates by the channel becoming inactive, the next storage location in PPM is set to zero. However, the word count is not affected by this empty word. Therefore, A holds the block length minus the number of real data words read.

During this instruction, address 0000 temporarily holds P while m is held in the P register. P advances by one to give the address of the next word as each word is read from the PPM.

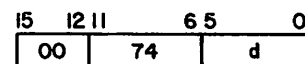
NOTE

If this instruction executes when the data channel is inactive, no output operation is accomplished, and the program continues at P plus 2.

During this instruction, address 0000 temporarily holds P while m is held in the P register. P advances by one to hold the address for the next word as each word is stored.

74d Activate Channel d

ACN d



This instruction activates the channel specified by d and sends the active signal on the channel to equipment connected to the channel. Activating a channel, which must precede a 70 through 73 instruction, prepares I/O equipment for the exchange of data.

NOTE

If this instruction executes when the data channel is inactive, no input operation is accomplished, and the program continues at P plus 2. However, the location specified by m is set to zero.

Table 4-6. PP Instruction Timing (Sheet 1 of 3)

Instruction Code	Description	Execution Time in 250-ns Cycles	Timing Notes
00xx	Pass	1	-
01dm	Long jump to m + (d)	3	-
02dm	Return jump to m + (d)	4	-
03d	Unconditional jump d	1	-
04d	Zero jump d	1	-
05d	Nonzero jump d	1	-
06d	Plus jump d	1	-
07d	Minus jump d	1	-
10d	Shift d	1	-
11d	Logical difference d	1	-
12d	Logical product d	1	-
13d	Selective clear d	1	-
14d	Load d	1	-
15d	Load complement d	1	-
16d	Add d	1	-
17d	Subtract d	1	-
20dm	Load dm	2	-
21dm	Add dm	2	-
22dm	Logical product dm	2	-
23dm	Logical difference dm	2	-
24d	Load R register from (d) and (d) + 1	3	-
25d	Store R register at (d) and (d) + 1	4	-
260x	Exchange jump	2	1
261x	Monitor exchange jump	2	1
262x	Monitor exchange jump to MA	2	1
27d	Pass	1	-
30d	Load (d)	2	-
31d	Add (d)	2	-
32d	Subtract (d)	2	-
33d	Logical difference (d)	2	-
34d	Store (d)	2	-
35d	Replace add (d)	4	-

Timing Notes:

1. No assembly-disassembly unit (ADU) conflicts and no outstanding CYBER 170 exchange jump request in the ADU.

Table 4-6. PP Instruction Timing (Sheet 2 of 3)

Instruction Code	Description	Execution Time in 250-ns Cycles	Timing Notes
36d	Replace add one (d)	5	-
37d	Replace subtract one (d)	5	-
40d	Load ((d))	3	-
41d	Add ((d))	3	-
42d	Subtract ((d))	3	-
43d	Logical difference ((d))	3	-
44d	Store ((d))	3	-
45d	Replace add ((d))	5	-
46d	Replace add one ((d))	6	-
47d	Replace subtract one ((d))	6	-
50dm	Load (m + (d))	4	-
51dm	Add (m + (d))	4	-
52dm	Subtract (m + (d))	4	-
53dm	Logical difference (m + (d))	4	-
54dm	Store (m + (d))	4	-
55dm	Replace add (m + d))	6	-
56dm	Replace add one (m + (d))	7	-
57dm	Replace subtract one (m + (d))	7	-
60d	Central read from (A) to d	12	2
61dm	Central read (d) words from (A) to m	-	2,3
62d	Central write to (A) from d	6	2
63dm	Central write (d) words to (A) from m	-	2,4
640cm	Jump to m if channel c active	2	-
641cm	Test and set channel c flag	2	-
650cm	Jump to m if channel c inactive	2	-
651cm	Clear channel c flag	2	-
660cm	Jump to m if channel c full	2	-
661cm	Jump to m if channel c error flag set	2	-

Timing Notes:

2. No ADU conflicts. No central memory conflicts. Add a possible trip due to resynchronization (CM read instructions only).
3. 7 major cycles for instruction set-up and instruction exit. 5 major cycles for every CM word.
4. 6 major cycles for instruction set-up and instruction exit. 5 major cycles for every CM word.

Table 4-6. PP Instruction Timing (Sheet 3 of 3)

Instruction Code	Description	Execution Time in 250-ns Cycles	Timing Notes
670cm	Jump to m if channel c empty	2	-
671cm	Jump to m if channel c error flag clear	2	-
70d	Input to A from channel d	2	-
71dm	Input A words to m from channel d	-	5
72d	Output from A on channel d	2	-
73dm	Output (A) words from m on channel d	-	5
74d	Activate channel d	2	-
75d	Deactivate channel d	2	-
76d	Function A on channel d	2	-
77dm	Function m on channel d	2	-
<p>Timing Notes:</p> <p>5. 5 major cycles for instruction set-up and exit. 1 major cycle per word (nonconflict case) or 2 major cycles per word (conflict case).</p> <p>Nonconflict case is when two PPs communicating to each other are not in the slot at the same time.</p> <p>Conflict case is when two PPs communicating with each other are in the slot at the same time.</p>			

Table 1. Daily water consumption by sex and age

Sex	Age (years)	Water consumption (liters/day)	Standard deviation
Male	15-17	2.5	0.5
Male	18-20	3.0	0.6
Male	21-23	3.5	0.7
Male	24-26	4.0	0.8
Male	27-29	4.5	0.9
Male	30-32	5.0	1.0
Male	33-35	5.5	1.1
Male	36-38	6.0	1.2
Male	39-41	6.5	1.3
Male	42-44	7.0	1.4
Male	45-47	7.5	1.5
Male	48-50	8.0	1.6
Male	51-53	8.5	1.7
Male	54-56	9.0	1.8
Male	57-59	9.5	1.9
Male	60-62	10.0	2.0
Male	63-65	10.5	2.1
Male	66-68	11.0	2.2
Male	69-71	11.5	2.3
Male	72-74	12.0	2.4
Male	75-77	12.5	2.5
Male	78-80	13.0	2.6
Male	81-83	13.5	2.7
Male	84-86	14.0	2.8
Male	87-89	14.5	2.9
Male	90-92	15.0	3.0
Male	93-95	15.5	3.1
Male	96-98	16.0	3.2
Male	99-101	16.5	3.3
Male	102-104	17.0	3.4
Male	105-107	17.5	3.5
Male	108-110	18.0	3.6
Male	111-113	18.5	3.7
Male	114-116	19.0	3.8
Male	117-119	19.5	3.9
Male	120-122	20.0	4.0
Male	123-125	20.5	4.1
Male	126-128	21.0	4.2
Male	129-131	21.5	4.3
Male	132-134	22.0	4.4
Male	135-137	22.5	4.5
Male	138-140	23.0	4.6
Male	141-143	23.5	4.7
Male	144-146	24.0	4.8
Male	147-149	24.5	4.9
Male	150-152	25.0	5.0
Male	153-155	25.5	5.1
Male	156-158	26.0	5.2
Male	159-161	26.5	5.3
Male	162-164	27.0	5.4
Male	165-167	27.5	5.5
Male	168-170	28.0	5.6
Male	171-173	28.5	5.7
Male	174-176	29.0	5.8
Male	177-179	29.5	5.9
Male	180-182	30.0	6.0
Male	183-185	30.5	6.1
Male	186-188	31.0	6.2
Male	189-191	31.5	6.3
Male	192-194	32.0	6.4
Male	195-197	32.5	6.5
Male	198-200	33.0	6.6
Male	201-203	33.5	6.7
Male	204-206	34.0	6.8
Male	207-209	34.5	6.9
Male	210-212	35.0	7.0
Male	213-215	35.5	7.1
Male	216-218	36.0	7.2
Male	219-221	36.5	7.3
Male	222-224	37.0	7.4
Male	225-227	37.5	7.5
Male	228-230	38.0	7.6
Male	231-233	38.5	7.7
Male	234-236	39.0	7.8
Male	237-239	39.5	7.9
Male	240-242	40.0	8.0
Male	243-245	40.5	8.1
Male	246-248	41.0	8.2
Male	249-251	41.5	8.3
Male	252-254	42.0	8.4
Male	255-257	42.5	8.5
Male	258-260	43.0	8.6
Male	261-263	43.5	8.7
Male	264-266	44.0	8.8
Male	267-269	44.5	8.9
Male	270-272	45.0	9.0
Male	273-275	45.5	9.1
Male	276-278	46.0	9.2
Male	279-281	46.5	9.3
Male	282-284	47.0	9.4
Male	285-287	47.5	9.5
Male	288-290	48.0	9.6
Male	291-293	48.5	9.7
Male	294-296	49.0	9.8
Male	297-299	49.5	9.9
Male	300-302	50.0	10.0
Male	303-305	50.5	10.1
Male	306-308	51.0	10.2
Male	309-311	51.5	10.3
Male	312-314	52.0	10.4
Male	315-317	52.5	10.5
Male	318-320	53.0	10.6
Male	321-323	53.5	10.7
Male	324-326	54.0	10.8
Male	327-329	54.5	10.9
Male	330-332	55.0	11.0
Male	333-335	55.5	11.1
Male	336-338	56.0	11.2
Male	339-341	56.5	11.3
Male	342-344	57.0	11.4
Male	345-347	57.5	11.5
Male	348-350	58.0	11.6
Male	351-353	58.5	11.7
Male	354-356	59.0	11.8
Male	357-359	59.5	11.9
Male	360-362	60.0	12.0
Male	363-365	60.5	12.1
Male	366-368	61.0	12.2
Male	369-371	61.5	12.3
Male	372-374	62.0	12.4
Male	375-377	62.5	12.5
Male	378-380	63.0	12.6
Male	381-383	63.5	12.7
Male	384-386	64.0	12.8
Male	387-389	64.5	12.9
Male	390-392	65.0	13.0
Male	393-395	65.5	13.1
Male	396-398	66.0	13.2
Male	399-401	66.5	13.3
Male	402-404	67.0	13.4
Male	405-407	67.5	13.5
Male	408-410	68.0	13.6
Male	411-413	68.5	13.7
Male	414-416	69.0	13.8
Male	417-419	69.5	13.9
Male	420-422	70.0	14.0
Male	423-425	70.5	14.1
Male	426-428	71.0	14.2
Male	429-431	71.5	14.3
Male	432-434	72.0	14.4
Male	435-437	72.5	14.5
Male	438-440	73.0	14.6
Male	441-443	73.5	14.7
Male	444-446	74.0	14.8
Male	447-449	74.5	14.9
Male	450-452	75.0	15.0
Male	453-455	75.5	15.1
Male	456-458	76.0	15.2
Male	459-461	76.5	15.3
Male	462-464	77.0	15.4
Male	465-467	77.5	15.5
Male	468-470	78.0	15.6
Male	471-473	78.5	15.7
Male	474-476	79.0	15.8
Male	477-479	79.5	15.9
Male	480-482	80.0	16.0
Male	483-485	80.5	16.1
Male	486-488	81.0	16.2
Male	489-491	81.5	16.3
Male	492-494	82.0	16.4
Male	495-497	82.5	16.5
Male	498-500	83.0	16.6
Male	501-503	83.5	16.7
Male	504-506	84.0	16.8
Male	507-509	84.5	16.9
Male	510-512	85.0	17.0
Male	513-515	85.5	17.1
Male	516-518	86.0	17.2
Male	519-521	86.5	17.3
Male	522-524	87.0	17.4
Male	525-527	87.5	17.5
Male	528-530	88.0	17.6
Male	531-533	88.5	17.7
Male	534-536	89.0	17.8
Male	537-539	89.5	17.9
Male	540-542	90.0	18.0
Male	543-545	90.5	18.1
Male	546-548	91.0	18.2
Male	549-551	91.5	18.3
Male	552-554	92.0	18.4
Male	555-557	92.5	18.5
Male	558-560	93.0	18.6
Male	561-563	93.5	18.7
Male	564-566	94.0	18.8
Male	567-569	94.5	18.9
Male	570-572	95.0	19.0
Male	573-575	95.5	19.1
Male	576-578	96.0	19.2
Male	579-581	96.5	19.3
Male	582-584	97.0	19.4
Male	585-587	97.5	19.5
Male	588-590	98.0	19.6
Male	591-593	98.5	19.7
Male	594-596	99.0	19.8
Male	597-599	99.5	19.9
Male	600-602	100.0	20.0
Male	603-605	100.5	20.1
Male	606-608	101.0	20.2
Male	609-611	101.5	20.3
Male	612-614	102.0	20.4
Male	615-617	102.5	20.5
Male	618-620	103.0	20.6
Male	621-623	103.5	20.7
Male	624-626	104.0	20.8
Male	627-629	104.5	20.9
Male	630-632	105.0	21.0
Male	633-635	105.5	21.1
Male	636-638	106.0	21.2
Male	639-641	106.5	21.3
Male	642-644	107.0	21.4
Male	645-647	107.5	21.5
Male	648-650	108.0	21.6
Male	651-653	108.5	21.7
Male	654-656	109.0	21.8
Male	657-659	109.5	21.9
Male	660-662	110.0	22.0
Male	663-665	110.5	22.1
Male	666-668	111.0	22.2
Male	669-671	111.5	22.3
Male	672-674	112.0	22.4
Male	675-677	112.5	22.5
Male	678-680	113.0	22.6
Male	681-683	113.5	22.7
Male	684-686	114.0	22.8
Male	687-689	114.5	22.9
Male	690-692	115.0	23.0
Male	693-695	115.5	23.1
Male	696-698	116.0	23.2
Male	699-701	116.5	23.3
Male	702-704	117.0	23.4
Male	705-707	117.5	23.5
Male	708-710	118.0	23.6
Male	711-713	118.5	23.7
Male	714-716	119.0	23.8
Male	717-719	119.5	23.9
Male	720-722	120.0	24.0
Male	723-725	120.5	24.1
Male	726-728	121.0	24.2
Male	729-731	121.5	24.3
Male	732-734	122.0	24.4
Male	735-737	122.5	24.5
Male	738-740	123.0	24.6
Male	741-743	123.5	24.7
Male	744-746	124.0	24.8
Male	747-749	124.5	24.9
Male	750-752	125.0	25.0
Male	753-755	125.5	25.1
Male	756-758	126.0	25.2
Male	759-761	126.5	25.3
Male	762-764	127.0	25.4
Male	765-767	127.5	25.5
Male	768-770	128.0	25.6
Male	771-773	128.5	25.7
Male	774-776	129.0	25.8
Male	777-779	129.5	25.9
Male	780-782	130.0	26.0
Male	783-785	130.5	26.1
Male	786-788	131.0	26.2
Male	789-791	131.5	26.3
Male	792-794	132.0	26.4
Male	795-797	132.5	26.5
Male	798-800	133.0	26.6
Male	801-803	133.5	

This section contains special programming information about the CP, CM, PPs, display station, real-time clock, two-port multiplexer, and maintenance channel.

CP PROGRAMMING

CYBER 170 EXCHANGE JUMP

The CP operates in either CYBER 170 job mode, which can be interrupted, or CYBER 170 monitor mode, which cannot be interrupted. A hardware flag called the CYBER 170 monitor flag (MF) indicates the mode in which the CP is executing a job.

The CP uses a CYBER 170 exchange jump operation to switch from CYBER 170 job mode to CYBER 170 monitor mode and back again. The execution of a CYBER 170 exchange jump permits the CP to send pertinent information from the operating and control registers to CM and permits CM to send new information to the same registers. The information that flows from and into the operating and control registers during a CYBER 170 exchange jump is called a CYBER 170 exchange package (figure 5-1).

A CYBER 170 exchange jump operation is initiated by the CP 013 instruction and the PP 2600, 2610, and 2620 instructions. A CYBER 170 exchange jump instruction starts or interrupts the CP and provides CM with the first address of a 16-word exchange package. For the 013 instruction with MF set (CP in monitor mode) the starting address of the CYBER 170 exchange package is Bj plus K. With MF clear (CP in job mode), the address is the monitor address (MA). For the 2600 instruction, the CYBER 170 exchange package address is A plus R when bit 17 of the A register is set. When this bit is clear, the address is A. For the 2610 instruction with MF set, the instruction is a pass. With MF clear, the CYBER 170 exchange package address is A plus R when bit 17 of the A register is set. When this bit is clear, the address is A. For the 2620 instruction with MF set, the instruction is a pass. With MF clear, the CYBER 170 exchange package address is MA of the outgoing CYBER 170 exchange package.

The CYBER 170 exchange package contains the following registers which provide information for program execution.

- 18-bit program address (P) register.
- 21-bit reference address for CM (RAC) register.
- 21-bit field length for CM (FLC) register.

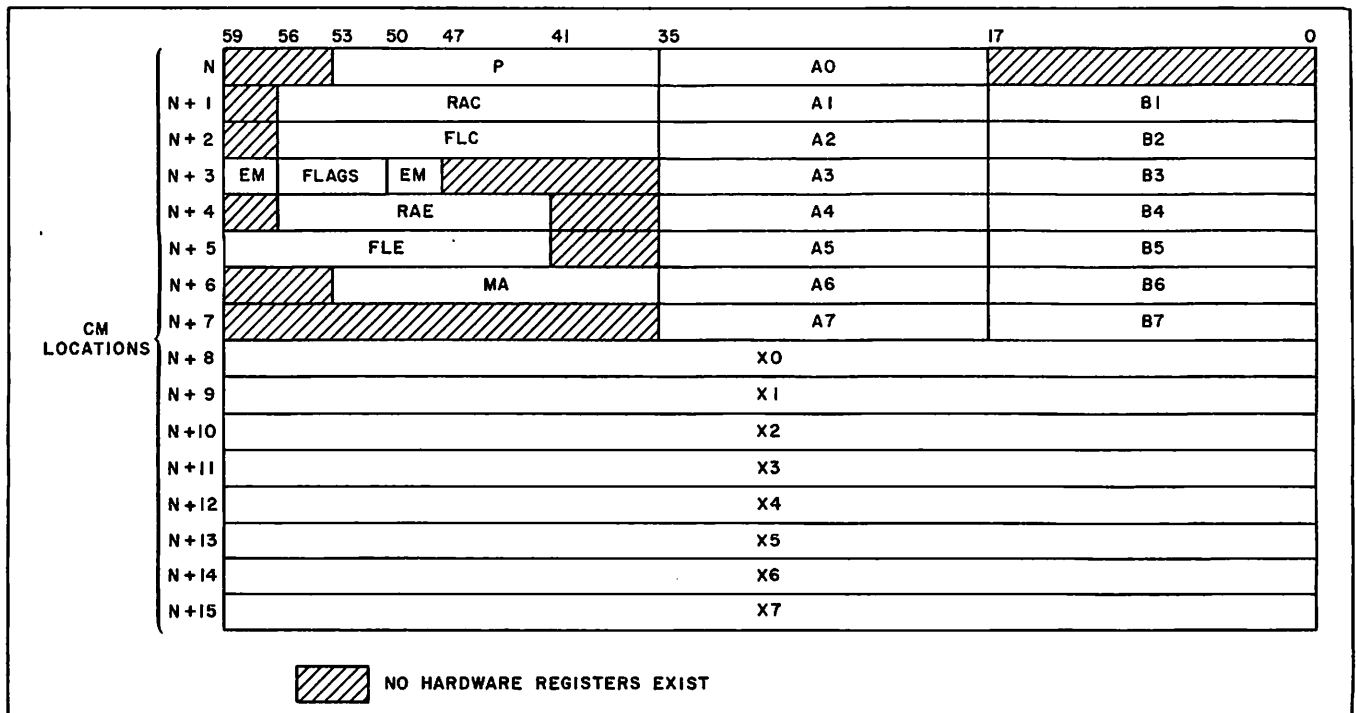


Figure 5-1. CYBER 170 Exchange Package

- 6-bit exit mode (EM) register.
- 6-bit flag register.
- 21- or 24-bit reference address for UEM (RAE); 21 bits with lower 6 bits assumed to be zero in standard addressing mode; 24 bits right-shifted with 6 assumed zeros in expanded addressing mode.
- 21- or 24-bit field length for UEM (FLE); 21 bits in standard addressing mode and 24 bits in expanded addressing mode; lower 6 bits are assumed to be zero.
- 18-bit monitor address (MA) register.
- Initial contents of eight 60-bit X registers.
- Initial contents of eight 18-bit A registers.
- Initial contents of 18-bit B registers B1 through B7, B0 contains constant 0.

The time that a particular CYBER 170 exchange package resides in the CP hardware registers is the execution interval. The execution interval begins with a CYBER 170 exchange jump that swaps the CYBER 170 exchange package information in CM with the information contained in the CP registers. The execution interval ends with the next CYBER 170 exchange jump.

