Programmers Reference Manual
Part Two
860-7319-000

by FPS Technical Publications Staff
### Table of Contents

**AP-120B Instructions**

**Unconditional Fields**

Each of the following fields may be used in any given instruction word.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>B</th>
<th>SOP</th>
<th>SOP1</th>
<th>SH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octal Code</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>NOP</td>
<td>SOP1</td>
<td>E-23</td>
<td>NOP</td>
</tr>
<tr>
<td>1</td>
<td>&amp;</td>
<td>SPEC</td>
<td>E-13</td>
<td>L</td>
</tr>
<tr>
<td>2</td>
<td>ADD</td>
<td>E-17</td>
<td></td>
<td>RR</td>
</tr>
<tr>
<td>3</td>
<td>SUB</td>
<td>E-18</td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>4</td>
<td>MOV</td>
<td>E-19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>AND</td>
<td>E-20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>OR</td>
<td>E-21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>EQV</td>
<td>E-22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>SPS (S-Pad Source Reg.)</th>
<th>SPD (S-Pad Dest. Reg.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octal Code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 to 17</td>
<td>(0-17)</td>
<td>(0-17)</td>
</tr>
</tbody>
</table>

**S-Pad Timing Rules**

E-12

<table>
<thead>
<tr>
<th>Field Name</th>
<th>FAOD</th>
<th>FAOD1</th>
<th>A1</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octal Code</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>FAOD</td>
<td>NOP</td>
<td>NC</td>
<td>NC</td>
</tr>
<tr>
<td>1</td>
<td>FSUBR</td>
<td>FIX</td>
<td>FM</td>
<td>FA</td>
</tr>
<tr>
<td>2</td>
<td>FSUBR</td>
<td>FIXT</td>
<td>DPX</td>
<td>DPA</td>
</tr>
<tr>
<td>3</td>
<td>FAOD</td>
<td>FSCLT</td>
<td>DPT</td>
<td>DPT</td>
</tr>
<tr>
<td>4</td>
<td>FEQV</td>
<td>FSM2C</td>
<td>DM</td>
<td>MD</td>
</tr>
<tr>
<td>5</td>
<td>FAND</td>
<td>F2CSM</td>
<td>ZER</td>
<td>ZER</td>
</tr>
<tr>
<td>6</td>
<td>FOR</td>
<td>FSCALE</td>
<td>ZER</td>
<td>ZER</td>
</tr>
<tr>
<td>7</td>
<td>IO</td>
<td>FABS</td>
<td>ZER</td>
<td>ZER</td>
</tr>
</tbody>
</table>
### Unconditional Fields

Each of the following fields may be used in any given instruction word.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>COND</th>
<th>DISP (Branch Displacement)</th>
<th>D PX</th>
<th>D PY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octal Code</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>NOP</td>
<td>(0-37)</td>
<td>NOP</td>
<td>NOP</td>
</tr>
<tr>
<td>1</td>
<td>f</td>
<td>E-68</td>
<td>DB</td>
<td>E-123</td>
</tr>
<tr>
<td>2</td>
<td>BR</td>
<td>E-69</td>
<td>FA</td>
<td>E-124</td>
</tr>
<tr>
<td>3</td>
<td>SINTRQ</td>
<td>E-70</td>
<td>FM</td>
<td>E-125</td>
</tr>
<tr>
<td>4</td>
<td>BION</td>
<td>E-70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>BIOZ</td>
<td>E-71</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>BFPE</td>
<td>E-71</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>RETURN</td>
<td>E-72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>BFQ</td>
<td>E-73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>BFNE</td>
<td>E-74</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>BFGE</td>
<td>E-75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>BFST</td>
<td>E-76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>BEQ</td>
<td>E-77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>BNE</td>
<td>E-78</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>BGE</td>
<td>E-79</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>BGT</td>
<td>E-80</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>DPBS</th>
<th>XR (DPX Read Index)</th>
<th>YR (DPY Read Index)</th>
<th>XW (DPX Write Index)</th>
<th>YW (DPY Write Index)</th>
<th>FM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octal Code</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>ZERO</td>
<td>E-129</td>
<td>(0-7)</td>
<td>(0-7)</td>
<td>(0-7)</td>
<td>NOP</td>
</tr>
<tr>
<td>1</td>
<td>INBS</td>
<td>E-130</td>
<td></td>
<td></td>
<td></td>
<td>FMU</td>
</tr>
<tr>
<td>2</td>
<td>VALUE*</td>
<td>E-131</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DPX</td>
<td>E-132</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>DPY</td>
<td>E-133</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>MD</td>
<td>E-134</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SPFN</td>
<td>E-135</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>TM</td>
<td>E-136</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>M1</th>
<th>M2</th>
<th>MI</th>
<th>MA</th>
<th>DPA</th>
<th>TMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octal Code</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>FM</td>
<td>E-99</td>
<td>FA</td>
<td>E-100</td>
<td>NOP</td>
<td>NOP</td>
</tr>
<tr>
<td>1</td>
<td>DPX</td>
<td>E-99</td>
<td>DPX</td>
<td>E-100</td>
<td>FA</td>
<td>INCM  E-137</td>
</tr>
<tr>
<td>2</td>
<td>DPY</td>
<td>E-99</td>
<td>DPY</td>
<td>E-100</td>
<td>FM</td>
<td>DECM  E-138</td>
</tr>
<tr>
<td>3</td>
<td>TM</td>
<td>E-99</td>
<td>MD</td>
<td>E-100</td>
<td>DB</td>
<td>SETMA  E-139</td>
</tr>
</tbody>
</table>

