CDC® CYBER 170
COMPUTER SYSTEMS
MODELS 835 AND 855

HARDWARE REFERENCE MANUAL
## REVISION RECORD

<table>
<thead>
<tr>
<th>REVISION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Preliminary issue.</td>
</tr>
<tr>
<td>(05-15-81)</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Manual revised to add support of model 855.</td>
</tr>
<tr>
<td>(04-23-82)</td>
<td></td>
</tr>
</tbody>
</table>

Publication No. 60469290

REVISION LETTERS I, O, Q, S, X AND Z ARE NOT USED

© 1981, 1982 by Control Data Corporation
All rights reserved
Printed in the United States of America

Address comments concerning this manual to:
Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112
or use Comment Sheet in the back of this manual.
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.

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

\[
\begin{array}{c|c}
59 & 0 \\
\hline
\end{array}
\]

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

\[
\begin{array}{c|c}
0 & 63 \\
\hline
\end{array}
\]

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

<table>
<thead>
<tr>
<th>Control Data Publication</th>
<th>Publication Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOS Version 2 Operator/Analyst Handbook</td>
<td>60459310</td>
</tr>
<tr>
<td>NOS Version 2 Systems Programmer’s Instant</td>
<td>60459370</td>
</tr>
<tr>
<td>NOS Version 1 Operator’s Guide</td>
<td>60457700</td>
</tr>
<tr>
<td>NOS Version 1 Systems Programmer’s Instant</td>
<td>60457790</td>
</tr>
<tr>
<td>NOS/BE Version 1 Operator’s Guide</td>
<td>60457380</td>
</tr>
<tr>
<td>NOS/BE Version 1 System Programmer’s Reference Manual, Volume 1</td>
<td>60458480</td>
</tr>
</tbody>
</table>

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
   Introduction
   1-1
   Physical Characteristics
   1-1
   Functional Characteristics
   1-2
   Model 835 Characteristics
   1-2
   Central Processor
   1-2
   Central Memory
   1-2
   Input/Output Unit
   1-3
   Model 855 Characteristics
   1-3
   Central Processor
   1-3
   Central Memory
   1-3
   Input/Output Unit
   1-3
   Major System Component Descriptions
   1-4
   Central Processor
   1-4
   Control Section
   1-4
   Registers
   1-4
   Execution Section
   1-4
   Cache Memory
   1-4
   Addressing Section
   1-4
   Central Memory Control
   1-4
   (Model 855 Only)
   Central Memory
   1-4
   Input/Output Unit
   1-5
   Display Station
   1-5

2. FUNCTIONAL DESCRIPTIONS
   2-1
   Central Processor
   2-1
   Control Section
   2-1
   Model 835 Instruction Lookahead
   2-1
   Model 855 Instruction Lookahead
   2-1
   Maintenance Access Control
   2-1
   Instruction Control Sequences
   2-1
   Registers
   2-4
   Operating Registers
   2-4
   Support Registers
   2-4
   Execution Section
   2-5
   Cache Memory
   2-7
   Addressing Section
   2-7
   Central Memory Control
   2-7
   (Model 855 Only)
   Central Memory
   2-7
   Address Format
   2-7
   CM Access and Cycle Times
   2-7
   Model 835
   2-7
   Model 855
   2-7
   CM Ports and Priorities
   2-8
   SECDED
   2-8
   CM Layout
   2-8
   CM Bounds Register
   2-8
   Central Memory Reconfiguration
   2-8
   Input/Output Unit
   2-11
   Peripheral Processor
   2-11
   Deadstart
   2-11
   Barrel and Slot
   2-11
   PP Registers
   2-11
   PP Numbering
   2-13
   PP Memory
   2-13
   I/O Channels
   2-13
   Real-Time Clock
   2-13
   Two-Port Multiplexer
   2-13
   Maintenance Channel
   2-13
   Central Memory Access
   2-13

3. OPERATING INSTRUCTIONS
   3-1
   Controls and Indicators
   3-1
   Deadstart Panel Controls/Indicators
   3-1
   Central Memory Controls
   3-1
   Power-On and Power-Off Procedures
   3-4
   Operating Procedures
   3-4
   Control Checks
   3-4
   Deadstart Sequences
   3-4
   IOU Reconfiguration
   3-6

4. INSTRUCTION DESCRIPTIONS
   4-1
   CP Instructions
   4-1
   CP Instruction Formats
   4-1
   CP Operating Modes
   4-2
   CP Instruction Descriptions
   4-2
   Instruction Execution Timing
   4-20
   PP Instructions
   4-27
   PP Instruction Formats
   4-27
   PP Data Format
   4-27
   PP Relocation Register Format
   4-27
   PP Instruction Descriptions
   4-28
   Instruction Execution Timing
   4-36

5. PROGRAMMING INFORMATION
   5-1
   CP Programming
   5-1
   CYBER 170 Exchange Jump
   5-1
   Executive State
   5-2
   Floating-Point Arithmetic
   5-2
   Format
   5-2
   Packing
   5-2
   Overflow
   5-3
   Underflow
   5-3
   Indefinite
   5-3
   Nonstandard Operands
   5-4
   Normalized Number
   5-4
   Rounding
   5-4
   Double-Precision Results
   5-4
   Fixed-Point Arithmetic
   5-4
   Integer Arithmetic
   5-6
   Compare/Move Arithmetic
   5-6
   Instruction Lookahead Purge Control
   5-6
   Model 835 Purge Control
   5-6
   Model 855 Purge Control
   5-7
   Error Response
   5-7
   Illegal Instructions
   5-7
   Hardware Errors
   5-7
   Conditional Software Errors
   5-12
   Memory Programming
   5-12
   Addressing Modes
   5-12
   Direct Read/Write Instructions (014, 015, 660, 670)
   5-12
   Block Copy Instructions (011, 012)
   5-12
   PP Programming
   5-14
   Central Memory Addressing by PPs
   5-14
   PP Memory Addressing by PPs
   5-14
   Direct 6-Bit Operand
   5-14
   Direct 18-Bit Operand
   5-14
Central Memory Read/Write Instructions

PP Central Memory Read
Instructions (60, 61)

PP Central Memory Write
Instructions (62, 63)

Input/Output Channel Communications

Inter-PP Communications

PP Program Timing Considerations

Channel Operation

Channel Control Flags
Channel Active/Inactive Flag
Register Full/Empty Flag
Channel (Marker) Flag Instructions
(641, 651)

Error Flag Instructions (661, 671)

Channel Transfer Timing

Input/Output Transfers
Data Input Sequence
Data Output Sequence

Display Station Programming
Keyboard
Data Display
Character Mode
Dot Mode
Codes

Programming Example
Programming Timing Considerations

Real-Time Clock Programming
Two-Port Multiplexer Programming

Two-Port Multiplexer Operation
Terminal Deselect (6XXX)
Read Status Summary (00XX)
PP Read Terminal Data (01XX)
PP Write Output Buffer (02XX)
Set Operation Mode to the
Terminal (03XX)
Set/Clear Data Terminal Ready (04XX)
Set/Clear Request to Send (05XX)
Master Clear (07XX)

Programming Considerations
Output Data
Input Data
Request to Send and Data Terminal
Ready

Maintenance Channel Programming

Maintenance Channel
MCH Function Words
MCH Control Words

MCH Programming for Halt/Start
(Opcod 0/1)
MCH Programming for Read/Write
(Opcod 4/5)
MCH Programming for Master Clear/
Clear Errors (Opcod 6/7)
MCH Programming for Read IOU Status
Summary (Opcod C, IOU only)

APPENDIX

A. GLOSSARY

INDEX

FIGURES

1-1 Models 835 and 855 Computer Systems
1-2 Model 835 Chassis Configuration
(Top Cutaway View)
1-3 Model 855 Chassis Configuration
(Top Cutaway View)
1-4 Model 835 Computer System
1-5 Model 855 Computer System
2-1 CYBER 170 Exchange Package
2-2 Address Format
2-3 CM Layout
2-4 Barrel and Slot
2-5 Formation of Absolute CM Address
3-1 Deadstart Panel
3-2 CM Configuration Switches
4-1 CP Instruction Parcel Arrangement
4-2 PP Instruction Formats
4-3 PP Data Format

4-4 PP Relocation (R) Register Format
4-5 CYBER 170 Exchange Package
5-1 Floating-Point Format
5-2 Floating-Add Result Format
5-3 Multiply Result Format
5-4 Format of Exit Condition Register
at (RAC)
5-5 Memory Map
5-6 Channel Transfer Timing
5-7 Data Input Sequence Timing
5-8 Data Output Sequence Timing
5-9 Display Station Output Function
Code
5-10 Coordinate Data Word
5-11 Character Data Word
5-12 Receive and Display Program Flowchart

APPENDIX

A-1

INDEX

FIGURES

1-1 Models 835 and 855 Computer Systems
1-2 Model 835 Chassis Configuration
(Top Cutaway View)
1-3 Model 855 Chassis Configuration
(Top Cutaway View)
1-4 Model 835 Computer System
1-5 Model 855 Computer System
2-1 CYBER 170 Exchange Package
2-2 Address Format
2-3 CM Layout
2-4 Barrel and Slot
2-5 Formation of Absolute CM Address
3-1 Deadstart Panel
3-2 CM Configuration Switches
4-1 CP Instruction Parcel Arrangement
4-2 PP Instruction Formats
4-3 PP Data Format

4-4 PP Relocation (R) Register Format
4-5 CYBER 170 Exchange Package
5-1 Floating-Point Format
5-2 Floating-Add Result Format
5-3 Multiply Result Format
5-4 Format of Exit Condition Register
at (RAC)
5-5 Memory Map
5-6 Channel Transfer Timing
5-7 Data Input Sequence Timing
5-8 Data Output Sequence Timing
5-9 Display Station Output Function
Code
5-10 Coordinate Data Word
5-11 Character Data Word
5-12 Receive and Display Program Flowchart

APPENDIX

A-1

INDEX

FIGURES
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Port Priority</td>
</tr>
<tr>
<td>2-2</td>
<td>Model 835 SECDED Syndrome Codes/Corrected Bits</td>
</tr>
<tr>
<td>2-3</td>
<td>Model 855 SECDED Syndrome Codes/Corrected Bits</td>
</tr>
<tr>
<td>3-1</td>
<td>Deadstart Panel Controls/Indicators</td>
</tr>
<tr>
<td>3-2</td>
<td>Central Memory Reconfiguration</td>
</tr>
<tr>
<td>3-3</td>
<td>Barrel Numbering Table</td>
</tr>
<tr>
<td>4-1</td>
<td>Central Processor Instruction Designators</td>
</tr>
<tr>
<td>4-2</td>
<td>Collate Table</td>
</tr>
<tr>
<td>4-3</td>
<td>Model 835 CP Instruction Timing</td>
</tr>
<tr>
<td>4-4</td>
<td>Model 855 CP Instruction Timing</td>
</tr>
<tr>
<td>4-5</td>
<td>PP Nomenclature</td>
</tr>
<tr>
<td>4-6</td>
<td>PP Instruction Timing</td>
</tr>
<tr>
<td>5-1</td>
<td>Bits 50 and 59 Configurations</td>
</tr>
<tr>
<td>5-2</td>
<td>Xj Plus Xk (30, 32, 34 Instructions)</td>
</tr>
<tr>
<td>5-3</td>
<td>Xj Minus Xk (31, 33, 35 Instructions)</td>
</tr>
<tr>
<td>5-4</td>
<td>Xj Multiplied by Xk (40, 41, 42 Instructions)</td>
</tr>
<tr>
<td>5-5</td>
<td>Xj Divided by Xk (44, 45 Instructions)</td>
</tr>
<tr>
<td>5-6</td>
<td>Contents of Exit Condition Register at (RAC)</td>
</tr>
<tr>
<td>2-8</td>
<td>Error Exits in CYBER 170 Monitor Mode (MF=1)</td>
</tr>
<tr>
<td>2-9</td>
<td>Error Exits in CYBER 170 Job Mode (MF=0)</td>
</tr>
<tr>
<td>2-10</td>
<td>Keyboard Character Codes</td>
</tr>
<tr>
<td>3-2</td>
<td>Display Character Codes</td>
</tr>
<tr>
<td>3-4</td>
<td>MCH Function Word Bit Assignments</td>
</tr>
<tr>
<td>3-6</td>
<td>Model 835 CP Internal Address Assignments</td>
</tr>
<tr>
<td>4-2</td>
<td>Model 855 CP Internal Address Assignments</td>
</tr>
<tr>
<td>4-16</td>
<td>Model 855 CP Internal Address Assignments</td>
</tr>
<tr>
<td>5-13</td>
<td>Model 835 CM Internal Address Assignments</td>
</tr>
<tr>
<td>5-14</td>
<td>Model 855 CP Internal Address Assignments</td>
</tr>
<tr>
<td>5-15</td>
<td>Model 855 CM Internal Address Assignments</td>
</tr>
<tr>
<td>4-24</td>
<td>Model 835 CP Register Bit Assignments</td>
</tr>
<tr>
<td>5-5</td>
<td>Model 835 CP Register Bit Assignments</td>
</tr>
<tr>
<td>5-5</td>
<td>Model 855 CM Register Bit Assignments</td>
</tr>
<tr>
<td>5-5</td>
<td>Model 855 CM Register Bit Assignments</td>
</tr>
<tr>
<td>5-5</td>
<td>IOU Register Bit Assignments</td>
</tr>
<tr>
<td>5-7</td>
<td>Error Exits in CYBER 170 Monitor Mode (MF=1)</td>
</tr>
<tr>
<td>5-8</td>
<td>Error Exits in CYBER 170 Job Mode (MF=0)</td>
</tr>
<tr>
<td>5-9</td>
<td>Keyboard Character Codes</td>
</tr>
<tr>
<td>5-10</td>
<td>Display Character Codes</td>
</tr>
<tr>
<td>5-11</td>
<td>MCH Function Word Bit Assignments</td>
</tr>
<tr>
<td>5-12</td>
<td>Model 835 CP Internal Address Assignments</td>
</tr>
<tr>
<td>5-13</td>
<td>Model 855 CP Internal Address Assignments</td>
</tr>
<tr>
<td>5-14</td>
<td>Model 855 CP Internal Address Assignments</td>
</tr>
<tr>
<td>5-15</td>
<td>Model 855 CM Internal Address Assignments</td>
</tr>
<tr>
<td>5-16</td>
<td>IOU Internal Address Assignments</td>
</tr>
<tr>
<td>5-17</td>
<td>Model 835 CP Register Bit Assignments</td>
</tr>
<tr>
<td>5-18</td>
<td>Model 835 CM Register Bit Assignments</td>
</tr>
<tr>
<td>5-19</td>
<td>Model 855 CP Register Bit Assignments</td>
</tr>
<tr>
<td>5-20</td>
<td>Model 855 CM Register Bit Assignments</td>
</tr>
<tr>
<td>5-21</td>
<td>IOU Register Bit Assignments</td>
</tr>
<tr>
<td>5-28</td>
<td>Model 835 CM Internal Address Assignments</td>
</tr>
<tr>
<td>5-29</td>
<td>Model 855 CM Internal Address Assignments</td>
</tr>
<tr>
<td>5-30</td>
<td>Model 835 CM Internal Address Assignments</td>
</tr>
<tr>
<td>5-31</td>
<td>Model 835 CP Register Bit Assignments</td>
</tr>
<tr>
<td>5-34</td>
<td>Model 835 CM Register Bit Assignments</td>
</tr>
<tr>
<td>5-36</td>
<td>Model 835 CP Register Bit Assignments</td>
</tr>
<tr>
<td>5-40</td>
<td>Model 855 CM Register Bit Assignments</td>
</tr>
<tr>
<td>5-42</td>
<td>IOU Register Bit Assignments</td>
</tr>
</tbody>
</table>
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.
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.

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.

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.

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 prefetched 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 multiplexed 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 I/O (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 CF.