EXECUTIVE STATE

The executive state uses a combination of hardware, software, and microcode to handle the following:

- System initialization.
- Compare/move instructions.
- Software errors and unimplemented instructions that occur in CYBER 170 monitor mode.
- Processor-detected hardware errors.
- Hardware integrity verification (diagnostics).

In general, executive state determines the cause of an interrupt and decides whether to return the CP to the interrupted mode, to halt the CP, or to simulate a CYBER 170 exchange and return control to CYBER 170 monitor mode. Refer to Error Response, this section.

FLOATING-POINT ARITHMETIC

Format

Floating-point arithmetic expresses a number in the form kB^n .

- k Coefficient
- B Base number
- n Exponent or power to which the base number is raised

B is assumed to be 2 for binary-coded quantities. In the 60-bit floating-point format (figure 5-2), the binary point is considered to be to the right of the coefficient. The lower 48 bits express the integer coefficient, which is the equivalent of 15 decimal digits. The sign of the coefficient is separated from the rest of the coefficient and appears in the highest-order bit of the packed word. Negative numbers are represented in one's complement notation. The exponent is biased by complementing the exponent sign bit.

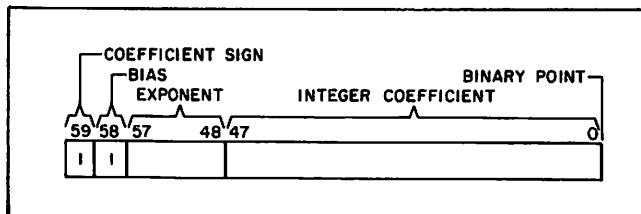


Figure 5-2. Floating-Point Format

Table 5-1 summarizes the configurations of bits 58 and 59 and the implications regarding signs of the possible combinations.

Table 5-1. Bits 58 and 59 Configurations

Bit 59	Bit 58	Coefficient Sign	Exponent Sign
0	1	Positive	Positive
0	0	Positive	Negative
1	0	Negative	Positive
1	1	Negative	Negative

Packing

Packing refers to the conversion of numbers in the form kB^n to floating-point format. A shortcut method of packing exponents can be derived by considering the representation of negative and positive zero exponents. Assuming a positive coefficient, zero exponents are packed as follows:

Positive zero exponent	2000x,...,x
Negative zero exponent	1777x,...,x

Since positive exponents are expressed in true form, begin with a bias of 2000 (positive zero) and add the magnitude of the exponent. The range of positive exponents is 0000 through 1777. In packed form, the range is 2000 through 3777.

When the coefficient is negative, the packed positive exponent is complemented to become 5777 through 4000.

Negative exponents are expressed in complement form by beginning with a bias of 1777 (negative zero) and then subtracting the magnitude of the exponent. The range of negative exponents is negative 0000 through negative 1777. In packed form, the range is 1777 through 0000.

When the coefficient is negative, the packed negative exponent is complemented to become 6000 through 7777.

Examples of packed and unpacked floating-point numbers are shown in octal notation to illustrate the packing process. Examples 1 and 2 are different forms of the integer positive 1. Example 3 is positive 100 (decimal), and example 4 is negative 100 (decimal). Examples 5 and 6 are large and small positive numbers. The unpacked values are shown as they might appear in the X and B registers prior to a pack operation.

The packed negative zero exponent is not used for normal operation. Instead, 1777 is used to indicate the special error condition of indefinite.

1. Unpacked coefficient	0000	0000	0000	0000	0001
Unpacked exponent	00	0000			
Packed format	2000	0000	0000	0000	0001
2. Unpacked coefficient	0000	4000	0000	0000	0000
Unpacked exponent	77	7720			
Packed format	1720	4000	0000	0000	0000
3. Unpacked coefficient	0000	6200	0000	0000	0000
Unpacked exponent	77	7726			
Packed format	1726	6200	0000	0000	0000
4. Unpacked coefficient	7777	1577	7777	7777	7777
Unpacked exponent	77	7726			
Packed format	6051	1577	7777	7777	7777
5. Unpacked coefficient	0000	4771	3000	0044	7021
Unpacked exponent	00	1363			
Packed format	3363	4771	3000	0044	7021

6. Unpacked coefficient	0000	6301	0277	4315	6033
Unpacked exponent	77	6210			
Packed format	0210	6301	0277	4315	6033

Overflow

Overflow of the floating-point range is indicated by an exponent value of positive 1777 (3777 or 4000 in packed form). This is the largest exponent value that can be represented in the floating-point format. This exponent value may result from the calculation in which this exponent value, together with the computed coefficient value, is a correct representation of the result. This situation is called a partial overflow. However, further computation using this result generates an overflow.

A complete overflow occurs whenever a result requires an exponent larger than positive 1777. In this case, a complete overflow value results. This result has a positive 1777 exponent and a zero coefficient. The sign of the coefficient is the same as that which generates if the result had not overflowed the floating-point range.

Underflow

Underflow of the floating-point range is indicated by an exponent value of negative 1777 (0000 or 7777 in packed form). This is the smallest exponent value that can be represented in the floating-point format. This exponent value may result from the calculation in which this exponent value, together with the computed coefficient value, is a correct representation of the result. This situation is called a partial underflow. Further computation using this result may be detected as an underflow.

A complete underflow occurs whenever a result requires an exponent smaller than negative 1777. In this case, a complete underflow value results. This result has a negative 1777 exponent and a zero coefficient. The complete underflow indicator is a word of all zeros, and it is the same as a zero word in integer format.

Indefinite

An indefinite result indicator generates whenever the calculation cannot be resolved. An example is division when the divisor is 0 and the dividend is also 0. Another example is multiplication of an overflow number times an underflow number. The indefinite result indicator is a value that cannot occur in normal floating-point calculations. This indicator corresponds to a negative 0 exponent and a 0 coefficient (177770,...,0 in packed form).

Any indefinite indicator used as an operand generates an indefinite result no matter what the other operand value is. Although indefinite indicators always generate with a positive sign, they may occur as operands with a negative sign.

Nonstandard Operands

In summary, the special operand forms in octal are:

Positive overflow (+ ∞)	3777x,...,x
Negative overflow (- ∞)	4000x,...,x
Positive indefinite (+IND)	1777x,...,x
Negative indefinite (-IND)	6000x,...,x
Positive underflow (+0)	0000x,...,x
Negative underflow (-0)	7777x,...,x

Tables 5-2 through 5-5 indicate the resulting forms when various combinations of underflow, overflow, and indefinite forms are used in floating-point operations. The designations W and N are defined as follows:

W	Any word except + ∞ and + IND
N	Any word except + ∞ , + IND, and + 0

Normalized Number

A normalized floating-point number has as large a coefficient and as small an exponent as possible. A floating-point number in packed format is normalized if the coefficient sign bit is different from bit 47. This condition indicates that the coefficient has been left shifted until bit 47 contains the most significant bit in the coefficient; therefore, the floating-point number has no leading sign bits in the coefficient. The normalized instructions perform the coefficient shift. The floating-multiply and floating-divide instructions deliver normalized results when provided with normalized operands. The floating-add instructions may deliver unnormalized results even when both operands are normalized. Therefore, it is necessary to perform the normalize operation after each sequence of floating-add or floating-subtract operations if the result is to be kept in a normalized form.

Rounding

Floating-point instructions round the results in single-precision computation. These instructions execute in the same amount of time as the unrounded versions. The operands are modified to accomplish the rounding function. The amount of bias introduced by the rounding operation varies and is affected by the coefficient value in the operands. The descriptions of the round instructions define the effects of rounding in detail.

Double-Precision Results

The floating-point arithmetic instructions generate double-precision results. Use of unrounded instructions allows separate recovery of upper and lower half results with proper exponents. Rounded instructions allow only upper half results to be obtained. Two instructions, one single-precision and one double-precision, are required to retrieve an entire double-precision result.

To add or subtract two floating-point numbers, the coefficient having the smaller exponent enters the upper half of an accumulator and is right shifted by the difference of the exponents. The other coefficient is then added into the upper half of the accumulator. The result is a double-length register with the format shown in figure 5-3.

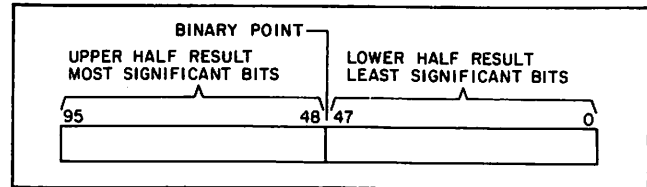


Figure 5-3. Floating-Add Result Format

If single precision is selected, the upper 48 bits of the 96-bit result and the larger exponent are returned as the result. Selecting double precision causes only the lower 48 bits of the 96-bit result and the larger exponent minus 60 (octal) to be returned as the result. The subtraction of 60 (octal) is necessary because the binary point is effectively moved from the right of bit 48 to the right of bit 0.

A 96-bit product generates from two 48-bit coefficients. The result of a multiply is a double-length register with the format shown in figure 5-4.

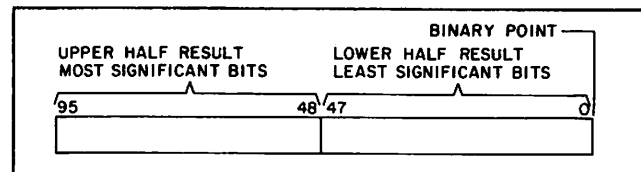


Figure 5-4. Multiply Result Format

If single precision is selected, the upper 48 bits of the product and the sum of the exponents plus 60 (octal) are returned as the result. The addition of 60 (octal) is necessary because the binary point effectively moves from the right of bit 0 to the right of bit 48 when the upper half of the 96-bit result is selected. If double precision is selected, the result is the lower 48 bits of the product and the sum of the exponents.

FIXED-POINT ARITHMETIC

Fixed-point addition and subtraction of 60-bit numbers are handled by the long-add instructions (36 and 37). Negative numbers are represented in one's complement notation, and overflows are ignored. The sign bit is in the high-order bit position (bit 59), and the binary point is to the right of the low-order bit position (bit 0).

Fixed-point addition and subtraction of 18-bit numbers are handled by the increment instructions (50 through 77). Negative numbers are represented in one's complement notation, and overflows are ignored. The sign bit is in the high-order bit position (bit 17), and the binary point is to the right of the low-order position (bit 0).

Table 5-2. Xj Plus Xk
(30, 32, 34 Instructions)

		Xk			
		W	+∞	-∞	+IND
Xj	W	/	+∞	-∞	IND
	+∞	+∞	+∞	IND	IND
	-∞	-∞	IND	-∞	IND
	±IND	IND	IND	IND	IND

Table 5-3. Xj Minus Xk
(31, 33, 35 Instructions)

		Xk			
		W	+∞	-∞	+IND
Xj	W	/	-∞	+∞	IND
	+∞	+∞	IND	+∞	IND
	-∞	-∞	-∞	IND	IND
	±IND	IND	IND	IND	IND

Table 5-4. Xj Multiplied by Xk (40, 41, 42 Instructions)

		Xk						
		+N	-N	+0	-0	+∞	-∞	+IND
Xj	+N	/	/	0	0	+∞	-∞	IND
	-N	/	/	0	0	-∞	+∞	IND
	+0	0	0	Integer † multiply		IND	IND	IND
	-0	-0	0			IND	IND	IND
	+∞	+∞	-∞	IND	IND	+∞	-∞	IND
	-∞	-∞	+∞	IND	IND	-∞	+∞	IND
	±IND	IND	IND	IND	IND	IND	IND	IND

† If both operands used in the integer multiply are normalized, an underflow results.

Table 5-5. Xj Divided by Xk (44, 45 Instructions)

		Xk						
		+N	-N	+0	-0	+∞	-∞	+IND
Xj	+N	/	/	+∞	-∞	0	0	IND
	-N	/	/	-∞	+∞	0	0	IND
	+0	0	0	IND	IND	0	0	IND
	-0	0	0	IND	IND	0	0	IND
	+∞	+∞	-∞	+∞	-∞	IND	IND	IND
	-∞	-∞	+∞	-∞	+∞	IND	IND	IND
	±IND	IND	IND	IND	IND	IND	IND	IND

Integer multiplication is handled as a subset operation of the floating-multiply (42) instruction. The integer multiply requires that both 47-bit integer operands have zero exponents and are not normalized. The result is 48 bits with sign extension. Normalized operands cause underflow results to be reported. If the results exceed 48 bits, overflow is not detected.

An integer divide takes several steps. For example, an integer quotient X1 equal to X2/X3 is produced by the following steps.

<u>Instructions</u>	<u>Remarks</u>
1. Pack X2 from X2 and B0	Pack X2
2. Pack X3 from X3 and B0	Pack X3
3. Normalize X3 in X0 and B0	Normalize X3 (divisor)
4. Normalize X2 in X2 and B0	Normalize X2 (dividend)
5. Floating quotient of X2 and X0 to X1	Divide
6. Unpack X1 to X1 and B7	Unpack quotient
7. Shift X1 nominally left B7 places	Shift to integer position

The divide requires that both integer (247 maximum) operands be in floating-point format, and the dividend coefficient must be less than two times the divisor coefficient. The normalize X3 instruction ensures this condition.

The normalize X3 instruction left shifts the divisor n places ($n > 0$), providing a divisor exponent of negative n. The quotient exponent is then 0 minus (-n) minus 48 equals n minus 48 (<0).

After unpacking and left shifting nominally, the negative (or zero) value in B7 right shifts the quotient 48 minus n places, producing an integer quotient in X1. A remainder may be obtained by an integer multiply of X1 and X3 and subtracting the result from X2.

INTEGER ARITHMETIC

Integer divide packs the integers into floating-point format using the pack instruction with a zero-exponent value.

In integer multiplication, a 48-bit product can be formed by using the double-precision multiply instruction. Both operands must have an exponent value of +0, and the coefficients cannot both be normalized. The result is sign-extended to 60 bits and sent to an X register.

In integer division, the divisor must be normalized but the dividend need not be normalized. The resulting quotient must be unpacked and the coefficient shifted by the amount of the unpacked exponent using the left shift (22) instruction to obtain the integer quotient.

COMPARE/MOVE ARITHMETIC

The compare/move arithmetic provides multiple character manipulation. The characters are 6 bits long. Characters can be moved from one CM location to another, and fields of characters can be compared either directly or through a collate table.

The move direct instruction moves a field of up to 127 characters from one location to another location as specified in the instruction. The move indirect instruction performs the same kind of move, but a CM reference is used to obtain the parameters. The move indirect instruction moves a field of up to 8181 characters.

The compare collated instruction compares two fields of up to 127 characters. When two characters are unequal, the characters are referenced in a collate table, and the values are compared. If those values are unequal, the field with the larger character is indicated. The compare uncollated instruction compares two fields of up to 127 characters and indicates the larger of the first character pair that is found to be unequal.

CMU instructions are provided for compatibility with previous systems. For better performance, recompile jobs to avoid use of CMU instructions.

INSTRUCTION LOOKAHEAD PURGE CONTROL

Prefetching of instructions at a branch target address by instruction lookahead hardware can lead to program failures if a program modifies its own code dynamically. Under normal conditions, the lookahead registers are purged by execution of a return jump instruction (010), UEM read instruction (011), exchange jump instruction (013), or unconditional branch instruction (02). These conditions can be extended by selecting extended purge control. When extended purge control is in effect, lookahead registers are also purged by execution of any conditional jump instruction (03 through 07) or any CM store instruction (50 through 57 when i equals 6 or 7). To enable extended purge control, the system sets bit 52 of the flag register in the CYBER 170 exchange package. When self-modifying code is present, it may be helpful to set extended purge control; however, the additional purging does cause a degradation in execution and does not cover all cases of code modification.

Model 835 Purge Control

If normal purge conditions are in effect, a store instruction that modifies a sequential instruction must modify at least P plus five words ahead to ensure execution of the modified code. A store instruction followed by a branch to a modified instruction will execute the modified code only if that code is at least at the branch's target address plus two words, or the branch is at least at P plus four words following the store instruction.

If the extended purge option is selected, a store instruction can modify the next sequential instruction and be assured of executing the modified instruction. Likewise, a store instruction followed by a branch to a modified instruction always executes the modified code.

Model 855 Purge Control

If normal purge conditions are in effect, a store instruction that modifies a sequential instruction must modify at least P plus six words ahead to ensure execution of the modified code. In addition, a store instruction followed by a branch to a modified instruction will execute the modified code only if there are at least 12 executed instructions between the store and the modified code.

If the extended purge option is selected, a store instruction can modify the next sequential instruction and be assured of executing the modified instruction. Likewise, a store instruction followed by a branch to a modified instruction always executes the modified code.

ERROR RESPONSE

When the CP detects or is informed of an error, it records the error. Depending upon the type of error and the exit mode selection bits set in the EM register, the program in execution may be interrupted. If the error is an illegal instruction or an address-range error on an RNI or branch, the program interruption is unconditional. For other types of errors, the exit mode selection bits determine whether or not the program is interrupted. If the exit mode selection bit is set and the corresponding condition is detected, the program is interrupted. The exit mode selection bits are contained in word N plus 3 of the exchange package. Figure 5-5 shows the format of the exit condition register at (RAC). Table 5-6 describes the possible contents of the register. Tables 5-7 and 5-8 list CP error responses.

The CP has the following error conditions: illegal instructions, hardware errors, and conditional software errors.

Illegal Instructions

An instruction is illegal when it has an illegal operating code, an illegal operating parameter, or when it is positioned so that it begins in one instruction word and extends into the next instruction word. In the CYBER 170 job mode, illegal instructions cause an exchange to the CYBER 170 monitor mode. In the CYBER 170 monitor mode, they cause a jump to executive state; the CP stops. CP illegal instructions are:

- 017.
- 011, 012, 013, 464, 465, 466, 467 if they do not begin at parcel 0.

- 011, 012, 014, 015 if the UEM enable flag in the flag register of the CYBER 170 exchange package is clear.
- Any 30-bit instruction which begins at parcel 3.

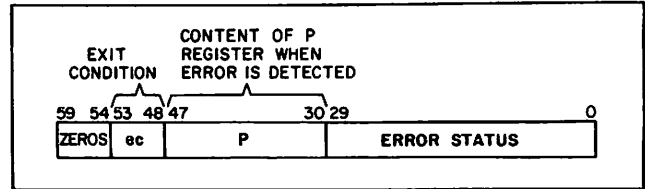


Figure 5-5. Format of Exit Condition Register at (RAC)

Table 5-6. Contents of Exit Condition Register at (RAC)

Field	Description
ec	6-bit exit condition code
	Code Condition
	00 ₈ Illegal instruction
	01 ₈ Address-range error (bit 48)
	02 ₈ Floating-point infinite (bit 49)
	04 ₈ Floating-point infinite (bit 50)
	20 ₈ Processor-detected malfunction
	67 ₈ Hardware malfunction
P	When an error exit occurs, the content of the P register may not correspond to the address of the instruction that caused the error exit. The P register may have been incremented prior to the execution of the instruction.
ERROR STATUS	Nonzero information in bits 0 through 29 is error status for customer engineering and maintenance.

Hardware Errors

CP/CM hardware errors are: data parity errors, address parity errors, and double bit errors. If the CP is in CYBER 170 job mode, a hardware error causes a jump to executive state which returns to CYBER 170 monitor mode. If the CP is in CYBER 170 monitor mode, a hardware error causes a jump to executive state; the CP halts. The instruction being executed when such a fault is detected is not necessarily connected with the fault.

Table 5-7. Error Exits in CYBER 170 Monitor Mode (MF=1) (Sheet 1 of 2)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Illegal instruction or 00 instruction.	<ol style="list-style-type: none"> 1. The instruction is not executed. 2. Store P and exit condition bits (00) at location RAC. P equals address of illegal instruction. 3. Interrupt to executive state. 4. CP stops in executive state. 	<ol style="list-style-type: none"> 1. N/A (exit mode is always selected).
Exit condition bit 48 set by an incremental read with an address out of range (AOR).	<ol style="list-style-type: none"> 1. The X register is unchanged. 2. The A register contains the AOR address. 3. Store P and exit condition bits (01) at location RAC. P equals address of increment instruction or address of instruction following the increment. 4. Interrupt to executive state. 5. CP stops in executive state. 	<ol style="list-style-type: none"> 1. Inhibit read, X unchanged. 2. Continue execution.
Exit condition bit 48 set by an incremental write with an address out of range (AOR).	<ol style="list-style-type: none"> 1. Block write operation; content of CM is unchanged. 2. The A register contains the AOR address. 3. Store P and exit condition bits (01) at location RAC. P equals address of instruction or address of instruction following the increment. 4. Interrupt to executive state. 5. CP stops in executive state. 	<ol style="list-style-type: none"> 1. Inhibit write, CM unchanged. 2. Continue execution.
Exit condition bit 48 set by an RNI or branch address out of range.	<ol style="list-style-type: none"> 1. Inhibit execution. 2. Store P and exit condition bits (01) at location RAC. P equals address of instruction required by RNI or address of branch destination instruction. 3. Interrupt to executive state. 4. CP stops in executive state. 	<ol style="list-style-type: none"> 1. N/A (exit mode is always selected regardless of status of EM register bit 48).