*This instruction uses a 16-bit immediate VALUE as a constant or address (in bits 48-63 of this instruction). The YW, FM, M1, M2, MI, TMA, and DPA fields are then disabled for this instruction word.
SPE C fields
One of the SPE C fields may be used per instruction word. The S-Pad fields (B, SOP, SOP1, SH, SPS, and SPD) are then disabled for this instruction.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>SPEC</th>
<th>STE Test</th>
<th>HostPNl</th>
<th>SETPSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octal Code</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>STEST</td>
<td>BFLT</td>
<td>PHLLIT</td>
<td>JMPA*</td>
</tr>
<tr>
<td>1</td>
<td>HOSTPNl</td>
<td>BLT</td>
<td>DBLLIT</td>
<td>JSRA*</td>
</tr>
<tr>
<td>2</td>
<td>SPMDA</td>
<td>BNC</td>
<td>DBHLLIT</td>
<td>JMP*</td>
</tr>
<tr>
<td>3</td>
<td>NOP</td>
<td>BZC</td>
<td>DBLLLIT</td>
<td>JSR*</td>
</tr>
<tr>
<td>4</td>
<td>NOP</td>
<td>BDIN</td>
<td>NOP</td>
<td>JMPT</td>
</tr>
<tr>
<td>5</td>
<td>NOP</td>
<td>BDIZ</td>
<td>NOP</td>
<td>JSRT</td>
</tr>
<tr>
<td>6</td>
<td>NOP</td>
<td>BFIN</td>
<td>NOP</td>
<td>JMPP</td>
</tr>
<tr>
<td>7</td>
<td>NOP</td>
<td>BFZ</td>
<td>NOP</td>
<td>JSRP</td>
</tr>
<tr>
<td>10</td>
<td>SETPSA</td>
<td>NOP</td>
<td>SWDB</td>
<td>NOP</td>
</tr>
<tr>
<td>11</td>
<td>PSEVEN</td>
<td>NOP</td>
<td>SWDBE</td>
<td>NOP</td>
</tr>
<tr>
<td>12</td>
<td>PSODD</td>
<td>NOP</td>
<td>SWDBH</td>
<td>NOP</td>
</tr>
<tr>
<td>13</td>
<td>PS</td>
<td>NOP</td>
<td>SWDBL</td>
<td>NOP</td>
</tr>
<tr>
<td>14</td>
<td>SETEXIT</td>
<td>BFLD</td>
<td>NOP</td>
<td>NOP</td>
</tr>
<tr>
<td>15</td>
<td>NOP</td>
<td>BFL1</td>
<td>NOP</td>
<td>NOP</td>
</tr>
<tr>
<td>16</td>
<td>NOP</td>
<td>BFL2</td>
<td>NOP</td>
<td>NOP</td>
</tr>
<tr>
<td>17</td>
<td>NOP</td>
<td>BFL3</td>
<td>NOP</td>
<td>NOP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>PSEVEN</th>
<th>PSODD</th>
<th>PS</th>
<th>SETEXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octal Code</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>RPSOA*</td>
<td>RPSIA*</td>
<td>RPSLA*</td>
<td>NOP</td>
</tr>
<tr>
<td>1</td>
<td>RPS2A*</td>
<td>RPS3A*</td>
<td>RPSFA*</td>
<td>SETEXA*</td>
</tr>
<tr>
<td>2</td>
<td>RPSO*</td>
<td>RPS1*</td>
<td>RPSL*</td>
<td>NOP</td>
</tr>
<tr>
<td>3</td>
<td>RPS2*</td>
<td>RPS3*</td>
<td>RPSF*</td>
<td>SETEX*</td>
</tr>
<tr>
<td>4</td>
<td>RPS0T</td>
<td>RPSIT</td>
<td>RPSLT</td>
<td>SETEX</td>
</tr>
<tr>
<td>5</td>
<td>RPS2T</td>
<td>RPS3T</td>
<td>RPSFT</td>
<td>E-67</td>
</tr>
<tr>
<td>6</td>
<td>NOP</td>
<td>NOP</td>
<td>RPSLP</td>
<td>NOP</td>
</tr>
<tr>
<td>7</td>
<td>NOP</td>
<td>NOP</td>
<td>RPSFP</td>
<td>E-61</td>
</tr>
<tr>
<td>10</td>
<td>WPSOA*</td>
<td>WPSIA*</td>
<td>LPSLA*</td>
<td>E-62</td>
</tr>
<tr>
<td>11</td>
<td>WPS2A*</td>
<td>WPS3A*</td>
<td>LPSRA*</td>
<td>E-63</td>
</tr>
<tr>
<td>12</td>
<td>WPSO*</td>
<td>WPS1*</td>
<td>LPSL*</td>
<td>E-64</td>
</tr>
<tr>
<td>13</td>
<td>WPS2*</td>
<td>WPS3*</td>
<td>LPSR*</td>
<td>E-64</td>
</tr>
<tr>
<td>14</td>
<td>WPS0T</td>
<td>WPS1T</td>
<td>LPSLT</td>
<td>E-64</td>
</tr>
<tr>
<td>15</td>
<td>WPS2T</td>
<td>WPS3T</td>
<td>LPSRT</td>
<td>E-65</td>
</tr>
<tr>
<td>16</td>
<td>NOP</td>
<td>NOP</td>
<td>LPSLP</td>
<td>NOP</td>
</tr>
<tr>
<td>17</td>
<td>NOP</td>
<td>NOP</td>
<td>LPSRP</td>
<td>E-65</td>
</tr>
</tbody>
</table>

Formats for partial words (PSEVEN, PSODD, PS Fields) . . . . E-51

*This instruction uses a 16-bit integer VALUE (in bits 48-63 of the instruction word). The YM, FM, M1, M3, MI, MA, TMA, and DPA fields are then disabled for this instruction word.
I/O Fields
One of the I/O Fields may be used per instruction word. The Floating Adder Fields (FADO, FADDI, A1, and A2) are then disabled for this instruction word.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>I/O</th>
<th>LDREP</th>
<th>RDREP</th>
<th>INOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Name</td>
<td>Octal Code</td>
<td>I/O</td>
<td>LDRd</td>
<td>RDRd</td>
</tr>
<tr>
<td>0</td>
<td>LDRE2</td>
<td>E-101</td>
<td>NOP</td>
<td>RPSA</td>
</tr>
<tr>
<td>1</td>
<td>LDRE2</td>
<td>E-105</td>
<td>LDSPD</td>
<td>RSPD</td>
</tr>
<tr>
<td>2</td>
<td>SPH2AV</td>
<td>E-7</td>
<td>LDMA</td>
<td>RMA</td>
</tr>
<tr>
<td>3</td>
<td>SPH2AV</td>
<td>E-7</td>
<td>LDMA</td>
<td>RTMA</td>
</tr>
<tr>
<td>4</td>
<td>SPH2AV</td>
<td>E-7</td>
<td>LDMA</td>
<td>RTMA</td>
</tr>
<tr>
<td>5</td>
<td>SPH2AV</td>
<td>E-109</td>
<td>LDQPA</td>
<td>RDPA</td>
</tr>
<tr>
<td>6</td>
<td>SENSE</td>
<td>E-115</td>
<td>LDSP</td>
<td>RSPFN</td>
</tr>
<tr>
<td>6</td>
<td>SENSE</td>
<td>E-115</td>
<td>LDSP</td>
<td>RAP</td>
</tr>
<tr>
<td>7</td>
<td>SENSE</td>
<td>E-115</td>
<td>LDSP</td>
<td>RAP</td>
</tr>
<tr>
<td>7</td>
<td>SENSE</td>
<td>E-115</td>
<td>LDSP</td>
<td>RAP</td>
</tr>
<tr>
<td>7</td>
<td>SENSE</td>
<td>E-115</td>
<td>LDSP</td>
<td>RAP</td>
</tr>
<tr>
<td>7</td>
<td>SENSE</td>
<td>E-115</td>
<td>LDSP</td>
<td>RAP</td>
</tr>
</tbody>
</table>