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 1-5. Model 855 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
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:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Logical product ((X_j)) and ((X_k)) (BX_i X_j \times X_k) to (X_i)</td>
</tr>
<tr>
<td>12</td>
<td>Logical sum of ((X_j)) and ((X_k)) (BX_i X_j + X_k) to (X_i)</td>
</tr>
<tr>
<td>13</td>
<td>Logical difference of ((X_j)) (BX_i X_j - X_k) and ((X_k)) to (X_i)</td>
</tr>
<tr>
<td>15</td>
<td>Logical product of ((X_j)) with complement of ((X_k)) (BX_i -X_k \times X_j) to (X_i)</td>
</tr>
<tr>
<td>16</td>
<td>Logical sum of ((X_j)) with complement of ((X_k)) (BX_i -X_k + X_j) to (X_i)</td>
</tr>
<tr>
<td>17</td>
<td>Logical difference of ((X_j)) (BX_i -X_k - X_j) with complement of ((X_k)) to (X_i)</td>
</tr>
</tbody>
</table>
The instructions requiring transmissive operations are:

10 Transmit \((X_j)\) to \(X_i\) \(BX_i X_j\)
14 Transmit complement of \((X_k)\) to \(X_i\) \(BX_i -X_k\)

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 \((X_i)\) by \(jk\) \(LX_i jk\)
21 Right shift \((X_i)\) by \(jk\) \(AX_i jk\)
22 Left shift \((X_k)\) nominally \((Bj)\) places to \(X_i\) \(LX_i Bj X_k\)
23 Right shift \((X_k)\) nominally \((Bj)\) places to \(X_i\) \(AX_i Bj X_k\)
43 Form mask of \(jk\) bits to \(X_i\) \(MX_i jk\)

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 \(X_k\) register, delivers the coefficient to the \(X_i\) register, and delivers the exponent to the \(B_j\) register. The unpack and pack instructions are:

26 Unpack \((X_k)\) to \(X_i\) and \(B_j\) \(UX_i Bj X_k\)
27 Pack \((X_k)\) and \((B_j)\) to \(X_i\) \(PX_i Bj X_k\)

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 \((X_k)\) to \(X_i\) and \(B_j\) \(NX_i Bj X_k\)
25 Round normalize \((X_k)\) to \(X_i\) \(ZX_i Bj X_k\)

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 \((X_j)\) and \((X_k)\) to \(X_i\) \(FX_i X_j + X_k\)
31 Floating difference of \((X_j)\) and \((X_k)\) to \(X_i\) \(FX_i X_j - X_k\)
32 Floating double-precision sum of \((X_j)\) and \((X_k)\) to \(X_i\) \(DX_i X_j + X_k\)
33 Floating double-precision difference of \((X_j)\) and \((X_k)\) to \(X_i\) \(DX_i X_j - X_k\)
34 Round floating sum of \((X_j)\) and \((X_k)\) to \(X_i\) \(RX_i X_j + X_k\)
35 Round floating difference of \((X_j)\) and \((X_k)\) to \(X_i\) \(RX_i X_j - X_k\)

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 \((X_j)\) and \((X_k)\) to \(X_i\) \(FX_i X_j \times X_k\)
41 Round floating product of \((X_j)\) and \((X_k)\) to \(X_i\) \(RX_i X_j \times X_k\)
42 Floating double-precision product of \((X_j)\) and \((X_k)\) to \(X_i\) \(DX_i X_j \times X_k\)

The divide instructions are:

44 Floating divide \((X_j)\) by \((X_k)\) to \(X_i\) \(FX_i X_j/X_k\)
45 Round floating divide \((X_j)\) by \((X_k)\) to \(X_i\) \(RX_i X_j/X_k\)

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

47 Population count of \((X_k)\) to \(X_i\) \(CX_i X_k\)

Increment Sequence

The increment sequence controls the one's complement addition and subtraction of 16-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 \(A_i\) to \((A_j) + K\) \(SA_i Aj + K\)
51 Set \(A_i\) to \((B_j) + K\) \(SA_i Bj + K\)
52 Set \(A_i\) to \((X_j) + K\) \(SA_i X_j + K\)
53 Set \(A_i\) to \((X_j) + (B_k)\) \(SA_i X_j + B_k\)
54 Set \(A_i\) to \((A_j) + (B_k)\) \(SA_i Aj + B_k\)
55 Set \(A_i\) to \((A_j) - (B_k)\) \(SA_i Aj - B_k\)
56 Set \(A_i\) to \((B_j) + (B_k)\) \(SA_i Bj + B_k\)
57 Set \(A_i\) to \((B_j) - (B_k)\) \(SA_i Bj - B_k\)
60 Set \(B_i\) to \((A_j) + K\) \(SB_i Aj + K\)

The increment sequence also controls the 60-bit one's complement sum and difference values for long-add instructions 36 and 37.
61 Set Bi to (Bj) + K
62 Set Bi to (Xj) + K
63 Set Bi to (Xj) + (Bk)
64 Set Bi to (Aj) + (Bk)
65 Set Bi to (Aj) - (Bk)
66 Set Bi to (Bj) + (Bk)
67 Set Bi to (Bj) - (Bk)
70 Set Xi to (Aj) + K
71 Set Xi to (Bj) + K
72 Set Xi to (Xj) + K
73 Set Xi to (Xj) + (Bk)
74 Set Xi to (Aj) + (Bk)
75 Set Xi to (Aj) - (Bk)
76 Set Xi to (Bj) + (Bk)
77 Set Xi to (Bj) - (Bk)

The long-add instructions are:
36 Integer sum of (Xj) and (Xk) to Xi
37 Integer difference of (Xj) and (Xk) to Xi

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
465 Move direct
466 Compare collated
467 Compare uncollated

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 CI register that specifies the character position or offset of the first CM word of the source field.

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
012 Block copy Bj + K words from CM to UEM

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
015 Write one word from Xj to Xk
UEM at (Xk + RAE)

660 Read central memory at (Xk) to Xj

670 Write Xj into central memory at (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 Bj 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](image)
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:

<table>
<thead>
<tr>
<th>Mode Selection Bit</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>Address out of range</td>
</tr>
<tr>
<td>49</td>
<td>Infinite operand</td>
</tr>
<tr>
<td>50</td>
<td>Indefinite operand</td>
</tr>
<tr>
<td>57</td>
<td>Hardware error</td>
</tr>
<tr>
<td>58</td>
<td>Hardware error</td>
</tr>
<tr>
<td>59</td>
<td>Hardware error</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>Hardware error bit.</td>
</tr>
<tr>
<td>52</td>
<td>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.</td>
</tr>
<tr>
<td>53</td>
<td>CMU interrupted flag. If set, one of instructions 464 through 467 has been interrupted. The information necessary to resume operation has been saved.</td>
</tr>
<tr>
<td>54</td>
<td>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.</td>
</tr>
<tr>
<td>55</td>
<td>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.</td>
</tr>
<tr>
<td>56</td>
<td>UEM enable flag. If set, UEM is available. This flag must be set to allow 011, 012, 014, and 015 instructions to access UEM.</td>
</tr>
</tbody>
</table>

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.
- SECDED 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 SECDED 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 SECDED generators generate the SECDED code bits stored with each word. SECDED 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.

![Address Format](image)

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).

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

<table>
<thead>
<tr>
<th>Maximum Request Lockout Time in Bank Cycles</th>
<th>Port</th>
<th>Read or Write Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Port 0</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Port 1</td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>

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.

SECDDED

The SECDDED logic corrects single-bit errors during a CM read, permitting unimpeded computer operation. The SECDDED 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 SECDDED 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.

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 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.

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

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 SECDED Syndrome Codes/Corrected Bits

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

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 PPO, 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

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

Figure 2-5. Formation of Absolute CM Address
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:

<table>
<thead>
<tr>
<th>Barrel</th>
<th>PPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00 to 04</td>
</tr>
<tr>
<td>1</td>
<td>05 to 11 (octal)</td>
</tr>
<tr>
<td>2</td>
<td>20 to 24 (octal)</td>
</tr>
<tr>
<td>3</td>
<td>25 to 31 (octal)</td>
</tr>
</tbody>
</table>

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 PP0. Since Q is the channel selector, assigns PP0 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 PPU 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 10g. 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 14g. 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 15g.

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 17g, 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.
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
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>
<thead>
<tr>
<th>Panel Nomenclature</th>
<th>Description</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>2° through 215 by 1 through 20g</td>
<td>Toggle switch matrix (two-position switches)</td>
<td>Provides a 16-word deadstart program for PPO. Switches 20 through 211 set 12 bits for each of the program words, labeled 1 through 20 (octal). Switches 212 through 215 are set to zero. Up position sets bit. Down position clears bit.</td>
</tr>
<tr>
<td>PP NO</td>
<td>Octal display</td>
<td>Shows the PP selected by PP SELECT switches.</td>
</tr>
<tr>
<td>DATA</td>
<td>Octal display</td>
<td>Shows the content of the register selected by REGISTER SELECT switches.</td>
</tr>
<tr>
<td>PP SELECT</td>
<td>Toggle switches (two-position)</td>
<td>Selects the PP whose register is to be displayed.</td>
</tr>
<tr>
<td>REGISTER SELECT</td>
<td>Toggle switches (two-position)</td>
<td>Selects the register to be displayed (00 = P, 01 = Q, 10 = K, 11 = A).</td>
</tr>
<tr>
<td>A, K, Q, P</td>
<td>Indicators</td>
<td>One of these lights to indicate which register is selected by REGISTER SELECT switches.</td>
</tr>
<tr>
<td>LAMP TEST</td>
<td>Toggle switch (two-position)</td>
<td>Lights all indicators and display segments.</td>
</tr>
<tr>
<td>CLEAR AUTO</td>
<td>Toggle switch (two-position)</td>
<td>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.</td>
</tr>
<tr>
<td>FREQ MARGIN</td>
<td>Toggle switch (three-position)</td>
<td>Determines the frequency margin selected (FAST/NORMAL/SLOW). The setting of this switch is sensed only at deadstart time.</td>
</tr>
<tr>
<td>RECONFIGURATION, BARREL</td>
<td>Toggle switches (two-position)</td>
<td>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.)</td>
</tr>
<tr>
<td>RECONFIGURATION, PPM</td>
<td>Toggle switches (two-position)</td>
<td>Selects the physical PP memory which is logical PPMD. 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.</td>
</tr>
</tbody>
</table>
### Table 3-1. Deadstart Panel Controls/Indicators (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Panel Nomenclature</th>
<th>Description</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>LONG/SHORT DEAD START SEQUENCE</td>
<td>Toggle switch (two-position)</td>
<td>Selects the LONG/SHORT deadstart sequence. The setting of this switch is sensed only at deadstart.</td>
</tr>
<tr>
<td>DEAD START</td>
<td>Toggle switch (three-position, center is off)</td>
<td>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.)</td>
</tr>
<tr>
<td>L.D.S. ERROR-A</td>
<td>Indicator</td>
<td>Remains lit when long deadstart branch tests are not completed within 10.25 microseconds.</td>
</tr>
<tr>
<td>L.D.S. ERROR-B</td>
<td>Indicator</td>
<td>Remains lit when a long deadstart sequence does not go to completion.</td>
</tr>
<tr>
<td>DEAD START ERROR</td>
<td>Indicator</td>
<td>Lights in case of long deadstart ROM address/data parity error.</td>
</tr>
<tr>
<td>M, CH, A, PQ, I, C</td>
<td>Indicators</td>
<td>Lights in case of hardware failures as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M: PP memory failure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CH: I/O channel failure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A: A barrel failure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PQ: P or Q barrel failure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I: Firmware or control failure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C: 12/16 conversion failure</td>
</tr>
</tbody>
</table>

### Figure 3-2. CM Configuration Switches

- **CENTRAL MEMORY**
  - A
  - B
  - C
  - D
  - E
  - F
- **MODEL 835**
  - SW3
  - SW4
- **MODEL 865**
  - SW5
  - SW6
Table 3-2. Central Memory Reconfiguration

<table>
<thead>
<tr>
<th>Original CM</th>
<th>Reconfigured CM</th>
<th>Location of Failing CM</th>
<th>Reconfiguration Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Size</td>
<td>Words</td>
<td>Bit 23</td>
</tr>
<tr>
<td>524K</td>
<td>0-1 777 777</td>
<td>262K</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>262K</td>
<td>1</td>
</tr>
<tr>
<td>1049K</td>
<td>0-3 777 777</td>
<td>524K</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>524K</td>
<td>1</td>
</tr>
<tr>
<td>1573K</td>
<td>0-5 777 777</td>
<td>524K</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>524K</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1049K</td>
<td>1</td>
</tr>
<tr>
<td>2097K</td>
<td>0-7 777 777</td>
<td>1049K</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1049K</td>
<td>1</td>
</tr>
</tbody>
</table>

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:

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

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:

<table>
<thead>
<tr>
<th>Channel</th>
<th>Active/Inactive</th>
<th>Full/Empty</th>
<th>Channel Error Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Inactive</td>
<td>Empty</td>
<td>Clear</td>
</tr>
<tr>
<td>10</td>
<td>Active</td>
<td>Full</td>
<td>Set</td>
</tr>
<tr>
<td>14</td>
<td>Active</td>
<td>Full</td>
<td>Set</td>
</tr>
<tr>
<td>15</td>
<td>Active</td>
<td>Empty</td>
<td>Clear</td>
</tr>
<tr>
<td>17</td>
<td>Active</td>
<td>Empty</td>
<td>Clear</td>
</tr>
<tr>
<td>Other installed channels</td>
<td>Active</td>
<td>Empty</td>
<td>Clear</td>
</tr>
<tr>
<td>Noninstalled channels</td>
<td>Inactive</td>
<td>Empty</td>
<td>Clear</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Initialization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>007100₈</td>
<td>Instruction display on deadstart panel</td>
</tr>
<tr>
<td>P</td>
<td>00777₇₈</td>
<td>Causes block input to start from location 0</td>
</tr>
<tr>
<td>A</td>
<td>10,000₈</td>
<td>Count of 4096 words</td>
</tr>
<tr>
<td>Q</td>
<td>0, 1, 2...</td>
<td>I/O channel numbers (PPO: 0, PP1: 1, and so on)</td>
</tr>
</tbody>
</table>

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 777₇₈ in all PP memories and sets the P register of PPO to 6000₈. 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 6200₈ within 10.25 microseconds. Indicator B remains lit until the test program reaches location 777₆₈. 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 20₈, and clear PPO location 21₈. 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 21₈. 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

<table>
<thead>
<tr>
<th>Barrels Installed</th>
<th>Physical Barrel</th>
<th>Logical PPs in Physical Barrel with BARREL RECONFIGURATION Switch Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Barrels (20 PPs)</td>
<td>0</td>
<td>0-4 25-31 20-24 5-11</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5-11 0-4 25-31 20-24</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>20-24 5-11 0-4 25-31</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>25-31 20-24 5-11 0-4</td>
</tr>
<tr>
<td>3 Barrels (15 PPs)</td>
<td>0</td>
<td>0-4 20-24 5-11 (0-4)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5-11 0-4 20-24 (5-11)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>20-24 5-11 0-4 (20-24)</td>
</tr>
<tr>
<td>2 Barrels (10 PPs)</td>
<td>0</td>
<td>0-4 5-11 (0-4) (0-4)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5-11 0-4 (5-11) (5-11)</td>
</tr>
<tr>
<td>1 Barrel (5 PPs)</td>
<td>0</td>
<td>0-4 (0-4) (0-4) (0-4)</td>
</tr>
</tbody>
</table>
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 462xx. 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.

Table 4-1. Central Processor Instruction Designators

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

*Applicable to compare/move instructions only.

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.

**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 O0xxx and 010xK instruction formats, indicate unused bits. The unused bits are ignored by the CP.

**00xxx** Error Exit to MA when CYBER 17 PS MF Clear
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

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 \( K + 1 \).

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.

\[
\begin{array}{cccc}
K & 0400x & xxxx & 00000 \\
K + 1 & yyyy & yyyy & yyyy \\
\end{array}
\]

Subroutine exit

\[
\begin{array}{cccc}
K + 1 & yyyy & yyyy & yyyy \\
\end{array}
\]

Subroutine entrance

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.

This instruction copies a block of \( Bj \) plus \( K \) consecutive words from CM to UEM. The source CM address is either \( A0 \) 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.

\[
\begin{array}{cccc}
59 & 51 & 47 & 30 29 0
\end{array}
\]

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. 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 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 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.

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.

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

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 \((X_j) \neq 0\)

032JK Branch to K if \((X_j)\) is Positive

033JK Branch to K if \((X_j)\) is Negative

034JK Branch to K if \((X_j)\) is in Range

035JK Branch to K if \((X_j)\) is Out of Range

036JK Branch to K if \((X_j)\) is Definite

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 \(X_j\). The branch decision for this instruction is based on the value of the sign bit in \(X_j\).

Jump to K if: Bit 59 of \(X_j\) = 0 (positive)

Continue if: Bit 59 of \(X_j\) = 1 (negative)
The program sequence continues only on the following conditions. The branch to address K occurs for all other cases.

Continue if: \((X_j) = 1777 \ldots \) \((+\text{indefinite})\)
\((X_j) = 6000 \ldots \) \((-\text{indefinite})\)

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 \((X_j)\) is Indefinite

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: \((X_j) = 1777 \ldots \) \((+\text{indefinite})\)
\((X_j) = 6000 \ldots \) \((-\text{indefinite})\)

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 \((B_i) = (B_j)\)

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 branch to address K occurs for all other cases.

041JK Branch to K if \((B_i) \neq (B_j)\)

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 branch to address K occurs for all other cases.

051JK Branch to K if \((B_i) \neq (B_j)\)

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 branch to address K occurs for all other cases.

061JK Branch to K if \((B_i) \geq (B_j)\)

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 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.

071JK Branch to K if \((B_i) < (B_j)\)

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 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.

101JX Transmit \((X_j)\) to \(X_i\)

This instruction transfers a 60-bit word from \(X_j\) into \(X_i\).

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 \( (Xk) \) to Xi

\[ BXi = Xj \times Xk \]

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.

\[(Xj) = 0123 7777 0123 4567 1010\]
\[(Xk) = 0123 4567 7777 3210 1100\]
\[(Xi) = 0000 3210 7654 7777 0110\]

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

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

\[ BXi = Xj + Xk \]

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.

\[(Xj) = 0000 7777 0123 4567 1010\]
\[(Xk) = 0123 4567 7777 3210 1100\]
\[(Xi) = 0123 7777 7777 4567 1110\]

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

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

\[ BXi = Xj - Xk \]

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.

14ijk Transmit Complement of (Xk) to Xi

\[ BXi = -Xk \]

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 Complement of (Xk) to Xi

\[ BXi = Xj \times -Xk \]

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.

\[(Xj) = 7777 7000 0123 4567 1010\]
\[(Xk) = 0123 4567 7777 0000 1100\]
\[(Xi) = 7654 3000 0120 0067 0010\]

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

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

\[ BXi = Xj + -Xk \]

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.

\[(Xj) = 7777 7000 0123 4567 1010\]
\[(Xk) = 0123 4567 0000 7777 3210\]
\[(Xi) = 7654 3000 0120 0067 0010\]

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

\[(Xj) = 0000\ 7777\ 0123\ 4567\ 1010\]
\[(Xk) = 0123\ 4567\ 7777\ 0000\ 1100\]
\[(XI) = 7777\ 7654\ 7777\ 0123\ 7777\]

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

17ijk Logical Difference of \((Xj)\) with Complement of \((Xk)\) to XI

\[BXi = Xk - Xj\]

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.

\[(Xj) = 0123\ 7777\ 0123\ 4567\ 1010\]
\[(Xk) = 0123\ 4567\ 7777\ 0000\ 1100\]
\[(XI) = 7777\ 7654\ 7777\ 0123\ 7777\]

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.
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.

\[
\begin{align*}
(Xk) &= 23236600000000111 \\
(Bj) &= 000012 \\
(Xi) &= 754000000002464 \\
(Xk) &= 132760000033332422 \\
(Bj) &= 777771 \\
(Xi) &= 0013276000003324
\end{align*}
\]

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.

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.

\[
\begin{align*}
(Xk) &= 132760000033332422 \\
(Bj) &= 000006 \\
(Xi) &= 0013276000003324 \\
(Xk) &= 23236600000000111 \\
(Bj) &= 777765 \\
(Xi) &= 754000000002464
\end{align*}
\]

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.

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.

\[
\begin{align*}
(X_k) &= 2034 \ 0047 \ 6500 \ 0000 \ 2262 \\
(X_i) &= 2026 \ 4765 \ 0000 \ 0222 \ 6200 \\
(B_j) &= 00 \ 0006 \\

(X_k) &= 5743 \ 7730 \ 1277 \ 7777 \ 5515 \\
(X_i) &= 5751 \ 3012 \ 7777 \ 7755 \ 1577 \\
(B_j) &= 00 \ 0006
\end{align*}
\]

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.