Table 5-7. Error Exits in CYBER 170 Monitor Mode (MF=1) (Sheet 2 of 2)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
<p>Exit condition bit 48 set on CMU instruction.</p> <p>1. C1 or C2 greater than 9.</p> <p>2. K1 or K2 address out of range.</p>	<p>1. Detected by executive state during the execution of compare/move instruction.</p> <p>2. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted.</p> <p>3. Store P and exit bits (01) at RAC.</p> <p>4. CP stops in executive state.</p>	<p>1. Detected by executive state during the execution of compare/move instruction.</p> <p>2. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted.</p> <p>3. Continue with next instruction.</p>
<p>Exit condition bit 48 set by a UEM address range check for instructions 011 and 012.</p>	<p>1. Execute instruction as a pass.</p> <p>2. Store P and exit bits (01) at RAC.</p> <p>3. Interrupt to executive state.</p> <p>4. CP stops in executive state.</p>	<p>1. Execute instruction as a pass.</p> <p>2. Exit to next 60-bit word and continue execution.</p>
<p>Exit condition bit 48 set by a UEM address range check for instructions 014 and 015.</p>	<p>1. Execute instruction as a pass.</p> <p>2. Store P and exit condition bits (01) at RAC. P equals address of following instruction.</p> <p>3. Interrupt to executive state.</p> <p>4. CP stops in executive state.</p>	<p>1. Execute instruction as a pass.</p> <p>2. Exit to next parcel and continue execution.</p>
<p>Exit condition bit 49 set by infinite condition, or bit 50 set by indefinite condition.</p>	<p>1. Store P and exit condition bits (02 for infinite or 04 for indefinite). P equals address of arithmetic instruction or address of instruction following.</p> <p>2. Interrupt to executive state.</p> <p>3. CP stops in executive state.</p>	<p>1. Continue execution.</p>
<p>Any hardware parity error or double SECEDED error.</p>	<p>1. Interrupt to executive state.</p> <p>2. Executive state stores P and exit condition bits (20) at RAC.</p> <p>3. CP stops in executive state.</p>	<p>1. Interrupt to executive state.</p> <p>2. Executive state stores P and exit condition bits (20) at RAC.</p> <p>3. CP stops in executive state.</p>

Table 5-8. Error Exits in CYBER 170 Job Mode (MF=0) (Sheet 1 of 2)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Illegal instruction or 00 instruction.	<ol style="list-style-type: none"> 1. The instruction is not executed. 2. Store P and exit condition bits (00) at location RAC. P equals address of illegal instruction. 3. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. N/A (exit mode is always selected).
Exit condition bit 48 set by an incremental read with an address out of range (AOR).	<ol style="list-style-type: none"> 1. The X register is unchanged. 2. The A register contains the AOR address. 3. Store P and exit condition bits (01) at location RAC. P equals address of increment instruction or address of instruction following the increment. 4. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. Inhibit read, X unchanged. 2. Continue execution.
Exit condition bit 48 set by an incremental write with an address out of range (AOR).	<ol style="list-style-type: none"> 1. Block write operation; content of CM is unchanged. 2. The A register contains the AOR address. 3. Store P and exit condition bits (01) at location RAC. P equals address of instruction or address of instruction following the increment. 4. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. Inhibit write, CM unchanged. 2. Continue execution.
Exit condition bit 48 set by an RNI or branch address out of range.	<ol style="list-style-type: none"> 1. Inhibit execution. 2. Store P and exit condition bits (01) at location RAC. P equals address of instruction required by RNI or address of branch destination instruction. 3. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. N/A (exit mode is always selected regardless of status of EM register bit 48).

Table 5-8. Error Exits in CYBER 170 Job Mode (MF=0) (Sheet 2 of 2)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
<p>Exit condition bit 48 set on CMU instruction.</p> <ol style="list-style-type: none"> 1. C1 or C2 greater than 9. 2. K1 or K2 address out of range. 	<ol style="list-style-type: none"> 1. Detected by executive state during the execution of compare/move instruction. 2. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted. 3. Store P and exit bits (01) at RAC. 4. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. Detected by executive state during the execution of compare/move instruction. 2. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted. 3. Continue with next instruction.
<p>Exit condition bit 48 set by a UEM address range check for instructions 011 and 012.</p>	<ol style="list-style-type: none"> 1. Execute instruction as a pass. 2. Store P and exit bits (01) at RAC. 3. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. Execute instruction as a pass. 2. Exit to next 60-bit word and continue execution.
<p>Exit condition bit 48 set by a UEM address range check for instructions 014 and 015.</p>	<ol style="list-style-type: none"> 1. Execute instruction as a pass. 2. Stop CP. 3. Store P and exit condition bits (01) at location RAC. 4. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. Execute instruction as a pass. 2. Exit to next parcel and continue execution.
<p>Exit condition bit 49 set by infinite condition, or bit 50 set by indefinite condition.</p>	<ol style="list-style-type: none"> 1. Store P and exit condition bits (02 for infinite or 04 for indefinite). P equals address of arithmetic instruction or address of instruction following. 2. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. Continue execution.
<p>Any hardware parity error or double SECEDED error.</p>	<ol style="list-style-type: none"> 1. Interrupt to executive state. 2. Executive state stores P and exit condition bits (20) at RAC. 3. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. Interrupt to executive state. 2. Executive state stores P and exit condition bits (20) at RAC. 3. Exchange jump to MA and set CYBER 170 MF.

Conditional Software Errors

Conditional software errors are caused by address-range errors, and floating-point infinite/indefinite operands or results. A conditional software error causes action depending on bits set in the EM field in the current CYBER 170 exchange package. If the bit reserved for use with the specific type of error is clear, the error is ignored in both CYBER 170 job and CYBER 170 monitor modes. If the bit is set and the error occurs in the CYBER 170 job mode, it causes an exchange to the CYBER 170 monitor mode.

If the bit is set and the error occurs in the CYBER 170 monitor mode, it causes an interrupt to executive state.

MEMORY PROGRAMMING

All references to CM by the CP for instructions or read/write data are made relative to RAC. The RAC defines the lower limit of the addresses of a program in CM. The upper limit of the program addresses is defined by FLC added to RAC.

All references to UEM by the CP for instructions or read/write data are made relative to RAE. The RAE defines the lower limit of the addresses of a program/data in UEM. The upper limit of the addresses is defined by FLE added to RAE.

The field length is a number of 60-bit words established by the operating system prior to program execution. All references to CM or UEM for a program/data must be within the field established for that program.

During a CYBER 170 exchange jump, RAC and FLC are loaded into respective registers to define the CM limits of the program that is initiated by the CYBER 170 exchange jump. RAE and FLE are loaded to define the UEM limits of a program.

Figure 5-6 shows the absolute and relative memory addresses, RAC, FLC, RAE, and FLE register relationships. For a program to operate within the established limits, the following conditions must exist.

For absolute memory addresses:

$$RAC \leq (RAC + P) < (RAC + FLC)$$

For relative memory addresses:

$$0 \leq P < FLC$$

ADDRESSING MODES

UEM can be used in either of two addressing modes: standard or expanded. Standard addressing mode provides addressing up to 21 bits in a 24-bit format. Expanded addressing mode provides addressing up to 24 bits in a 30-bit format. Addressing mode is determined by the expanded addressing select flag, bit 55 of word 3 in the CYBER 170 exchange package.

DIRECT READ/WRITE INSTRUCTIONS (014, 015, 660, 670)

These instructions transfer one 60-bit word between the selected X register and a memory location, using a 21-bit relative address. Instructions 660 and 670 use the memory address Xk (21 bits) plus RAC (21 bits) to address CM. Instructions 014 and 015 use the memory address Xk (21 bits) plus RAE (21 bits) to address UEM.

BLOCK COPY INSTRUCTIONS (011, 012)

These instructions transfer up to 131 071 60-bit words between fields in CM and UEM. The UEM address is X0 plus RAE (bits 0 through 22 in standard addressing mode; bits 0 through 28 in expanded addressing mode). The CM address is A0 plus RAC (if the block copy flag is clear in the CYBER 170 exchange package) or X0, bits 30 through 50, plus RAC (if the block copy flag is set).

The transfers occur in blocks of up to 64 words, during which other CP activities are suspended.

These instructions are 30-bit instructions which must start at parcel 0. If the UEM address has bit 21 or bit 22 set in standard addressing mode (bit 28 if in expanded addressing mode), zeros are transferred to CM and the next instruction is taken from parcel 2 of the same instruction word. If this is not the case on a block read, the next instruction is taken from parcel 0 of the next instruction word. A transfer of all zeros can be made to central memory using the 011 instruction and setting bit 21 or 22 (or bit 28) of the address (X0 + RAE) when FLE is sufficiently large.

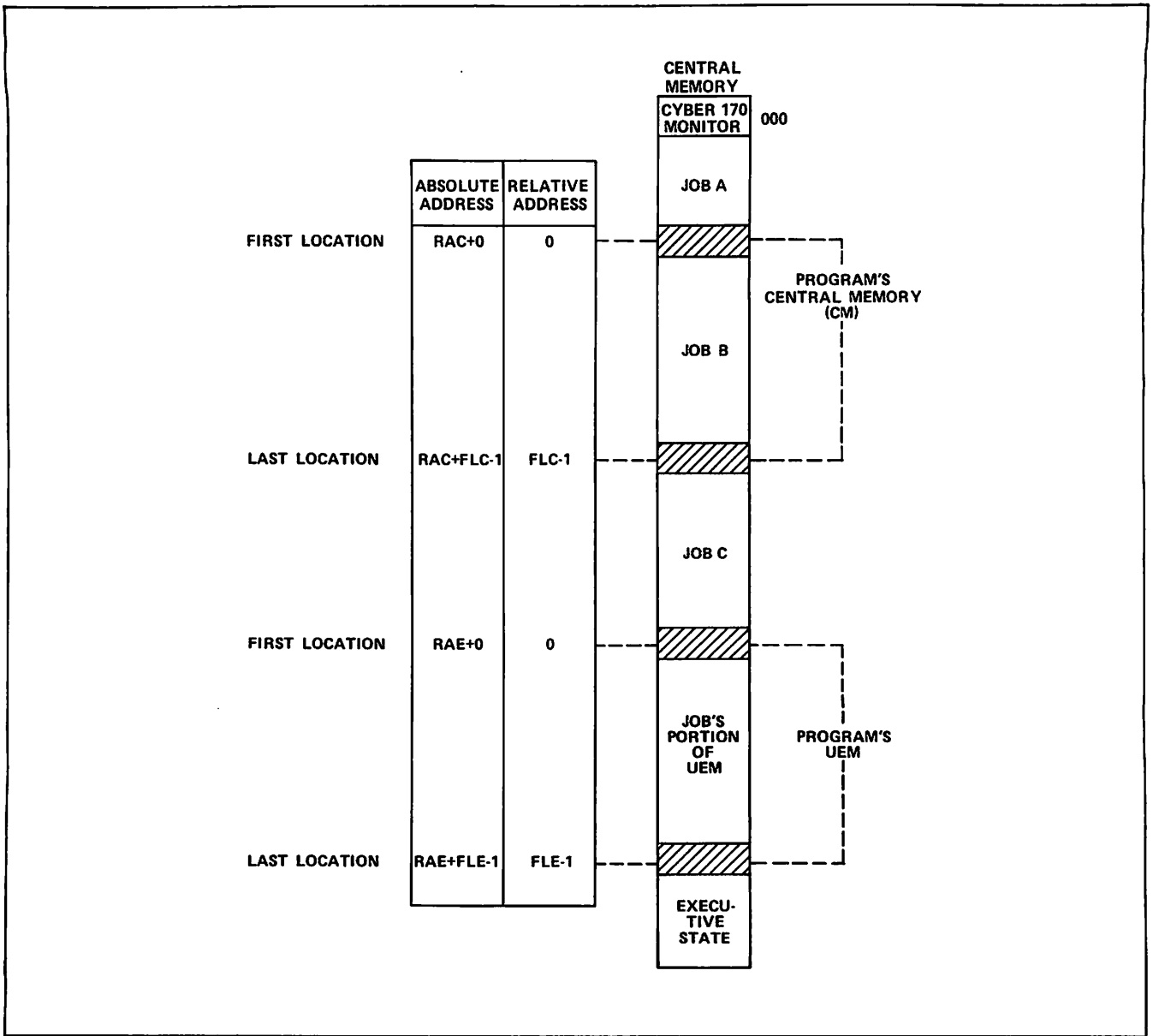


Figure 5-6. Memory Map

PP PROGRAMMING

The PPs have access to all CM storage locations. One 64-bit word or a block of 64-bit words can be transferred from a peripheral processor memory (PPM) to CM or from CM to PPM. (Five 12-bit PP words equal one 64-bit CM word, with the leftmost 4 bits undefined.) Data from external devices is read into a PPM, and with additional instructions, is transferred to CM. Conversely, data is transferred from CM to a PPM and is then transferred by additional instructions to external devices. Addresses sent to CM from PPs are absolute or relocation addresses.

CENTRAL MEMORY ADDRESSING BY PPs

Pps address central memory using either absolute or relocation addressing. Every PP can read all central memory locations without restriction. Every PP has write access to central memory. The bounds register in central memory may also be set to limit write access from the IOU.

Instructions 24/25 load/store the relocation (R) register. If bit 17 of the A register is zero, bits 0 through 16 of A specify an absolute central memory address 0 through 377 777₈. If bit 17 of A is one, bits 0 through 16 of A are added to the 28-bit R register to specify an absolute central memory address 0 through 0 007 777 777₈. If bit 17 of A changes during a transfer, the addressing mode also changes accordingly. The leftmost 7 bits of R represent extra addressing capacity which is unused. The rightmost 6 bits of R are appended zeros. Instruction 24 loads R from two consecutive PP memory locations. Instruction 25 stores R into two PP memory locations. Figure 4-4 shows how R is stored in PP memory.

PP MEMORY ADDRESSING BY PPs

PP instructions use 6-bit or 18-bit direct operands, or access PP memory through direct, indirect, or indexed addressing.

Direct 6-Bit Operand

PP instructions in this category are no-address instructions. They have the format OPCODEd. The d field is used as a 6-bit direct operand, zero-extended to 18 bits in calculations.

Direct 18-Bit Operand

PP instructions in this category are constant address instructions. They have the format OPCODEdm. The combined d and m fields are used as an 18-bit operand.

Direct 6-Bit Address

PP instructions in this category are direct address instructions. They have the format OPCODEd. The d field is used as a 6-bit direct address, accessing PP memory locations 0 to 77₈.

Direct 12-Bit Address

PP instructions in this category are indexed direct address instructions, with zero index. They have the format OPCODEdm, d equals 0. The m field is used as a 12-bit direct address, accessing PP memory locations 0 through 777₈.

Indexed 12-Bit Address

PP instructions in this category are indexed direct address instructions. They have the format OPCODEdm, d equals 0. The m field is used as a 12-bit direct address (base address). The d field specifies a PP memory location from 1 to 77₈, the contents of which is a 12-bit one's complement number index. The indexed direct address is formed by adding the index to the base address as signed one's complement numbers, ignoring overflow. When m plus (d) equals 777, the result is set to 0000, except as follows: adding 777 plus 777 equals 777. In general, adding 0000 or 777 leaves the other number unchanged, except when the other number is also 0000 or 777.

Indirect 6-Bit Address

PP instructions in this category are indirect address instructions. They have the format OPCODEd. The 6-bit d field is used to read a 12-bit number from PP locations 0 through 77₈; this number is used as a 12-bit address to access PP memory locations 0 through 777₈.

CENTRAL MEMORY READ/WRITE INSTRUCTIONS

PP instructions can read and write to central memory either single words or blocks of words.

PP Central Memory Read Instructions. (60, 61)

Instruction 60 transfers one CM word into five 12-bit PP memory words. Instruction 61 transfers a block of 1 through 811 CM words into 5 through 4095 12-bit PP words; it is possible to transfer up to 4096 CM words overwriting PP memory cyclically; location 0, however, has special properties. Refer to instruction 61.

PP Central Memory Write Instructions (62, 63)

Instruction 62 transfers five 12-bit PP memory words into one CM word. Instruction 63 transfers 5 through 4095 PP memory words into 1 through 811 CM words. It is possible to transfer up to 20 480 PP memory words, repeating information from PP memory cyclically.

INPUT/OUTPUT CHANNEL COMMUNICATIONS

Data transfers to and from external devices are controlled by PP instructions 64 through 77. The assignment of PPs, transfer priorities and resolution of conflicts are software responsibilities.

Channel parity and reservation must be provided for, using the channel marker flag and/or software interlocks in central memory. After any conflicts have been resolved, proceed as follows:

<u>Action</u>	<u>Typical Instruction</u>
1. Clear error flag.	Jump if error flag set, and clear flag (661).
2. Verify inactive status.	Jump if active (640).
3. Verify read status.	
Prepare for reading summary status.	Function m (77).
Verify that the device responded.	Jump if active (640).
Activate channel.	Activate (74).
Read summary status.	Input to A (70).
Verify error flag clear.	Jump if error flag set (661).
Analyze summary status.	Logical product (12). Zero jump (04).
4. Enter number of words to A.	Load d (14).
5. Prepare for input/output.	
Verify inactive status.	Jump if active (640).
Prepare for read/write.	Function m (77).
Verify that the device responded.	Jump if active (640).
6. Read/write data.	
Activate channel.	Activate (74).
Read/write data.	Input/output A words (71/73).
If write, loop until empty.	Jump if full (660).
Disconnect channel.	Deactivate (75).
Verify inactive status.	Jump if active (640).

<u>Action</u>	<u>Typical Instruction</u>
7. Verify transfer integrity.	
Verify A words were transferred (refer to note).	Nonzero jump (05).
Verify error flag clear.	Jump if error flag set (661).
Verify inactive status.	Jump if active (640).
Prepare for reading device status.	Function m (77).
Verify that the device responded.	Jump if active (640).
Activate channel.	Activate (74).
Read device status.	Input to A (70).
Verify error flag clear.	Jump if error flag set (661).
Analyze device status.	Logical product (12). Nonzero jump (05).
Disconnect channel.	Deactivate (75).

NOTE

If A equals the original value, no words were transferred.

If A is not equal to 0, the device or another PP ended the transfer.

INTER-PP COMMUNICATIONS

Any PP can communicate with any other PP using any channel (except the real-time clock) by omitting the conditioning of the external devices of that channel for a data transfer. Both single word and block transfers can be used. Either the sending or the receiving PP can activate the channel used, after which the sending PP outputs data into the channel register of the channel concerned and the receiving PP inputs data from the same register. The transfer rate is one word every 250 nanoseconds, except when the transfer is between PPs in different barrels but in the same time slot. In such a case the transfer rate is one word every 500 nanoseconds. PPs which use the same time slots are as follows:

<u>Slot</u>	<u>PP Number</u>
1	0, 5, 20, 25
2	1, 6, 21, 26
3	2, 7, 22, 27
4	3, 10, 23, 30
5	4, 11, 24, 31

Software resolves priority and reservation problems arising in inter-PP communications by interlocks stored in CM or by other means.

PP PROGRAM TIMING CONSIDERATIONS

Some external equipment may require timing considerations in issuing function, activate, and input instructions. Refer to the applicable external equipment reference manual. Such timing considerations may, for example, be required to ensure that the equipment attains a proper speed before data is sent (required by some magnetic tape equipment). Also, equipment which terminates a data transfer by resetting the active flag to inactive often requires timing considerations in issuing the next function instruction.

CHANNEL OPERATION

Channel Control Flags

Channel operation is affected by the channel active/inactive and full/empty flags and, depending on the status of these two flags, the channel is said to be active, inactive, full, or empty. Each channel also has a marker flag for software use, and an error flag for indicating transmission parity errors.

Channel Active/Inactive Flag

A channel is normally activated by a function (76 or 77) instruction or by an activate channel (74) instruction. The channel can also be activated by an external device.

A function instruction conditions the external device for a coming data or status information transfer. The instruction places a 12-bit function word plus parity in the channel register and sets the active and full flags. The function word and a function signal are sent to the external device. No active or full signals are sent during a function instruction. The external device accepts the function word and sends an inactive signal which clears the channel active and full flags, clearing the channel register.

An activate channel instruction prepares a channel for data transfer and sends an active signal to the external device. Subsequent input or output instructions transfer data. A disconnect channel (75) instruction after a data transfer returns the channel to an inactive state, and an inactive signal is sent to the external device.