ALSO, I/SH
I/SPS
I/SPD
FADO
I/A1
I/A2
COND
DISP

<table>
<thead>
<tr>
<th>Field Name</th>
<th>SENSE</th>
<th>FLAG</th>
<th>CONTROL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Name</td>
<td>Octal Code</td>
<td>SENSE</td>
<td>FLAG</td>
</tr>
<tr>
<td>0</td>
<td>SNSA</td>
<td>E-115</td>
<td>SFLO</td>
</tr>
<tr>
<td>1</td>
<td>SPNSA</td>
<td>E-115</td>
<td>SFL1</td>
</tr>
<tr>
<td>2</td>
<td>SNSSA</td>
<td>E-115</td>
<td>SFL2</td>
</tr>
<tr>
<td>2</td>
<td>SPNSA</td>
<td>E-115</td>
<td>SFL3</td>
</tr>
<tr>
<td>4</td>
<td>SNSB</td>
<td>E-118</td>
<td>CFL0</td>
</tr>
<tr>
<td>5</td>
<td>SPNSB</td>
<td>E-118</td>
<td>CFL1</td>
</tr>
<tr>
<td>6</td>
<td>SNSSBD</td>
<td>E-119</td>
<td>CFL2</td>
</tr>
<tr>
<td>7</td>
<td>SPNSBD</td>
<td>E-120</td>
<td>CFL3</td>
</tr>
</tbody>
</table>

| NOP | E-5 |

AP-120B Instruction Field Layout

<table>
<thead>
<tr>
<th>B</th>
<th>SOP</th>
<th>SN</th>
<th>SPS</th>
<th>SPD</th>
<th>FADO</th>
<th>A1</th>
<th>A2</th>
<th>COND</th>
<th>DISP</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-Pad Group</td>
<td>Adder Group</td>
<td>Branch Group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOP1</td>
<td>FADDI</td>
<td>I/O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| DPX | DFY | DPA | DPRS | XR | YR | XW | YM | FM | MI | M2 | MI | MA | DPA | THA |
|-----|-----|-----|------|----|----|----|----|----|----|----|----|----|-----|-----|-----|
| Data Pad Group | Multiply Group | Memory Group |

VALUE

E - 4
all zeros NOP  No-operation

Assembler format:  NOP

Effect:  No operation is performed

Description: The assembler recognizes this mnemonic and will insert an all zeros instruction which is a NOP.
CONTROL (from SPEC  )

<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 10 11 12 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 SH SPS SPS SOCPL</td>
</tr>
</tbody>
</table>

VALUE

2

SPMDA  SPIN WHILE MAIN DATA BUSY

Assembler
Format:  SPMDA

Effect:  "SPIN" while MAIN DATA BUSY

Description: When specified, SPMDA causes the AP-120B to suspend program execution until MAIN DATA MEMORY (MD) completes its READ or WRITE cycle and becomes available for the next READ/WRITE operation. Using this op-code in an instruction immediately following one that initiates an MD READ operation, results in the data from that operation being available for use during the present instruction. It has no effect on a MD READ/WRITE operation in the same instruction.

Thus:  LDMA; DB=100
       SPMDA; DPX(O)\(\text{MD}\)

results in the contents of MD(100) being loaded into DPX(0).
VALUE

2

SPMDAV

SPIN WHILE MAIN DATA BUSY

Assembler Format: SPMDAV
Effect: "SPIN" while MAIN DATA BUSY

Description: When specified, SPMDA causes the AP-1208 to suspend program execution until MAIN DATA MEMORY (MD) completes its READ or WRITE cycle and becomes available for the next READ/WRITE operation. Using this op-code in an instruction immediately following one that initiates an MD READ operation, results in the data from that operation being available for use during the present instruction. It has no effect on a MD READ/WRITE operation in the same instruction.

Thus: LDMA; DB=100
SPMDA; DPX(O)<MD

results in the contents of MD(100) being loaded into DPX(0).

3

REXIT

READ EXIT

Assembler Format: REXIT
Effect: SRS(SRA)→PNLBS

Description: This instruction is similar to SETEXIT, but allows reading instead. The SRA is NOT changed by this instruction. This is generally used with a LDSPNL to load into S-PAD register.
Halt AP-120

Assembler
Format: HALT

example: INCMA; MI<FA; HALT

Effect: $1 + FN_{\text{bit } \emptyset}$; clear RUN INDICATOR

Description: The AP-120B program execution will be halted after completion of the current instruction word. (See note.) AP-120B RUN INDICATOR (RUN) cleared and PANEL FUNCTION REGISTER(bit 0) set. When halted, PSA will point to the next instruction to be executed and it will have been entered into the instruction register. SPFN will reflect the operation in that instruction.

^NOTE: if the current instruction "SPINS" while waiting for I/O or MEMORY, HALT will not be effective until the "SPIN" cycle is finished and the instruction completed (as in the above example).

WARNING: Due to timing problems in the JSR instruction, it should not follow a HALT. Thus, the recommended programming practice is to place a NOP (no operation, all zeros instruction) after every HALT instruction.
CONTROL FIELD (CONTROL)

1  IORST  RESET I/O DEVICES IMMEDIATELY

Assembler
Format: IORST
Effect: Clear I/O device logic and timing and all four of the general FLAGS.

Description: Effects are device dependent. No effect on Host interface, TMRAM, or IOP16.

2  INTEN  INTERRUPT ENABLE

Assembler
Format: INTEN
Effect: If CTL Bit 05 is already set to "1", generate interrupt to HOST-CPU. If not, no effect.

Description: This is used in conjunction with the CTL05 interrupt. (See I/O, PROGRAMMED INTERRUPTS.)

When an INTEN is executed, the AP-120B will attempt to set CTL05 interrupt. If CTL(Bit 05) is already set, then the AP-120B will generate an interrupt to HOST-CPU. The state of CTL(Bit 05) is not altered by this instruction.
CONTROL FIELD (CONTROL)

VALUE

3

INTA

INTERRUPT ACKNOWLEDGE

Assembler Format: INTA

Effect: (DA interrupting IODEVICE) + INBS

Description: The interrupting I/O DEVICE enables its Device Address onto the INBS MANTISSA (Bits 20 to 27) for the current instruction cycle. Used to identify the interrupting I/O device after an interrupt is detected via the BINTRQ instruction.

4

REFR

MEMORY REFRESH REQUEST

Assembler Format: REFR

Effect: REFRESH MD; reset REFR CTR

Description: REFR initiates a REFRESH cycle to MAIN DATA MEMORY (MD). The REFRESH COUNTER (REFR CTR) is reset to zero. This has the effect of synchronizing the REFRESH timing with a running AP-120B program. It can be used either to eliminate REFRESH interference with programmed memory accesses or simply to stabilize the REFRESH timing in order to facilitate hardware fault tracing with an oscilloscope. Floating Point Systems supplied APEX drivers always generate a REFR before starting the user called micro-code.

NOTE: NON-PROGRAMMED REFRESH cycles occur every 31usec for 8K bank-pair MD memory and every 15.5usec for 32K bank-pair MD memory.
**CONTROL FIELD (CONTROL)**

<table>
<thead>
<tr>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>A1</td>
<td>A2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **MANDATORY FIELDS**
- **OPTIONAL FIELDS**
- **DISABLED FIELDS**

---

**VALUE**

5

**WRTEX**  
WRITE EXPONENT ONLY

**Assembler Format:** WRTEX  

**Example:** DPX (2)<TM; WRTEX

**Effect:** Restricts DPX, DPY, or MI fields to write exponent bits only.

**Description:** When specified with a concurrent WRITE DPX, DPY, or MI operation, WRTEX restricts the writing to EXPONENT(Bits 02-11) only. (See S-PAD group, WRTEXP for further description of effect.)

6

**WRTMAN**  
WRITE MANTISSA ONLY

**Assembler Format:** WRTMAN  

**Example:** MOV 5,5; SETMA; MI<FA; WRTMAN

**Effect:** Restricts DPX, DPY, or MI fields to WRITE MANTISSA bits only.

**Description:** When specified with a concurrent WRITE DPX, DPY or MI operation, WRTMAN restricts the writing to MANTISSA(Bits 00-27) only. (See S-PAD group for further discussion.)
CONTROL FIELD (CONTROL)
continued...