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

25ijk Round Normalize (Xk) to Xi and Bj

\[
\begin{array}{cccc}
14 & 98 & 65 & 32 & 0 \\
25 & i & j & k
\end{array}
\]

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.

\[
\begin{align*}
(X_k) &= 2034 \ 0047 \ 6500 \ 0000 \ 2262 \\
(X_i) &= 2026 \ 4765 \ 0000 \ 0222 \ 6200 \\
(B_j) &= 00 \ 0006 \\

(X_k) &= 5743 \ 7730 \ 1277 \ 7777 \ 5515 \\
(X_i) &= 5751 \ 3012 \ 7777 \ 7755 \ 1537 \\
(B_j) &= 00 \ 0006
\end{align*}
\]

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

\[
\begin{array}{cccc}
14 & 98 & 65 & 32 & 0 \\
26 & i & j & k
\end{array}
\]

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.

\[
\begin{align*}
(X_k) &= 2034 \ 4500 \ 3333 \ 2000 \ 0077 \\
(X_i) &= 0000 \ 4500 \ 3333 \ 2000 \ 0077 \\
(B_j) &= 00 \ 0034 \\

(X_k) &= 1743 \ 4500 \ 3333 \ 2000 \ 0077 \\
(X_i) &= 0000 \ 4500 \ 3333 \ 2000 \ 0077 \\
(B_j) &= 77 \ 7743 \\

(X_k) &= 5743 \ 3277 \ 4444 \ 5777 \ 7700 \\
(X_i) &= 7777 \ 3277 \ 4444 \ 5777 \ 7700
\end{align*}
\]
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.

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

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 Xk 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 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.

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.

\[
\begin{align*}
Xk &= 0000 4500 3333 2000 0077 \\
Bj &= 00 0034 \\
Xi &= 2034 4500 3333 2000 0077 \\
Xk &= 0000 4500 3333 2000 0077 \\
Bj &= 77 7743 \\
Xi &= 1743 4500 3333 2000 0077 \\
Xk &= 7777 3277 4444 5777 7700 \\
Bj &= 00 0034 \\
Xi &= 5743 3277 4444 5777 7700 \\
Xk &= 7777 3277 4444 5777 7700 \\
Bj &= 77 7743 \\
Xi &= 6034 3277 4444 5777 7700
\end{align*}
\]

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.

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.

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.

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.

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

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

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.

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
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

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 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

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.

33ijk Floating Double-Precision Difference DXi Xj - Xk to Xi

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.
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 (3777xxx...x or 6000xxx...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

<table>
<thead>
<tr>
<th>14</th>
<th>98</th>
<th>65</th>
<th>32</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>l</td>
<td>j</td>
<td>k</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>14</th>
<th>98</th>
<th>65</th>
<th>32</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>l</td>
<td>j</td>
<td>k</td>
<td></td>
</tr>
</tbody>
</table>

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.
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

\[
\text{DXI } \text{Xj} \times \text{Xk} \text{ to Xi}
\]

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.
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 occurs. 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.

451jk Round Floating Divide (Xj) RXI Xj/Xk by (Xk) to Xi

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

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 Kl designator specifies which CM word contains the first character of the source data field, and a Cl 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 Kl=1000 and Cl=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 Cl 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 (177g) characters. The maximum data field length for the move indirect instruction is 8191 (17777g) characters. If L (LU and LL combined) is zero, the compare/move operation having a decremented field length greater than 377 (octal).

464 jK Move Indirect

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.

<table>
<thead>
<tr>
<th>59 5150 4847</th>
<th>30 29</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>464</td>
<td>K</td>
<td></td>
</tr>
</tbody>
</table>

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 XO clears at the end of the execution.

465 Move Direct

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

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

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.

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.

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.

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.

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.
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.

\[ i = 0 \quad \text{No CM reference} \]
\[ i = 1,2,3,4,5 \quad \text{Read from CM to Xi} \]
\[ i = 6,7 \quad \text{Write into CM from Xi} \]

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

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.

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.

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.

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.

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.

\[ i = 0 \quad \text{No CM reference} \]
\[ i = 1,2,3,4,5 \quad \text{Read from CM to Xi} \]
\[ i = 6,7 \quad \text{Write into CM from Xi} \]

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

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.

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.

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

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)

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)

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)

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

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.

71ijK Set Xi to (Bj) + K

This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Bk, 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.

72ijk Set Xi to (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.

73ijk Set Xi to (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.

74ijk Set Xi to (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.

75ijk Set Xi to (Aj) - (Bk)

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.

76ijk Set Xi to (Bj) + (Bk)

This instruction reads operands from Bk 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 \( X_i \).

\[ 77\text{jk} \text{ Set } X_i \text{ to } (B_j) - (B_k) \]

This instruction reads operands from \( B_j \) and \( B_k \), subtracts the \( B_k \) operand from the \( B_j \) operand, and delivers the result to \( X_i \). 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 \( X_i \).

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

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

**Timing Notes:**

1. First time shown if branch was not taken; second time shown if branch was taken.
<table>
<thead>
<tr>
<th>Instruction Code</th>
<th>Description</th>
<th>Execution Time in 56-ns Cycles</th>
<th>Timing Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>037jK</td>
<td>Branch to K if ((X_j)) is indefinite</td>
<td>4 or 11-14</td>
<td>1</td>
</tr>
<tr>
<td>041jK</td>
<td>Branch to K if ((B_i) = (B_j))</td>
<td>6 or 11-14</td>
<td>1</td>
</tr>
<tr>
<td>051jK</td>
<td>Branch to K if ((B_i) \neq (B_j))</td>
<td>6 or 11-14</td>
<td>1</td>
</tr>
<tr>
<td>061jK</td>
<td>Branch to K if ((B_i) \geq (B_j))</td>
<td>6 or 11-14</td>
<td>1</td>
</tr>
<tr>
<td>071jK</td>
<td>Branch to K if ((B_i) &lt; (B_j))</td>
<td>6 or 11-14</td>
<td>1</td>
</tr>
<tr>
<td>101jx</td>
<td>Transmit ((X_j)) to (X_i)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>111jk</td>
<td>Logical product of ((X_j)) and ((X_k)) to (X_i)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>121jk</td>
<td>Logical sum of ((X_j)) and ((X_k)) to (X_i)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>131jk</td>
<td>Logical difference of ((X_j)) and ((X_k)) to (X_i)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>141jk</td>
<td>Transmit complement of ((X_k)) to (X_i)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>151jk</td>
<td>Logical product of ((X_j)) with complement of ((X_k)) to (X_i)</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>161jk</td>
<td>Logical sum of ((X_j)) with complement of ((X_k)) to (X_i)</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>171jk</td>
<td>Logical difference of ((X_j)) with complement of ((X_k)) to (X_i)</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>204jk</td>
<td>Left shift ((X_i)) by (j_k)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>211jk</td>
<td>Right shift ((X_i)) by (j_k)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>221jk</td>
<td>Left shift ((X_k)) nominally ((B_j)) places to (X_i)</td>
<td>6 or 7-10</td>
<td>2</td>
</tr>
<tr>
<td>231jk</td>
<td>Right shift ((X_k)) nominally ((B_j)) places to (X_i)</td>
<td>6 or 7-10</td>
<td>2</td>
</tr>
<tr>
<td>241jk</td>
<td>Normalize ((X_k)) to (X_i) and (B_j)</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>251jk</td>
<td>Round normalize ((X_k)) to (X_i) and (B_j)</td>
<td>11-14</td>
<td>-</td>
</tr>
<tr>
<td>261jk</td>
<td>Unpack ((X_k)) to (X_i) and (B_j)</td>
<td>7-10</td>
<td>-</td>
</tr>
<tr>
<td>271jk</td>
<td>Pack ((X_k)) and ((B_j)) to (X_i)</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>301jk</td>
<td>Floating sum of ((X_j)) and ((X_k)) to (X_i)</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>311jk</td>
<td>Floating difference of ((X_j)) and ((X_k)) to (X_i)</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>321jk</td>
<td>Floating double-precision sum of ((X_j)) and ((X_k)) to (X_i)</td>
<td>40-60</td>
<td>-</td>
</tr>
<tr>
<td>331jk</td>
<td>Floating double-precision difference of ((X_j)) and ((X_k)) to (X_i)</td>
<td>40-60</td>
<td>-</td>
</tr>
<tr>
<td>341jk</td>
<td>Round floating sum of ((X_j)) and ((X_k)) to (X_i)</td>
<td>7-10</td>
<td>-</td>
</tr>
<tr>
<td>351jk</td>
<td>Round floating difference of ((X_j)) and ((X_k)) to (X_i)</td>
<td>7-10</td>
<td>-</td>
</tr>
<tr>
<td>361jk</td>
<td>Integer sum of ((X_j)) and ((X_k)) to (X_i)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>371jk</td>
<td>Integer difference of ((X_j)) and ((X_k)) to (X_i)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>401jk</td>
<td>Floating product of ((X_j)) and ((X_k)) to (X_i)</td>
<td>11-14</td>
<td>-</td>
</tr>
</tbody>
</table>

**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>
<thead>
<tr>
<th>Instruction Code</th>
<th>Description</th>
<th>Execution Time in 56-ns Cycles</th>
<th>Timing Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>411jk</td>
<td>Round floating product of (Xj) and (Xk) to Xi</td>
<td>11-14</td>
<td></td>
</tr>
<tr>
<td>421jk</td>
<td>Floating double-precision product of (Xj) and (Xk) to Xi</td>
<td>11-14</td>
<td></td>
</tr>
<tr>
<td>431jk</td>
<td>Form mask of jk bits to Xi</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>441jk</td>
<td>Floating divide (Xj) by (Xk) to Xi</td>
<td>40-60</td>
<td></td>
</tr>
<tr>
<td>451jk</td>
<td>Round floating divide (Xj) by (Xk) to Xi</td>
<td>40-60</td>
<td></td>
</tr>
<tr>
<td>460xx</td>
<td>Pass</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>464jk</td>
<td>Move indirect</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>465</td>
<td>Move direct</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>466</td>
<td>Compare collated</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>467</td>
<td>Compare uncollated</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>471xk</td>
<td>Population count of (Xk) to Xi</td>
<td>20-25</td>
<td>4</td>
</tr>
<tr>
<td>501jK</td>
<td>Set Ai to (Aj) + K</td>
<td>3, 7-10, or 20-25</td>
<td>5</td>
</tr>
<tr>
<td>511jK</td>
<td>Set Ai to (Bj) + K</td>
<td>3, 7-10, or 20-25</td>
<td>5</td>
</tr>
<tr>
<td>521jK</td>
<td>Set Ai to (Xj) + K</td>
<td>3,7-10, or 20-25</td>
<td>5</td>
</tr>
<tr>
<td>531jK</td>
<td>Set Ai to (Xj) + (Bk)</td>
<td>3, 7-10, or 20-25</td>
<td>5</td>
</tr>
<tr>
<td>541jK</td>
<td>Set Ai to (Aj) + (Bk)</td>
<td>3, 7-10, or 20-25</td>
<td>5</td>
</tr>
<tr>
<td>551jK</td>
<td>Set Ai to (Aj) - (Bk)</td>
<td>3, 7-10, or 20-25</td>
<td>5</td>
</tr>
<tr>
<td>561jK</td>
<td>Set Ai to (Bj) + (Bk)</td>
<td>3, 7-10, or 20-25</td>
<td>5</td>
</tr>
<tr>
<td>571jK</td>
<td>Set Ai to (Bj) - (Bk)</td>
<td>3, 7-10, or 20-25</td>
<td>5</td>
</tr>
<tr>
<td>601jK</td>
<td>Set Bi to (Aj) + K</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>611jK</td>
<td>Set Bi to (Bj) + K</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>621jK</td>
<td>Set Bi to (Xj) + K</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>631jK</td>
<td>Set Bi to (Xj) + (Bk)</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>641jK</td>
<td>Set Bi to (Aj) + (Bk)</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>651jK</td>
<td>Set Bi to (Aj) - (Bk)</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>660jk</td>
<td>Read CM at (Xk) to Xj</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Instruction Code</th>
<th>Description</th>
<th>Execution Time in 56-ns Cycles</th>
<th>Timing Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>66ijk</td>
<td>Set Bi to (Bj) + (Bk)</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>670jk</td>
<td>Write Xj into CM at (Xk)</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>671jk</td>
<td>Set Bi to (Bj) - (Bk)</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>701jk</td>
<td>Set Xi to (Aj) + K</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>711jk</td>
<td>Set Xi to (Bj) + K</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>721jk</td>
<td>Set Xi to (Xj) + K</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>731jk</td>
<td>Set Xi to (Xj) + (Bk)</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>741jk</td>
<td>Set Xi to (Aj) + (Bk)</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>751jk</td>
<td>Set Xi to (Aj) - (Bk)</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>761jk</td>
<td>Set Xi to (Bj) + (Bk)</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>771jk</td>
<td>Set Xi to (Bj) - (Bk)</td>
<td>5</td>
<td>-</td>
</tr>
</tbody>
</table>
## Table 4-4. Model 855 CP Instruction Timing (Sheet 1 of 3)

<table>
<thead>
<tr>
<th>Instruction Code</th>
<th>Description</th>
<th>Execution Time in 64-ns Cycles</th>
<th>Timing Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>00xxx</td>
<td>Error exit to MA or interrupt to executive mode</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>010xK</td>
<td>Return jump to K</td>
<td>10-20</td>
<td>-</td>
</tr>
<tr>
<td>011jK</td>
<td>Block copy Bj + K words from UEM to CM</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>012jK</td>
<td>Block copy Bj + K words from CM to UEM</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>013jK</td>
<td>Central exchange jump to Bj + K (CYBER 170 monitor flag set)</td>
<td>125</td>
<td>-</td>
</tr>
<tr>
<td>013xx</td>
<td>Monitor exchange jump to MA (CYBER 170 monitor flag clear)</td>
<td>125</td>
<td>-</td>
</tr>
<tr>
<td>014jk</td>
<td>Read one word from UEM to Xj</td>
<td>10-20</td>
<td>-</td>
</tr>
<tr>
<td>015jk</td>
<td>Write one word from Xj to UEM</td>
<td>10-20</td>
<td>-</td>
</tr>
<tr>
<td>016jk</td>
<td>Read free running counter</td>
<td>10-20</td>
<td>-</td>
</tr>
<tr>
<td>017jk</td>
<td>Illegal instruction</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>021xK</td>
<td>Jump to (B1) + K</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>030jK</td>
<td>Branch to K if (Xj) = 0</td>
<td>1 or 6</td>
<td>2</td>
</tr>
<tr>
<td>031jK</td>
<td>Branch to K if (Xj) ≠ 0</td>
<td>1 or 6</td>
<td>2</td>
</tr>
<tr>
<td>032jK</td>
<td>Branch to K if (Xj) is positive</td>
<td>1 or 6</td>
<td>2</td>
</tr>
<tr>
<td>033jK</td>
<td>Branch to K if (Xj) is negative</td>
<td>1 or 6</td>
<td>2</td>
</tr>
<tr>
<td>034jK</td>
<td>Branch to K if (Xj) is in range</td>
<td>1 or 6</td>
<td>2</td>
</tr>
<tr>
<td>035jK</td>
<td>Branch to K if (Xj) is out of range</td>
<td>1 or 6</td>
<td>2</td>
</tr>
<tr>
<td>036jK</td>
<td>Branch to K if (Xj) is definite</td>
<td>1 or 6</td>
<td>2</td>
</tr>
<tr>
<td>037jK</td>
<td>Branch to K if (Xj) is indefinite</td>
<td>1 or 6</td>
<td>2</td>
</tr>
<tr>
<td>041jK</td>
<td>Branch to K if (B1) = (Bj)</td>
<td>1 or 6</td>
<td>2</td>
</tr>
<tr>
<td>051jK</td>
<td>Branch to K if (B1) ≠ (Bj)</td>
<td>1 or 6</td>
<td>2</td>
</tr>
<tr>
<td>061jK</td>
<td>Branch to K if (B1) ≥ (Bj)</td>
<td>1 or 6</td>
<td>2</td>
</tr>
<tr>
<td>071jK</td>
<td>Branch to K if (B1) &lt; (Bj)</td>
<td>1 or 6</td>
<td>2</td>
</tr>
<tr>
<td>101jx</td>
<td>Transmit (Xj) to Xi</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>111jk</td>
<td>Logical product of (Xj) and (Xk) to Xi</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>121jk</td>
<td>Logical sum of (Xj) and (Xk) to Xi</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>131jk</td>
<td>Logical difference of (Xj) and (Xk) to Xi</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>141xk</td>
<td>Transmit complement of (Xk) to Xi</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>151jk</td>
<td>Logical product of (Xj) with complement of (Xk) to Xi</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>

**Timing Notes:**

1. Execution time varies depending on number of words and number of 16-word blocks. Execution time is: 36 + 24\* (number of words + 5\* (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>
<thead>
<tr>
<th>Instruction Code</th>
<th>Description</th>
<th>Execution Time in 64-ns Cycles</th>
<th>Timing Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>16ijk</td>
<td>Logical sum of ((X_j)) with complement of ((X_k)) to (X_i)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>17ijk</td>
<td>Logical difference of ((X_j)) with complement of ((X_k)) to (X_i)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>20ijk</td>
<td>Left shift ((X_i)) by (j_k)</td>
<td>4-5</td>
<td>-</td>
</tr>
<tr>
<td>21ijk</td>
<td>Right shift ((X_i)) by (j_k)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>22ijk</td>
<td>Left shift ((X_k)) nominally ((B_j)) places to (X_i)</td>
<td>4-5</td>
<td>-</td>
</tr>
<tr>
<td>23ijk</td>
<td>Right shift ((X_k)) nominally ((B_j)) places to (X_i)</td>
<td>4-5</td>
<td>-</td>
</tr>
<tr>
<td>24ijk</td>
<td>Normalize ((X_k)) to (X_i) and (B_j)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>25ijk</td>
<td>Round normalize ((X_k)) to (X_i) and (B_j)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>26ijk</td>
<td>Unpack ((X_k)) to (X_i) and (B_j)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>27ijk</td>
<td>Pack ((X_k)) and ((B_j)) to (X_i)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>30ijk</td>
<td>Floating sum of ((X_j)) and ((X_k)) to (X_i)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>31ijk</td>
<td>Floating difference of ((X_j)) and ((X_k)) to (X_i)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>32ijk</td>
<td>Floating double-precision sum of ((X_j)) and ((X_k)) to (X_i)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>33ijk</td>
<td>Floating double-precision difference of ((X_j)) and ((X_k)) to (X_i)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>34ijk</td>
<td>Round floating sum of ((X_j)) and ((X_k)) to (X_i)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>35ijk</td>
<td>Round floating difference of ((X_j)) and ((X_k)) to (X_i)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>36ijk</td>
<td>Integer sum of ((X_j)) and ((X_k)) to (X_i)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>37ijk</td>
<td>Integer difference of ((X_j)) and ((X_k)) to (X_i)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>40ijk</td>
<td>Floating product of ((X_j)) and ((X_k)) to (X_i)</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>41ijk</td>
<td>Round floating product of ((X_j)) and ((X_k)) to (X_i)</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>42ijk</td>
<td>Floating double-precision product of ((X_j)) and ((X_k)) to (X_i)</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>43ijk</td>
<td>Form mask of (j_k) bits to (X_i)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>44ijk</td>
<td>Floating divide ((X_j)) by ((X_k)) to (X_i)</td>
<td>10-20</td>
<td>-</td>
</tr>
<tr>
<td>45ijk</td>
<td>Round floating divide ((X_j)) by ((X_k)) to (X_i)</td>
<td>10-20</td>
<td>-</td>
</tr>
<tr>
<td>460xx-463xx</td>
<td>Pass</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>464ijk</td>
<td>Move indirect</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>465</td>
<td>Move direct</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>466</td>
<td>Compare collated</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>467</td>
<td>Compare uncollated</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>471xx</td>
<td>Population count of ((X_k)) to (X_i)</td>
<td>10-20</td>
<td>-</td>
</tr>
</tbody>
</table>