Register Full/Empty Flag

A register is full when it contains a function or data word for an external device or contains a word received from the external device. The register is empty when the flag clears. The flag is turned on

or off as the register changes state. A channel can only be full when it is active.

On data output, the processor places a word in the channel register (the channel should be active and empty) and sets a full flag. The data word plus parity and a full signal are sent to the external device. The external device accepts the word and sends an empty signal to the channel which clears the full flag, clearing the channel register. The active and empty status of the channel signals the PP to send the next word to the register.

On data input, the external device sends a word and a full signal to the data channel. The word is placed in the channel register, and the full flag sets. The PP stores the word and clears the full flag, clearing the data register. An empty signal is sent to the external device signaling it to send

Channel (Marker) Flag Instructions (641, 651)

This flag is used by software as a marker and does not affect hardware operation. When PPs in the same time slot use this flag, priority conflicts exist. For channel 17g (maintenance channel) marker flag, priority problems are resolved by hardware. For other channels, such conflicts must be resolved by software. Any five consecutively numbered PPs are not in the same time slot.

Error Flag Instructions (661, 671)

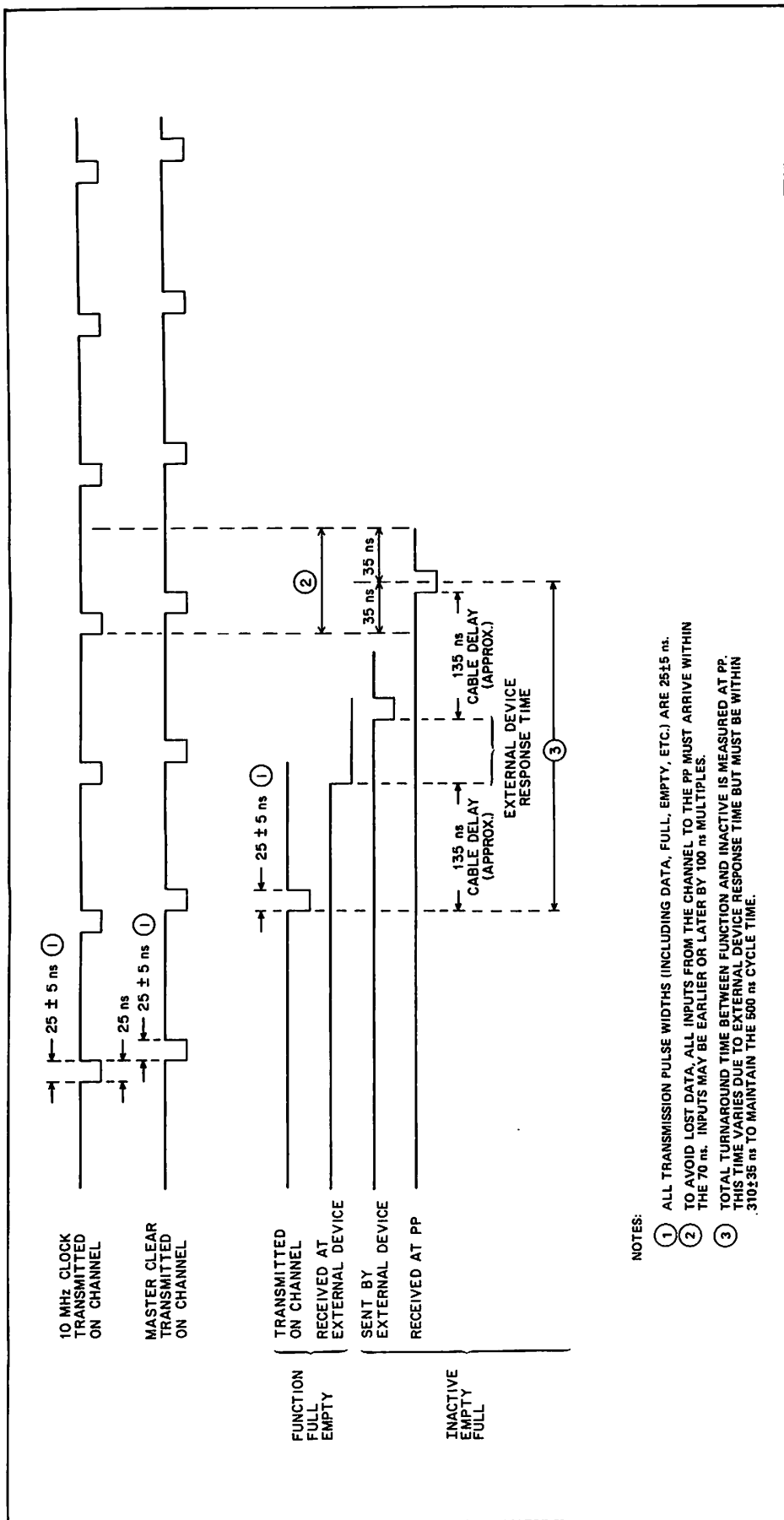
This flag indicates an input data parity error on the specific channel being tested. It also indicates an output data parity error on channels which have the capability of sending an error signal to the IOU in case of such an error. The status register of the device concerned must be read to verify output data integrity.

Channel Transfer Timing

Figure 5-7 shows channel transfer timing. All signal pulses are 25 ± 5 nanoseconds in width and occur 25 ± 5 nanoseconds following the 10-megahertz clock.

To maintain the fastest possible cycle time (250 nanoseconds), a function/full/empty pulse from the PP must be answered with an inactive/empty/full pulse, respectively, within 310 ± 35 nanoseconds. If the maximum speed is not required, this response time may be increased by multiples of 100 nanoseconds.

The PP master clock frequency can be varied by ± 2 percent. The peripheral devices used must tolerate this frequency variation.



NOTES:

- ① ALL TRANSMISSION PULSE WIDTHS (INCLUDING DATA, FULL, EMPTY, ETC.) ARE $25 \pm 5 \text{ ns}$.
- ② TO AVOID LOST DATA, ALL INPUTS FROM THE CHANNEL TO THE PP MUST ARRIVE WITHIN THE 70 ns . INPUTS MAY BE EARLIER OR LATER BY 100 ns MULTIPLES.
- ③ TOTAL TURNAROUND TIME BETWEEN FUNCTION AND INACTIVE IS MEASURED AT PP. THIS TIME VARIES DUE TO EXTERNAL DEVICE RESPONSE TIME BUT MUST BE WITHIN $310 \pm 35 \text{ ns}$ TO MAINTAIN THE 500 ns CYCLE TIME.

Figure 5-7. Channel Transfer Timing

INPUT/OUTPUT TRANSFERS

Data Input Sequence

The external device sends data (figure 5-8) to the PP via the controller as follows:

1. The PP places a function word in the channel register and sets the full flag and the channel active flag. At the same time, the PP sends the first of a group of words and function signals to all controllers. The function signals cause all controllers to sample the words and identify the words as function codes rather than data words. Connect codes select controllers and modes of operation and clear nonselected controllers. Only selected controllers are connected.
2. The controller sends an inactive signal to the PP, indicating acceptance of the function code. The signal drops the channel active flag, which in turn drops the full flag and clears the channel register.
3. The PP sets the channel active flag and sends an active signal to the controller which signals the input equipment to start sending data.
4. The input equipment reads a 12-bit data word plus one parity bit and then sends the word with parity to the channel register with a full signal which sets the channel full flag (10 to 15 nanoseconds after the data arrives).
5. The PP stores the word, drops the full flag, and returns an empty signal, indicating acceptance of the word. The input equipment clears its data register and prepares to send the next word.
6. Steps 4 and 5 repeat for each word transferred.
7. At the end of the transfer, the controller clears its active condition and sends an inactive signal to the PP to indicate the end of the data. The signal clears the channel active flag to disconnect the controller and the PP from the channel.

8. As an alternative, the PP may choose to disconnect from the channel before the input equipment has sent all its data. The PP does this by dropping the active flag and sending an inactive signal to the controller which immediately clears its active condition and sends no more data, although the input equipment may continue to the end of its record or cycle (for example, a magnetic tape unit would continue to end-of-record and stop in the record gap).

Data Output Sequence

The PP sends data (figure 5-9) to the external device as follows:

1. The PP places a function word in the channel register and sets the full flag and the channel active flag. The function signal causes all controllers to sample the word and identify the word as a function code rather than a data word. Connect codes select controllers and modes of operation and clear nonselected controllers. Only selected controllers are connected.
2. The controller sends an inactive signal to the PP, indicating acceptance of the function code. The signal drops the channel active flag, which in turn drops the full flag and clears the channel register.
3. The PP sets the channel active flag and sends an active signal to the controller which signals the output equipment that data flow is starting.
4. The PP places a 12-bit data word plus one parity bit in the channel register and sets the full flag. Coincidentally, the PP sends a word with parity and a full signal to the controller.
5. The controller accepts the word and sends an empty signal to the PP where the signal clears the channel register and drops the full flag.
6. Steps 4 and 5 repeat for each PP word.
7. After the last word is transferred and acknowledged by the controller empty signal, the PP drops the channel active flag and turns off the controller with an inactive signal.

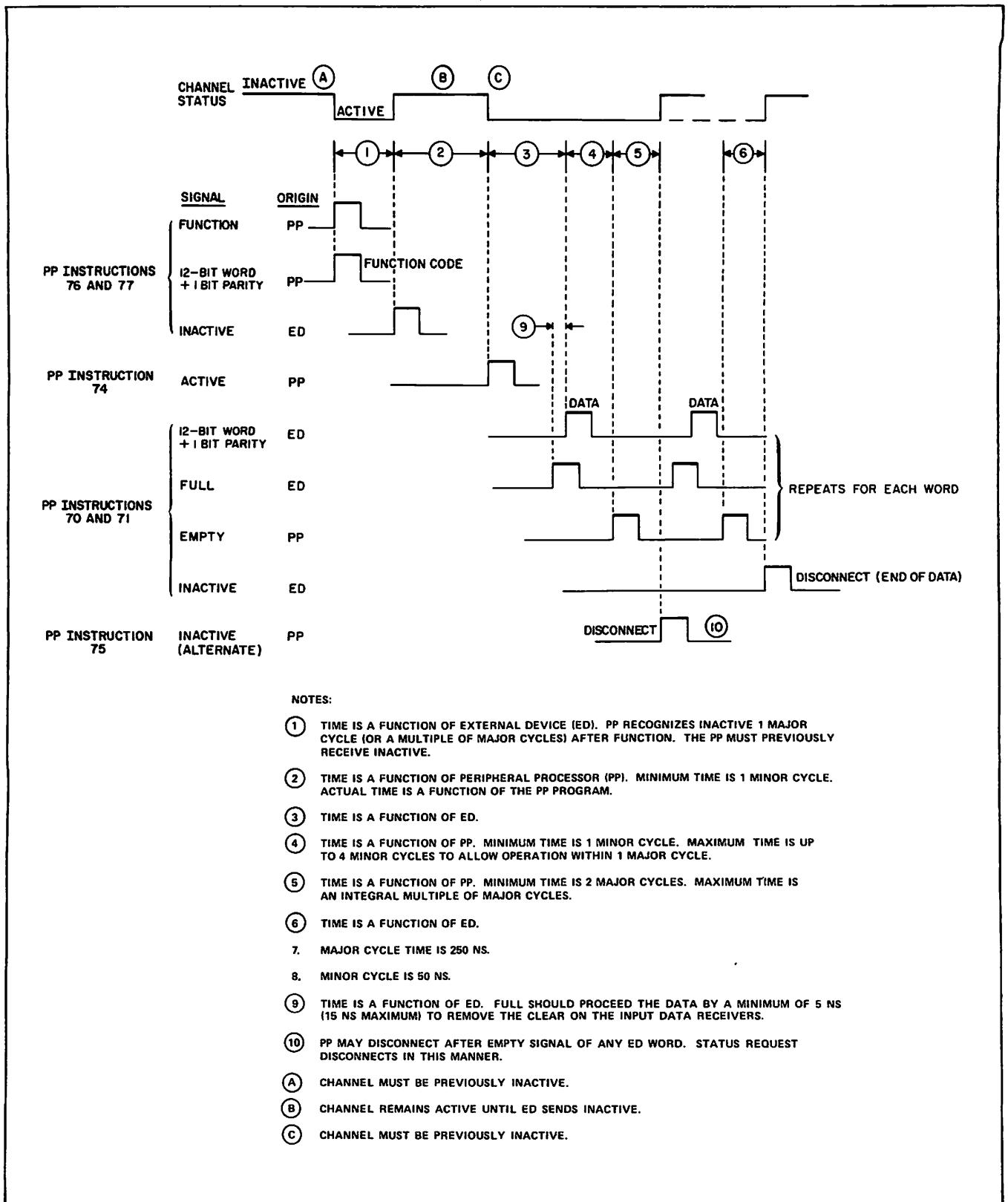
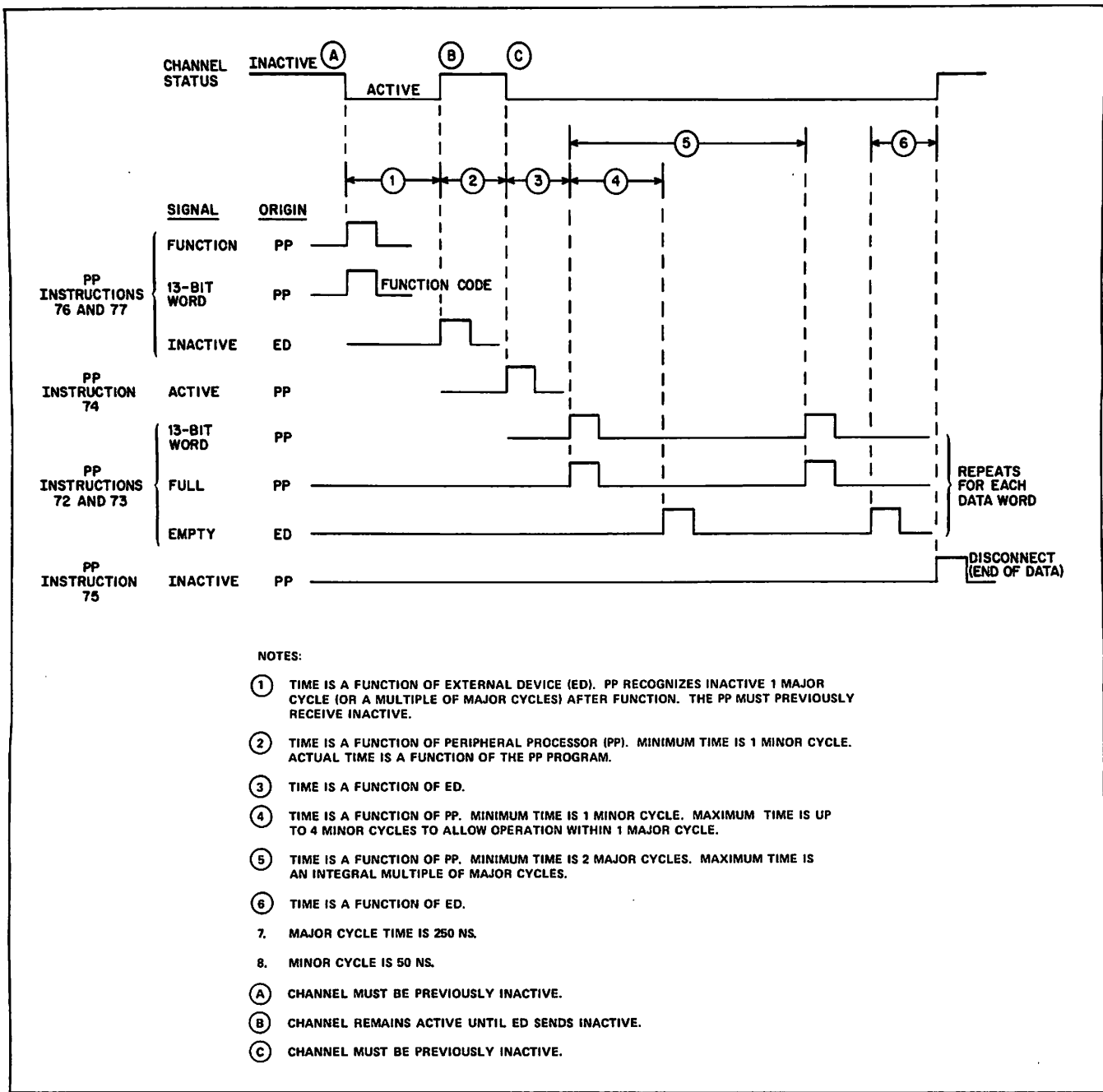


Figure 5-8. Data Input Sequence Timing



NOTES:

- ① TIME IS A FUNCTION OF EXTERNAL DEVICE (ED). PP RECOGNIZES INACTIVE 1 MAJOR CYCLE (OR A MULTIPLE OF MAJOR CYCLES) AFTER FUNCTION. THE PP MUST PREVIOUSLY RECEIVE INACTIVE.
- ② TIME IS A FUNCTION OF PERIPHERAL PROCESSOR (PP). MINIMUM TIME IS 1 MINOR CYCLE. ACTUAL TIME IS A FUNCTION OF THE PP PROGRAM.
- ③ TIME IS A FUNCTION OF ED.
- ④ TIME IS A FUNCTION OF PP. MINIMUM TIME IS 1 MINOR CYCLE. MAXIMUM TIME IS UP TO 4 MINOR CYCLES TO ALLOW OPERATION WITHIN 1 MAJOR CYCLE.
- ⑤ TIME IS A FUNCTION OF PP. MINIMUM TIME IS 2 MAJOR CYCLES. MAXIMUM TIME IS AN INTEGRAL MULTIPLE OF MAJOR CYCLES.
- ⑥ TIME IS A FUNCTION OF ED.
7. MAJOR CYCLE TIME IS 250 NS.
8. MINOR CYCLE IS 50 NS.
- Ⓐ CHANNEL MUST BE PREVIOUSLY INACTIVE.
- Ⓑ CHANNEL REMAINS ACTIVE UNTIL ED SENDS INACTIVE.
- Ⓒ CHANNEL MUST BE PREVIOUSLY INACTIVE.

Figure 5-9. Data Output Sequence Timing

DISPLAY STATION PROGRAMMING

KEYBOARD

A PP transmits function code 7020_g to request data from the keyboard of the display station. The PP then activates the input channel and inputs one character from the keyboard. This character enters as the lower 6 bits of the word; the upper bits are cleared. There is no status report by the keyboard. Table 5-9 lists the keyboard character codes.

Table 5-9. Keyboard Character Codes

Character	Code	Character	Code
No data	00	0	33
A	01	1	34
B	02	2	35
C	03	3	36
D	04	4	37
E	05	5	40
F	06	6	41
G	07	7	42
H	10	8	43
I	11	9	44
J	12	+	45
K	13	-	46
L	14	*	47
M	15	/	50
N	16	(51
O	17)	52
P	20	Left blank key	53
Q	21	=	54
R	22	Right blank key	55
S	23	,	56
T	24	.	57
U	25	Carriage return	60
V	26	Backspace	61
W	27	Space	62
X	30		
Y	31		
Z	32		

DATA DISPLAY

Data is displayed within an 8- by 11-inch area of a cathode-ray tube (CRT). The display can be in character mode (alphanumeric) and/or dot mode (graphic). Two presentation areas (left and right) are displayed. Each is made up of 262 144 dot locations arranged in a 512-by-512 dot format. Each dot position is determined by the intersection of X

and Y coordinates. The lower left corner dot is octal address X=6000 and Y=7000, and the upper right corner dot is octal address X=6777 and Y=7777.

Character Mode

In character mode, three sizes are provided. Large characters are arranged in a 32-by-32 dot format with 16 characters per line. Medium characters are arranged in a 16-by-16 dot format with 32 characters per line. Small characters are arranged in an 8-by-8 dot format with 64 characters per line. Table 5-10 lists the display character codes.

Table 5-10. Display Character Codes

Character	Code	Character	Code
Space	00	0	33
A	01	1	34
B	02	2	35
C	03	3	36
D	04	4	37
E	05	5	40
F	06	6	41
G	07	7	42
H	10	8	43
I	11	9	44
J	12	+	45
K	13	-	46
L	14	*	47
M	15	/	50
N	16	(51
O	17)	52
P	20	Space	53
Q	21	=	54
R	22	Space	55
S	23	,	56
T	24	.	57
U	25		
V	26		
W	27		
X	30		
Y	31		
Z	32		

Dot Mode

In dot mode, display dots are positioned by the X and Y coordinates. The X coordinates position the dots horizontally. The Y coordinates position the dots vertically and unblank the CRT for each dot. Horizontal lines are formed by a series of X and Y coordinates. Vertical lines are formed by a single X coordinate and a series of Y coordinates.