I/O INTERRUPT ACKNOWLEDGE

Assembler
Format: IOINTA

Effect: (DA interrupting IODEVICE) INBUS

Description: The interrupting I/O Device enables its Device Address onto the INBS Mantissa (Bits 20 to 27) for the current instruction cycle.
S-PAD TIMING RULES

1. SPFN for an instruction with an S-PAD operation is the result of that operation.

2. SPFN is stored back into SP(SPD) only once - at the end of the instruction in which the S-PAD operation took place (not stored at all if No-Load specified). Similarly, the N, Z, and C S-PAD condition bits are set only once for each S-PAD operation.

3. SPFN for an instruction without an S-PAD operation is the result of performing the last previous S-PAD instruction over again, using the current value of SP(SPD) as possibly modified by the original S-PAD operation. SP(SPD) is not altered if no S-PAD operation is specified. This modified SPFN value would be apparent if an SPFN utilizing instruction were executed (e.g., RSPFN, SETMA, etc.).
**BIT REVERSE FIELD**

<table>
<thead>
<tr>
<th>B</th>
<th>SOP</th>
<th>SP1</th>
<th>SPS</th>
<th>SPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ø</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

**VALUE**

Ø  

No Operation

1  

BIT REVERSE the contents of S-PAD SOURCE REGISTER before using

**Assembler Format:**

< & > (Brackets indicate optional use with S-PAD operations.)

**Effect:**

Example: ADD & 6,5

**Effect:**

BIT-REVERSE(SP_{SPS}) \rightarrow SOURCE INPUT FOR CURRENT S-PAD OPERATION

**Description:** The contents of the S-PAD SOURCE REGISTER (SP[SPS]) are BIT-REVERSED and shifted before being used as the SOURCE OPERAND in the current S-PAD operation.

The number of shifts performed depends on the size of the complex data array being processed. The programmer must load the applicable shift value into the BIT-REVERSE field of the APSTATUS Register before specifying the BIT-REVERSE operation. (See S-PAD SUMMARY BIT-REVERSE FIELD for more details.) (See also APSTATUS SUMMARY.)
SHIFT FIELD

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>SOP</td>
<td>*</td>
<td>SH</td>
<td>SPS</td>
<td>SPF</td>
<td>SPD</td>
<td>SOP1</td>
<td>*</td>
<td>Spec Oper</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MANDATORY FIELDS
OPTIONAL FIELDS
DISABLED FIELDS

* MAY BE USED WITH EITHER SOP OR SOP1 FIELDS

VALUE

Ø

No Operation

1

L

LEFT SHIFT S-PAD OUTPUT (SPFN) ONCE. ZERO FILL.

Assembler
Format: <L> (Brackets indicate optional use with S-PAD operations)
Example: SUBL 5,6
Effect: SPFNF LEFT SHIFTED ONCE+ SPFN

Description: The S-PAD RESULT (SPFN) is logically shifted left one place. The right-most bit is set to zero. The bit shifted off the left end is stored in the S-PAD CARRY BIT, (C) - overriding any carry that resulted from the specified arithmetic operation.

Excepting possible OVERFLOW, the shift has the effect of a multiplication by two. The carry bit (C), bit 7 of the AP INTERNAL STATUS REGISTER (APSTATUS), may be tested during the next instruction cycle.
SHIFT FIELD (SH)

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1</td>
</tr>
<tr>
<td>2 3</td>
</tr>
<tr>
<td>4 5</td>
</tr>
<tr>
<td>6 7</td>
</tr>
<tr>
<td>8 9</td>
</tr>
<tr>
<td>10 11</td>
</tr>
<tr>
<td>12 13</td>
</tr>
</tbody>
</table>

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
</table>
| SOP  *
| SH    |
| SPS   |
| SPD   |

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

* MAY BE USED WITH EITHER
SOP OR SOP1 FIELDS

VALUE

2

R

RIGHT SHIFT S-PAD FUNCTION (SPFN) ONCE. ZERO FILL.

Assembler Format: < R > (Brackets indicate optional use with S-PAD operations)

Example: SUBR 5,6

Effect: SPFN right-shifted once → SPFN

Description: The S-PAD RESULT (SPFN) is logically shifted right one place. A zero is shifted into the left-most bit. The bit shifted off the right end is set into the S-PAD CARRY BIT.

The instruction has the effect for unsigned numbers, of a division by two. Bit C of the AP INTERNAL STATUS REGISTER (bit 7, ASTATUS) reflects the condition of S-PAD CARRY and may be tested during the next instruction cycle.
SHIFT FIELD

<table>
<thead>
<tr>
<th>S</th>
<th>O</th>
<th>P</th>
<th>*</th>
<th>SH</th>
<th>SPS</th>
<th>SPD</th>
</tr>
</thead>
</table>

| MANDATORY FIELDS |
| OPTIONAL FIELDS |
| DISABLED FIELDS |

* MAY BE USED WITH EITHER SOP OR SOP1 FIELDS

VALUE

3

RR

RIGHT SHIFT S-PAD FUNCTION (SPFN) twice. Zero fill.

Assembler Format:

<RR> (Brackets indicate optional use with S-PAD operations)

Example: SUBRR 5,6

Effect:

SPFN → right shifted twice → SPFN

Description: The contents of the S-PAD ALU RESULT are logically shifted right two times before being enabled onto the SPFN data path. Zeros are filled into the left-most two bits. The second bit shifted off the end is set into the S-PAD ALU CARRY BIT.

The instruction has the effect for unsigned numbers of a division by four. Bit C (bit 7) of the AP INTERNAL STATUS REGISTER (APSTATUS) reflects the condition of the S-PAD CARRY BIT and may be tested during the next instruction cycle.

\[ \emptyset \rightarrow \emptyset \rightarrow SPFN \rightarrow C \]

- REFLECTS THE CONTENTS OF THE SECOND BIT SHIFTED "OFF THE END"

FIRST BIT SHIFTED "OFF THE END" IS LOST
S-PAD OPERATIONS FIELD

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td>SH</td>
<td>SPS</td>
<td>SPD</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

VALUE

∅

See, S-PAD OPERATIONS

1

See, SPECIAL OPERATIONS

2

ADD

ADD S-PAD SOURCE REGISTER AND S-PAD DESTINATION REGISTER

Assembler

Format: ADD<#> <&> SPS, SPD

Effect:

(SPSP)† plus (SPSP)†† SPFN††;

SPFN+(SPSP) unless S-PAD NO-LOAD(#) is specified

Description: The contents of S-PAD SOURCE REGISTER (SPSP) are added with the contents of S-PAD DESTINATION REGISTER (SPSP). The result of the operation, (SPFN) is stored back into the specified S-PAD DESTINATION REGISTER unless an S-PAD NO-LOAD(#) is specified.

Appropriate bits are set in the AP INTERNAL STATUS REGISTER (APSTATUS) and may be tested during the next instruction cycle.