**Timing Notes:**

3. CMU instructions are simulated. For best results, recompile to avoid use of these instructions.
<table>
<thead>
<tr>
<th>Instruction Code</th>
<th>Description</th>
<th>Execution Time in 64-ns Cycles</th>
<th>Timing Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>50ijK</td>
<td>Set Ai to (Aj) + K</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>511jK</td>
<td>Set Ai to (Bj) + K</td>
<td>4-5</td>
<td>4</td>
</tr>
<tr>
<td>521jK</td>
<td>Set Ai to (Xj) + K</td>
<td>4-5</td>
<td>4</td>
</tr>
<tr>
<td>531jK</td>
<td>Set Ai to (Xj) + (Bk)</td>
<td>4-5</td>
<td>4</td>
</tr>
<tr>
<td>541jK</td>
<td>Set Ai to (Aj) + (Bk)</td>
<td>4-5</td>
<td>4</td>
</tr>
<tr>
<td>551jK</td>
<td>Set Ai to (Aj) - (Bk)</td>
<td>4-5</td>
<td>4</td>
</tr>
<tr>
<td>561jK</td>
<td>Set Ai to (Bj) + (Bk)</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>571jK</td>
<td>Set Ai to (Bj) - (Bk)</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>601jK</td>
<td>Set Bi to (Aj) + K</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>611jK</td>
<td>Set Bi to (Bj) + K</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>621jK</td>
<td>Set Bi to (Xj) + K</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>631jK</td>
<td>Set Bi to (Xj) + (Bk)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>641jK</td>
<td>Set Bi to (Aj) + (Bk)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>651jK</td>
<td>Set Bi to (Aj) - (Bk)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>660jK</td>
<td>Read CM at (Xk) to Xj</td>
<td>4-5</td>
<td>4</td>
</tr>
<tr>
<td>661jK</td>
<td>Set Bi to (Bj) + (Bk)</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>670jK</td>
<td>Write Xj into CM at (Xk)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>671jK</td>
<td>Set Bi to (Bj) - (Bk)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>701jK</td>
<td>Set Xi to (Aj) + K</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>711jK</td>
<td>Set Xi to (Bj) + K</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>721jK</td>
<td>Set Xi to (Xj) + K</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>731jK</td>
<td>Set Xi to (Xj) + (Bk)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>741jK</td>
<td>Set Xi to (Aj) + (Bk)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>751jK</td>
<td>Set Xi to (Aj) - (Bk)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>761jK</td>
<td>Set Xi to (Bj) + (Bk)</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>771jK</td>
<td>Set Xi to (Bj) - (Bk)</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>

Timing Notes:
4. No map miss and cache hit rate of 75% is assumed.
5. No map miss is assumed.
6. i#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

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

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.

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).
RELOCATION REGISTER IN PP HARDWARE

LOCATION 15 12 9 0
d + 1
ZEROS a

RELOCATION REGISTER IN PP MEMORY

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

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)

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)

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

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

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

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. Positive zero is treated as a positive quantity. For interpretation of d, refer to the 03 instruction.

06d Plus Jump 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. For interpretation of d, refer to the 03 instruction.

07d Minus Jump 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. Negative zero is treated as a negative quantity. For interpretation of d, refer to the 03 instruction.
This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the contents of the A register is negative, the jump is taken. If the contents 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.

20d  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.

21d  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.

22d  Logical Product dm  LPC dm

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.
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

\[
\begin{array}{cccc}
31 & 28 & 27 & 22 \\
16 & 15 & 12 & 11 \\
00 & 23 & d & 00 \\
(P) & (P+1) & m & \\
\end{array}
\]

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

\[
\begin{array}{cccc}
15 & 12 & 11 & 6 \\
00 & 24 & d & 0 \\
\end{array}
\]

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

\[
\begin{array}{cccc}
15 & 12 & 11 & 6 \\
00 & 25 & d & 0 \\
\end{array}
\]

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.

26d Monitor Exchange Jump EXN

\[
\begin{array}{cccc}
15 & 12 & 11 & 6 \\
00 & 26 & 00 & 0 \\
\end{array}
\]

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

\[
\begin{array}{cccc}
15 & 12 & 11 & 6 \\
00 & 26 & 10 & 0 \\
\end{array}
\]

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

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

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.

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.

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 FPM, and the word read is used as the operand address.

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

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 FPM, and the word read is used as the operand address.

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 FPM, and the word read is used as the operand address.

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.

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.
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.

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.

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. 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.

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.

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.

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

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.

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.
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.

This instruction stores P plus 1 into PP location 0000 before beginning the transfer. 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 37776g 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.

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.

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

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

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

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

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  
SPFM 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  
ELJM 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 full. The next instruction is at P plus 2 if the channel is 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.

70d Input to A from Channel 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 or is deactivated before a full is received, the instruction exits. The word is not accepted, and the A register clears.

71dm Input A Words to m from Channel 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 input operation is accomplished, and the program continues at P plus 2.

72d Output from A on Channel d

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

**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.

73dm Output A Words from m on Channel 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. 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.

74d Activate Channel 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 already active and if bit 5 of d is set, the program continues at P plus 1. Otherwise, activating an already active channel causes the PP to wait until the channel goes inactive. The PP hangs if the channel does not go inactive.

75d Deactivate Channel d  
DCN d  

This instruction deactivates the channel specified by d. As a result, the I/O data transfer stops.

**NOTE**

If this instruction executes when the data channel is already inactive and bit 5 of d is set, the program continues at P plus 1. The channel remains inactive, and no inactive signal is sent to the I/O equipment. Deactivating an already inactive channel causes the PP to hang until the channel becomes active.

If an output instruction is followed by a disconnect instruction without first establishing that the information has been accepted by the input device (check for channel empty), the last word transmitted may be lost.

Do not deactivate a channel before putting a useful program in the associated PP. PPs other than 0 are hung on an input instruction (71) after deadstart. Deactivating a channel after deadstart causes an exit to the address specified by the content of location 0000 plus 1 and execution of that program. If the channel is deactivated without a valid program in that PP, the PP executes whatever program was left in PPM. Therefore, the PP could run wild.

76d Function A on Channel d  
FAN d  

This instruction sends the external function code in the lower 12 bits of the A register on channel d.

**NOTE**

If this instruction executes with bit 5 of d clear and the channel active, PP execution stops until a deadstart or another PP causes the channel to become inactive. If bit 5 of d is set and the channel is active, the program continues at P plus 1. Neither the function signal nor the function word transmits. The channel remains active, and execution continues.

77dm Function m on Channel d  
FNC m,d  

This instruction sends the external function code specified by m on channel d.

**NOTE**

If this instruction executes with bit 5 of d clear and the channel active, PP execution stops until a deadstart or another PP causes the channel to become inactive. If bit 5 of d is set and the channel is active, the program continues at P plus 2. Neither the function signal nor the function word transmits. The channel remains active, and execution continues.

### INSTRUCTION EXECUTION TIMING

Approximate execution times for the PP instructions are listed in table 4-6. These times are listed with the assumption that no conflicts occur. The numbers in the timing notes column refer to the notes at the end of the table. Execution times are given in 250-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>
<thead>
<tr>
<th>Instruction Code</th>
<th>Description</th>
<th>Execution Time in 250-ns Cycles</th>
<th>Timing Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>00xx</td>
<td>Pass</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>01dm</td>
<td>Long jump to m + (d)</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>02dm</td>
<td>Return jump to m + (d)</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>03d</td>
<td>Unconditional jump d</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>04d</td>
<td>Zero jump d</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>05d</td>
<td>Nonzero jump d</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>06d</td>
<td>Plus jump d</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>07d</td>
<td>Minus jump d</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>10d</td>
<td>Shift d</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>11d</td>
<td>Logical difference d</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>12d</td>
<td>Logical product d</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>13d</td>
<td>Selective clear d</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>14d</td>
<td>Load d</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>15d</td>
<td>Load complement d</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>16d</td>
<td>Add d</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>17d</td>
<td>Subtract d</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>20dm</td>
<td>Load dm</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>21dm</td>
<td>Add dm</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>22dm</td>
<td>Logical product dm</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>23dm</td>
<td>Logical difference dm</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>24d</td>
<td>Load R register from (d) and (d) + 1</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>25d</td>
<td>Store R register at (d) and (d) + 1</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>260x</td>
<td>Exchange jump</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>261x</td>
<td>Monitor exchange jump</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>262x</td>
<td>Monitor exchange jump to MA</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>27d</td>
<td>Pass</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>30d</td>
<td>Load (d)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>31d</td>
<td>Add (d)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>32d</td>
<td>Subtract (d)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>33d</td>
<td>Logical difference (d)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>34d</td>
<td>Store (d)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>35d</td>
<td>Replace add (d)</td>
<td>4</td>
<td>-</td>
</tr>
</tbody>
</table>

Timing Notes:

1. No assembly-disassembly unit (ADU) conflicts and no outstanding CYBER 170 exchange jump request in the ADU.
### Table 4-6. FP Instruction Timing (Sheet 2 of 3)

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

**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)

<table>
<thead>
<tr>
<th>Instruction Code</th>
<th>Description</th>
<th>Execution Time in 250-ns Cycles</th>
<th>Timing Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>670cm</td>
<td>Jump to m if channel c empty</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>671cm</td>
<td>Jump to m if channel c error flag clear</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>70d</td>
<td>Input to A from channel d</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>71dm</td>
<td>Input A words to m from channel d</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>72d</td>
<td>Output from A on channel d</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>73dm</td>
<td>Output (A) words from m on channel d</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>74d</td>
<td>Activate channel d</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>75d</td>
<td>Deactivate channel d</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>76d</td>
<td>Function A on channel d</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>77dm</td>
<td>Function m on channel d</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Timing Notes:

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).

Nonconflict case is when two PPs communicating to each other are not in the slot at the same time.

Conflict case is when two PPs communicating with each other are in the slot at the same time.
<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/1</td>
<td>12:00</td>
<td>Meeting</td>
<td>Office</td>
</tr>
<tr>
<td>1/2</td>
<td>14:00</td>
<td>Seminar</td>
<td>Auditorium</td>
</tr>
<tr>
<td>1/3</td>
<td>09:00</td>
<td>Workshop</td>
<td>Conference Room</td>
</tr>
</tbody>
</table>

Note: Times are in 24-hour format.
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](image)

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

<table>
<thead>
<tr>
<th>Bit 59</th>
<th>Bit 58</th>
<th>Coefficient Sign</th>
<th>Exponent Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>Positive</td>
<td>Positive</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Positive</td>
<td>Negative</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Negative</td>
<td>Positive</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Negative</td>
<td>Negative</td>
</tr>
</tbody>
</table>

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 $2000x, \ldots, x$
- Negative zero exponent $1777x, \ldots, x$

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
   Unpacked exponent
   Packed format
   0000 0000 0000 0000 0001
   00 0000
   2000 0000 0000 0000 0001

2. Unpacked coefficient
   Unpacked exponent
   Packed format
   0000 4000 0000 0000 0000
   77 7720
   1720 4000 0000 0000 0000

3. Unpacked coefficient
   Unpacked exponent
   Packed format
   0000 6200 0000 0000 0000
   77 7726
   1726 6200 0000 0000 0000

4. Unpacked coefficient
   Unpacked exponent
   Packed format
   7777 1577 7777 7777 7777
   77 7726
   6051 1577 7777 7777 7777

5. Unpacked coefficient
   Unpacked exponent
   Packed format
   0000 4771 3000 0044 7021
   00 1363
   3363 4771 3000 0044 7021

6. Unpacked coefficient
   Unpacked exponent
   Packed format
   0000 6301 0277 4315 6033
   77 6210
   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 normalization 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 60), 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 (30 through 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 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)

<table>
<thead>
<tr>
<th>Xk</th>
<th>W</th>
<th>+∞</th>
<th>-∞</th>
<th>+ IND</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>+∞</td>
<td>-∞</td>
<td>IND</td>
<td></td>
</tr>
<tr>
<td>+∞</td>
<td>+∞</td>
<td>-∞</td>
<td>IND</td>
<td></td>
</tr>
<tr>
<td>-∞</td>
<td>-∞</td>
<td>+∞</td>
<td>IND</td>
<td></td>
</tr>
<tr>
<td>± IND</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Xk</th>
<th>W</th>
<th>-∞</th>
<th>+∞</th>
<th>+ IND</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>-∞</td>
<td>+∞</td>
<td>IND</td>
<td></td>
</tr>
<tr>
<td>+∞</td>
<td>+∞</td>
<td>IND</td>
<td>IND</td>
<td></td>
</tr>
<tr>
<td>-∞</td>
<td>-∞</td>
<td>IND</td>
<td>IND</td>
<td></td>
</tr>
<tr>
<td>± IND</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Xk</th>
<th>+N</th>
<th>-N</th>
<th>+0</th>
<th>-0</th>
<th>+∞</th>
<th>-∞</th>
<th>+ IND</th>
</tr>
</thead>
<tbody>
<tr>
<td>+N</td>
<td>0</td>
<td>0</td>
<td>+∞</td>
<td>-∞</td>
<td>IND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-N</td>
<td>0</td>
<td>0</td>
<td>-∞</td>
<td>+∞</td>
<td>IND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+0</td>
<td>0</td>
<td>0</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0</td>
<td>0</td>
<td>0</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+∞</td>
<td>+∞</td>
<td>-∞</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-∞</td>
<td>-∞</td>
<td>+∞</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>± IND</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Xk</th>
<th>+N</th>
<th>-N</th>
<th>+0</th>
<th>-0</th>
<th>+∞</th>
<th>-∞</th>
<th>+ IND</th>
</tr>
</thead>
<tbody>
<tr>
<td>+N</td>
<td>+∞</td>
<td>-∞</td>
<td>0</td>
<td>0</td>
<td>IND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-N</td>
<td>-∞</td>
<td>+∞</td>
<td>0</td>
<td>0</td>
<td>IND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+0</td>
<td>0</td>
<td>0</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0</td>
<td>0</td>
<td>0</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+∞</td>
<td>+∞</td>
<td>-∞</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-∞</td>
<td>-∞</td>
<td>+∞</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>± IND</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td>IND</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An integer divide takes several steps. For example, an integer quotient $X_1$ equal to $X_2/X_3$ is produced by the following steps.

<table>
<thead>
<tr>
<th>Instructions</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Pack $X_2$ from $X_2$ and $B_0$</td>
<td>Pack $X_2$</td>
</tr>
<tr>
<td>2. Pack $X_3$ from $X_3$ and $B_0$</td>
<td>Pack $X_3$</td>
</tr>
<tr>
<td>3. Normalize $X_3$ in $X_0$ and $B_0$</td>
<td>Normalize $X_3$ (divisor)</td>
</tr>
<tr>
<td>4. Normalize $X_2$ in $X_2$ and $B_0$</td>
<td>Normalize $X_2$ (dividend)</td>
</tr>
<tr>
<td>5. Floating quotient of $X_2$ and $X_0$ to $X_1$</td>
<td>Divide</td>
</tr>
<tr>
<td>6. Unpack $X_1$ to $X_1$ and $B_7$</td>
<td>Unpack quotient</td>
</tr>
<tr>
<td>7. Shift $X_1$ nominally left 87 places</td>
<td>Shift to integer position</td>
</tr>
</tbody>
</table>

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 $X_3$ instruction ensures this condition.

The normalize $X_3$ instruction left shifts the divisor $n$ places ($n \geq 0$), 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 $X_1$. A remainder may be obtained by an integer multiply of X1 and X3 and subtracting the result from $X_2$.