Codes

A single function word is transmitted to select the presentation, mode, and character size (character mode only). Figure 5-10 illustrates the function word format. The word following the function word specifies the starting coordinates for the display (for either mode). Figure 5-11 illustrates the coordinate data word. In character mode, the words that follow are display character codes. Figure 5-12 illustrates the character data word.

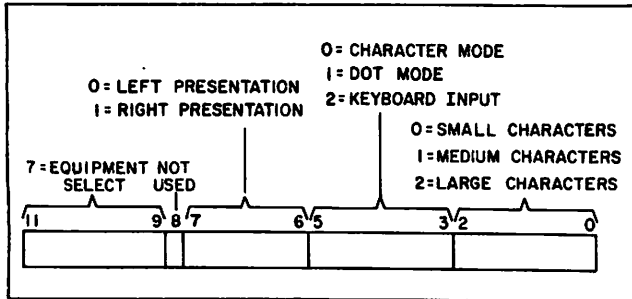


Figure 5-10. Display Station Output Function Code

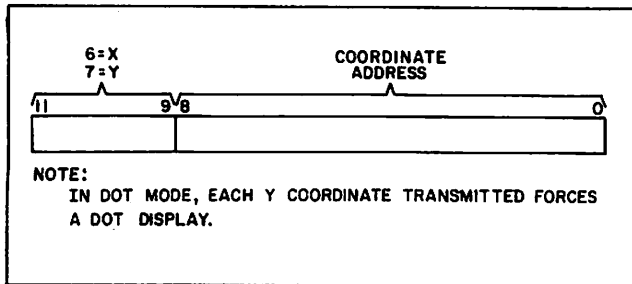


Figure 5-11. Coordinate Data Word

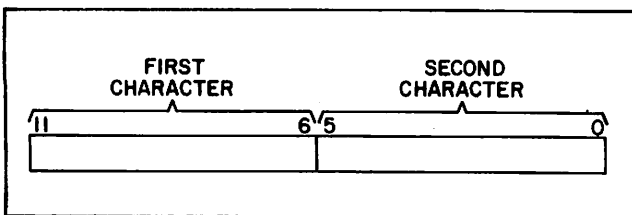


Figure 5-12. Character Data Word

When the display operation has started, the controller regulates character spacing on the line. A new coordinate data word must be sent to start each line. If new coordinates are not specified, data is written on the line specified by the active coordinate word, and information already on that line is overwritten. Character sizes can be mixed by sending a new function word and coordinate word for each size change. Spacing on a line can be varied by sending a coordinate word for the character which is to be spaced differently.

PROGRAMMING EXAMPLE

The following programming example (figure 5-13) requests an input of one line of data from the display station and displays this data on the CRT as it is being typed.

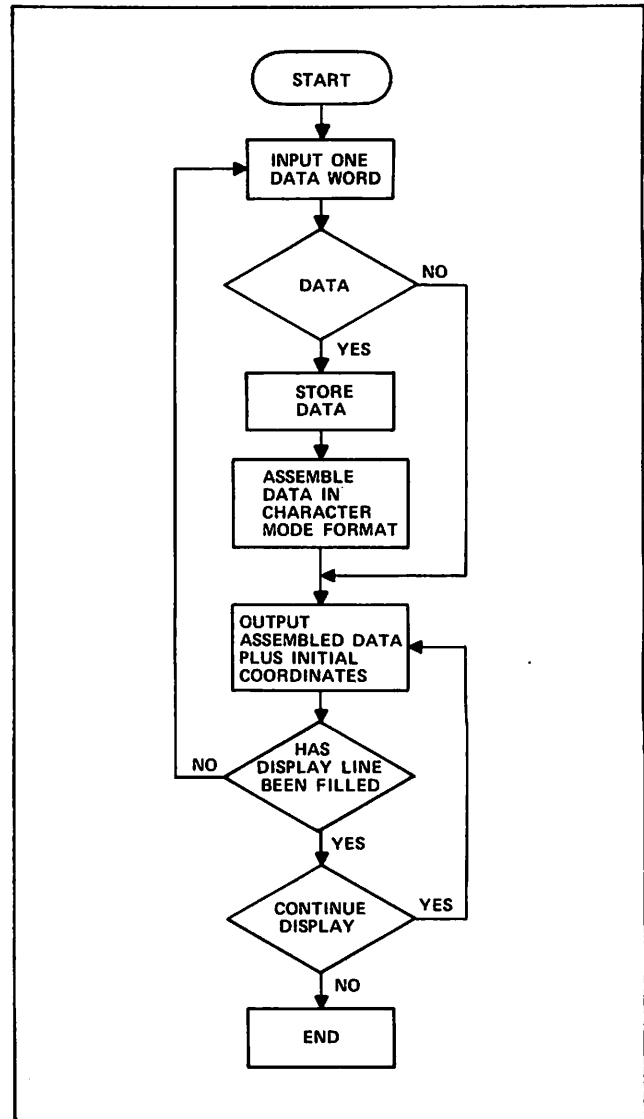


Figure 5-13. Receive and Display Program Flowchart

PROGRAMMING TIMING CONSIDERATIONS

When performing an output operation, the computer must wait at the end of the output for a channel empty condition to prevent a loss of coordinates or data. A full jump at the end of the output ensures that the channel is empty and the display controller accepts the last word of the output before disconnecting from the channel.

REAL-TIME CLOCK PROGRAMMING

Channel 14_g is reserved for the real-time clock. This channel is always active and full and may be read at any time. The real-time clock is a 12-bit free-running counter incrementing at a 1-megahertz rate from 0 through 4095₁₀.

TWO-PORT MULTIPLEXER PROGRAMMING

NOTE

For two-port multiplexer programming, bit numbering within words is 0 through 63 from left to right.

Channel 15_g is reserved for communications with one or two external devices through the two-port multiplexer. One port is reserved for maintenance purposes and the other is reserved for future use. The two-port multiplexer can communicate with all external devices which use EIA standard RS-232C serial interface. The multiplexer can accommodate data with odd/even parity, 5 to 8 bits per character and 1 or 2 stop bits. The format is set by issuing appropriate function codes. The rate is switch selectable for each channel for operation between 110 and 9600 baud. These switches are located internally on the two-port multiplexer.

TWO-PORT MULTIPLEXER OPERATION

The two-port multiplexer uses the rightmost 12 bits on channel 15_g. A 12-bit (octal) function word from the PP is translated to specify the following operating conditions.

<u>Code</u>	<u>Function</u>
7XXX	Terminal select
6XXX	Terminal deselect
00XX	Read status summary
01XX	Read terminal data
02XX	Write output buffer
03XX	Set operation mode to terminal
04XX	Set/clear terminal control signal, data terminal ready (DTR)
05XX	Set/clear terminal control signal, request to send (RTS)
06XX	Not used
07XX	Master clear selected port

Terminal Select (7XXX)

The PP sends this select code to specify the terminal to which the function codes and data transmissions apply. Code 7000 selects port 0 (for future use) and code 7001 selects port 1 (maintenance console).

Terminal Deselect (6XXX)

The PP sends this code which deselects the two-port multiplexer from channel 15_g so the 16-bit channel is available for inter-PP communications.

Read Status Summary (00XX)

This code permits the PP to input status from the selected terminal. One-word input must follow to read the status response. The response is 12 bits.

<u>Bit</u>	<u>Status</u>
52-58	Not used
59	Output buffer not full
60	Input ready
61	Data carrier detect or carrier on
62	Data set ready
63	Ring indication

PP Read Terminal Data (01XX)

This code permits the PP to input the terminal data from the selected terminal. Channel 15_g must be activated and a one-word input must follow to read in the terminal data. The data word is 12 bits.

<u>Bit</u>	<u>Status</u>
52	Data set ready
53	Data set ready and data carrier detector
54	Over run
55	Framing or parity error
56-63	8-bit data

Data Set Ready (Bit 52)

When the data set ready signal is active, this bit sets.

**Data Set Ready (DSR) and Data Carrier Detector (DCD)
(Bit 53)**

When both data set ready and data carrier detector signals are active, this bit sets.

Over Run (Bit 54)

When the previously received character is not read by the PP before the present character is transferred to the data holding register, the over run bit sets.

Framing or Parity Error (Bit 55)

When the received character does not have a valid stop bit (framing error), or when this bit sets, the received character parity does not agree with the select parity (parity error).

Data Character (Bits 56 through 63)

The lower 8 bits of the input word form the data character. The multiplexer forms this character directly from the Universal Asynchronous Receiver and Transmitter (UART).

PP Write Output Buffer (02XX)

This code prepares the multiplexer for an output operation to the 64-character output buffer memory. Before an output operation can proceed, channel 15g must be activated. The output operation is terminated when the multiplexer receives an inactive signal from the PP, or when no more locations are available in the output buffer. In the latter case, an inactive (instead of empty) signal is sent back to the channel, which in turn will terminate the output operations.

Set Operation Mode to the Terminal (03XX)

This code permits the PP to set the terminal operation mode register. A 12-bit function code word from the PP specifies the operation of the terminal. This word is decoded in the function register. Segments of the word define the mode as follows:

<u>Bit</u>	<u>Status</u>
58	Not used
59	No parity
	When this bit is set, it eliminates the parity bit from the transmitted and received characters. The stop bit(s) immediately follow the last data bit.

Bit Status

60 Number of stop bits
This bit selects the number of stop bits, 1 or 2, to be appended immediately after the parity bit. When this bit is clear, it inserts 1 stop bit and when set, it inserts 2 stop bits.

61-62 Number of bits per character
These 2 bits are internally decoded to select 5, 6, 7, or 8 data bits per character.

<u>Bit 61</u>	<u>Bit 62</u>	<u>Bits per Character</u>
0	0	5
0	1	6
1	0	7
1	1	8

63 Odd/even parity select
This bit selects the type of parity which will be appended immediately after the data bits. It also determines the parity that will be checked on read data.

Set/Clear Data Terminal Ready (04XX)

This code permits the PP to set or clear the terminal control signal, data terminal ready (DTR). When bit 63 is set, DTR is active, and when bit 63 is clear, DTR is inactive.

Set/Clear Request to Send (05XX)

This code permits the PP to set or clear the terminal control signal, request to send (RTS). When bit 63 is set, RTS is active, and when bit 63 is clear, RTS is inactive.

Master Clear (07XX)

This code permits the PP to master clear the selected port including its output buffer memory and UART. The terminal operation mode register and terminal control signals are not cleared.

PROGRAMMING CONSIDERATIONS

Channel 15g communicates with the terminals connected to the external interface, one at a time. To establish communications between a PP and the terminal, the PP issues a function for select. The function word for select is formed by the least significant 12 bits, sent to channel 15g, and specifies the following information.

- A select code to select the multiplexer (7XXX).
- The terminal with which the PP would like to establish communication (7XXX).

When the connect is established, the two-port multiplexer routes all data to the terminal designated by the select code. The multiplexer responds with the inactive signal to acknowledge the receipt of the function code of 7XXX for select, 6XXX for deselect, and 0XXX for operation. Otherwise, the function is ignored by the multiplexer.

Output Data

The multiplexer accepts a maximum data block length of 64 characters per terminal. During the block data transfer, the multiplexer terminates the output operation either when it receives an inactive signal from the channel or when the output buffer is full. When the output buffer is full, the multiplexer sends back an inactive signal instead of an empty signal to the channel on the last output word. The signal indicates the output buffer accepts the last output word and it cannot receive anymore data from the PP. Output to a full buffer is not allowed by the multiplexer. The multiplexer sends back an inactive signal to deactivate channel 15g after the multiplexer decodes the previous function code which is 02XX (PP write output buffer), and receives an activate signal from the PP.

Input Data

The multiplexer does not store the input data from the terminal. A lost data condition exists if the PP does not input the previous data before the new data arrives from the terminal. The multiplexer allows input from an empty input buffer.

Request to Send and Data Terminal Ready

Request to send and data terminal ready are brought up automatically by the hardware under the following conditions regardless of the software RTS and DTR bits.

- Data in the UART output register.
- Data in the FIFO output register.

When no data is in the FIFO or UART, the software bit determines RTS and DTR.

MAINTENANCE CHANNEL PROGRAMMING

NOTE

Maintenance registers are numbered 0 through 63 from left to right.

MAINTENANCE CHANNEL

A PP in the IOU can perform any or all of the following operations through the maintenance channel (MCH) to each system element, such as the CP, IOU, and CM.

- Initializing registers, controls, and memories.
- Monitoring and recording error information.
- Verifying error detection and correction hardware.

The maintenance channel consists of the maintenance channel interface on channel 17g, a maintenance channel interface in each system element, and a set of interconnecting cables.

The IOU maintenance channel interface contains a selector that connects to one of up to seven system elements. The IOU is element 0 and its maintenance access control is internally connected to the selector. All other system elements are assigned arbitrary element numbers. Each maintenance access control is connected to the selector by a single cable. This arrangement results in a radial connection that allows any system element to be shut down or removed without affecting communication with the other elements.

MCH FUNCTION WORDS

The MCH function word consists of the connect, opcode, and type fields which are used as described in the next three paragraphs and table 5-11.

The connect field specifies the unit to which the MCH is connected (CP, CM, or IOU), controlling selection within the IOU only. The unit remains connected until another connect code selects a different unit. Connect codes 10g to 17g leave the MCH unconnected; in this state the interface can be used for PP to PP communications.

The OPCODE field controls the unit selected by the connect code, preparing the unit for a coming read/write/echo operation, or causing the unit to halt, start, clear, or deadstart.

The use of the TYPE field depends on the connected unit. When the CP is the connected unit, type codes 1 through A₁₆ (model 835) or 1 through 7 (model 855) specify the data type in the operation to be performed. Also, for the CP, type code 0 specifies that the internal address of the CP register to be connected is specified in a control word which is sent as two data words immediately following the function word. When IOU is the connected unit, type codes 0 through 7 specify the starting byte number for read/write operations. The exceptions are reading the options installed and element identifier registers. On the model 835, CM ignores the type code. On the model 855, CM uses A₁₆ to access the maintenance registers.

Table 5-11. MCH Function Word Bit Assignments (Sheet 1 of 2)

Field	Description	
MCH Function Word to Model 835 CP		
CONNECT (bits 8-11)	Code 2 ₁₆ =	Connect CP maintenance registers
OPCODE (bits 4-7)	Code 0 ₁₆ = 1 ₁₆ = 4 ₁₆ = 5 ₁₆ = 6 ₁₆ = 7 ₁₆ =	Halt processor Start processor Prepare for read (control word required) Prepare for write (control word required) Master clear Clear errors
TYPE (bits 0-3)	Code 0 ₁₆ =	Control word required
MCH Function Word to Model 835 CM		
CONNECT (bits 8-11)	Code 1 ₁₆ =	Connect CM maintenance registers
OPCODE (bits 4-7)	Code 4 ₁₆ = 5 ₁₆ = 6 ₁₆ = 7 ₁₆ =	Prepare for read (control word required) Prepare for write (control word required) Master clear Clear fault status register
MCH Function Word to Model 855 CP and CM		
CONNECT (bits 8-11)	Code 1 ₁₆ =	Required for model 855 CP and CM
TYPE (bits 0-3)	Code 0 ₁₆ =	CP and CP registers
OPCODE (bits 4-7)	Code 0 ₁₆ = 1 ₁₆ = 4 ₁₆ = 5 ₁₆ = 6 ₁₆ = 7 ₁₆ =	Halt processor Start processor Prepare for read Prepare for write Master clear Clear errors
TYPE (bits 0-3)	Code 1 ₁₆ =	Control store memory
OPCODE (bits 4-7)	Code 4 ₁₆ = 5 ₁₆ =	Prepare for read Prepare for write
TYPE (bits 0-3)	Code 3-7 ₁₆ =	Internal memories
OPCODE (bits 4-7)	Code 4 ₁₆ = 5 ₁₆ =	Prepare for read Prepare for write
TYPE (bits 0-3)	Code A ₁₆ =	CM and CM registers
OPCODE (bits 4-7)	Code 4 ₁₆ = 5 ₁₆ = 6 ₁₆ = 7 ₁₆ =	Prepare for read Prepare for write Master clear Clear errors

Table 5-11. MCH Function Word Bit Assignments (Sheet 2 of 2)

Field		Description
MCH Function Word to IOU		
CONNECT (bits 8-11)	Code 0 ₁₆ =	Connect IOU maintenance registers
OPCODE (bits 4-7)	Code 4 ₁₆ =	Prepare for read (control word required)
	5 ₁₆ =	Prepare for write (control word required)
	6 ₁₆ =	Master clear
	7 ₁₆ =	Clear fault status registers
	C ₁₆ =	Read IOU status summary (reads one byte, control word not required)
TYPE (bits 0-3)	Codes 0-7 ₁₆ =	IOU registers are read circularly (byte 0 follows byte 7) from the byte specified by the TYPE field

MCH CONTROL WORDS

Some function words must be followed by two 8-bit control words which specify the internal address of the register to be accessed. This is accomplished by transmitting two PP words where the rightmost 8 bits in each word are used. Control words are required for the following:

- Function words to a model 835 CP with opcodes 4/5 (read/write) and type code 0.
- Function words to a model 855 CP with opcodes 4/5.
- Function words to CM and IOU with opcodes 4/5.
- Function words to CP, CM, and IOU with opcode 8 (echo).

Refer to tables 5-12 through 5-16 for CP, CM, and IOU internal address assignments.

MCH Programming for Halt/Start (Opcode 0/1)

These operations consist of the output of a function word. A halt opcode halts the processor without damaging the process being executed, including the integrity of the interunit communication of the halted processor such as CDC CYBER 170 exchange request communication, central memory communications, and the process state. If the process is subsequently restarted without performing any other MCH operations, or after performing read/write with certain precautions as described in Operating Systems Manual, the process continues without damage.

MCH Programming for Read/Write (Opcode 4/5)

Refer to Programming for PP Data Input/Output in this section for a more complete procedure. In general terms, proceed as follows:

1. Issue function with opcode 4/5.
2. Output first control word.

3. Verify error flag clear.
4. Output second control word.
5. Verify error flag clear.
6. Input/output required number of data words.
7. Verify error flag clear.

Reading a nonexistent register returns all zeros. Writing to a read-only register, or to a nonexistent register, does not alter any register. Most registers are read/write as 64-bit (eight-byte) registers, requiring the input/output of eight MCH data words. Most registers which are physically smaller than eight bytes are right-justified with zero-fill. Exceptions are as follows:

- Reading a status summary register repeats the status information in each byte.
- The IOU may disconnect the MCH without affecting subsequent MCH operations in the following cases:

- After reading one to eight bytes from any maintenance register.
- After writing one byte to a corrected error log register.
- After writing one byte to an uncorrected error log register.

The following MCH operations on CP registers can be performed with the CP running or halted.

- Read CP status summary register.
- Read CP fault status register.
- Read CP corrected error log registers.
- Read CP options installed registers.
- Read CP element identifier register.
- Read/write CP dependent environmental control register.

- Read/write test mode control registers.
- Clear errors.

arbitrarily clears the connected unit, without regard to possible information loss. Clear errors clears the error indicators in the connected unit; to avoid loss of error information while the errors are cleared, the unit concerned should be halted.

To read/write other CP registers, the CP must be running since these registers are accessed by microcode. Refer to tables 5-17 through 5-21 for register bit assignments.

MCH Programming for Master Clear/Clear Errors (Opcode 6/7)

These operations consist of the output of a single function word. The master clear immediately and

MCH Programming for Read IOU Status Summary (Opcode C, IOU Only)

This operation is an alternative, faster means of reading the IOU status summary register.

1. Issue function with opcode C.
2. Input status summary byte.

Table 5-12. Model 835 CP Internal Address Assignments

Internal Address (1)		Type		Description
Hex	Octal	(2)	(3)	
00	000	R	A	Status summary register
10	020	R	A	Element identifier register
30	060	R	A	Dependent environment control register
42	082	R	M	Monitor condition register
80	200	R	A	Processor fault status register
81	201	R	A	Control store errors register
90	220	R	A	Retry corrected error log register
92	222	R	A	Cache corrected error log register
93	223	R	A	Map corrected error log register

NOTES:

(1) The internal address is the second byte of two 8-bit control words which must be supplied after a function word output with OPCODE = 4/5. The first byte is discarded.

(2) R = read, W = write

(3) A = always accessible, M = microcode accessible

Table 5-13. Model 835 CM Internal Address Assignments

Internal Address (1)		Type (2)	Description
Hex	Octal		
00	000	R	Status summary register
10	020	R	Element identifier register
12	022	R	Options installed register
A0	240	R/W	Corrected error log register
A4	244	R/W	Uncorrected error log 1 register
A8	250	R/W	Uncorrected error log 2 register

NOTES:

(1) The internal address is the second byte of two 8-bit control words which must be issued after a function word output with OPCODE = 4/5. The first byte is discarded.