CARRY BIT EQUATION: If (SPSP)+(SPSP)≥216 then carry=1

† (SPSP) may be optionally BIT-REVERSED, (see BIT-REVERSE FIELD)

†† SPFN from the ADD may be optionally shifted, (see SHIFT FIELD)
S-PAD OPERATIONS FIELD

VALUE

3

SUB

SUBtract S-PAD SOURCE REGISTER from S-PAD
DESTINATION REGISTER

Assembler Format: SUB <sh> <#> < $ > sps, spd

Effect: 
(SP<SPD>) minus (SP<SPS>) \(\rightarrow\) SPFN \(^{*}\); SPFN \(\rightarrow\) (SP<SPD>) unless S-PAD NO-LOAD (#) is specified

Description: The contents of the S-PAD SOURCE REGISTER are subtracted from the contents of the S-PAD DESTINATION REGISTER. The result of the operation is stored back into the S-PAD DESTINATION REGISTER unless a S-PAD NO-LOAD (#) is specified.

Appropriate bits (N, Z, C) are set in the AP INTERNAL STATUS REGISTER (APSTATUS) and may be tested during the next instruction cycle.

CARRY BIT EQUATION: If \((SP[SPD]) + (SP[SPS]) + 1 \geq 2^{16}\) then \(C=1\) else 0. If a shift is specified, then \(C\) is set to the carry from that shift.

\(^{*}(SP[SPS])\) may be optionally BIT-REVERSED. See BIT-REVERSE FIELD.

\(^{*}\)SPFN from the SUB may be optionally shifted. See SHIFT FIELD.
S-PAD OPERATIONS FIELD

<table>
<thead>
<tr>
<th>#</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOP</td>
<td>SH</td>
<td>SPS</td>
<td>SPD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VALUE

4

**MOV**

**MOVE S-PAD SOURCE REGISTER TO S-PAD DESTINATION REGISTER**

Assembler Format:

MOV <sh> <#> <&> sps, spd

Effect:

(SP<SPS>)↑→ SPFN ↑↑; SPFN→ (SP<SPD>) unless S-PAD NO-LOAD is specified.

Description: SPFN is set to the contents of the S-PAD SOURCE REGISTER (SP[SPS]); SPFN is stored into the S-PAD DESTINATION REGISTER unless an S-PAD NO-LOAD (#) is specified.

Appropriate bits are set in the AP INTERNAL STATUS REGISTER (APSTATUS) and may be tested during the next instruction cycle.

CARRY BIT EQUATION: If [(SP[SPD]) AND (SP[SPS])] + [(SP[SPD])] OR (SP[SPS]) ≥ 216

then, C=1 else 0

↑(SP[SPS]) may be optionally BIT-REVERSED. See BIT-REVERSE FIELD.

↑↑ SPFN from the MOV may be optionally shifted. See SHIFT FIELD.
S-PAD OPERATIONS FIELD

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td>SH</td>
<td>SPS</td>
<td>SPD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

VALUE

5

AND

AND S-PAD SOURCE REGISTER to S-PAD DESTINATION REGISTER

Assembler Format:

AND <sh> <#> <&> sps,spd

Effect:

\((SP_{SPS})^+ \text{ AND } (SP_{SPD})^+ \to SPFN^{++};

SPFN \to (SP_{SPD}) \text{ unless S-PAD NO-LOAD is specified.}

Description: The contents of the S-PAD SOURCE REGISTER \((SP_{SPS})\) are logically ANDed with the contents of the S-PAD DESTINATION REGISTER \((SP_{SPD})\). A bit by bit comparison is made between the contents of the two operands and if both respective bits are "1", a "1" is recorded into the corresponding bit of the result \((SPFN)\). All other combinations result in "0" being recorded into the respective bit of \(SPFN\).

The result of the operation \((SPFN)\) is stored into \(SP_{SPD}\) unless an S-PAD NO-LOAD \((#)\) is specified.

The appropriate bits are set in the AP INTERNAL STATUS REGISTER \((APSTATUS)\) and may be tested during the next instruction cycle.

CARRY BIT EQUATION: If \([(SP_{SPD}) \text{ AND } (SP_{SPS})] + (SP_{SPD}) > 2^{16}\) then CARRY=1

TRUTH TABLE

<table>
<thead>
<tr>
<th>SP_{SPS}</th>
<th>SP_{SPD}</th>
<th>SPFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\cdot)</td>
<td>(\cdot)</td>
<td>(\cdot)</td>
</tr>
<tr>
<td>(\cdot)</td>
<td>(1)</td>
<td>(\cdot)</td>
</tr>
<tr>
<td>(1)</td>
<td>(\cdot)</td>
<td>(\cdot)</td>
</tr>
<tr>
<td>(1)</td>
<td>(1)</td>
<td>(1)</td>
</tr>
</tbody>
</table>

\(^+ (SP_{SPS}) \text{ may be optionally BIT-REVERSED, See BIT-REVERSE FIELD}\\

\(^{++} SPFN \text{ from the AND may be optionally shifted, See SHIFT FIELD}
S-PAD OPERATIONS FIELD

VALUE

6  

OR S-PAD SOURCE REGISTER to S-PAD DESTINATION REGISTER

Assembler
Format: OR <sh> <#> <& > sps, spd
Effect: (SP
SPS)† OR (SP
SPD)†† SPFN;
SPFN→(SP
SPD) unless NO-LOAD is specified.

Description: The contents of the S-PAD SOURCE REGISTER (SP[SPS]) are logically ORed with the contents of the S-PAD DESTINATION REGISTER (SP[SPD]). A bit-by-bit comparison is made between the contents of the two operands and if either one of the respective bits = "1," then a "1" is recorded in the corresponding bit of the result (SPFN). All other combinations result in a "0" being recorded into the respective SPFN bit position. The result of the operation (SPFN) is stored into SP(SPD) unless S-PAD NO-LOAD (#) is specified.

Additionally, S-PAD ALU CARRY BIT is set to "0." The appropriate bits of the AP INTERNAL STATUS REGISTER (APSTATUS) are set and may be tested during the next instruction cycle.

TRUTH TABLE

<table>
<thead>
<tr>
<th>SP SPS</th>
<th>SP SPD</th>
<th>SPFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ø</td>
<td>Ø</td>
<td>Ø</td>
</tr>
<tr>
<td>Ø</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Ø</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

†(SP[SPS]) may be optionally BIT-REVERSED. See BIT-REVERSE FIELD.
††SPFN from the OR may be optionally shifted. See SHIFT FIELD.
S-PAD OPERATIONS FIELD

VALUE

7

EQUIVALENCE S-PAD SOURCE REGISTER to S-PAD DESTINATION REGISTER

Assembler Format:

EQV <sh><#> <&> sps, spd

Effect:

\( (SP_{SPS})^{\dagger} \oplus (SP_{SPD}) \Rightarrow SPFN^{\dagger\dagger}; \)

\( SPFN \Rightarrow (SP_{SPD}) \) unless NO-LOAD is specified.

Description: The contents of the S-PAD SOURCE REGISTER (SP[SPS]) and the S-PAD DESTINATION REGISTER (SP[SPD]) are compared on a corresponding bit position basis for equal value. If the corresponding bits both equal "0," or both equal "1," then the respective bit of the result (SPFN) is set to "1." All other combinations result in a "0" being set into the corresponding bit of SPFN. The result of the operation (SPFN) is then written into (SP[SPD]) unless S-PAD NO-LOAD (#) is specified.