**INTEGER ARITHMETIC**

Multiply packets 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 F 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)](image)

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ec</td>
<td>6-bit exit condition code</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Illegal instruction</td>
</tr>
<tr>
<td>01</td>
<td>Address-range error (bit 48)</td>
</tr>
<tr>
<td>02</td>
<td>Floating-point infinite (bit 49)</td>
</tr>
<tr>
<td>04</td>
<td>Floating-point indefinite (bit 50)</td>
</tr>
<tr>
<td>20</td>
<td>Processor-detected malfunction</td>
</tr>
<tr>
<td>67</td>
<td>Hardware malfunction</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Error Condition</th>
<th>Exit Mode Selected</th>
<th>Exit Mode Not Selected</th>
</tr>
</thead>
</table>
| Illegal instruction or 00 instruction.                                          | 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.                                                                                                                                                                                                                                                   | 1. N/A (exit mode is always selected).                                                                                                                                                                                                 |
| Exit condition bit 48 set by an incremental read with an address out of range (AOR). | 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.                                                                                                                                                                                                                                                   | 1. Inhibit read, X unchanged.  
2. Continue execution.                                                                                                                                                                                                                                                                                                                                |
| Exit condition bit 48 set by an incremental write with an address out of range (AOR). | 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.                                                                                                                                                                                                                                                   | 1. Inhibit write, CM unchanged.  
2. Continue execution.                                                                                                                                                                                                                                                                                                                                |
| Exit condition bit 48 set by an RNI or branch address out of range.             | 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.                                                                                                                                                                                                                                                   | 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)

<table>
<thead>
<tr>
<th>Error Condition</th>
<th>Error Response</th>
<th>Exit Mode Not Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit condition bit 48 set on CMU instruction.</td>
<td>1. Detected by executive state during the execution of compare/move instruction.</td>
<td>1. Detected by executive state during the execution of compare/move instruction.</td>
</tr>
<tr>
<td>1. Cl or C2 greater than 9.</td>
<td>2. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted.</td>
<td>2. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted.</td>
</tr>
<tr>
<td>2. K1 or K2 address out of range.</td>
<td>3. Store P and exit bits (01) at RAC.</td>
<td>3. Continue with next instruction.</td>
</tr>
<tr>
<td></td>
<td>4. CP stops in executive state.</td>
<td></td>
</tr>
<tr>
<td>Exit condition bit 48 set by a UEM address range check for instructions 011 and 012.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. Execute instruction as a pass.</td>
<td>1. Execute instruction as a pass.</td>
</tr>
<tr>
<td></td>
<td>2. Store P and exit bits (01) at RAC.</td>
<td>2. Exit to next 60-bit word and continue execution.</td>
</tr>
<tr>
<td></td>
<td>3. Interrupt to executive state.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. CP stops in executive state.</td>
<td></td>
</tr>
<tr>
<td>Exit condition bit 48 set by a UEM address range check for instructions 014 and 015.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. Execute instruction as a pass.</td>
<td>1. Execute instruction as a pass.</td>
</tr>
<tr>
<td></td>
<td>2. Store P and exit condition bits (01) at RAC. P equals address of following instruction.</td>
<td>2. Exit to next parcel and continue execution.</td>
</tr>
<tr>
<td></td>
<td>3. Interrupt to executive state.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. CP stops in executive state.</td>
<td></td>
</tr>
<tr>
<td>Exit condition bit 49 set by infinite condition, or bit 50 set by indefinite condition.</td>
<td>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.</td>
<td>1. Continue execution.</td>
</tr>
<tr>
<td></td>
<td>2. Interrupt to executive state.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. CP stops in executive state.</td>
<td></td>
</tr>
<tr>
<td>Any hardware parity error or double SECDED error.</td>
<td>1. Interrupt to executive state.</td>
<td>1. Interrupt to executive state.</td>
</tr>
<tr>
<td></td>
<td>2. Executive state stores P and exit condition bits (20) at RAC.</td>
<td>2. Executive state stores P and exit condition bits (20) at RAC.</td>
</tr>
<tr>
<td></td>
<td>3. CP stops in executive state.</td>
<td>3. CP stops in executive state.</td>
</tr>
<tr>
<td>Error Condition</td>
<td>Exit Mode Selected</td>
<td>Exit Mode Not Selected</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>--------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Illegal instruction or 00 instruction.</td>
<td>1. The instruction is not executed.</td>
<td>1. N/A (exit mode is always selected).</td>
</tr>
<tr>
<td></td>
<td>2. Store P and exit condition bits (00) at location RAC. P equals address of illegal instruction.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Exchange jump to MA and set CYBER 170 MF.</td>
<td></td>
</tr>
<tr>
<td>Exit condition bit 48 set by an incremental read with an address out of range (AOR).</td>
<td>1. The X register is unchanged.</td>
<td>1. Inhibit read, X unchanged.</td>
</tr>
<tr>
<td></td>
<td>2. The A register contains the AOR address.</td>
<td>2. Continue execution.</td>
</tr>
<tr>
<td></td>
<td>3. Store P and exit condition bits (01) at location RAC. P equals address of increment instruction or address of instruction following the increment.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Exchange jump to MA and set CYBER 170 MF.</td>
<td></td>
</tr>
<tr>
<td>Exit condition bit 48 set by an incremental write with an address out of range (AOR).</td>
<td>1. Block write operation; content of CM is unchanged.</td>
<td>1. Inhibit write, CM unchanged.</td>
</tr>
<tr>
<td></td>
<td>2. The A register contains the AOR address.</td>
<td>2. Continue execution.</td>
</tr>
<tr>
<td></td>
<td>3. Store P and exit condition bits (01) at location RAC. P equals address of instruction or address of instruction following the increment.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Exchange jump to MA and set CYBER 170 MF.</td>
<td></td>
</tr>
<tr>
<td>Exit condition bit 48 set by an RNI or branch address out of range.</td>
<td>1. Inhibit execution.</td>
<td>1. N/A (exit mode is always selected regardless of status of EM register bit 48).</td>
</tr>
<tr>
<td></td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Exchange jump to MA and set CYBER 170 MF.</td>
<td></td>
</tr>
<tr>
<td>Error Condition</td>
<td>Exit Mode Selected</td>
<td>Exit Mode Not Selected</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------</td>
<td>------------------------</td>
</tr>
</tbody>
</table>
| Exit condition bit 48 set on CMU instruction. | 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. | 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. |
| Exit condition bit 48 set by a UEM address range check for instructions 011 and 012. | 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. | 1. Execute instruction as a pass.  
2. Exit to next 60-bit word and continue execution. |
| Exit condition bit 48 set by a UEM address range check for instructions 014 and 015. | 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. | 1. Execute instruction as a pass.  
2. Exit to next parcel and continue execution. |
| Exit condition bit 49 set by infinite condition, or bit 50 set by indefinite condition. | 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. | 1. Continue execution. |
| Any hardware parity error or double SECDED error. | 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. | 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 3777777g. 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 0007777777g. 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 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 77g.

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 7777g.

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 77g, 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 7777, the result is set to 0000, except as follows: adding 7777 plus 7777 equals 7777. In general, adding 0000 or 7777 leaves the other number unchanged, except when the other number is also 0000 or 7777.

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 77g; this number is used as a 12-bit address to access PP memory locations 0 through 7777g.

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:

<table>
<thead>
<tr>
<th>Action</th>
<th>Typical Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Verify read status.</td>
<td>Function m (77).</td>
</tr>
<tr>
<td>Prepare for reading summary status.</td>
<td>Jump if active (640).</td>
</tr>
<tr>
<td>Activate channel.</td>
<td>Activate (74).</td>
</tr>
<tr>
<td>Read summary status.</td>
<td>Input to A (70).</td>
</tr>
<tr>
<td>Verify error flag clear.</td>
<td>Jump if error flag set (661).</td>
</tr>
<tr>
<td>Analyze summary status.</td>
<td>Logical product (12). Nonzero jump (04).</td>
</tr>
</tbody>
</table>

4. Enter number of words to A.

5. Prepare for input/output.

6. Read/write data.

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:

<table>
<thead>
<tr>
<th>Slot</th>
<th>PP Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0, 5, 20, 25</td>
</tr>
<tr>
<td>2</td>
<td>1, 6, 21, 26</td>
</tr>
<tr>
<td>3</td>
<td>2, 7, 22, 27</td>
</tr>
<tr>
<td>4</td>
<td>3, 10, 23, 30</td>
</tr>
<tr>
<td>5</td>
<td>4, 11, 24, 31</td>
</tr>
</tbody>
</table>

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.
Figure 5-7. Channel Transfer Timing

NOTES:

1. All transmission pulse widths (including data, full, empty, etc.) are 25±5 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±35 ns to maintain the 500 ns cycle time.
**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 functions 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. Coincidently, 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.
9. TIME IS A FUNCTION OF ED. FULL SHOULD PROCEED THE DATA BY A MINIMUM OF 5 NS (15 NS MAXIMUM) TO REMOVE THE CLEAR ON THE INPUT DATA RECEIVERS.
10. PP MAY DISCONNECT AFTER EMPTY SIGNAL OF ANY ED WORD. STATUS REQUEST DISCONNECTS IN THIS MANNER.

A. CHANNEL MUST BE PREVIOUSLY INACTIVE.
B. CHANNEL REMAINS ACTIVE UNTIL ED SENDS INACTIVE.
C. CHANNEL MUST BE PREVIOUSLY INACTIVE.
INSTRUCTIONS 76 AND 77

PP INSTRUCTIONS 72 AND 73

PP INSTRUCTION 75

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 TIME IS 50 NS.

A. CHANNEL MUST BE PREVIOUSLY INACTIVE.

B. CHANNEL REMAINS ACTIVE UNTIL ED SENDS INACTIVE.

C. CHANNEL MUST BE PREVIOUSLY INACTIVE.

Figure 5-9. Data Output Sequence Timing
DISPLAY STATION PROGRAMMING

KEYBOARD

A PP transmits function code 7020g 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>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>No data</td>
<td>00</td>
</tr>
<tr>
<td>A</td>
<td>01</td>
</tr>
<tr>
<td>B</td>
<td>02</td>
</tr>
<tr>
<td>C</td>
<td>03</td>
</tr>
<tr>
<td>D</td>
<td>04</td>
</tr>
<tr>
<td>E</td>
<td>05</td>
</tr>
<tr>
<td>F</td>
<td>06</td>
</tr>
<tr>
<td>G</td>
<td>07</td>
</tr>
<tr>
<td>H</td>
<td>10</td>
</tr>
<tr>
<td>I</td>
<td>11</td>
</tr>
<tr>
<td>J</td>
<td>12</td>
</tr>
<tr>
<td>K</td>
<td>13</td>
</tr>
<tr>
<td>L</td>
<td>14</td>
</tr>
<tr>
<td>M</td>
<td>15</td>
</tr>
<tr>
<td>N</td>
<td>16</td>
</tr>
<tr>
<td>O</td>
<td>17</td>
</tr>
<tr>
<td>P</td>
<td>20</td>
</tr>
<tr>
<td>Q</td>
<td>21</td>
</tr>
<tr>
<td>R</td>
<td>22</td>
</tr>
<tr>
<td>S</td>
<td>23</td>
</tr>
<tr>
<td>T</td>
<td>24</td>
</tr>
<tr>
<td>U</td>
<td>25</td>
</tr>
<tr>
<td>V</td>
<td>26</td>
</tr>
<tr>
<td>W</td>
<td>27</td>
</tr>
<tr>
<td>X</td>
<td>30</td>
</tr>
<tr>
<td>Y</td>
<td>31</td>
</tr>
<tr>
<td>Z</td>
<td>32</td>
</tr>
</tbody>
</table>

Table 5-9. Keyboard Character Codes

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>00</td>
</tr>
<tr>
<td>A</td>
<td>01</td>
</tr>
<tr>
<td>B</td>
<td>02</td>
</tr>
<tr>
<td>C</td>
<td>03</td>
</tr>
<tr>
<td>D</td>
<td>04</td>
</tr>
<tr>
<td>E</td>
<td>05</td>
</tr>
<tr>
<td>F</td>
<td>06</td>
</tr>
<tr>
<td>G</td>
<td>07</td>
</tr>
<tr>
<td>H</td>
<td>10</td>
</tr>
<tr>
<td>I</td>
<td>11</td>
</tr>
<tr>
<td>J</td>
<td>12</td>
</tr>
<tr>
<td>K</td>
<td>13</td>
</tr>
<tr>
<td>L</td>
<td>14</td>
</tr>
<tr>
<td>M</td>
<td>15</td>
</tr>
<tr>
<td>N</td>
<td>16</td>
</tr>
<tr>
<td>O</td>
<td>17</td>
</tr>
<tr>
<td>P</td>
<td>20</td>
</tr>
<tr>
<td>Q</td>
<td>21</td>
</tr>
<tr>
<td>R</td>
<td>22</td>
</tr>
<tr>
<td>S</td>
<td>23</td>
</tr>
<tr>
<td>T</td>
<td>24</td>
</tr>
<tr>
<td>U</td>
<td>25</td>
</tr>
<tr>
<td>V</td>
<td>26</td>
</tr>
<tr>
<td>W</td>
<td>27</td>
</tr>
<tr>
<td>X</td>
<td>30</td>
</tr>
<tr>
<td>Y</td>
<td>31</td>
</tr>
<tr>
<td>Z</td>
<td>32</td>
</tr>
</tbody>
</table>

Table 5-10. Display Character Codes

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>00</td>
</tr>
<tr>
<td>A</td>
<td>01</td>
</tr>
<tr>
<td>B</td>
<td>02</td>
</tr>
<tr>
<td>C</td>
<td>03</td>
</tr>
<tr>
<td>D</td>
<td>04</td>
</tr>
<tr>
<td>E</td>
<td>05</td>
</tr>
<tr>
<td>F</td>
<td>06</td>
</tr>
<tr>
<td>G</td>
<td>07</td>
</tr>
<tr>
<td>H</td>
<td>10</td>
</tr>
<tr>
<td>I</td>
<td>11</td>
</tr>
<tr>
<td>J</td>
<td>12</td>
</tr>
<tr>
<td>K</td>
<td>13</td>
</tr>
<tr>
<td>L</td>
<td>14</td>
</tr>
<tr>
<td>M</td>
<td>15</td>
</tr>
<tr>
<td>N</td>
<td>16</td>
</tr>
<tr>
<td>O</td>
<td>17</td>
</tr>
<tr>
<td>P</td>
<td>20</td>
</tr>
<tr>
<td>Q</td>
<td>21</td>
</tr>
<tr>
<td>R</td>
<td>22</td>
</tr>
<tr>
<td>S</td>
<td>23</td>
</tr>
<tr>
<td>T</td>
<td>24</td>
</tr>
<tr>
<td>U</td>
<td>25</td>
</tr>
<tr>
<td>V</td>
<td>26</td>
</tr>
<tr>
<td>W</td>
<td>27</td>
</tr>
<tr>
<td>X</td>
<td>30</td>
</tr>
<tr>
<td>Y</td>
<td>31</td>
</tr>
<tr>
<td>Z</td>
<td>32</td>
</tr>
</tbody>
</table>

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.

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.
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.

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.

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.

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 14g 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 4095.

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 15g 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.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>52-58</td>
<td>Not used</td>
</tr>
<tr>
<td>59</td>
<td>Output buffer not full</td>
</tr>
<tr>
<td>60</td>
<td>Input ready</td>
</tr>
<tr>
<td>61</td>
<td>Data carrier detect or carrier on</td>
</tr>
<tr>
<td>62</td>
<td>Data set ready</td>
</tr>
<tr>
<td>63</td>
<td>Ring indication</td>
</tr>
</tbody>
</table>

PP Read Terminal Data (01XX)

This code permits the PP to input the terminal data from the selected terminal. Channel 15g must be activated and a one-word input must follow to read in the terminal data. The data word is 12 bits.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
<td>Data set ready</td>
</tr>
<tr>
<td>53</td>
<td>Data set ready and data carrier detector</td>
</tr>
<tr>
<td>54</td>
<td>Over run</td>
</tr>
<tr>
<td>55</td>
<td>Framing or parity error</td>
</tr>
<tr>
<td>56-63</td>
<td>8-bit data</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>58</td>
<td>Not used</td>
</tr>
<tr>
<td>59</td>
<td>No parity</td>
</tr>
</tbody>
</table>

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.

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

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.

NOTE

Maintenance registers are numbered 0 through 63 from left to right.
Table 5.11. MCH Function Word Bit Assignments (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MCH Function Word to Model 835 CP</strong></td>
<td></td>
</tr>
<tr>
<td>CONNECT (bits 8-11)</td>
<td>Code: 2₁₆ = Connect CP maintenance registers</td>
</tr>
<tr>
<td>OPCODE (bits 4-7)</td>
<td>Code: 0₁₆ = Halt processor, 1₁₆ = Start processor, 4₁₆ = Prepare for read (control word required), 5₁₆ = Prepare for write (control word required), 6₁₆ = Master clear, 7₁₆ = Clear errors</td>
</tr>
<tr>
<td>TYPE (bits 0-3)</td>
<td>Code: 0₁₆ = Control word required</td>
</tr>
<tr>
<td><strong>MCH Function Word to Model 835 CM</strong></td>
<td></td>
</tr>
<tr>
<td>CONNECT (bits 8-11)</td>
<td>Code: 1₁₆ = Connect CM maintenance registers</td>
</tr>
<tr>
<td>OPCODE (bits 4-7)</td>
<td>Code: 4₁₆ = Prepare for read (control word required), 5₁₆ = Prepare for write (control word required), 6₁₆ = Master clear, 7₁₆ = Clear fault status register</td>
</tr>
<tr>
<td><strong>MCH Function Word to Model 855 CP and CM</strong></td>
<td></td>
</tr>
<tr>
<td>CONNECT (bits 8-11)</td>
<td>Code: 1₁₆ = Required for model 855 CP and CM</td>
</tr>
<tr>
<td>TYPE (bits 0-3)</td>
<td>Code: 0₁₆ = CP and CP registers</td>
</tr>
<tr>
<td>OPCODE (bits 4-7)</td>
<td>Code: 0₁₆ = Halt processor, 1₁₆ = Start processor, 4₁₆ = Prepare for read, 5₁₆ = Prepare for write, 6₁₆ = Master clear, 7₁₆ = Clear errors</td>
</tr>
<tr>
<td>TYPE (bits 0-3)</td>
<td>Code: 1₁₆ = Control store memory</td>
</tr>
<tr>
<td>OPCODE (bits 4-7)</td>
<td>Code: 4₁₆ = Prepare for read, 5₁₆ = Prepare for write</td>
</tr>
<tr>
<td>TYPE (bits 0-3)</td>
<td>Code: 3-7₁₆ = Internal memories</td>
</tr>
<tr>
<td>OPCODE (bits 4-7)</td>
<td>Code: 4₁₆ = Prepare for read, 5₁₆ = Prepare for write</td>
</tr>
<tr>
<td>TYPE (bits 0-3)</td>
<td>Code: 4₁₆ = CM and CM registers</td>
</tr>
<tr>
<td>OPCODE (bits 4-7)</td>
<td>Code: 4₁₆ = Prepare for read, 5₁₆ = Prepare for write, 6₁₆ = Master clear, 7₁₆ = Clear errors</td>
</tr>
</tbody>
</table>
Table 5-11. MCH Function Word Bit Assignments (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CONNECT (bits 8-11)</strong></td>
<td>Code 0_16 = Connect IOU maintenance registers</td>
</tr>
<tr>
<td><strong>OPCODE (bits 4-7)</strong></td>
<td>Code 4_16 = Prepare for read (control word required)</td>
</tr>
<tr>
<td></td>
<td>5_16 = Prepare for write (control word required)</td>
</tr>
<tr>
<td></td>
<td>6_16 = Master clear</td>
</tr>
<tr>
<td></td>
<td>7_16 = Clear fault status registers</td>
</tr>
<tr>
<td><strong>TYPE (bits 0-3)</strong></td>
<td>Code 0-7_16 = IOU registers are read circularly (byte 0 follows byte 7) from the byte specified by the TYPE field</td>
</tr>
</tbody>
</table>

**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.

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 clears immediately and 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.