(2) R = read, W = write

Table 5-14. Model 855 CP Internal Address Assignments

Internal Address (1)		Type		Description
Hex	Octal	(2)	(3)	
00	000	R	A	Status summary register
10	020	R	A	Element identifier register
30	060	R	A	Dependent environment control register
42	082	R	M	Monitor condition register
80-89	200-211	R	A	Processor fault status registers 1 through 9

NOTES:

(1) The internal address is the second byte of two 8-bit control words which must be supplied after a function word output with OPCODE = 4/5. The first byte is discarded.

(2) R = read, W = write

(3) A = always accessible, M = microcode accessible

Table 5-15. Model 855 CM Internal Address Assignments

Internal Address (1)		Type (2)	Description
Hex	Octal		
00	000	R	Status summary register
10	020	R	Element identifier register
12	022	R	Options installed register
A0	240	R/W	Corrected error log register
A4	244	R/W	Uncorrected error log 1 register
A8	250	R/W	Uncorrected error log 2 register

NOTES:

(1) The internal address is the second byte of two 8-bit control words which must be issued after a function word output with OPCODE = 4/5. The first byte is discarded.

(2) R = read, W = write

Table 5-16. IOU Internal Address Assignments

Internal Address (1)		Type (2)	Description
Hex	Octal		
00	000	R	Status summary register
10	020	R	Element identifier register
12	022	R	Options installed register
18	030	R/W	Fault status mask register
40	100	R	Status register
80	200	R/W	Fault status 1 register
81	201	R/W	Fault status 2 register

NOTES:

(1) The internal address is the second byte of two 8-bit control words which must be issued after a function word output with OPCODE = 4/5. The first byte is discarded.

(2) R = read, W = write

Table 5-17. Model 835 CP Register Bit Assignments † (Sheet 1 of 3)

Channel Bit	Description
CP Status Summary Register (MCH Address 00 ₁₆)	
59 60 61 62 63	Short warning, imminent power failure CP halted Uncorrected error Corrected error Long warning
CP Element Identifier Register (MCH Address 10 ₁₆)	
32-39 40-47 48-63	Element type (00 ₁₆ = CP) Element model (20 ₁₆ = model 835) Element serial number
CP Dependent Environment Control Register (MCH Address 30 ₁₆)	
08 09 10 11 12 13 14 15	Cache memory 1st quarter enabled (0-1 kiloword) Cache memory 2nd quarter enabled (0-1 kiloword) Cache memory 3rd quarter enabled (0-1 kiloword) Cache memory 4th quarter enabled (0-1 kiloword) Cache memory conflict register 0 enabled Cache memory conflict register 1 enabled Cache memory conflict register 2 enabled Cache memory conflict register 3 enabled
CP Monitor Condition Register (MCH Address 42 ₁₆)	
48 50	Uncorrectable error Short warning
CP Processor Fault Status Registers (MCH Address 80 ₁₆ , 81 ₁₆) CP Retry Corrected Error Log Register (MCH Address 90 ₁₆)	
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 20 21 22 23 27 28 29	Address and data cache input 0 Address and data cache input 1 Address and data cache input 2 Address and data cache input 3 Cache output 0 Cache output 1 Cache output 2 Cache output 3 Data port 0-0 Data port 0-1 Data port 0-2 Data port 0-3 Data port 1-0 Data port 1-1 Data port 1-2 Data port 1-3 Indent, port 0 Indent, port 1 Indent, partial write, cache input Indent, cache output Response code = 1 error Response code = 5 error Response code = 7 error Cache time outs No overload on simulated response buffer Function code, cache input
† Register bits are numbered 0 through 63 from left to right.	

Table 5-17. Model 835 CP Register Bit Assignments† (Sheet 2 of 3)

Channel Bit	Description
CP Processor Fault Status Registers (MCH Address 80 ₁₆ , 81 ₁₆) CP Retry Corrected Error Log Register (MCH Address 90 ₁₆) (Contd)	
30 31 32 33 34 35 36 37 38 39 41 42 43 44 48 50 51 52 53 54 56 57 58 59 60 61	Increment indent, cache input Maintenance access control ROMs Register file byte 0 Register file byte 1 Register file byte 2 Register file byte 3 Register file byte 4 Register file byte 5 Register file byte 6 Register file byte 7 I mux, B mux, address I mux, B mux, address I mux, B mux, address I mux, B mux, address Invalidation address, exchange address J stream K stream Byte output Byte control Byte branch ROM Floating point trap ROM Exponent adder ROM and partial writes A data selection ROMs Ident from cache Immediate control ROMs
CP Control Store Parity Fault Status Register (MCH Address 81 ₁₆)	
0 8 16 24 32 40 48 56	Parity of micrand bytes 0/8 Parity of micrand bytes 1/9 Parity of micrand bytes 2/10 Parity of micrand bytes 3/11 Parity of micrand bytes 4/12 Parity of micrand bytes 5/13 Parity of micrand bytes 6/14 Parity of micrand bytes 7/15
CP Cache Corrected Error Log Register (MCH Address 92 ₁₆)	
0 1 2 3 4-6 7 8 9 10 11 12 13 14 15	This log contains valid information Unlogged error Not used Parity error, LRU status array counters Not used No multiple hit in tag arrays No parity errors, tag array blocks 0 and 1 No parity errors, tag array blocks 2 and 3 No parity errors, tag array blocks 4 and 5 No parity errors, tag array blocks 6 and 7 No parity error, tag array blocks 0 and 1 No parity error, tag array blocks 2 and 3 No parity error, tag array blocks 4 and 5 No parity error, tag array blocks 6 and 7
† Register bits are numbered 0 through 63 from left to right.	

Table 5-17. Model 835 CP Register Bit Assignments † (Sheet 3 of 3)

Channel Bit	Description
CP Cache Corrected Error Log Register (MCH Address 92 ₁₆) (Contd)	
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	Compare bits block 0 Compare bits block 1 Compare bits block 2 Compare bits block 3 Compare bits block 4 Compare bits block 5 Compare bits block 6 Compare bits block 7 Compare bits block 0 Compare bits block 1 Compare bits block 2 Compare bits block 3 Compare bits block 4 Compare bits block 5 Compare bits block 6 Compare bits block 7
CP Map Corrected Error Log (MCH Address 93 ₁₆)	
0-31 32 33 34-36 37 38 39 40-63	Reserved Valid map corrected error log (CEL) entry Unlogged error Not used Parity error, segment associative section Multiple hit Multiple hit Parity group fault identifier
† Register bits are numbered 0 through 63 from left to right.	

Table 5-18. Model 835 CM Register Bit Assignments† (Sheet 1 of 2)

Channel Bit	Description
CM Status Summary Register (MCH Address 00 ₁₆)	
56-57 58 61 62 63	Oscillator code (00 = normal, 01 = -2%, 10 = +2%) Clock set to tuning mode Valid uncorrectable error Corrected error Long warning (low/high temperature, condensing unit fault, blower 1/blower 2 fault)
NOTE: The byte is duplicated in each byte.	
CM Element Identifier Register (MCH Address 10 ₁₆)	
32-39 40-47 48-63	Element type (01 ₁₆ = CM) Element model (20 ₁₆ = model 835) Element serial number
CM Options Installed Register (MCH Address 12 ₁₆)	
03 07 09 11 16 19 20 21 22	524K memory installed 1049K memory installed 1573K memory installed 2097K memory installed A memory configuration switch is in the up position Top switch (SW3) is not in the center position Next to top switch (SW4) is not in the center position Next to bottom switch (SW5) is not in the center position Bottom switch (SW6) is not in the center position
CM Corrected Error Log Register (MCH Address A0 ₁₆)	
00 01 08-09 13-14 15-16 17-23 24-30 31-33 35 36 37 42-49	This log contains valid information Unlogged single-bit error Port code (0/1 = port 0/1, 4/5 = not used, 6 = no request, 7 = refresh) Array PAK select address Chip row select address Chip column address Row address Bank address Address parity P5 Address parity P6 Address parity P7 Syndrome bits
CM Uncorrected Error Log 1 Register (MCH Address A4 ₁₆)	
00 01 02 03 04 05	This log contains valid information Unlogged uncorrected error Illegal function Double-bit error Bounds fault First level parity error
† Register bits are numbered 0 through 63 from left to right.	

Table 5-18. Model 835 CM Register Bit Assignments† (Sheet 2 of 2)

Channel Bit	Description
CM Uncorrected Error Log 1 Register (MCH Address A4 ₁₆) (Contd)	
06 07 08-10 13-14 15-16 17-23 24-30 31-33 35 36 37 38-41 50-57 58 59-62 63	Second level parity error Buffer request Port code (0/1/2/3 = port, 4/5 = not used, 6 = no request, 7 = refresh) Array PAK select address Chip row select address Chip column address Chip row address Bank address Address parity P5 Address parity P6 Address parity P7 Parity error byte position code (0 = tag PE, 1/2/3/4/5/6/7/8 = byte 7/6/5/4/3/2/1/0 data PE, 9/A/B = address bits 56-60/48-55/40-47 PE, C = not used, D = mask PE, E = function code PE, F = no error) Mark bits associated with error word Mark parity Function bits associated with error word Function parity
CM Uncorrected Error Log 2 Register (MCH Address A8 ₁₆)	
00 01 02 03 08-10 13-14 15-16 17-23 24-30 31-33 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54	This log contains valid information Unlogged uncorrected error 2 CM partial write parity error Data read parity error Port code (0/1/2/3 = port 0/1/2/3, 4/5 = not used, 6 = no request, 7 = refresh) PAK select address Chip row select address Chip column address Chip row address Bank address Address parity P5 Address parity P6 Address parity P7 Byte 0 read PE (1) Byte 1 read PE (1) Byte 2 read PE (1) Byte 3 read PE (1) Byte 4 read PE (1) Byte 5 read PE (1) Byte 6 read PE (1) Byte 7 read PE (1) Byte 0 PE, partial write Byte 1 PE, partial write Byte 2 PE, partial write Byte 3 PE, partial write Byte 4 PE, partial write Byte 5 PE, partial write Byte 6 PE, partial write Byte 7 PE, partial write Tag out parity error
<p>NOTE:</p> <p>(1) If bit 3 is set in both uncorrected error logs and SECEDED is discarded, these bits point to the bytes with errors for a read data parity error.</p> <p>†Register bits are numbered 0 through 63 from left to right.</p>	

Table 5-19. Model 855 CP Register Bit Assignments† (Sheet 1 of 4)

Channel Bit	Description
CP Status Summary Register (MCH Address 00 ₁₆)	
59 60 61 62 63	Short warning, imminent power failure CP halted Uncorrected error Corrected error Long warning
CP Element Identifier Register (MCH Address 10 ₁₆)	
32-39 40-47 48-63	Element type (00 ₁₆ = CP) Element model (30 ₁₆ = model 855) Element serial number
CP Options Installed Register (MCH Address 12 ₁₆)	
62	32K-byte cache installed
CP Dependent Environment Control Register (MCH Address 30 ₁₆)	
33 35 42 44 47 48 49 50 51 52-56 57-60 61	Test mode Disable corrected error to CP status summary Control store off-line (sweep) Enable control store breakpoint Disable detected uncorrectable errors (processor-detected malfunctions) Apply wide clock margins Apply narrow clock margins Enable cache lookahead Disable unconditional cache lookahead Error retry limit plus parity; bit 56 is parity Enable cache configuration sets 0 through 3 where bit 57 is set 0, bit 58 is set 1, and so on. Cache fake central memory
CP Monitor Condition Register (MCH Address 42 ₁₆)	
48 50	Uncorrectable error Short warning
CP Processor Fault Status Register 0 (MCH Address 80)	
0 1 2-7 8-15 16-23 24-31	Uncorrected error One of the following has occurred: corrected error, soft error, or bypass of a processor fault Address fan-in parity error, bytes 0 through 5 where bit 2 indicates byte 0 and so on Input data stream parity error, bytes 0 through 7 where bit 8 indicates byte 0 and so on A data stream holding register parity error, bytes 0 through 7 where bit 16 indicates byte 0 and so on B data stream holding register parity error, bytes 0 through 7 where bit 24 indicates byte 0 and so on
† Register bits are numbered 0 through 63 from left to right.	

Table 5-19. Model 855 CP Register Bit Assignments † (Sheet 2 of 4)

Channel Bit	Description
CP Processor Fault Status Register 0 (MCH Address 80) (Contd)	
38-39 40-43 47 48 49 50 51 52 54 55 56 57 58 59 61 62 63	Operation code parity error where bit 38 is byte 0 and bit 39 is byte 1 Recovery address parity error, bytes 0 through 3 where bit 40 indicates byte 0 and so on Shift count parity error A length counter parity error B length counter parity error AJ in MUX parity error AK in MUX parity error Left bit shift parity error AK port input parity error AJ port input parity error Register file A address counter parity error Register file B address counter parity error Register file A data parity error Register file B data parity error Limit register stage 3 parity error Stage 7 parity error PFS PAK 0 parity error
CP Processor Fault Status Register 1 (MCH Address 81 ₁₆)	
0 1 2 3 4 5 6 7 8 9 10 11	Input stream buffer address parity error Output stage 2 parity error RAM PAK 1 - memory A or memory B address parity error RAM PAK 1 - memory A or memory B data parity error Port J second data latch parity error Port K second data latch parity error AJ descriptor latch parity error AK descriptor latch parity error RAM PAK 0 - memory A address parity error RAM PAK 0 - memory A data parity error RAM PAK 0 - memory B address parity error RAM PAK 0 - memory B data parity error
CP Processor Fault Status Register 2 (MCH Address 82 ₁₆)	
0 1 2-7 8-15 16 17 20 21 23 24 25 26 27 28-31 32-35 52 56 57 58	Scale counter parity error Edit mask parity error Cache address register parity error, bytes 0 through 5 where bit 2 indicates byte 0 and so on Cache write data parity error, bytes 0 through 7 where bit 8 indicates byte 0 and so on Cache multiple hit Cache multiple allocate error LM MUX pointer, direct CMC data LM MUX pointer, cache read data LM MUX pointer, buffered CMC data Cache write data from CP parity error Cache write data from CM parity error Cache associative tag parity error Cache mark parity error Car error pointer parity error, bytes 0 through 3 where bit 28 indicates byte 0 and so on LM input control parity error, bytes 0 through 3 where bit 32 indicates byte 0 and so on Exchange tag parity error CM corrected write CM corrected read CM uncorrected write
† Register bits are numbered 0 through 63 from left to right.	

Table 5-19. Model 855 CP Register Bit Assignments † (Sheet 3 of 4)

Channel Bit	Description
CP Processor Fault Status Register 2 (MCH Address 82 ₁₆) (Contd)	
59 60 61 62 63	CM uncorrected read CM reject CM response code parity error CM tag parity error PFS PAK 1 parity error
CP Processor Fault Status Register 3 (MCH Address 83 ₁₆)	
0-3 4-7 8-11	Cache address parity error, bytes 0 through 3 where bit 0 indicates byte 0 and so on Cache tag RAM parity error, bytes 0 through 3 where bit 4 indicates byte 0 and so on Cache MUX parity error, bytes 0 through 3 where bit 8 indicates byte 0 and so on
CP Processor Fault Status Register 4 (MCH Address 84 ₁₆)	
0 1 12-15 16-19 20 24 25 26 32-33 36-39 41 42 45 46 56 57-58	Set 0 parity error Set 1 parity error New P register parity error, byte 0 through 3 where bit 12 indicates byte 0 and so on Address input parity error, bytes 0 through 3 where bit 16 indicates byte 0 and so on Bypass register parity error Input purge code parity error Data type rank 32 parity error JK rank 32 parity error Write data parity error where bit 32 is 0 through 7, and bit 33 is 8 through 15 P register parity error Successful entry Deadman time-out Retry counter parity error Exchange halt Write address minipipe parity error Register file out parity error where bit 57 is bytes 0 through 3 and bit 58 is bytes 4 through 7
CP Processor Fault Status Register 5 (MCH Address 85 ₁₆)	
13	Tag parity error
CP Processor Fault Status Register 6 (MCH Address 86 ₁₆)	
32 33 34 35 36 37 40 41 42 48 49 50	Maintenance access control (MAC) read path parity error Maintenance channel (MCH) parity error MAC data fanout parity error MAC data bus parity error RAM address MUX parity error Address translation MUX parity error CYBER 170 monitor or job mode odd RAM PAK 0 parity error CYBER 170 monitor or job mode even RAM PAK 0 parity error Executive state RAM PAK 0 parity error CYBER 170 monitor or job mode even RAM PAK 1 parity error CYBER 170 monitor or job mode even RAM PAK 1 parity error Executive state RAM PAK 1 parity error
† Register bits are numbered 0 through 63 from left to right.	

Table 5-19. Model 855 CP Register Bit Assignments† (Sheet 4 of 4)

Channel Bit	Description
CP Processor Fault Status Register 7 (MCH Address 87 ₁₆)	
0-3	Branch address address
CP Processor Fault Status Register 8 (MCH Address 88 ₁₆)	
14 15 16-23 24-31 32-43 44-55 56 58-59 60-61	Soft control register parity error Soft control register parity error Multiply C register parity error, bytes 0 through 7 where bit 16 indicates byte 0 and so on Multiply B register parity error, bytes 0 through 7 where bit 24 indicates byte 0 and so on General network adder parity error, bytes 0 through 11 where bit 32 indicates byte 0 and so on General network adder carry miscompare, bytes 0 through 11 where bit 44 indicates byte 0 and so on Multisoft control parity error, byte 0 Shift parity error where bit 58 is byte 0 and bit 59 is byte 1 Multiply carry miscompare where bit 60 is byte 0 and bit 61 is byte 1
CP Processor Fault Status Register 9 (MCH Address 89 ₁₆)	
0-6	Multiply carry miscompare, bytes 2 through 8 where bit 0 indicates byte 2 and so on
†Register bits are numbered 0 through 63 from left to right.	

Table 5-20. Model 855 CM Register Bit Assignments† (Sheet 1 of 2)

Channel Bit	Description																									
CM Status Summary Register (MCH Address 00 ₁₆)																										
56-57 58 61 62 63	Oscillator code (00 = normal, 01 = -2%, 10 = +2%) Clock set to tuning mode Valid uncorrectable error Corrected error Long warning (low/high temperature, condensing unit fault, blower 1/blower 2 fault)																									
CM Element Identifier Register (MCH Address 10 ₁₆)																										
32-39 40-47 48-63	Element type (01 ₁₆ = CM) Element model (30 ₁₆ = model 855) Element serial number																									
CM Options Installed Register (MCH Address 12 ₁₆)																										
0-15 16 17-22	<p>Installed memory size in megabytes. Only one bit can be set at a time.</p> <table border="0"> <thead> <tr> <th><u>Bit set</u></th> <th><u>Memory size</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>524K words</td> </tr> <tr> <td>1</td> <td>1049K words</td> </tr> <tr> <td>2</td> <td>1573K words</td> </tr> <tr> <td>3</td> <td>2097K words</td> </tr> </tbody> </table> <p>12-15 Model-independent options</p> <p>Set when any of the configuration switches in central memory are in the up position.</p> <p>Set when a particular configuration switch is not in the center position.</p> <table border="0"> <thead> <tr> <th><u>Bit</u></th> <th><u>Switch</u></th> <th><u>Memory size</u></th> </tr> </thead> <tbody> <tr> <td>19</td> <td>SW3</td> <td>2097K words</td> </tr> <tr> <td>20</td> <td>SW4</td> <td>1573K words</td> </tr> <tr> <td>21</td> <td>SW5</td> <td>1049K words</td> </tr> <tr> <td>22</td> <td>SW6</td> <td>524K words</td> </tr> </tbody> </table>	<u>Bit set</u>	<u>Memory size</u>	0	524K words	1	1049K words	2	1573K words	3	2097K words	<u>Bit</u>	<u>Switch</u>	<u>Memory size</u>	19	SW3	2097K words	20	SW4	1573K words	21	SW5	1049K words	22	SW6	524K words
<u>Bit set</u>	<u>Memory size</u>																									
0	524K words																									
1	1049K words																									
2	1573K words																									
3	2097K words																									
<u>Bit</u>	<u>Switch</u>	<u>Memory size</u>																								
19	SW3	2097K words																								
20	SW4	1573K words																								
21	SW5	1049K words																								
22	SW6	524K words																								
CM Environment Control Register (MCH address 20 ₁₆)																										
0 1 3-4 5-6	<p>Disable parity checking Disable SECDED Write check/read check/read syndrome bits</p> <table border="0"> <thead> <tr> <th><u>Value</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>00</td> <td>Normal</td> </tr> <tr> <td>01</td> <td>Write byte 0 to correction byte</td> </tr> <tr> <td>10</td> <td>Read correction byte to byte 0</td> </tr> <tr> <td>11</td> <td>Read syndrome to byte 0</td> </tr> </tbody> </table> <p>Timing margins</p> <table border="0"> <thead> <tr> <th><u>Value</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>00</td> <td>Normal</td> </tr> <tr> <td>01</td> <td>Narrow</td> </tr> <tr> <td>10</td> <td>Wide</td> </tr> <tr> <td>11</td> <td>Wide</td> </tr> </tbody> </table>	<u>Value</u>	<u>Description</u>	00	Normal	01	Write byte 0 to correction byte	10	Read correction byte to byte 0	11	Read syndrome to byte 0	<u>Value</u>	<u>Description</u>	00	Normal	01	Narrow	10	Wide	11	Wide					
<u>Value</u>	<u>Description</u>																									
00	Normal																									
01	Write byte 0 to correction byte																									
10	Read correction byte to byte 0																									
11	Read syndrome to byte 0																									
<u>Value</u>	<u>Description</u>																									
00	Normal																									
01	Narrow																									
10	Wide																									
11	Wide																									
† Register bits are numbered 0 through 63 from left to right.																										