The appropriate bits are set in the AP INTERNAL STATUS REGISTER (APSTATUS) and may be tested during the next instruction cycle.

CARRY BIT EQUATION: If \((SP[SPD]) + (SP[SPS]) \geq 2^{16}\) then CARRY=1

<table>
<thead>
<tr>
<th>(\sim)</th>
<th>(\sim)</th>
<th>(\sim)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>(\emptyset)</td>
<td>1</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>1</td>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\(\dagger\) (SP[SPS]) may be optionally BIT-REVERSED. See BIT-REVERSE FIELD.

\(\dagger\dagger\) SPFN from the EQV may be optionally shifted. See SHIFT FIELD.
S-PAD OPERATION 1 FIELD

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>SOP</td>
<td>SH</td>
<td>SPS</td>
<td>SPD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SOP1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SPEC OPER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

VALUE

1

WRTEXP

RESTRICT WRITE TO EXPONENT only into DPX, DPY or MI

Assembler Format: WRTEXP

Example: DPX (-2) FA; WRTEXP

Effect: Restricts DPX,DPY or MI field to write EXPONENT bits only.

Description: WRTEXP restricts writing of the pertinent MEMORY INPUT REGISTER into EXPONENT bits 02-11 only. WRTEXP used in conjunction with a DPX, DPY or MI WRITE operation.

When used in conjunction with a WRITE DPX or WRITE DPY operation, this operation has the effect of concatenating a portion of the input data with the value most recently written into DPX or DPY irrespective of XW or YW. Thus, if the last WRITE into DPX placed a floating point 1.0 into DPX(-2) and in this instruction we WRITE DPX(0) in conjunction with the WRTEXP Op-Code, the net effect is to concatenate the EXPONENT portion of the current input with the MANTISSA from the 1.0 of the preceding DPX WRITE operation and place the result in DPX(0). WRTHMN, WRTLMN act in a similar fashion with the exception that they use different portions of the input argument. WRTEX and WRTMAN from the I/O group also work in a similar manner.
S-PAD OPERATIONS 1 FIELD

<table>
<thead>
<tr>
<th>B</th>
<th>SOP</th>
<th>SH</th>
<th>SPS</th>
<th>SPD</th>
<th>SOP1</th>
<th>SPEC OPER</th>
</tr>
</thead>
</table>

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

Value

2

WRTHMN

Restrict write to high mantissa only into DPX, DPY or MI

Assembler Format: WRTHMN

Example: WRTHMN ; DPY<FM

Effect: Restricts DPX, DPY or MI fields to write high mantissa bits only (mantissa bits 00-11).

Description: WRTHMN restricts the writing to the high mantissa only, (mantissa bits 00-11) of the pertinent memory input register. WRTHMN is used in conjunction with a DPX, DPY or MI write operation. (See example above).

NOTE: See WRTEXP for a description of the effect of this operation on DPX or DPY.

3

WRTL MN

Restrict write to low mantissa only into DPX, DPY or MI fields

Assembler Format: WRTL MN

Example: WRTL MN ; SETMA; MI<MD

Effect: Restricts DPX, DPY or MI fields to write low mantissa only (mantissa bits 12-27).

Description: WRTL MN restricts writing to the low mantissa only (mantissa[bits 12-27]) of the pertinent memory input register. WRTL MN is used in conjunction with a DPX, DPY or MI write operation. (See example above.)

NOTE: See WRTEXP for a description of the effect of this operation on DPX or DPY.
S-PAD OPERATIONS 1 FIELD

<table>
<thead>
<tr>
<th>B</th>
<th>SOP</th>
<th>SH</th>
<th>SPS</th>
<th>SPD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Mandatory Fields**

**Optional Fields**

**Disabled Fields**

---

### VALUE

- **4 through 7**
  - No Operation

### 10

**CLR**

**Clear S-PAD Destination Register**

**Assembler Format:**

CLR <sh><#> spd

**Effect:**

\( \emptyset \rightarrow SPFN; \emptyset \rightarrow SP_{SPD} \) unless NO-LOAD (\#) is specified.

**Description:**

The S-PAD OUTPUT (SPFN) is forced to all zeros and bit "Z" of the AP INTERNAL STATUS REGISTER is set to "1" (bit 5, APSTATUS). SP(SPD) is cleared unless S-PAD NO-LOAD (\#) is specified.

**CARRY BIT EQUATION:**

If \( SP(SPD) \) is negative then CARRY=1.

### 11

**INC**

**Increment S-PAD Destination Register**

**Assembler Format:**

INC<sh><#>spd

**Effect:**

\( SP_{SPD} + 1 \rightarrow SPFN; \) and, unless NO-LOAD is specified,

\( SPFN \rightarrow (SP_{SPD})' \).

**Description:**

The contents of the S-PAD DESTINATION REGISTER (SP[SPD]), plus ONE are enabled onto the S-PAD FUNCTION (SPFN). SPFN is stored into the S-PAD DESTINATION REGISTER unless S-PAD NO-LOAD is specified.

The appropriate bits of the APSTATUS Register are set and may be tested during the next instruction cycle.

**CARRY BIT EQUATION:**

If (SP[SPD]) was -1, then CARRY=1 else 0.

---

E - 25
S-PAD OPERATIONS 1 FIELD

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>SOP</td>
<td>SH</td>
<td>SPS</td>
<td>SPD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SOP1</td>
<td>SPEC OPER</td>
<td></td>
<td></td>
<td>MANDATORY FIELDS</td>
<td>OPTIONAL FIELDS</td>
<td>DISABLED FIELDS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VALUE

12

DEC

DECREMENT S-PAD DESTINATION REGISTER

Assembler Format: DEC <sh> <#> spd

Effect: \((SP_{SPD}) - 1 + SPFN; \) and, unless NO-LOAD is specified,
\((SPFN) + SP_{SPD}\)

Description: The contents of the S-PAD DESTINATION REGISTER minus ONE is set to the S-PAD Function (SPFN). The result (SPFN) is stored into the S-PAD DESTINATION REGISTER unless S-PAD NO-LOAD (#) is specified.

The appropriate bits of the AP INTERNAL STATUS REGISTER (APSTATUS) will be set and may be tested during the next instruction cycle.

CARRY BIT EQUATION: Unless \(SP_{SPD}\) was \(\emptyset\), CARRY=1.

13

COM

COMPLEMENT S-PAD DESTINATION REGISTER

Assembler Format: COM <sh> <#> spd

Effect: \((SP_{SPD}) \rightarrow SPFN; \) unless NO-LOAD is specified,
\(SPFN \rightarrow (SP_{SPD})\)

Description: The ONE's COMPLEMENT of the contents of S-PAD DESTINATION REGISTER are enabled onto the S-PAD Function. The result (SPFN) is stored into \(SP_{SPD}\) unless S-PAD NO-LOAD (#) is specified in the instruction.

The appropriate bits of the AP INTERNAL STATUS REGISTER (APSTATUS) will be set and may be tested during the next instruction cycle.