**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**

<table>
<thead>
<tr>
<th>Internal Address (1)</th>
<th>Type (2) (3)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex</td>
<td>Octal</td>
<td>Description</td>
</tr>
<tr>
<td>00</td>
<td>000</td>
<td>R A</td>
</tr>
<tr>
<td>10</td>
<td>020</td>
<td>R A</td>
</tr>
<tr>
<td>30</td>
<td>060</td>
<td>R A</td>
</tr>
<tr>
<td>42</td>
<td>082</td>
<td>R M</td>
</tr>
<tr>
<td>80</td>
<td>200</td>
<td>R A</td>
</tr>
<tr>
<td>81</td>
<td>201</td>
<td>R A</td>
</tr>
<tr>
<td>90</td>
<td>220</td>
<td>R A</td>
</tr>
<tr>
<td>92</td>
<td>222</td>
<td>R A</td>
</tr>
<tr>
<td>93</td>
<td>223</td>
<td>R A</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Internal Address (1)</th>
<th>Type (2)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex</td>
<td>Octal</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>000</td>
<td>R</td>
</tr>
<tr>
<td>10</td>
<td>020</td>
<td>R</td>
</tr>
<tr>
<td>12</td>
<td>022</td>
<td>R</td>
</tr>
<tr>
<td>A0</td>
<td>240</td>
<td>R/W</td>
</tr>
<tr>
<td>A4</td>
<td>244</td>
<td>R/W</td>
</tr>
<tr>
<td>A8</td>
<td>250</td>
<td>R/W</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Internal Address (1)</th>
<th>Type (2) (3)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex</td>
<td>Octal</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>000</td>
<td>R A</td>
</tr>
<tr>
<td>10</td>
<td>020</td>
<td>R A</td>
</tr>
<tr>
<td>30</td>
<td>060</td>
<td>R A</td>
</tr>
<tr>
<td>42</td>
<td>082</td>
<td>R M</td>
</tr>
<tr>
<td>80-89</td>
<td>200-211</td>
<td>R A</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Internal Address (1)</th>
<th>Type (2)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>R</td>
<td>Status summary register</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>Element identifier register</td>
</tr>
<tr>
<td>12</td>
<td>R</td>
<td>Options installed register</td>
</tr>
<tr>
<td>A0</td>
<td>R/W</td>
<td>Corrected error log register</td>
</tr>
<tr>
<td>A4</td>
<td>R/W</td>
<td>Uncorrected error log 1 register</td>
</tr>
<tr>
<td>A8</td>
<td>R/W</td>
<td>Uncorrected error log 2 register</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Internal Address (1)</th>
<th>Type (2)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>R</td>
<td>Status summary register</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>Element identifier register</td>
</tr>
<tr>
<td>12</td>
<td>R</td>
<td>Options installed register</td>
</tr>
<tr>
<td>18</td>
<td>R/W</td>
<td>Fault status mask register</td>
</tr>
<tr>
<td>40</td>
<td>R</td>
<td>Status register</td>
</tr>
<tr>
<td>80</td>
<td>R/W</td>
<td>Fault status 1 register</td>
</tr>
<tr>
<td>81</td>
<td>R/W</td>
<td>Fault status 2 register</td>
</tr>
</tbody>
</table>

**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)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td>Short warning, imminent power failure</td>
</tr>
<tr>
<td>60</td>
<td>CP halted</td>
</tr>
<tr>
<td>61</td>
<td>Uncorrected error</td>
</tr>
<tr>
<td>62</td>
<td>Corrected error</td>
</tr>
<tr>
<td>63</td>
<td>Long warning</td>
</tr>
</tbody>
</table>

- **CP Status Summary Register (MCH Address 0016)**
  - 59: Short warning, imminent power failure
  - 60: CP halted
  - 61: Uncorrected error
  - 62: Corrected error
  - 63: Long warning

- **CP Element Identifier Register (MCH Address 1016)**
  - 32-39: Element type (0016 = CP)
  - 40-47: Element model (2016 = model 835)
  - 48-63: Element serial number

- **CP Dependent Environment Control Register (MCH Address 3016)**
  - 08: Cache memory 1st quarter enabled (0-1 kiloword)
  - 09: Cache memory 2nd quarter enabled (0-1 kiloword)
  - 10: Cache memory 3rd quarter enabled (0-1 kiloword)
  - 11: Cache memory 4th quarter enabled (0-1 kiloword)
  - 12: Cache memory conflict register 0 enabled
  - 13: Cache memory conflict register 1 enabled
  - 14: Cache memory conflict register 2 enabled
  - 15: Cache memory conflict register 3 enabled

- **CP Monitor Condition Register (MCH Address 4216)**
  - 48: Uncorrectable error
  - 50: Short warning

- **CP Processor Fault Status Registers (MCH Address 8016, 8116)**
  - **CP Retry Corrected Error Log Register (MCH Address 9016)**

  - 0: Address and data cache input 0
  - 1: Address and data cache input 1
  - 2: Address and data cache input 2
  - 3: Address and data cache input 3
  - 4: Cache output 0
  - 5: Cache output 1
  - 6: Cache output 2
  - 7: Cache output 3
  - 8: Data port 0-0
  - 9: Data port 0-1
  - 10: Data port 0-2
  - 11: Data port 0-3
  - 12: Data port 1-0
  - 13: Data port 1-1
  - 14: Data port 1-2
  - 15: Data port 1-3
  - 16: Indent, port 0
  - 17: Indent, port 1
  - 18: Indent, partial write, cache input
  - 19: Indent, cache output
  - 20: Response code = 1 error
  - 21: Response code = 5 error
  - 22: Response code = 7 error
  - 23: Cache time outs
  - 24: No overload on simulated response buffer
  - 25: 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)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>Increment indent, cache input</td>
</tr>
<tr>
<td>31</td>
<td>Maintenance access control ROMs</td>
</tr>
<tr>
<td>32</td>
<td>Register file byte 0</td>
</tr>
<tr>
<td>33</td>
<td>Register file byte 1</td>
</tr>
<tr>
<td>34</td>
<td>Register file byte 2</td>
</tr>
<tr>
<td>35</td>
<td>Register file byte 3</td>
</tr>
<tr>
<td>36</td>
<td>Register file byte 4</td>
</tr>
<tr>
<td>37</td>
<td>Register file byte 5</td>
</tr>
<tr>
<td>38</td>
<td>Register file byte 6</td>
</tr>
<tr>
<td>39</td>
<td>Register file byte 7</td>
</tr>
<tr>
<td>40</td>
<td>I mux, B mux, address</td>
</tr>
<tr>
<td>41</td>
<td>I mux, B mux, address</td>
</tr>
<tr>
<td>42</td>
<td>I mux, B mux, address</td>
</tr>
<tr>
<td>43</td>
<td>I mux, B mux, address</td>
</tr>
<tr>
<td>44</td>
<td>I mux, B mux, address</td>
</tr>
<tr>
<td>48</td>
<td>Invalidation address, exchange address</td>
</tr>
<tr>
<td>50</td>
<td>J stream</td>
</tr>
<tr>
<td>51</td>
<td>K stream</td>
</tr>
<tr>
<td>52</td>
<td>Byte output</td>
</tr>
<tr>
<td>53</td>
<td>Byte control</td>
</tr>
<tr>
<td>54</td>
<td>Byte branch ROM</td>
</tr>
<tr>
<td>55</td>
<td>Floating point trap ROM</td>
</tr>
<tr>
<td>56</td>
<td>Exponent adder</td>
</tr>
<tr>
<td>57</td>
<td>ROM and partial writes</td>
</tr>
<tr>
<td>58</td>
<td>A data selection ROMs</td>
</tr>
<tr>
<td>59</td>
<td>Ident from cache</td>
</tr>
<tr>
<td>60</td>
<td>Immediate control ROMs</td>
</tr>
<tr>
<td>61</td>
<td></td>
</tr>
</tbody>
</table>

CP Processor Fault Status Registers (MCH Address 8016, 8116)
CP Retry Corrected Error Log Register (MCH Address 9016) (Contd)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Parity of micrand bytes 0/8</td>
</tr>
<tr>
<td>8</td>
<td>Parity of micrand bytes 1/9</td>
</tr>
<tr>
<td>16</td>
<td>Parity of micrand bytes 2/10</td>
</tr>
<tr>
<td>24</td>
<td>Parity of micrand bytes 3/11</td>
</tr>
<tr>
<td>32</td>
<td>Parity of micrand bytes 4/12</td>
</tr>
<tr>
<td>40</td>
<td>Parity of micrand bytes 5/13</td>
</tr>
<tr>
<td>48</td>
<td>Parity of micrand bytes 6/14</td>
</tr>
<tr>
<td>56</td>
<td>Parity of micrand bytes 7/15</td>
</tr>
</tbody>
</table>

CP Control Store Parity Fault Status Register (MCH Address 8116)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This log contains valid information</td>
</tr>
<tr>
<td>1</td>
<td>Unlogged error</td>
</tr>
<tr>
<td>2</td>
<td>Not used</td>
</tr>
<tr>
<td>3</td>
<td>Parity error, LRU status array counters</td>
</tr>
<tr>
<td>4-6</td>
<td>Not used</td>
</tr>
<tr>
<td>7</td>
<td>No multiple hit in tag arrays</td>
</tr>
<tr>
<td>8</td>
<td>No parity errors, tag array blocks 0 and 1</td>
</tr>
<tr>
<td>9</td>
<td>No parity errors, tag array blocks 2 and 3</td>
</tr>
<tr>
<td>10</td>
<td>No parity errors, tag array blocks 4 and 5</td>
</tr>
<tr>
<td>11</td>
<td>No parity errors, tag array blocks 6 and 7</td>
</tr>
<tr>
<td>12</td>
<td>No parity error, tag array blocks 0 and 1</td>
</tr>
<tr>
<td>13</td>
<td>No parity error, tag array blocks 2 and 3</td>
</tr>
<tr>
<td>14</td>
<td>No parity error, tag array blocks 4 and 5</td>
</tr>
<tr>
<td>15</td>
<td>No parity error, tag array blocks 6 and 7</td>
</tr>
</tbody>
</table>

CP Cache Corrected Error Log Register (MCH Address 9216)

†Register bits are numbered 0 through 63 from left to right.
Table 5-17. Model 835 CP Register Bit Assignments† (Sheet 3 of 3)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CP Cache Corrected Error Log Register (MCH Address 9216) (Contd)</td>
</tr>
<tr>
<td>16</td>
<td>Compare bits block 0</td>
</tr>
<tr>
<td>17</td>
<td>Compare bits block 1</td>
</tr>
<tr>
<td>18</td>
<td>Compare bits block 2</td>
</tr>
<tr>
<td>19</td>
<td>Compare bits block 3</td>
</tr>
<tr>
<td>20</td>
<td>Compare bits block 4</td>
</tr>
<tr>
<td>21</td>
<td>Compare bits block 5</td>
</tr>
<tr>
<td>22</td>
<td>Compare bits block 6</td>
</tr>
<tr>
<td>23</td>
<td>Compare bits block 7</td>
</tr>
<tr>
<td>24</td>
<td>Compare bits block 0</td>
</tr>
<tr>
<td>25</td>
<td>Compare bits block 1</td>
</tr>
<tr>
<td>26</td>
<td>Compare bits block 2</td>
</tr>
<tr>
<td>27</td>
<td>Compare bits block 3</td>
</tr>
<tr>
<td>28</td>
<td>Compare bits block 4</td>
</tr>
<tr>
<td>29</td>
<td>Compare bits block 5</td>
</tr>
<tr>
<td>30</td>
<td>Compare bits block 6</td>
</tr>
<tr>
<td>31</td>
<td>Compare bits block 7</td>
</tr>
<tr>
<td>32</td>
<td>Reserved</td>
</tr>
<tr>
<td>33</td>
<td>Valid map corrected error log (CEL) entry</td>
</tr>
<tr>
<td>34-36</td>
<td>Unlogged error</td>
</tr>
<tr>
<td>37</td>
<td>Not used</td>
</tr>
<tr>
<td>38</td>
<td>Parity error, segment associative section</td>
</tr>
<tr>
<td>39</td>
<td>Multiple hit</td>
</tr>
<tr>
<td>40-63</td>
<td>Multiple hit</td>
</tr>
<tr>
<td></td>
<td>Parity group fault identifier</td>
</tr>
</tbody>
</table>

†Register bits are numbered 0 through 63 from left to right.
### Table 5-18. Model 835 CM Register Bit Assignments† (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>56-57</td>
<td>Oscillator code (00 = normal, 01 = -2%, 10 = +2%)</td>
</tr>
<tr>
<td>58</td>
<td>Clock set to tuning mode</td>
</tr>
<tr>
<td>61</td>
<td>Valid uncorrectable error</td>
</tr>
<tr>
<td>62</td>
<td>Corrected error</td>
</tr>
<tr>
<td>63</td>
<td>Long warning (low/high temperature, condensing unit fault, blower 1/blower 2 fault)</td>
</tr>
<tr>
<td>32-39</td>
<td>Element type (0116 = CM)</td>
</tr>
<tr>
<td>40-47</td>
<td>Element model (2016 = model 835)</td>
</tr>
<tr>
<td>48-63</td>
<td>Element serial number</td>
</tr>
<tr>
<td>00</td>
<td>524K memory installed</td>
</tr>
<tr>
<td>07</td>
<td>1049K memory installed</td>
</tr>
<tr>
<td>09</td>
<td>1573K memory installed</td>
</tr>
<tr>
<td>11</td>
<td>2097K memory installed</td>
</tr>
<tr>
<td>16</td>
<td>A memory configuration switch is in the up position</td>
</tr>
<tr>
<td>19</td>
<td>Top switch (SW3) is not in the center position</td>
</tr>
<tr>
<td>20</td>
<td>Next to top switch (SW4) is not in the center position</td>
</tr>
<tr>
<td>21</td>
<td>Next to bottom switch (SW5) is not in the center position</td>
</tr>
<tr>
<td>22</td>
<td>Bottom switch (SW6) is not in the center position</td>
</tr>
<tr>
<td>00</td>
<td>This log contains valid information</td>
</tr>
<tr>
<td>01</td>
<td>Unlogged single-bit error</td>
</tr>
<tr>
<td>08-09</td>
<td>Port code (0/1 = port 0/1, 4/5 = not used, 6 = no request, 7 = refresh)</td>
</tr>
<tr>
<td>13-14</td>
<td>Array PAK select address</td>
</tr>
<tr>
<td>15-16</td>
<td>Chip row select address</td>
</tr>
<tr>
<td>17-23</td>
<td>Chip column address</td>
</tr>
<tr>
<td>24-30</td>
<td>Row address</td>
</tr>
<tr>
<td>31-33</td>
<td>Bank address</td>
</tr>
<tr>
<td>35</td>
<td>Address parity P5</td>
</tr>
<tr>
<td>36</td>
<td>Address parity P6</td>
</tr>
<tr>
<td>37</td>
<td>Address parity P7</td>
</tr>
<tr>
<td>42-49</td>
<td>Syndrome bits</td>
</tr>
</tbody>
</table>

**NOTE:**

The byte is duplicated in each byte.

**CM Options Installed Register (MCH Address 1216)**

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>03</td>
<td>524K memory installed</td>
</tr>
<tr>
<td>07</td>
<td>1049K memory installed</td>
</tr>
<tr>
<td>09</td>
<td>1573K memory installed</td>
</tr>
<tr>
<td>11</td>
<td>2097K memory installed</td>
</tr>
<tr>
<td>16</td>
<td>A memory configuration switch is in the up position</td>
</tr>
<tr>
<td>19</td>
<td>Top switch (SW3) is not in the center position</td>
</tr>
<tr>
<td>20</td>
<td>Next to top switch (SW4) is not in the center position</td>
</tr>
<tr>
<td>21</td>
<td>Next to bottom switch (SW5) is not in the center position</td>
</tr>
<tr>
<td>22</td>
<td>Bottom switch (SW6) is not in the center position</td>
</tr>
<tr>
<td>00</td>
<td>This log contains valid information</td>
</tr>
<tr>
<td>01</td>
<td>Unlogged uncorrected error</td>
</tr>
<tr>
<td>02</td>
<td>Illegal function</td>
</tr>
<tr>
<td>03</td>
<td>Double-bit error</td>
</tr>
<tr>
<td>04</td>
<td>Bounds fault</td>
</tr>
<tr>
<td>05</td>
<td>First level parity error</td>
</tr>
</tbody>
</table>

†Register bits are numbered 0 through 63 from left to right.
Table 5-18. Model 835 CM Register Bit Assignments† (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>06</td>
<td>Second level parity error</td>
</tr>
<tr>
<td>07</td>
<td>Buffer request</td>
</tr>
<tr>
<td>08-10</td>
<td>Port code (0/1/2/3 = port, 4/5 = not used, 6 = no request, 7 = refresh)</td>
</tr>
<tr>
<td>13-14</td>
<td>Array PAK select address</td>
</tr>
<tr>
<td>15-16</td>
<td>Chip row select address</td>
</tr>
<tr>
<td>17-23</td>
<td>Chip column address</td>
</tr>
<tr>
<td>24-30</td>
<td>Chip row address</td>
</tr>
<tr>
<td>31-33</td>
<td>Bank address</td>
</tr>
<tr>
<td>35</td>
<td>Address parity P5</td>
</tr>
<tr>
<td>36</td>
<td>Address parity P6</td>
</tr>
<tr>
<td>37</td>
<td>Address parity P7</td>
</tr>
<tr>
<td>38-41</td>
<td>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)</td>
</tr>
<tr>
<td>50-57</td>
<td>Mark bits associated with error word</td>
</tr>
<tr>
<td>58</td>
<td>Mark parity</td>
</tr>
<tr>
<td>59-62</td>
<td>Function bits associated with error word</td>
</tr>
<tr>
<td>63</td>
<td>Function parity</td>
</tr>
</tbody>
</table>

CM Uncorrected Error Log 2 Register (MCH Address A816)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>This log contains valid information</td>
</tr>
<tr>
<td>01</td>
<td>Unlogged uncorrected error 2</td>
</tr>
<tr>
<td>02</td>
<td>CM partial write parity error</td>
</tr>
<tr>
<td>03</td>
<td>Data read parity error</td>
</tr>
<tr>
<td>08-10</td>
<td>Port code (0/1/2/3 = port 0/1/2/3, 4/5 = not used, 6 = no request, 7 = refresh)</td>
</tr>
<tr>
<td>13-14</td>
<td>PAK select address</td>
</tr>
<tr>
<td>15-16</td>
<td>Chip row select address</td>
</tr>
<tr>
<td>17-23</td>
<td>Chip column address</td>
</tr>
<tr>
<td>24-30</td>
<td>Chip row address</td>
</tr>
<tr>
<td>31-33</td>
<td>Bank address</td>
</tr>
<tr>
<td>35</td>
<td>Address parity P5</td>
</tr>
<tr>
<td>36</td>
<td>Address parity P6</td>
</tr>
<tr>
<td>37</td>
<td>Address parity P7</td>
</tr>
<tr>
<td>38</td>
<td>Byte 0 read PE (1)</td>
</tr>
<tr>
<td>39</td>
<td>Byte 1 read PE (1)</td>
</tr>
<tr>
<td>40</td>
<td>Byte 2 read PE (1)</td>
</tr>
<tr>
<td>41</td>
<td>Byte 3 read PE (1)</td>
</tr>
<tr>
<td>42</td>
<td>Byte 4 read PE (1)</td>
</tr>
<tr>
<td>43</td>
<td>Byte 5 read PE (1)</td>
</tr>
<tr>
<td>44</td>
<td>Byte 6 read PE (1)</td>
</tr>
<tr>
<td>45</td>
<td>Byte 7 read PE (1)</td>
</tr>
<tr>
<td>46</td>
<td>Byte 0 PE, partial write</td>
</tr>
<tr>
<td>47</td>
<td>Byte 1 PE, partial write</td>
</tr>
<tr>
<td>48</td>
<td>Byte 2 PE, partial write</td>
</tr>
<tr>
<td>49</td>
<td>Byte 3 PE, partial write</td>
</tr>
<tr>
<td>50</td>
<td>Byte 4 PE, partial write</td>
</tr>
<tr>
<td>51</td>
<td>Byte 5 PE, partial write</td>
</tr>
<tr>
<td>52</td>
<td>Byte 6 PE, partial write</td>
</tr>
<tr>
<td>53</td>
<td>Byte 7 PE, partial write</td>
</tr>
<tr>
<td>54</td>
<td>Tag out parity error</td>
</tr>
</tbody>
</table>

**NOTE:**

1. If bit 3 is set in both uncorrected error logs and SECDED is discarded, these bits point to the bytes with errors for a read data parity error.