Table 5-20. Model 855 CM Register Bit Assignments† (Sheet 2 of 2)

Channel Bit	Description										
CM Environment Control Register (MCH address 20 ₁₆) (Contd)											
32-35	Port disable										
	<table border="0"> <thead> <tr> <th style="text-align: left;"><u>Bit</u></th> <th style="text-align: left;"><u>Significance</u></th> </tr> </thead> <tbody> <tr> <td>32</td> <td>Port 0; disable CP1 ports</td> </tr> <tr> <td>33</td> <td>Port 1; disable IOU1 port</td> </tr> <tr> <td>34</td> <td>Port 2; disable CP2 ports</td> </tr> <tr> <td>35</td> <td>Port 3; disable IOU2 port</td> </tr> </tbody> </table>	<u>Bit</u>	<u>Significance</u>	32	Port 0; disable CP1 ports	33	Port 1; disable IOU1 port	34	Port 2; disable CP2 ports	35	Port 3; disable IOU2 port
<u>Bit</u>	<u>Significance</u>										
32	Port 0; disable CP1 ports										
33	Port 1; disable IOU1 port										
34	Port 2; disable CP2 ports										
35	Port 3; disable IOU2 port										
38	Suppress corrected error reporting via ports										
39	Disable corrected error log										
CM Corrected Error Log Register (MCH Address A0 ₁₆)											
0	Valid bit										
1	Unlogged corrected error										
8-10	Port number										
11-37	Address plus parity associated with the error										
42-49	Syndrome bits										
CM Uncorrected Error Log 1 Register (MCH Address A4 ₁₆)											
0	Valid bit										
1	Unlogged uncorrectable error										
2	Illegal function										
3	Multiple bit memory error										
4	Memory bounds fault										
5	CMC error										
6	CSU error										
7	Common memory select; memory address bit 01										
8-10	Port number										
11-37	Address plus parity										
38-41	Parity error byte position										
42-49	Data-in parity error bits										
50-58	Mark plus parity; bit 58 is mark parity										
59-63	Function plus parity; bit 63 is function parity										
CM Uncorrected Error Log 2 Register (MCH Address A8 ₁₆)											
0	Valid bit										
1	Unlogged uncorrectable error										
2	Partial write parity error										
3	Data out path parity error										
8-10	Port number										
11-37	Address plus parity										
38-45	Data-out path parity error, bytes 0 through 7 where bit 38 indicates byte 0 and so on										
46-53	Partial write parity error, bytes 0 through 7 where bit 46 indicates byte 0 and so on										
† Register bits are numbered 0 through 63 from left to right.											

Table 5-21. IOU Register Bit Assignments † (Sheet 1 of 4)

Channel Bit	Description
IOU Status Summary Register (MCH Address 00 ₁₆)	
59 60 61 63	Summary status Processor halt Valid uncorrected error Long warning
NOTE: The byte is duplicated in each byte.	
IOU Element Identifier Register (MCH Address 10 ₁₆)	
32-39 40-47 48-63	Element type (02 ₁₆ = IOU) Element model Element serial number
IOU Options Installed Register (MCH Address 12 ₁₆)	
20 21 22 23 24 25 26 27 28 29 30 31 32 34 36 37 38 39 40 41 42 43 44 45 46 47 52 53 54 55 59 60 61 62 63	Barrel 3 Barrel 2 Barrel 1 Barrel 0 Channel 7 Channel 6 Channel 5 Channel 4 Channel 3 Channel 2 Channel 1 Channel 0 Channel 17 Channel 15 Channel 13 Channel 12 Channel 11 Channel 10 Channel 27 Channel 26 Channel 25 Channel 24 Channel 23 Channel 22 Channel 21 Channel 20 Channel 33 Channel 32 Channel 31 Channel 30 Radial interface 5/6 Radial interface 3/4 Radial interface 1/2 Two-port multiplexer Display controller
† Register bits are numbered 0 through 63 from left to right.	

Table 5-21. IOU Register Bit Assignments† (Sheet 2 of 4)

Channel Bit	Description
IOU Fault Status Mask Register (MCH Address 18 ₁₆)	
03 04 05 06 07 11 12 13 14 15 19 20 21 22 23 27 28 29 30 31 32 33 34 35 36 37 38 39 40	Barrel 0, PP 4 Barrel 0, PP 3 Barrel 0, PP 2 Barrel 0, PP 1 Barrel 0, PP 0 Barrel 1, PP 4 Barrel 1, PP 3 Barrel 1, PP 2 Barrel 1, PP 1 Barrel 1, PP 0 Barrel 2, PP 4 Barrel 2, PP 3 Barrel 2, PP 2 Barrel 2, PP 1 Barrel 2, PP 0 Barrel 3, PP 4 Barrel 3, PP 3 Barrel 3, PP 2 Barrel 3, PP 1 Barrel 3, PP 0 Channel 7 Channel 6 Channel 5 Channel 4 Channel 3 Channel 2 Channel 1 Channel 0 Channel 17
IOU Status Register (MCH Address 40 ₁₆)	
59-60 61-63	Barrel reconfiguration switches PP reconfiguration switches
IOU Fault Status 1 Register (MCH Address 80 ₁₆)	
03 04 05 06 07 11 12 13 14 15	Barrel 0, PP 4 error Barrel 0, PP 3 error Barrel 0, PP 2 error Barrel 0, PP 1 error Barrel 0, PP 0 error Barrel 1, PP 4 error Barrel 1, PP 3 error Barrel 1, PP 2 error Barrel 1, PP 1 error Barrel 1, PP 0 error
† Register bits are numbered 0 through 63 from left to right	

Table 5-21. IOU Register Bit Assignments † (Sheet 3 of 4)

Channel Bit	Description
IOU Fault Status 1 Register (MCH Address 80 ₁₆) (Contd)	
19 20 21 22 23 27 28 29 30 31 32 33 34 35 36 37 38 39 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63	Barrel 2, PP 4 error Barrel 2, PP 3 error Barrel 2, PP 2 error Barrel 2, PP 1 error Barrel 2, PP 0 error Barrel 3, PP 4 error Barrel 3, PP 3 error Barrel 3, PP 2 error Barrel 3, PP 1 error Barrel 3, PP 0 error PP error 7VDO PP error 7VEO Firmware error PP memory data out error 7XF0 PP error 7VGO 12/16 conversion error 7VJ0 PP memory address error PP memory data-in error OS bounds violation OS boundary address PE ADU barrel priority ROM PE CM read buffer error Uncorrected CM read error Uncorrected CM write error CM reject Input CM tag error CM response code error CM data out error CM address out error CM data in error byte 0 CM data in error byte 1 CM data in error byte 2 CM data in error byte 3 CM data in error byte 4 CM data in error byte 5 CM data in error byte 6 CM data in error byte 7
IOU Fault Status 2 Register (MCH Address 81 ₁₆)	
32 33 34 35 36 37 38 39 40 42 44 45 46 47 48 49 50 51 52	Channel 7 error Channel 6 error Channel 5 error Channel 4 error Channel 3 error Channel 2 error Channel 1 error Channel 0 error Channel 17 error Channel 15 error Channel 13 error Channel 12 error Channel 11 error Channel 10 error Channel 27 error Channel 26 error Channel 25 error Channel 24 error Channel 23 error
†Register bits are numbered 0 through 63 from left to right.	

Table 5-21. IOU Register Bit Assignments † (Sheet 4 of 4)

Channel Bit	Description
IOU Fault Status 2 Register (MCH Address 18 ₁₆) (Contd)	
53 54 55 57 58 59 60 61 62 63	Channel 22 error Channel 21 error Channel 20 error Radial interface 5/6 error Radial interface 3/4 error Radial interface 1/2 error Channel 33 error Channel 32 error Channel 31 error Channel 30 error
†Register bits are numbered 0 through 63 from left to right.	

GLOSSARY

A

ADU	Assembly-disassembly unit	I/O	Input/output
AOR	Address out of range	IOU	Input/output unit
CEL	Corrected error log	MA	Monitor address
CIF	CMU interrupted flag	MCH	Maintenance channel
CM	Central memory	MF	Monitor flag
CMU	Compare/move unit	MOS	Metal oxide semiconductor
CP	Central processor	MUX	Multiplexer, selector
CRT	Cathode-ray tube	OS	Operating system
CTI	Common Test and Initialization	PE	Parity error
DSC	Display station	PP	Peripheral processor
DTR	Data terminal ready	PPM	Peripheral processor memory
ECC	Error correction code	RAC	Reference address, central memory
ECL	Emitter-coupled logic	RAE	Reference address, extended memory
EDS	Extended deadstart	RAM	Random access (read-write) memory
EIA	Electronic Industries Association	RNI	Read next instruction
EM, EMS	Exit mode selection	ROM	Read-only memory
EC	Exit condition code field at (RAC)	RTS	Request to send
FIFO	First in, first out	SECDED	Single-error correction double-error detection
FLC	Field length, central memory	UART	Universal Asynchronous Receiver and Transmitter
FLE	Field length, extended memory	UEM	Unified extended memory
HIVS	Hardware Initialization and Verification Software		
ILH	Instruction lookahead hardware		

INDEX

- A barrel failure 3-3
- A register, PP 2-11
 - Set instructions 2-2
- A registers, CP 1-4; 2-4
 - Set instructions 4-17,18
- Absolute CM address formation 2-6,11; 5-12
- Absolute UEM address formation 2-6
- Access, CM 2-7,13
- Activate channel 4-35; 5-16
- Active channel 5-16
- Add instructions 4-29,30,31,32
- Address designator 4-2
- Address format, CM 2-7
- Address out of range error 2-6; 5-8,10
- Address registers, see A registers
- Addressing
 - Absolute address formation 2-6,11; 5-12
 - By PPs 5-14
 - CM 5-14
 - CM by PPs 5-14
 - PPM by PPs 5-14
 - Relative address formation 5-12
 - Section in CP 1-4; 2-7
- Addressing mode
 - Expanded 1-4; 2-6; 5-2,12
 - Standard 1-4; 2-6; 5-2,12
- Alphanumeric characters 5-21
- Assembly/disassembly 2-13
- Auto mode bit 3-2

- B registers 1-4; 2-4
 - Set instructions 2-3; 4-18
- Bank cycle times 2-7
- Bank select field 2-7
- Banks, CM 1-4; 2-7
- Barrel and slot 2-11,12
- Barrels 1-5; 2-11
 - Logical barrel 0 selection 3-2
 - Reconfiguration 3-2
- Bit assignments, maintenance registers 5-31
- Bit numbering 5; 4-1
- Block copy flag 2-6
- Block copy from CM instruction 4-3; 5-12
- Block copy from UEM instruction 4-3; 5-12
- Block copy instruction sequence 2-3
- Block copy instructions 2-6
- Boolean instruction sequence 2-1
- Bounds register 1-5; 2-7,8
- Branch destination address 4-4
- Branch instructions 2-4
 - Definite 4-5
 - Equal to zero 4-4
 - In range 4-5
 - Indefinite 4-6
 - Negative 4-5
 - Not equal to zero 4-5
 - Out of range 4-5
 - Positive 4-5
 - Register greater than or equal 4-6
 - Register less than 4-6
 - Registers equal 4-6
 - Registers not equal 4-6
- Branch target address 2-1
- Buffers, queuing 1-4

- Cache corrected error log register 5-32
- Cache invalidation bus interface 1-5
- Cache memory 1-4; 2-7
- Cathode ray tube, see CRT
- CCEL register, see cache corrected error log register
- CEL register, see corrected error log register
- Central exchange jump instruction 4-4
- Central memory control 2-7
- Central memory control (CMC) 1-4
- Central memory, see CM
- Central processor, see CP
- Central read instruction 4-33
- Central write instruction 4-33
- Channel active flag 5-16
- Channel control flag 5-16
- Channel error flag 5-16
- Channel flag instruction 5-16
- Channel, I/O, see I/O channels
- Channel, maintenance, see maintenance channel
- Channel marker flag 5-16
- Channel transfer timing 5-16,17
- Character address designator 4-2
- Character codes 5-21
- Character manipulation 5-6
- Character mode 5-21,22
- Character size 5-21
- Characteristics,
 - Functional 1-2
 - Physical 1-1
- Chassis configuration
 - Model 835 1-2
 - Model 855 1-2
- Chip address field 2-7
- CIF (CMU interrupted flag) 2-6
- CLEAR AUTO switch 3-2
- Clear channel error flag instruction 4-34
- Clear channel flag instruction 4-34
- CM
 - Access 2-13
 - Access time, model 835 2-7
 - Access time, model 855 2-7
 - Address format 2-7
 - Address formation 2-6,11
 - Addressing by PPs 5-14
 - Banks 1-4; 2-7
 - Block copy instructions 5-12
 - Bounds register 1-5; 2-7,8
 - Configuration switches 3-3
 - Controls 3-1
 - Cycle times, model 835 2-7
 - Cycle times, model 855 2-8
 - Description 1-4
 - Distributor 1-4
 - Extended, see UEM
 - Field length register, see FLC register
 - Functional characteristics, model 835 1-2
 - Functional characteristics, model 855 1-3
 - Functional description 2-7
 - Layout 2-8

Maintenance registers, model 835 5-29,34
 Maintenance registers, model 855 5-30,40
 Map 5-13
 MCH function word 5-25
 Port priority 2-8
 Ports 1-4; 2-7,8
 Programming 5-12
 Read word from 4-19; 5-12
 Reconfiguration 2-8; 3-1,4
 Reconfiguration switches 3-1
 Reference address register, see RAC register
 Word 4-1
 Write word to 4-19; 5-12
 CMC 1-4; 2-7
 CMU error exit 5-9,11,13
 CMU instructions, see Compare/move instructions
 CMU interrupted flag 2-6
 Codes, display station 5-22
 Collate table for compare/move instruction 4-16
 Compare collated instruction 4-16; 5-6
 Compare/move arithmetic 5-6
 Compare/move instruction sequence 2-3
 Compare/move instructions 2-3; 4-15
 Compare collated 5-6
 Compare uncollated 5-6
 Designators within instructions 4-2
 Move direct 5-6
 Move indirect 5-6
 Support registers 2-3
 Compare uncollated instruction 4-17; 5-6
 Complement, instructions using 4-7,8
 Conditional software errors 5-12
 Configuration, mainframe 1-1
 Configuration switches, CM 3-1,3
 Constant address instructions 5-14
 Control checks 3-4
 Control failure 3-3
 Control flags 2-6; 5-16
 Control section 1-4; 2-1
 Control store parity fault status register 5-32
 Control words, maintenance channel 5-27
 Controls 3-1,2
 Conversion
 Fixed-point to floating-point 4-11
 Floating-point to fixed-point 4-10
 Coordinates, display station 5-21,22
 Corrected error log register
 Model 835 5-34
 Model 855 5-41
 Counter, running 4-4
 CP
 Addressing section 1-4; 2-7
 Cache memory 1-4; 2-7
 Control section 1-4; 2-1
 Description 1-4
 Execution section 1-4; 2-7
 Functional characteristics, model 835 1-2
 Functional characteristics, model 855 1-3
 Functional description 2-1
 Instruction descriptions 4-1
 Instruction designators 4-2
 Maintenance registers, model 835 5-28,31
 Maintenance registers, model 855 5-29,36
 MCH function word 5-25
 Operating modes 4-2
 Programming 5-1
 Registers 1-4; 2-4
 CRT 1-5; 5-21
 CYBER 170 exchange jump 2-3,11; 4-2,4; 5-1
 CYBER 170 exchange package 2-3,4; 4-4; 5-1
 Contents 5-1
 CYBER 170 exchange sequence 2-3
 CYBER 170 job mode 4-2; 5-1
 Error exits 5-10
 CYBER 170 monitor flag 2-7; 4-2; 5-1
 CYBER 170 monitor mode 4-2; 5-1
 Error exits 5-8
 Cycle times, CM
 Model 835 2-7
 Model 855 2-7
 Data carrier detector (DCD) 5-24
 Data character 5-24
 Data display 5-21
 Data formats 4-1
 PP 4-27
 Data input sequence 5-18,19
 Data output sequence 5-18,20
 Data parity error 5-16
 Data set ready (DSR) 5-23
 DATA switch 3-2
 Data terminal ready (DTR) 5-24,25
 DCD, see Data carrier detector
 Deactivate channel 4-36
 DEAD START ERROR indicator 3-3
 DEAD START switch 2-11; 3-3
 Deadstart
 Controls 3-1,2
 Indicators 3-1,2
 Matrix 3-2
 Panel 2-11; 3-1
 Program 3-2,5
 Sequence 2-11; 3-4
 Switches 3-1,2
 DEC register, see Dependent environment control register
 Dependent environment control register
 Model 835 5-31
 Model 855 5-36
 Description, system 1-1
 Direct address instructions 5-14
 Direct read/write instruction sequence 2-3
 Direct read/write instructions,
 CM 2-3; 5-12
 UEM 2-3; 5-12
 Direct 6-bit address 5-14
 Direct 6-bit operand 5-14
 Direct 12-bit address 5-14
 Direct 18-bit operand 5-14
 Display character codes 5-21
 Display controller interface 1-5
 Display station 1-5
 Character codes 5-21
 Character sizes 5-21
 Codes 5-22
 Controller 2-13
 Coordinates 5-21,22
 Keyboard 5-21
 Mode 5-21,22
 Output function code 5-22
 Presentation 5-22
 Programming 5-21
 Timing considerations 5-22
 Distributor, CM 1-4
 Dot mode 5-21
 Double-bit errors 2-7; 5-9,11
 Double-precision results 5-4
 DSC, see Display station controller
 DSR, see Data set ready
 DTR, see Data terminal ready
 EC register, see Environment control register
 ECC bits, see Error correction code bits
 ECL, see Emitter-coupled logic
 EID register, see Element identifier register

Element identifier register,
 CM, model 835 5-34
 CM, model 855 5-40
 CP, model 835 5-31
 CP, model 855 5-36
 IOU 5-42
 EM register 2-6
 EMC, see Exit mode conditions
 EMERGENCY OFF switch 3-4
 Emitter-coupled logic (ECL) 1-2
 Empty register 5-16
 EMS bits, see Exit mode selection bits
 Environment control register
 CM, model 855 5-40
 IOU 3-2
 Error conditions 2-6
 Error correction code bits 2-8; 5-7
 Error exit 4-2
 Error exit condition codes 2-6; 5-7
 Error exit to MA instruction 4-2
 Error exits 5-7
 In CYBER 170 job mode 5-10
 In CYBER 170 monitor mode 5-8
 Error flag 5-16
 Error flag instruction 5-16
 Error handling 5-7
 Error processing 2-6; 5-7
 Error response 5-7
 Exchange jump instruction 4-30
 Exchange jump, see CYBER 170 exchange jump
 Exchange package, see CYBER 170 exchange package
 Exchange sequence, see CYBER 170 exchange sequence
 Execution interval 2-4; 5-2
 Execution section 1-4; 2-7
 Execution times
 Model 835 CP 4-20
 Model 855 CP 4-24
 PP 4-36
 Executive state 2-8; 4-2; 5-2
 Executive state software, see Executive state
 Exit condition codes 5-7
 Exit condition register 5-7
 Exit mode conditions 5-7
 Exit mode register, see EM register
 Exit mode selection bits 2-6; 5-7
 Expanded addressing mode 1-4; 2-6; 5-2,12
 Expanded addressing select flag 2-6
 Extended purge control, see instruction lookahead
 purge control
 External interface, IOU 1-5; 2-13