CARRY BIT EQUATION: Unless \(SP_{SPD}\) was \(\emptyset\), CARRY=1.
VALUE

14  

LDSPNL  

LOAD S-PAD DESTINATION REGISTER from the PANEL BUS

Assembler Format:  
LDSPNL spd

Effect:  
\((SP_{SPD}) \rightarrow SPFN; \ PNLBS \rightarrow SP_{SPD}\)

Description: First, the S-PAD Function is set to the old contents of the S-PAD DESTINATION REGISTER. Then, whatever is enabled onto the PANEL BUS is loaded into the S-PAD DESTINATION REGISTER.

The appropriate bits of the AP INTERNAL STATUS REGISTER (APSTATUS) are set as determined by the previous contents of \(SP_{SPD}\) and may be tested during the next instruction. S-PAD CARRY is set to one. If no S-PAD operation is done in the next instruction, then SPFN for that instruction will be the new contents of \(SP_{SPD}\) as loaded by this instruction cycle.

FIRST:

\[
\begin{array}{c}
\emptyset \\
SP_{SPD} \\
\emptyset \downarrow \ 15 \\
SPFN
\end{array}
\]

THEN:

\[
\begin{array}{c}
\emptyset \\
SP_{SPD} \\
\emptyset \downarrow \ 15 \\
PNLBS
\end{array}
\]

E - 27
S-PAD OPERATIONS 1 FIELD

VALUE

15

LDSPE

Load S-PAD DESTINATION REGISTER from
DATA PAD BUS - EXPONENT

Assembler Format: 
LDSPE spd

Effect: 
(SP_{SPD}) \rightarrow SPFN; then, (DPBS_{EXP}) -512 \rightarrow SP_{SPD}

Description: First, the SPFN is set to the old contents of the S-PAD DESTINATION REGISTER. Then the EXPONENT portion of the DATA PAD BUS (bits 02-11), BIAS inverted, is loaded into the S-PAD DESTINATION REGISTER (bits 06-15). The inverted EXPONENT BIAS BIT is extended into the remaining portion of SP(SPD) (bits 00-05).

The appropriate bits of the AP INTERNAL STATUS REGISTER (APSTATUS) are set as determined by the previous contents of SP(SPD) and may be tested during the next instruction. S-PAD CARRY is set to one. If no S-PAD operation is done in the next instruction, then SPFN for that instruction will be the new contents of SP(SPD) as loaded by this instruction.

\(^{\dagger}\) This transformation converts a BIASED EXPONENT from a Floating Point word into its TWO'S COMPLEMENT equivalent.
S-PAD OPERATIONS 1 FIELD

<table>
<thead>
<tr>
<th>B</th>
<th>SOP</th>
<th>SH</th>
<th>SPS</th>
<th>SPD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SOP1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPEC OPER</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MANDATORY FIELDS
OPTIONAL FIELDS
DISABLED FIELDS

VALUE

16

LDSPI

Load S-PAD DESTINATION REGISTER from DATA PAD BUS - INTEGER

Assembler
Format: LDSPI spd

Effect: \((SP_{SPEC}) \rightarrow SPFN\), then \((DPBS_{LOW MANTISSA}) \rightarrow SP_{SPD}\)

Description: First, SPFN is set to the old contents of the S-PAD DESTINATION REGISTER. Then the contents of the DATA PAD BUS - LOW MANTISSA, a 16-bit integer, are loaded into the S-PAD DESTINATION REGISTER.

The appropriate bits of the AP INTERNAL STATUS REGISTER (APSTATUS) are set as determined by the previous contents of \(SP_{SPD}\) and may be tested during the next instruction. S-PAD CARRY is set to one. If no S-PAD operation is done in the next instruction, the SPFN for that instruction will be the new contents of \(SP_{SPD}\) as loaded by this instruction.
**S-PAD OPERATIONS 1 FIELD**

<table>
<thead>
<tr>
<th>B</th>
<th>SOP</th>
<th>SH</th>
<th>SPS</th>
<th>SPD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**MANDATORY FIELDS**

**OPTIONAL FIELDS**

**DISABLED FIELDS**

**VALUE**

17

**LDSPT**

**Load S-PAD DESTINATION REGISTER from DATA PAD BUS - TABLE LOOK UP BITS**

**Assembler Format:**

LDSPT spd

**Effect:**

\[(SP_{SPD}) \rightarrow SPFN; \text{ then } (DPBS \text{ MANTISSA}_{\text{Bits }02-08}) \rightarrow SP_{SPD}\]

**Description:**

First, the SPFN is set to the old contents of S-PAD DESTINATION REGISTER. Then the DATA PAD BUS - TABLE LOOK UP bits (MANTISSA [bits 02-08]) are loaded into bits 09-15 of the S-PAD DESTINATION REGISTER. SP(SPD) bits 00-08 are cleared to zero.

LDSPT may be used to calculate memory addresses for use with a look up table. It extracts the seven most significant unknown bits from a positive, normalized, non-zero Floating Point number.

The appropriate bits of the AP INTERNAL STATUS REGISTER (APSTATUS) are set as determined by the previous contents of SP(SPD) and may be tested during the next instruction. S-PAD CARRY is set to one. If no S-PAD operation is done in the next instruction, then SPFN for that instruction will be the new contents of SP(SPD) as loaded by this instruction.
BRANCH on FLOATING ADDER LESS THAN ZERO

Assembler Format: BFLT targ
Effect: If FA < 0; then (PSA) + (DISP + BIAS) + PSA (where BIAS = 20).

Description: CONDITIONAL RELATIVE BRANCH. BFLT will cause a program branch if the FADDR Result (FA) available during the previous instruction was less than zero.

This instruction tests the FA-NEGATIVE Bit (FN) of the APSTATUS Register. If FN is equal to "1" (indicating that FA was negative during the previous instruction) a program branch will occur to the BRANCH TARGET ADDRESS (targ) formed by adding the current contents of PSA with Biased contents of the DISP field of the current instruction word. If FN is equal to "0," this instruction will have no effect.

The BRANCH TARGET ADDRESS must lie within -20(8) to +17(8) locations relative to the current PROGRAM SOURCE ADDRESS.

DISP=Instruction Word (BITS 27-31) is computed as follows: DISP = targ - (PSA) + BIAS. Note that if FN was altered via a LDAPS instruction, at least one cycle must intervene before testing it with this instruction. This restriction applies to all BRANCH instructions that test conditions appearing in APSTATUS.
SPECIAL TEST FIELD (STEST)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

VALUE

1

BLT

BRANCH if S-PAD FUNCTION is LESS THAN ZERO

Assembler Format: BLT targ

Effect: If SPFN < 0; then (PSA) + (DISP + BIAS) + PSA (Where BIAS = 20₈).

Description: CONDITIONAL RELATIVE BRANCH. BLT will cause a program branch if the result of the last S-PAD operation (SPFN) was less than zero.

This instruction tests the condition of the SPFN-NEGATIVE Bit (N) of the APSTATUS Register. If "N" is equal to "1" (indicating that SPFN of the last previous instruction was negative), a program branch will occur to the BRANCH TARGET ADDRESS (targ) formed by adding the current contents of PSA with the BIASED contents of the DISP field of the current instruction word. If "N" is equal to "0," this instruction will have no effect.
SPECIAL TEST FIELD (STEST)

VALUE

2

BNC

BRANCH if S-PAD CARRY is equal to "1"

Assembler Format:

BNC targ

Effect:

If S-PAD CARRY = 1; then (PSA) + (DISP↑ - BIAS) → PSA
(Where BIAS = 208).