†Register bits are numbered 0 through 63 from left to right.
Table 5-19. Model 855 CP Register Bit Assignments† (Sheet 1 of 4)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CP Status Summary Register (MCH Address 00&lt;sub&gt;16&lt;/sub&gt;)</td>
</tr>
<tr>
<td>59</td>
<td>Short warning, imminent power failure</td>
</tr>
<tr>
<td>60</td>
<td>CP halted</td>
</tr>
<tr>
<td>61</td>
<td>Uncorrected error</td>
</tr>
<tr>
<td>62</td>
<td>Corrected error</td>
</tr>
<tr>
<td>63</td>
<td>Long warning</td>
</tr>
<tr>
<td></td>
<td>CP Element Identifier Register (MCH Address 10&lt;sub&gt;16&lt;/sub&gt;)</td>
</tr>
<tr>
<td>32-39</td>
<td>Element type (00&lt;sub&gt;16&lt;/sub&gt; = CP)</td>
</tr>
<tr>
<td>40-47</td>
<td>Element model (30&lt;sub&gt;16&lt;/sub&gt; = model 855)</td>
</tr>
<tr>
<td>48-63</td>
<td>Element serial number</td>
</tr>
<tr>
<td></td>
<td>CP Options Installed Register (MCH Address 12&lt;sub&gt;16&lt;/sub&gt;)</td>
</tr>
<tr>
<td>62</td>
<td>32K-byte cache installed</td>
</tr>
<tr>
<td></td>
<td>CP Dependent Environment Control Register (MCH Address 30&lt;sub&gt;16&lt;/sub&gt;)</td>
</tr>
<tr>
<td>33</td>
<td>Test mode</td>
</tr>
<tr>
<td>35</td>
<td>Disable corrected error to CP status summary</td>
</tr>
<tr>
<td>42</td>
<td>Control store off-line (sweep)</td>
</tr>
<tr>
<td>44</td>
<td>Enable control store breakpoint</td>
</tr>
<tr>
<td>47</td>
<td>Disable detected uncorrectable errors (processor-detected malfunctions)</td>
</tr>
<tr>
<td>48</td>
<td>Apply wide clock margins</td>
</tr>
<tr>
<td>49</td>
<td>Apply narrow clock margins</td>
</tr>
<tr>
<td>50</td>
<td>Enable cache lookahead</td>
</tr>
<tr>
<td>51</td>
<td>Disable unconditional cache lookahead</td>
</tr>
<tr>
<td>52-56</td>
<td>Error retry limit plus parity; bit 56 is parity</td>
</tr>
<tr>
<td>57-60</td>
<td>Enable cache configuration sets 0 through 3 where bit 57 is set 0, bit 58 is set 1, and so on.</td>
</tr>
<tr>
<td>61</td>
<td>Cache fake central memory</td>
</tr>
<tr>
<td></td>
<td>CP Monitor Condition Register (MCH Address 42&lt;sub&gt;16&lt;/sub&gt;)</td>
</tr>
<tr>
<td>48</td>
<td>Uncorrectable error</td>
</tr>
<tr>
<td>50</td>
<td>Short warning</td>
</tr>
<tr>
<td></td>
<td>CP Processor Fault Status Register 0 (MCH Address 80)</td>
</tr>
<tr>
<td>0</td>
<td>Uncorrected error</td>
</tr>
<tr>
<td>1</td>
<td>One of the following has occurred: corrected error, soft error, or bypass of a processor fault</td>
</tr>
<tr>
<td>2-7</td>
<td>Address fan-in parity error, bytes 0 through 5 where bit 2 indicates byte 0 and so on</td>
</tr>
<tr>
<td>8-15</td>
<td>Input data stream parity error, bytes 0 through 7 where bit 8 indicates byte 0 and so on</td>
</tr>
<tr>
<td>16-23</td>
<td>A data stream holding register parity error, bytes 0 through 7 where bit 16 indicates byte 0 and so on</td>
</tr>
<tr>
<td>24-31</td>
<td>B data stream holding register parity error, bytes 0 through 7 where bit 24 indicates byte 0 and so on</td>
</tr>
</tbody>
</table>

†Register bits are numbered 0 through 63 from left to right.
<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP Processor Fault Status Register 0 (MCH Address 80) (Contd)</td>
<td></td>
</tr>
<tr>
<td>38-39</td>
<td>Operation code parity error where bit 38 is byte 0 and bit 39 is byte 1</td>
</tr>
<tr>
<td>40-43</td>
<td>Recovery address parity error, bytes 0 through 3 where bit 40 indicates byte 0 and so on</td>
</tr>
<tr>
<td>47</td>
<td>Shift count parity error</td>
</tr>
<tr>
<td>48</td>
<td>A length counter parity error</td>
</tr>
<tr>
<td>49</td>
<td>B length counter parity error</td>
</tr>
<tr>
<td>50</td>
<td>AJ in MUX parity error</td>
</tr>
<tr>
<td>51</td>
<td>AK in MUX parity error</td>
</tr>
<tr>
<td>52</td>
<td>Left bit shift parity error</td>
</tr>
<tr>
<td>54</td>
<td>AK port input parity error</td>
</tr>
<tr>
<td>55</td>
<td>AJ port input parity error</td>
</tr>
<tr>
<td>56</td>
<td>Register file A address counter parity error</td>
</tr>
<tr>
<td>57</td>
<td>Register file B address counter parity error</td>
</tr>
<tr>
<td>58</td>
<td>Register file A data parity error</td>
</tr>
<tr>
<td>59</td>
<td>Register file B data parity error</td>
</tr>
<tr>
<td>61</td>
<td>Limit register stage 3 parity error</td>
</tr>
<tr>
<td>62</td>
<td>Stage 7 parity error</td>
</tr>
<tr>
<td>63</td>
<td>PFS PAK 0 parity error</td>
</tr>
</tbody>
</table>

| CP Processor Fault Status Register 1 (MCH Address 8116) |
| 0 | Input stream buffer address parity error |
| 1 | Output stage 2 parity error |
| 2 | RAM PAK 1 - memory A or memory B address parity error |
| 3 | RAM PAK 1 - memory A or memory B data parity error |
| 4 | Port J second data latch parity error |
| 5 | Port K second data latch parity error |
| 6 | AJ descriptor latch parity error |
| 7 | AK descriptor latch parity error |
| 8 | RAM PAK 0 - memory A address parity error |
| 9 | RAM PAK 0 - memory A data parity error |
| 10 | RAM PAK 0 - memory B address parity error |
| 11 | RAM PAK 0 - memory B data parity error |

| CP Processor Fault Status Register 2 (MCH Address 8216) |
| 0 | Scale counter parity error |
| 1 | Edit mask parity error |
| 2-7 | Cache address register parity error, bytes 0 through 5 where bit 2 indicates byte 0 and so on |
| 8-15 | Cache write data parity error, bytes 0 through 7 where bit 8 indicates byte 0 and so on |
| 16 | Cache multiple hit |
| 17 | Cache multiple allocate error |
| 20 | LM MUX pointer, direct CMC data |
| 21 | LM MUX pointer, cache read data |
| 23 | LM MUX pointer, buffered CMC data |
| 24 | Cache write data from CP parity error |
| 25 | Cache write data from CM parity error |
| 26 | Cache associative tag parity error |
| 27 | Cache mark parity error |
| 28-31 | Car error pointer parity error, bytes 0 through 3 where bit 28 indicates byte 0 and so on |
| 32-35 | LM input control parity error, bytes 0 through 3 where bit 32 indicates byte 0 and so on |
| 52 | Exchange tag parity error |
| 56 | CM corrected write |
| 57 | CM corrected read |
| 58 | 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)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td>CM uncorrected read</td>
</tr>
<tr>
<td>60</td>
<td>CM reject</td>
</tr>
<tr>
<td>61</td>
<td>CM response code parity error</td>
</tr>
<tr>
<td>62</td>
<td>CM tag parity error</td>
</tr>
<tr>
<td>63</td>
<td>PFS PAK 1 parity error</td>
</tr>
</tbody>
</table>

CP Processor Fault Status Register 2 (MCH Address 8216) (Contd)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-3</td>
<td>Cache address parity error, bytes 0 through 3 where bit 0 indicates byte 0 and so on</td>
</tr>
<tr>
<td>4-7</td>
<td>Cache tag RAM parity error, bytes 0 through 3 where bit 4 indicates byte 0 and so on</td>
</tr>
<tr>
<td>8-11</td>
<td>Cache MUX parity error, bytes 0 through 3 where bit 8 indicates byte 0 and so on</td>
</tr>
</tbody>
</table>

CP Processor Fault Status Register 3 (MCH Address 8316)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Set 0 parity error</td>
</tr>
<tr>
<td>1</td>
<td>Set 1 parity error</td>
</tr>
<tr>
<td>12-15</td>
<td>New P register parity error, byte 0 through 3 where bit 12 indicates byte 0 and so on</td>
</tr>
<tr>
<td>16-19</td>
<td>Address input parity error, bytes 0 through 3 where bit 16 indicates byte 0 and so on</td>
</tr>
<tr>
<td>20</td>
<td>Bypass register parity error</td>
</tr>
<tr>
<td>24</td>
<td>Input purge code parity error</td>
</tr>
<tr>
<td>25</td>
<td>Data type rank 32 parity error</td>
</tr>
<tr>
<td>26</td>
<td>JK rank 32 parity error</td>
</tr>
<tr>
<td>32-33</td>
<td>Write data parity error where bit 32 is 0 through 7, and bit 33 is 8 through 15</td>
</tr>
<tr>
<td>36-39</td>
<td>P register parity error</td>
</tr>
<tr>
<td>41</td>
<td>Successful entry</td>
</tr>
<tr>
<td>42</td>
<td>Deadman time-out</td>
</tr>
<tr>
<td>45</td>
<td>Retry counter parity error</td>
</tr>
<tr>
<td>46</td>
<td>Exchange halt</td>
</tr>
<tr>
<td>56</td>
<td>Write address minipipe parity error</td>
</tr>
<tr>
<td>57-58</td>
<td>Register file out parity error where bit 57 is bytes 0 through 3 and bit 58 is bytes 4 through 7</td>
</tr>
</tbody>
</table>

CP Processor Fault Status Register 4 (MCH Address 8416)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Tag parity error</td>
</tr>
</tbody>
</table>

CP Processor Fault Status Register 5 (MCH Address 8516)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>Maintenance access control (MAC) read path parity error</td>
</tr>
<tr>
<td>33</td>
<td>Maintenance channel (MCH) parity error</td>
</tr>
<tr>
<td>34</td>
<td>MAC data fanout parity error</td>
</tr>
<tr>
<td>35</td>
<td>MAC data bus parity error</td>
</tr>
<tr>
<td>36</td>
<td>RAM address MUX parity error</td>
</tr>
<tr>
<td>37</td>
<td>Address translation MUX parity error</td>
</tr>
<tr>
<td>40</td>
<td>CYBER 170 monitor or job mode odd RAM PAK 0 parity error</td>
</tr>
<tr>
<td>41</td>
<td>CYBER 170 monitor or job mode even RAM PAK 0 parity error</td>
</tr>
<tr>
<td>42</td>
<td>Executive state RAM PAK 0 parity error</td>
</tr>
<tr>
<td>48</td>
<td>CYBER 170 monitor or job mode even RAM PAK 1 parity error</td>
</tr>
<tr>
<td>49</td>
<td>CYBER 170 monitor or job mode even RAM PAK 1 parity error</td>
</tr>
<tr>
<td>50</td>
<td>Executive state RAM PAK 1 parity error</td>
</tr>
</tbody>
</table>

† Register bits are numbered 0 through 63 from left to right.
<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CP Processor Fault Status Register 7 (MCH Address 87_{16})</strong></td>
<td></td>
</tr>
<tr>
<td>0-3</td>
<td>Branch address address</td>
</tr>
<tr>
<td><strong>CP Processor Fault Status Register 8 (MCH Address 88_{16})</strong></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Soft control register parity error</td>
</tr>
<tr>
<td>15</td>
<td>Soft control register parity error</td>
</tr>
<tr>
<td>16-23</td>
<td>Multiply C register parity error, bytes 0 through 7 where bit 16 indicates byte 0 and so on</td>
</tr>
<tr>
<td>24-31</td>
<td>Multiply B register parity error, bytes 0 through 7 where bit 24 indicates byte 0 and so on</td>
</tr>
<tr>
<td>32-43</td>
<td>General network adder parity error, bytes 0 through 11 where bit 32 indicates byte 0 and so on</td>
</tr>
<tr>
<td>44-55</td>
<td>General network adder carry miscompare, bytes 0 through 11 where bit 44 indicates byte 0 and so on</td>
</tr>
<tr>
<td>56</td>
<td>Multisoft control parity error, byte 0</td>
</tr>
<tr>
<td>58-59</td>
<td>Shift parity error where bit 58 is byte 0 and bit 59 is byte 1</td>
</tr>
<tr>
<td>60-61</td>
<td>Multiply carry miscompare where bit 60 is byte 0 and bit 61 is byte 1</td>
</tr>
<tr>
<td><strong>CP Processor Fault Status Register 9 (MCH Address 89_{16})</strong></td>
<td></td>
</tr>
<tr>
<td>0-6</td>
<td>Multiply carry miscompare, bytes 2 through 8 where bit 0 indicates byte 2 and so on</td>
</tr>
</tbody>
</table>

*Register bits are numbered 0 through 63 from left to right.*
Table 5-20. Model 855 CM Register Bit Assignments† (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>56-57</td>
<td>Oscillator code (00 = normal, 01 = -2%, 10 = +2%)</td>
</tr>
<tr>
<td>58</td>
<td>Clock set to tuning mode</td>
</tr>
<tr>
<td>61</td>
<td>Valid uncorrectable error</td>
</tr>
<tr>
<td>62</td>
<td>Corrected error</td>
</tr>
<tr>
<td>63</td>
<td>Long warning (low/high temperature, condensing unit fault, blower 1/blower 2 fault)</td>
</tr>
</tbody>
</table>

**CM Status Summary Register (MCH Address 00\text{16})**

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-39</td>
<td>Element type (01\text{16} = CM)</td>
</tr>
<tr>
<td>40-47</td>
<td>Element model (30\text{16} = model 855)</td>
</tr>
<tr>
<td>48-63</td>
<td>Element serial number</td>
</tr>
</tbody>
</table>

**CM Element Identifier Register (MCH Address 10\text{16})**

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-15</td>
<td>Installed memory size in megabytes. Only one bit can be set at a time.</td>
</tr>
<tr>
<td>Bit set</td>
<td>Memory size</td>
</tr>
<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>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Switch</th>
<th>Memory size</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17-22</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CM Options Installed Register (MCH Address 12\text{16})**

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-15</td>
<td>Installed memory size in megabytes. Only one bit can be set at a time.</td>
</tr>
<tr>
<td>Bit set</td>
<td>Memory size</td>
</tr>
<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>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Switch</th>
<th>Memory size</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17-22</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CM Environment Control Register (MCH address 20\text{16})**

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disable parity checking</td>
</tr>
<tr>
<td>1</td>
<td>Disable SECDED</td>
</tr>
<tr>
<td>3-4</td>
<td>Write check/read check/read syndrome bits</td>
</tr>
</tbody>
</table>

**Value** | **Description** |
<table>
<thead>
<tr>
<th></th>
<th></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>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Timing margins</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-6</td>
<td></td>
</tr>
</tbody>
</table>

**Value** | **Description** |
<table>
<thead>
<tr>
<th></th>
<th></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>

†Register bits are numbered 0 through 63 from left to right.
Table 5-20. Model 855 CM Register Bit Assignments† (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM Environment Control Register (MCH address 2016) (Contd)</td>
<td></td>
</tr>
<tr>
<td>32-35</td>
<td>Port disable</td>
</tr>
<tr>
<td>Bit</td>
<td>Significance</td>
</tr>
<tr>
<td>32</td>
<td>Port 0; disable CPI ports</td>
</tr>
<tr>
<td>33</td>
<td>Port 1; disable IOU1 port</td>
</tr>
<tr>
<td>34</td>
<td>Port 2; disable CPI2 ports</td>
</tr>
<tr>
<td>35</td>
<td>Port 3; disable IOU2 port</td>
</tr>
<tr>
<td>38</td>
<td>Suppress corrected error reporting via ports</td>
</tr>
<tr>
<td>39</td>
<td>Disable corrected error log</td>
</tr>
</tbody>
</table>

| CM Corrected Error Log Register (MCH Address A016) |
| 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 A416) |
| 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 A816) |
| 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)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td>Summary status</td>
</tr>
<tr>
<td>60</td>
<td>Processor halt</td>
</tr>
<tr>
<td>61</td>
<td>Valid uncorrected error</td>
</tr>
<tr>
<td>63</td>
<td>Long warning</td>
</tr>
</tbody>
</table>

**NOTE:**

The byte is duplicated in each byte.