 Fast deadstart 3-3
 Fault status mask register 5-43
 Fault status registers 1 and 2 5-43,44
 Field length for CM register, see FLC register
 Field length for UEM register, see FLE register
 Firmware or control failure 3-3
 Fixed-point arithmetic 5-4
 Normalized numbers 5-6
 Overflow 5-4
 Fixed-point to floating-point conversion 4-11
 Flag register 2-6
 FLC register 2-6
 FLE register 2-6
 Floating-add instruction sequence 2-2
 Floating divide instruction 4-14
 Floating-divide instruction sequence 2-2
 Floating double-precision difference
 instruction 4-12
 Floating double-precision product instruction 4-14
 Floating double-precision sum instruction 4-12
 Floating-multiply instruction sequence 2-2

 Floating-point arithmetic 5-2
 Double-precision 5-4
 Format 5-2
 Indefinite 5-3
 Nonstandard operands 5-4
 Normalized numbers 5-4
 Overflow 5-3
 Packing 5-2
 Rounding 5-4
 Underflow 5-3
 Floating-point difference instruction 4-11
 Floating-point sum instruction 4-11
 Floating-point to fixed-point conversion 4-10
 Floating product instruction 4-13
 Form mask instruction 4-14
 Framing error 5-24
 Free running counter 4-4
 FREQ MARGIN switch 3-2
 Frequency margin 3-2
 FS register, see fault status register
 Full register flag 5-16
 Function instruction 5-16
 Function on channel instruction 4-36
 Function words, maintenance channel 5-27
 Functional characteristics
 Model 835 1-2
 Model 855 1-3
 Functional description
 CM 2-7
 CP 2-1
 IOU 2-11

 Glossary A-1
 Graphic mode 5-21

 Halt CP 5-27
 Hardware errors 5-7,9,11

 I/O channel failure 3-3
 I/O channels 2-11,13
 Activate 5-16
 Active/inactive 5-16
 Communications 5-15
 External interface 1-5; 2-13
 Initial state 3-5
 Internal interface 1-5; 2-13
 Operation 5-16
 Parity 5-15
 Reservation 5-15
 I/O transfers 5-16,18
 ILH, see Instruction lookahead
 Illegal instructions 4-2,4; 5-7,8
 Inactive channel 5-16
 Increment instruction sequence 2-2
 Indefinite error 5-9,11
 Indefinite, floating-point 5-3
 Indefinite operand error 2-6
 Index registers, see B registers
 Indexed direct address instructions 5-14
 Indexed 12-bit address 5-14
 Indicators 3-1,2
 Indirect address instructions 5-14
 Indirect 6-bit address 5-14
 Infinite operand error 2-6; 5-9,11
 Initialization
 IOU 2-11; 3-4
 Input data parity error 5-16
 Input from channel instruction 4-35
 Input/output channel, see I/O channel
 Input/output unit, see IOU
 Input sequence 5-18,19

Instruction control sequences 2-1
 Block copy sequence 2-3
 Boolean sequence 2-1
 Compare/move sequence 2-3
 CYBER 170 exchange sequence 2-3
 Direct read/write sequence 2-3
 Floating-add sequence 2-2
 Floating-divide sequence 2-2
 Floating-multiply sequence 2-2
 Increment sequence 2-2
 Normal jump sequence 2-4
 Return jump sequence 2-4
 Shift sequence 2-2
 Instruction descriptions 4-1
 CP 4-1
 PP 4-27
 Instruction designators
 CP 4-2
 PP 4-27
 Instruction execution times
 Model 835 CP 4-20
 Model 855 CP 4-24
 PP 4-36
 Instruction lookahead 2-1
 Purge control 5-6
 Instruction lookahead purge control 2-6
 Instruction lookahead purge flag 2-6
 Instruction prefetch, see Instruction lookahead
 Instruction purge flag 2-6
 Instruction word designators
 CP 4-2
 PP 4-27
 Instruction word format
 CP 4-1
 Instructions, CP (see also inside front cover)
 Block copy from CM 4-3; 5-12
 Block copy from UEM 4-3; 5-12
 Branch 4-4,5,6
 Central exchange jump 4-4
 Compare collated 4-16; 5-6
 Compare/move 4-15
 Compare uncollated 4-17; 5-6
 Complement 4-7,8
 Direct read/write of CM 5-12
 Direct read/write of UEM 5-12
 Error exit to MA 4-2
 Floating 4-11,12,13,14
 Floating divide 4-14
 Floating double-precision difference 4-12
 Floating double-precision product 4-14
 Floating double-precision sum 4-12
 Floating-point difference 4-11
 Floating-point sum 4-11
 Floating product 4-13
 Form mask 4-14
 Illegal 4-2,4; 5-7,8
 Integer difference 4-13
 Integer sum 4-13
 Interrupt to executive mode 4-2
 Jump 4-2,4
 Left shift 4-8
 Logical difference 4-7
 Logical product 4-7
 Logical sum 4-7
 Monitor exchange jump 4-4
 Move direct 4-16; 5-6
 Move indirect 4-16; 5-6
 Normalize 4-9,10
 Pack 4-11
 Pass 4-15
 Population count 4-17
 Read free running counter 4-4
 Read word from CM 4-19; 5-12
 Read word from UEM 4-4
 Return jump 4-2
 Right shift 4-8,9
 Round floating difference 4-12
 Round floating divide 4-15
 Round floating product 4-13
 Round floating sum 4-12
 Round normalize 4-10
 Set Ai 4-17,18
 Set Bi 4-18,19
 Set Xi 4-19,20
 Transfer word register to register 4-6
 Transmit complement 4-7
 Transmit word 4-6
 Unconditional jump 4-2,4
 Unpack 4-10
 Write one word to UEM 4-4
 Write word to CM 4-19; 5-12
 Instructions, PP (see also inside front cover) 4-27
 Activate channel 4-35
 Add 4-29,30,31,32
 Central read 4-33
 Central write 4-33
 Channel flag 5-16
 Clear channel error flag 4-34
 Clear channel flag 4-34
 CM read 5-14
 CM write 5-15
 Deactivate channel 4-36
 Error flag 5-16
 Exchange jump 4-30
 Function on channel 4-36
 Input from channel 4-35
 Jump if channel active 4-34
 Jump if channel empty 4-34
 Jump if channel error flag clear 4-34
 Jump if channel error flag set 4-34
 Jump if channel full 4-34
 Jump if channel inactive 4-34
 Load 4-29,30,31,32
 Load complement 4-29
 Load/store R register 4-30; 5-14
 Logical difference 4-29,30,31,32
 Logical product 4-29
 Long jump 4-28
 Minus jump 4-28
 Monitor exchange jump 4-30
 Monitor exchange jump to MA 4-30
 Nonzero jump 4-28
 Output on channel 4-35
 Pass 4-28; 30
 Plus jump 4-28
 Replace add 4-31; 32
 Replace add one 4-31; 32
 Replace subtract one 4-31; 32
 Return jump 4-28
 Selective clear 4-29
 Shift 4-29
 Store 4-31; 32
 Store R register 4-30; 5-14
 Subtract 4-29,30; 31
 Test and set channel flag 4-34
 Unconditional jump 4-28
 Zero jump 4-28
 Integer arithmetic 5-6
 Integer difference instruction 4-13
 Integer sum instruction 4-13
 Inter-PP communications 5-15
 Internal addresses, maintenance registers 5-28,29,30
 Internal interface, IOU 1-5; 2-13
 Interrupt to executive mode instruction 4-2
 IOU
 Barrel and slot 2-11,12
 Deadstart 2-11

Description 1-5
 Environmental control register 3-2
 External interface 1-5
 Functional characteristics 1-3
 Functional description 2-11
 I/O channels 1-5
 Initialization 2-11; 3-4,5
 Internal interface 1-5
 Maintenance registers 5-30,42
 MCH function word 5-25,27
 Peripheral processors, see PPs
 PPs 1-5; 2-11
 Reconfiguration 3-6

Job mode, see CYBER 170 job mode
 Jump if channel active instruction 4-34
 Jump if channel empty instruction 4-34
 Jump if channel error flag clear instruction 4-34
 Jump if channel error flag set instruction 4-34
 Jump if channel full instruction 4-34
 Jump if channel inactive 4-34
 Jump instructions 4-2,4

K register, PP 2-13
 Keyboard character codes 5-21

L. D. S. ERROR-A indicator 3-3
 L. D. S. ERROR-B indicator 3-3
 LAMP TEST switch 3-2
 Large scale integration (LSI) logic 1-2
 Least significant bit 4-1
 Left circular shift 4-8,9
 Left shift instruction 4-8
 Load complement instruction 4-29
 Load instruction 4-29,30,31,32
 Load R register 4-30; 5-14
 Lockout time, maximum request 2-8
 Logical difference instruction 4-7,8,29,30,31,32
 Logical product instruction 4-7,29
 Logical sum instruction 4-7
 Long-add instructions 2-3
 Long jump instruction 4-28
 Long/short deadstart sequence 3-3,5
 LONG/SHORT DEAD START SEQUENCE switch 3-3
 Lookahead
 Purge control 5-6
 Lookahead, instruction 2-1
 Lookahead purge control 2-6
 Lookahead purge flag 2-6
 LSI logic 1-2

MA register 2-7
 MAC, see Maintenance access control
 Mainframe configuration 1-1
 Maintenance access control 2-1,13
 Maintenance channel 2-13; 5-25
 Master clear 5-28
 Programming 5-25
 Register bit assignments 5-31
 Register internal addresses 5-28,29,30
 Maintenance channel interface 1-5; 2-7,13; 5-25
 Control words 5-27
 Function words 5-25,26,27
 Maintenance registers, see also individual registers
 by name
 Bit assignments 5-31
 CM, model 835 5-34
 CM, model 855 5-40
 CP, model 835 5-31
 CP, model 855 5-36
 Initialization 5-25
 Internal address 5-28,29,30
 IOU 5-42
 Read 5-27
 Write 5-27
 Map corrected error log register 5-33
 Marker flag, channel 5-16
 Mask designator 4-2
 Masking word 4-14
 Master clear 3-5
 Maintenance channel 5-28
 Two-port multiplexer 5-24
 MCEL register, see Map corrected error log register
 MCH, see Maintenance channel
 CM reconfiguration 3-1,4
 CM reconfiguration switches 3-1,5
 PP reconfiguration 3-6
 PP reconfiguration switches 3-2,6
 Programming 5-12
 Reconfiguration switches 2-8
 Memory see also CM
 CM reconfiguration 2-8; 3-1,4
 CM reconfiguration switches 3-1,5
 Map 5-13
 PP 2-13
 PP reconfiguration 2-13; 3-6
 PP reconfiguration switches 2-8; 3-2,6
 Programming 5-12
 Metal oxide semiconductor memory, see MOS
 MF, see CYBER 170 monitor flag
 Microcode 5-2
 Minus jump instruction 4-28
 Mode selection bits 5-7
 Modes of operation 4-2
 Monitor address register, see MA register
 Monitor condition register
 Model 835 5-31
 Model 855 5-36
 Monitor exchange jump instruction 4-4,30
 Monitor flag, see CYBER 170 monitor flag
 Monitor mode, see CYBER 170 monitor mode
 MOS 1-4
 Most insignificant bit 4-1
 Move direct instruction 4-16; 5-6
 Move indirect instruction 4-16; 5-6

No-address instructions 5-14
 Nonstandard operands 5-4
 Nonzero jump instruction 4-28
 Normal jump instruction sequence 2-4
 Normalize instruction 4-9,10
 Normalize operations 2-2; 4-9; 5-4
 Normalized numbers
 Fixed-point 5-6
 Floating-point 5-4

Offset designator for compare/move 4-2,15
 OI register, see Options installed register
 One's complement addition/subtraction 2-2
 Opcode designator 4-2
 Operand designator 4-2
 Operand registers, see X registers
 Operating instructions/procedures 3-1,4
 Operating modes, CP 4-2
 Operating registers 1-4; 2-4
 Operation code 4-2
 Options installed register
 CM, model 835 5-34
 CM, model 855 5-40
 CP, model 835 5-36
 IOU 5-42

Output data parity error 5-16
 Output on channel instruction 4-35
 Output sequence 5-18,20
 Overflow
 Fixed-point 5-4
 Floating-point 5-3
 Overrun 5-24

P or Q barrel failure 3-3
 P register, CP 2-6
 P register, PP 2-12
 Pack instruction 4-11
 Pack/unpack instructions 2-2; 4-10
 Packing numbers 5-2
 Panel, deadstart 3-1
 Parcels 2-1; 4-1
 Parity, channel 5-15
 Parity errors 5-9,11,16,24
 Partial overflow 5-3
 Partial underflow 5-3
 Pass instruction 4-15,28,30
 Peripheral processors, see PPs
 PFS register, see Processor fault status register
 Phased-bank operation 2-7
 Physical characteristics 1-1
 Plus jump instruction 4-28
 Population-count instructions 2-2; 4-17
 Port bounds register, see CM bounds register
 Port priority 2-8
 Ports, CM 1-4; 2-7,8
 Power-off procedures 3-4
 Power-on procedures 3-4
 PP NO switch 3-2
 PP SELECT switch 3-2
 PPM 2-13
 PPs 1-5
 Access to CM 2-13
 Addressing 5-14
 Barrels 1-5; 2-11
 Communications 2-11; 5-15
 CM read instructions 5-14
 CM write instructions 5-15
 Data format 4-27
 Description 2-11
 Initialization 3-5
 Instruction designators 4-27
 Instruction execution times 4-36
 Instruction word format 4-27
 Instructions 4-27
 Memory failure 3-3
 Memory (PPM) 2-13
 Numbering 2-13
 PPO selection
 Program timing considerations 5-16
 Programming 5-14
 Reconfiguration 2-13; 3-2,6
 Reconfiguration switches 3-2,6
 Registers 2-11; 3-5
 Relocation register format 4-27
 Selection switches 3-2
 To PP communications 5-15
 Write output buffer function 5-24

Prefetch of instructions, see Instruction lookahead
 Processor fault status register
 Model 835 5-31
 Model 855 5-36

Programming
 CM 5-12
 CP 5-1
 Display station 5-21
 Maintenance channel 5-25
 PP 5-14
 Real-time clock 5-23

 Two-port multiplexer 5-23
 Program address register, see P register
 Program mode 4-2
 Program timing considerations, PP 5-16
 Program word format 4-1
 Programming information 5-1
 Publication index 6
 Purge control
 Model 835 5-6
 Model 855 5-7

Q barrel failure 3-3
 Q register, PP 2-12
 Quadrant select field 2-7
 Queuing buffers 1-4

R register 2-11; 4-27,30
 Load instruction 5-14
 Store instruction 5-14
 RAC 5-7
 RAC register 2-6
 RAE register 2-6
 Ranks 2-11
 Read free running counter instruction 4-4
 Read maintenance registers 5-27
 Read-only memory 3-5
 Read status summary 5-28
 Read status summary function 5-23
 Read terminal data function 5-23
 Read word from CM instruction 4-19; 5-12
 Read word from UEM instruction 4-4
 Real-time clock interface 1-5; 2-13
 Programming 5-23

Reconfiguration,
 CM 2-8; 3-1,4
 PPs 2-13; 3-2,6
 RECONFIGURATION, BARREL switch 3-2
 RECONFIGURATION, PPM switch 3-2
 Reconfiguration switches
 CM 2-8; 3-1,3
 PP 3-2,6

Reference address for CM register, see RAC register
 Reference address for UEM register, see RAE register
 Register designator 4-2
 Register display, PP 3-2
 Register empty 5-16
 Register formats 4-1
 Register full flag 5-16
 REGISTER SELECT switch 3-2
 Registers, 2-4
 A, CP 1-4; 2-4; 4-17,18
 A, PP 2-11
 B 1-4; 2-4; 4-18,19
 Compare/move support 2-3
 CP 1-4; 2-4
 EM 2-6
 Exit mode, see EM register
 Field length, see FLC and FLE registers
 Flag 2-6
 FLC 2-6
 FLE 2-6
 General description 1-4
 K 2-13
 MA 2-7
 Maintenance, see Maintenance registers
 Monitor address register, see MA register
 Operating 1-4; 2-4
 P, CP 2-6
 P, PP 2-12
 PP 2-11
 Program address, see P register
 Q 2-12

R 2-11; 5-14
 RAC 2-6
 RAE 2-6
 Reference address, see RAC and RAE registers
 Support 1-4; 2-5
 X 1-4; 2-4; 4-19,20
 Relative address 5-12
 Relocation register, see R register
 Replace add instruction 4-31,32
 Replace add one instruction 4-31,32
 Replace subtract one instruction 4-31,32
 Request lockout time, maximum 2-8
 Request to send 5-24,25
 Reservation, channel 5-15
 Retry corrected error log register 5-31
 Return jump instruction 4-2,28
 Return jump instruction sequence 2-4
 Right shift instruction 4-8,9
 ROM 3-5
 Round floating difference instruction 4-12
 Round floating divide instruction 4-15
 Round floating product instruction 4-13
 Round floating sum instruction 4-12
 Round normalize instruction 4-10
 Rounding, floating-point 5-4
 Rounding operation 4-10; 5-4
 Row select field 2-7
 RTS, see Request to send
 Running counter 4-4

 S register, see Status register
 SECEDED 2-8
 SECEDED code bits
 Description 2-8
 Model 835 2-9
 Model 855 2-10
 SECEDED errors 5-9,11
 Selective clear instructions 4-29
 Selector 5-25
 Set A register instructions 2-2
 Set Ai instructions 4-17,18
 Set B register instructions 2-3
 Set Bi instructions 4-18,19
 Set X register instructions 2-3
 Set Xi instructions 4-19,20
 Shift designator 4-2
 Shift instruction 4-29
 Shift instruction sequence 2-2
 Short deadstart sequence 3-3,5
 Single-bit errors 2-8
 Single error correction/double error detection, see SECEDED
 Single-precision 5-4
 Slot, see Barrel and slot
 Slow deadstart 3-3
 Software errors, 5-12
 SS register, see Status summary register
 Stack 2-1
 Standard addressing mode 1-4; 2-6; 5-2,12
 Start CP 5-27
 Status register 5-43
 Status summary register
 CM, model 835 5-34
 CM, model 855 5-40
 CP, model 835 5-31
 CP, model 855 5-36
 IOU 5-42
 Read 5-28
 Read function 5-23
 Store instruction 4-31,32
 Store R register 4-30; 5-14
 Subtract instruction 4-29,30,31,32
 Support registers 1-4; 2-5

Compare/move 2-3
 Switches,
 CM reconfiguration 2-8; 3-1,3
 PP reconfiguration 3-2,6
 Syndrome codes 2-8
 Model 835 2-9
 Model 855 2-10
 System description 1-1
 System publication index 6

 Target address 2-1
 Terminal deselect function 5-23
 Terminal operation mode function 5-24
 Terminal select function 5-23
 Test and set channel flag instruction 4-34
 Timing considerations
 Channel 5-16
 Display station 5-22
 PP 5-16
 Transfer blocks 5-12
 Transfer single words 5-12
 Transfer timing 5-16,17
 Transfer word register to register 4-6
 Transfers, I/O 5-16
 Transmit complement instruction 4-7
 Transmit word instruction 4-6
 Two-port multiplexer interface 1-5; 2-13
 Input data 5-25
 Operation 5-23
 Output data 5-25
 Programming 5-23

 UART 5-24
 UEL register, see Uncorrected error log register
 UEM
 Block copy instructions 2-3; 5-12
 Description 1-4
 Direct read/write instructions 5-12
 Enable flag 2-6
 Field length register, see FLE register
 Instructions 2-3
 Read one word from 5-12
 Reference address register, see RAE register
 Write one word to 5-12
 Unconditional jump instruction 4-2,4,28
 Uncorrected error log registers 1 and 2
 Model 835 5-34,35
 Model 855 5-41
 Underflow, floating-point 5-3
 Unified extended memory, see UEM
 Universal Asynchronous Receiver and Transmitter 5-24
 Unpack instruction 4-10
 Unpack instructions 2-2

 Word
 Bit numbering 5
 Instruction 4-1
 Write maintenance registers 5-27
 Write output buffer 5-24
 Write word to CM instruction 4-19; 5-12
 Write word to UEM instruction 4-4

 X registers 1-4
 Set instructions 2-3; 4-19,20

 Zero jump instruction 4-28

 00 instruction
 Error exit 5-8,10
 12/16 conversion failure 3-3

COMMENT SHEET

CDC CYBER 170 Computer Systems
MANUAL TITLE: Models 835 and 855 Hardware Reference Manual

PUBLICATION NO.: 60469290

REVISION: A

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ STATE: _____ ZIP CODE: _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

Please Reply

No Reply Necessary

CUT ALONG LINE

AA3419 REV. 4/79 PRINTED IN U.S.A.

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD

FOLD

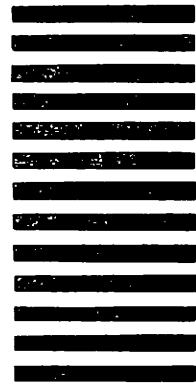


NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY
CONTROL DATA CORPORATION

Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112



CUT ALONG LINE

FOLD

FOLD