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Barrel 3</td>
</tr>
<tr>
<td>21</td>
<td>Barrel 2</td>
</tr>
<tr>
<td>22</td>
<td>Barrel 1</td>
</tr>
<tr>
<td>23</td>
<td>Barrel 0</td>
</tr>
<tr>
<td>24</td>
<td>Channel 7</td>
</tr>
<tr>
<td>25</td>
<td>Channel 6</td>
</tr>
<tr>
<td>26</td>
<td>Channel 5</td>
</tr>
<tr>
<td>27</td>
<td>Channel 4</td>
</tr>
<tr>
<td>28</td>
<td>Channel 3</td>
</tr>
<tr>
<td>29</td>
<td>Channel 2</td>
</tr>
<tr>
<td>30</td>
<td>Channel 1</td>
</tr>
<tr>
<td>31</td>
<td>Channel 0</td>
</tr>
<tr>
<td>32</td>
<td>Channel 17</td>
</tr>
<tr>
<td>33</td>
<td>Channel 15</td>
</tr>
<tr>
<td>34</td>
<td>Channel 13</td>
</tr>
<tr>
<td>35</td>
<td>Channel 12</td>
</tr>
<tr>
<td>36</td>
<td>Channel 11</td>
</tr>
<tr>
<td>37</td>
<td>Channel 10</td>
</tr>
<tr>
<td>38</td>
<td>Channel 7</td>
</tr>
<tr>
<td>39</td>
<td>Channel 6</td>
</tr>
<tr>
<td>40</td>
<td>Channel 5</td>
</tr>
<tr>
<td>41</td>
<td>Channel 4</td>
</tr>
<tr>
<td>42</td>
<td>Channel 3</td>
</tr>
<tr>
<td>43</td>
<td>Channel 2</td>
</tr>
<tr>
<td>44</td>
<td>Channel 1</td>
</tr>
<tr>
<td>45</td>
<td>Channel 0</td>
</tr>
<tr>
<td>46</td>
<td>Channel 17</td>
</tr>
<tr>
<td>47</td>
<td>Channel 15</td>
</tr>
<tr>
<td>48</td>
<td>Channel 13</td>
</tr>
<tr>
<td>49</td>
<td>Channel 12</td>
</tr>
<tr>
<td>50</td>
<td>Channel 11</td>
</tr>
<tr>
<td>51</td>
<td>Channel 10</td>
</tr>
<tr>
<td>52</td>
<td>Channel 7</td>
</tr>
<tr>
<td>53</td>
<td>Channel 6</td>
</tr>
<tr>
<td>54</td>
<td>Channel 5</td>
</tr>
<tr>
<td>55</td>
<td>Channel 4</td>
</tr>
<tr>
<td>56</td>
<td>Channel 3</td>
</tr>
<tr>
<td>57</td>
<td>Channel 2</td>
</tr>
<tr>
<td>58</td>
<td>Channel 1</td>
</tr>
<tr>
<td>59</td>
<td>Radial interface 5/6</td>
</tr>
<tr>
<td>60</td>
<td>Radial interface 3/4</td>
</tr>
<tr>
<td>61</td>
<td>Radial interface 1/2</td>
</tr>
<tr>
<td>62</td>
<td>Two-port multiplexer</td>
</tr>
<tr>
<td>63</td>
<td>Display controller</td>
</tr>
</tbody>
</table>

†Register bits are numbered 0 through 63 from left to right.
Table 5-21. IOU Register Bit Assignments† (Sheet 2 of 4)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>03</td>
<td>Barrel 0, PP 4</td>
</tr>
<tr>
<td>04</td>
<td>Barrel 0, PP 3</td>
</tr>
<tr>
<td>05</td>
<td>Barrel 0, PP 2</td>
</tr>
<tr>
<td>06</td>
<td>Barrel 0, PP 1</td>
</tr>
<tr>
<td>07</td>
<td>Barrel 0, PP 0</td>
</tr>
<tr>
<td>08</td>
<td>Barrel 0, PP 0</td>
</tr>
<tr>
<td>11</td>
<td>Barrel 1, PP 4</td>
</tr>
<tr>
<td>12</td>
<td>Barrel 1, PP 3</td>
</tr>
<tr>
<td>13</td>
<td>Barrel 1, PP 2</td>
</tr>
<tr>
<td>14</td>
<td>Barrel 1, PP 1</td>
</tr>
<tr>
<td>15</td>
<td>Barrel 1, PP 0</td>
</tr>
<tr>
<td>19</td>
<td>Barrel 2, PP 4</td>
</tr>
<tr>
<td>20</td>
<td>Barrel 2, PP 3</td>
</tr>
<tr>
<td>21</td>
<td>Barrel 2, PP 2</td>
</tr>
<tr>
<td>22</td>
<td>Barrel 2, PP 1</td>
</tr>
<tr>
<td>23</td>
<td>Barrel 2, PP 0</td>
</tr>
<tr>
<td>27</td>
<td>Barrel 3, PP 4</td>
</tr>
<tr>
<td>28</td>
<td>Barrel 3, PP 3</td>
</tr>
<tr>
<td>29</td>
<td>Barrel 3, PP 2</td>
</tr>
<tr>
<td>30</td>
<td>Barrel 3, PP 1</td>
</tr>
<tr>
<td>31</td>
<td>Barrel 3, PP 0</td>
</tr>
<tr>
<td>32</td>
<td>Channel 7</td>
</tr>
<tr>
<td>33</td>
<td>Channel 6</td>
</tr>
<tr>
<td>34</td>
<td>Channel 5</td>
</tr>
<tr>
<td>35</td>
<td>Channel 4</td>
</tr>
<tr>
<td>36</td>
<td>Channel 3</td>
</tr>
<tr>
<td>37</td>
<td>Channel 2</td>
</tr>
<tr>
<td>38</td>
<td>Channel 1</td>
</tr>
<tr>
<td>39</td>
<td>Channel 0</td>
</tr>
<tr>
<td>40</td>
<td>Channel 17</td>
</tr>
</tbody>
</table>

IOU Fault Status Mask Register (MCH Address 1816)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>03</td>
<td>Barrel 0, PP 4 error</td>
</tr>
<tr>
<td>04</td>
<td>Barrel 0, PP 3 error</td>
</tr>
<tr>
<td>05</td>
<td>Barrel 0, PP 2 error</td>
</tr>
<tr>
<td>06</td>
<td>Barrel 0, PP 1 error</td>
</tr>
<tr>
<td>07</td>
<td>Barrel 0, PP 0 error</td>
</tr>
<tr>
<td>11</td>
<td>Barrel 1, PP 4 error</td>
</tr>
<tr>
<td>12</td>
<td>Barrel 1, PP 3 error</td>
</tr>
<tr>
<td>13</td>
<td>Barrel 1, PP 2 error</td>
</tr>
<tr>
<td>14</td>
<td>Barrel 1, PP 1 error</td>
</tr>
<tr>
<td>15</td>
<td>Barrel 1, PP 0 error</td>
</tr>
</tbody>
</table>

IOU Status Register (MCH Address 4016)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>59-60</td>
<td>Barrel reconfiguration switches</td>
</tr>
<tr>
<td>61-63</td>
<td>PP reconfiguration switches</td>
</tr>
</tbody>
</table>

IOU Fault Status 1 Register (MCH Address 8016)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>03</td>
<td>Barrel 0, PP 4 error</td>
</tr>
<tr>
<td>04</td>
<td>Barrel 0, PP 3 error</td>
</tr>
<tr>
<td>05</td>
<td>Barrel 0, PP 2 error</td>
</tr>
<tr>
<td>06</td>
<td>Barrel 0, PP 1 error</td>
</tr>
<tr>
<td>07</td>
<td>Barrel 0, PP 0 error</td>
</tr>
<tr>
<td>11</td>
<td>Barrel 1, PP 4 error</td>
</tr>
<tr>
<td>12</td>
<td>Barrel 1, PP 3 error</td>
</tr>
<tr>
<td>13</td>
<td>Barrel 1, PP 2 error</td>
</tr>
<tr>
<td>14</td>
<td>Barrel 1, PP 1 error</td>
</tr>
<tr>
<td>15</td>
<td>Barrel 1, PP 0 error</td>
</tr>
</tbody>
</table>

† Register bits are numbered 0 through 63 from left to right
<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>Barrel 2, PP 4 error</td>
</tr>
<tr>
<td>20</td>
<td>Barrel 2, PP 3 error</td>
</tr>
<tr>
<td>21</td>
<td>Barrel 2, PP 2 error</td>
</tr>
<tr>
<td>22</td>
<td>Barrel 2, PP 1 error</td>
</tr>
<tr>
<td>23</td>
<td>Barrel 2, PP 0 error</td>
</tr>
<tr>
<td>27</td>
<td>Barrel 3, PP 4 error</td>
</tr>
<tr>
<td>28</td>
<td>Barrel 3, PP 3 error</td>
</tr>
<tr>
<td>29</td>
<td>Barrel 3, PP 2 error</td>
</tr>
<tr>
<td>30</td>
<td>Barrel 3, PP 1 error</td>
</tr>
<tr>
<td>31</td>
<td>Barrel 3, PP 0 error</td>
</tr>
<tr>
<td>32</td>
<td>PP error 7VD0</td>
</tr>
<tr>
<td>33</td>
<td>PP error 7VE0</td>
</tr>
<tr>
<td>34</td>
<td>Firmware error</td>
</tr>
<tr>
<td>35</td>
<td>PP memory data out error 7XF0</td>
</tr>
<tr>
<td>36</td>
<td>PP error 7VG0</td>
</tr>
<tr>
<td>37</td>
<td>12/16 conversion error 7VJ0</td>
</tr>
<tr>
<td>38</td>
<td>PP memory address error</td>
</tr>
<tr>
<td>39</td>
<td>PP memory data-in error</td>
</tr>
<tr>
<td>45</td>
<td>OS bounds violation</td>
</tr>
<tr>
<td>46</td>
<td>OS boundary address PE</td>
</tr>
<tr>
<td>47</td>
<td>ADU barrel priority ROM PE</td>
</tr>
<tr>
<td>48</td>
<td>CM read buffer error</td>
</tr>
<tr>
<td>49</td>
<td>Uncorrected CM read error</td>
</tr>
<tr>
<td>50</td>
<td>Uncorrected CM write error</td>
</tr>
<tr>
<td>51</td>
<td>CM reject</td>
</tr>
<tr>
<td>52</td>
<td>Input CM tag error</td>
</tr>
<tr>
<td>53</td>
<td>CM response code error</td>
</tr>
<tr>
<td>54</td>
<td>CM data out error</td>
</tr>
<tr>
<td>55</td>
<td>CM address out error</td>
</tr>
<tr>
<td>56</td>
<td>CM data in error byte 0</td>
</tr>
<tr>
<td>57</td>
<td>CM data in error byte 1</td>
</tr>
<tr>
<td>58</td>
<td>CM data in error byte 2</td>
</tr>
<tr>
<td>59</td>
<td>CM data in error byte 3</td>
</tr>
<tr>
<td>60</td>
<td>CM data in error byte 4</td>
</tr>
<tr>
<td>61</td>
<td>CM data in error byte 5</td>
</tr>
<tr>
<td>62</td>
<td>CM data in error byte 6</td>
</tr>
<tr>
<td>63</td>
<td>CM data in error byte 7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>Channel 7 error</td>
</tr>
<tr>
<td>33</td>
<td>Channel 6 error</td>
</tr>
<tr>
<td>34</td>
<td>Channel 5 error</td>
</tr>
<tr>
<td>35</td>
<td>Channel 4 error</td>
</tr>
<tr>
<td>36</td>
<td>Channel 3 error</td>
</tr>
<tr>
<td>37</td>
<td>Channel 2 error</td>
</tr>
<tr>
<td>38</td>
<td>Channel 1 error</td>
</tr>
<tr>
<td>39</td>
<td>Channel 0 error</td>
</tr>
<tr>
<td>40</td>
<td>Channel 17 error</td>
</tr>
<tr>
<td>41</td>
<td>Channel 16 error</td>
</tr>
<tr>
<td>42</td>
<td>Channel 15 error</td>
</tr>
<tr>
<td>43</td>
<td>Channel 14 error</td>
</tr>
<tr>
<td>44</td>
<td>Channel 13 error</td>
</tr>
<tr>
<td>45</td>
<td>Channel 12 error</td>
</tr>
<tr>
<td>46</td>
<td>Channel 11 error</td>
</tr>
<tr>
<td>47</td>
<td>Channel 10 error</td>
</tr>
<tr>
<td>48</td>
<td>Channel 27 error</td>
</tr>
<tr>
<td>49</td>
<td>Channel 26 error</td>
</tr>
<tr>
<td>50</td>
<td>Channel 25 error</td>
</tr>
<tr>
<td>51</td>
<td>Channel 24 error</td>
</tr>
<tr>
<td>52</td>
<td>Channel 23 error</td>
</tr>
</tbody>
</table>

†Register bits are numbered 0 through 63 from left to right.
Table 5-21. IOU Register Bit Assignments† (Sheet 4 of 4)

<table>
<thead>
<tr>
<th>Channel Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>Channel 22 error</td>
</tr>
<tr>
<td>54</td>
<td>Channel 21 error</td>
</tr>
<tr>
<td>55</td>
<td>Channel 20 error</td>
</tr>
<tr>
<td>57</td>
<td>Radial interface 5/6 error</td>
</tr>
<tr>
<td>58</td>
<td>Radial interface 3/4 error</td>
</tr>
<tr>
<td>59</td>
<td>Radial interface 1/2 error</td>
</tr>
<tr>
<td>60</td>
<td>Channel 33 error</td>
</tr>
<tr>
<td>61</td>
<td>Channel 32 error</td>
</tr>
<tr>
<td>62</td>
<td>Channel 31 error</td>
</tr>
<tr>
<td>63</td>
<td>Channel 30 error</td>
</tr>
</tbody>
</table>

†Register bits are numbered 0 through 63 from left to right.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADU</td>
<td>Assembly-disassembly unit</td>
<td>I/O</td>
<td>Input/output</td>
</tr>
<tr>
<td>AOR</td>
<td>Address out of range</td>
<td>IOU</td>
<td>Input/output unit</td>
</tr>
<tr>
<td>CEL</td>
<td>Corrected error log</td>
<td>MA</td>
<td>Monitor address</td>
</tr>
<tr>
<td>CIF</td>
<td>CMU interrupted flag</td>
<td>MCH</td>
<td>Maintenance channel</td>
</tr>
<tr>
<td>CM</td>
<td>Central memory</td>
<td>MF</td>
<td>Monitor flag</td>
</tr>
<tr>
<td>CMU</td>
<td>Compare/move unit</td>
<td>MOS</td>
<td>Metal oxide semiconductor</td>
</tr>
<tr>
<td>CP</td>
<td>Central processor</td>
<td>MUX</td>
<td>Multiplexer, selector</td>
</tr>
<tr>
<td>CKT</td>
<td>Cathode-ray tube</td>
<td>OS</td>
<td>Operating system</td>
</tr>
<tr>
<td>CTI</td>
<td>Common Test and Initialization</td>
<td>PE</td>
<td>Parity error</td>
</tr>
<tr>
<td>DSC</td>
<td>Display station</td>
<td>PP</td>
<td>Peripheral processor</td>
</tr>
<tr>
<td>DTR</td>
<td>Data terminal ready</td>
<td>PPM</td>
<td>Peripheral processor memory</td>
</tr>
<tr>
<td>ECC</td>
<td>Error correction code</td>
<td>RAC</td>
<td>Reference address, central memory</td>
</tr>
<tr>
<td>ECL</td>
<td>Emitter-coupled logic</td>
<td>RAE</td>
<td>Reference address, extended memory</td>
</tr>
<tr>
<td>EDS</td>
<td>Extended deadstart</td>
<td>RAM</td>
<td>Random access (read-write) memory</td>
</tr>
<tr>
<td>EIA</td>
<td>Electronic Industries Association</td>
<td>RNI</td>
<td>Read next instruction</td>
</tr>
<tr>
<td>EM, EMS</td>
<td>Exit mode selection</td>
<td>ROM</td>
<td>Read-only memory</td>
</tr>
<tr>
<td>EC</td>
<td>Exit condition code field at (RAC)</td>
<td>RTS</td>
<td>Request to send</td>
</tr>
<tr>
<td>FIFO</td>
<td>First in, first out</td>
<td>SECDED</td>
<td>Single-error correction double-error detection</td>
</tr>
<tr>
<td>FLC</td>
<td>Field length, central memory</td>
<td>UART</td>
<td>Universal Asynchronous Receiver and Transmitter</td>
</tr>
<tr>
<td>FLE</td>
<td>Field length, extended memory</td>
<td>UEM</td>
<td>Unified extended memory</td>
</tr>
<tr>
<td>HIVS</td>
<td>Hardware Initialization and Verification Software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ILH</td>
<td>Instruction lookahead hardware</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
Bound register 1-5; 2-7, 8
Branch destination address 4-4
Branch instructions 2-4
  Define 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 4-3
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
  Bound 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 CP
  Field length register, see PLC register
  Functional characteristics, model 835 1-2
  Functional characteristics, model 855 1-3
  Functional description 2-7
  Layout 2-8

60469290 A
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

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 5-15
Internal interface 1-5; 2-13
Operation 5-16
Parity 5-15
Reservation 5-15

I/O transfers 5-16,18
ILHR, 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
Linked 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

Index-3
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 4-2

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 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, 5-12
- Central exchange jump 4-4
- Compare collapsed 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

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 1-5; 2-13
- 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 PP's
PP's 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
MAG, 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 designer 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 3-13
  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 designer 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 5-31
  IOU 5-42

60469290 A
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 FFs
PPS 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 designers 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
PFO 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
Reference address, see RAC and RAE registers
Support 1-4; 2-5
X 1-4; 2-4; 4-19, 20

Support registers 1-4; 2-5

S register, see Status register
SECDED 2-8
SECDED code bits
Description 2-8
Model 835 2-9
Model 855 2-10
SECDED 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 SECDED
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
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