Description: CONDITIONAL RELATIVE BRANCH. BNC will cause a program branch if the S-PAD CARRY Bit (C) of the APSTATUS Register is equal to "1."

Bit "C" will be equal to "1" if either:

*the S-PAD CARRY Bit was set to "1" as a result of the last S-PAD operation and no S-PAD SHIFT was specified, or

*a shift occurred during the last S-PAD operation and the last bit shifted "off the end" of the S-PAD Result was equal to "1."

E - 33
SPECIAL TEST FIELD (STEST)

\[ \begin{array}{cccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\
\hline
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array} \]

**VALUE**

3

<table>
<thead>
<tr>
<th>BZC</th>
</tr>
</thead>
</table>

**BRANCH on S-PAD CARRY equal to ZERO**

**Assembler Format:**

BZC targ

**Effect:**

If S-PAD CARRY = \( \emptyset \); then (PSA) + (DISP\(^{\dagger}\) - BIAS) + PSA

(Where BIAS = 20\(_{16}\)).

**Description:** CONDITIONAL RELATIVE BRANCH. BZC will cause a program branch if the S-PAD CARRY Bit (C) of the APSTATUS Register is equal to zero.
SPECIAL TEST FIELD (STEST)

**Banner**

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>SH</td>
<td>SPS</td>
<td>SPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

**VALUE**

<table>
<thead>
<tr>
<th>4</th>
</tr>
</thead>
</table>

**BDEN**

BRANCH if DATA PAD BUS is NEGATIVE

**Assembler**

Format: BDBN targ

Effect: If (DB) < 0.0; then (PSA) + (DISP + BIAS) → PSA

(Where BIAS = 208).

**Description:** CONDITIONAL RELATIVE BRANCH. The sign of the DATA PAD BUS (MANTISSA) (DB[MANT]bit 00) is tested as to its state during the preceding instruction. If DB(MANT)Bit 00 was negative, (e.g., = 1), a program branch will occur to the BRANCH TARGET ADDRESS (targ) formed by adding the current contents of PSA with the BIASED contents of the DISP field of the current instruction word. If DB(MANT)Bit 00 was "0," this instruction will have no effect.

**Note:** Since any data enabled onto DB is not latched, the programmer must re-enable the particular data onto DB one instruction cycle before attempting to test it with this instruction.

Note that instructions in the PS field (RPSF, LPSL, etc.) cannot be used to enable data onto Data Pad Bus for testing with this instruction.
SPECIAL TEST FIELD (STEST)

### Description:

**BDBZ**

BRANCH if DATA PAD BUS is POSITIVE and UNNORMALIZED

**Assembler Format:**

```
BDBZ targ
```

**Effect:**

If DB\(^\text{MANT}\) Bits 0\(0,01\) = "0", then \((\text{PSA}) + (\text{DISP} + - \text{BIAS})\) + PSA

(Where BIAS = 208).

**Description:**

CONDITIONAL RELATIVE BRANCH. BDBZ will cause a program branch to occur to the BRANCH TARGET ADDRESS (targ), formed by adding the current contents of PSA with the BIASED contents of the DISP field of the current instruction word, if the sign of the DATA PAD BUS(MANTISSA)(DB[MANT]Bit 00), enabled during the preceding instruction, was positive (e.g., =0) and DB(MANT)Bit 01 was also equal to "0," (indicating an UNNORMALIZED MANTISSA). If either or both Bits equal(s) "1," this instruction will have no effect.

**NOTE:** Since any data enabled onto DB is not latched, the programmer must re-enable the particular data onto DB one cycle before attempting to test it by this instruction.
SPECIAL TEST FIELD (STEST)

VALUE

6

BIFN

BRANCH if INVERSE FFT FLAG = 1

Assembler Format: BIFN targ

Effect: If IFFT (APSTATUS Bit 11) = 1; then (PSA) + (DISP \textsuperscript{+} - BIAS) \rightarrow PSA
(Where BIAS = 208). 

Description: CONDITIONAL RELATIVE BRANCH. BIFN will cause a program branch if the Inverse FFT Flag (IFFT) of the APSTATUS Register is set to "1."

IFFT is APSTATUS(bit 11) and can be set by an LDAPS instruction. (See LDREG, I/O.) It is normally set to "1," along with APSTATUS(bit 12) (FFT), only during an INVERSE FAST FOURIER TRANSFORM.

7

BIFZ

BRANCH if IFFT FLAG = 0

Assembler Format: BIFZ targ

Effect: If IFFT (APSTATUS Bit 12) = \emptyset; then (PSA) = (DISP \textsuperscript{+} - BIAS) \rightarrow PSA
(Where BIAS = 208).

Description: CONDITIONAL RELATIVE BRANCH. BIFZ will cause a program branch if the INVERSE FFT Flag (IFFT) of the APSTATUS Register is cleared to zero.

E - 37
SPECIAL TEST FIELD (TEST)

Assembler Format:

Effect:

\[
\text{BFLn targ}
\]

If FLAG n=1; then \((PSA) + (\text{DISP}^\dagger - \text{BIAS}) \rightarrow PSA\)

(Where \(\text{BIAS} = 208\)).

Description: A branch will occur to program location "targ" (assembler format) if flag "n" is set to "1." Flag "n" must have been set or cleared two cycles before the current instruction cycle in order to be tested, i.e., at least one cycle must intervene between a set or clear flag instruction and a branch flag instruction.

Note: The CONDITIONAL RELATIVE BRANCH instructions test the condition of either of four GENERAL FLAGS (0,1,2,3) available for use in the AP. These flags may be "set" or "cleared" by software instructions. (See FLAG, I/0.)
HOST/ PANEL FIELD (HOSTPNL)

<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 10 11 12 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1</td>
</tr>
</tbody>
</table>

MANDATORY FIELDS
OPTIMAL FIELDS
DISABLED FIELDS

VALUE

0

PNLLIT
TRANSFER PANEL BUS to the LITES REGISTER

Assembler
Format: PNLLIT
Effect: (PNLBS) \rightarrow LITES

Description: The current data enabled onto the 16-bit PANEL BUS are loaded into the 16-bit LITES REGISTER.

1

DBELIT
TRANSFER DATA PAD BUS\textsuperscript{EXONENT} to the LITES REGISTER, via PANEL BUS

Assembler
Format: DBELIT
Effect: (DB^{\text{EXP}}) PNLBS \rightarrow LITES; right justified.

Description: The current data enabled onto the 10-bit DATA PAD BUS\textsuperscript{EXONENT} are loaded into the 16-bit LITES REGISTER - right justified. The transfer is via the PANEL BUS.

E - 39
### HOST/PANEL FIELD (HOSTPNL)

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### MANDATORY FIELDS

#### OPTIONAL FIELDS

#### DISABLED FIELDS

### VALUE

**2**

**DBHLIT**  
TRANSFER DATA PAD BUS - HIGH MANTISSA to the LITES REGISTER, via PANEL BUS

**Assembler Format:**  
```
DBHLIT
```

**Effect:**  
```
(DB\_HMANT) + PNLBS + LITES; right justified
```

**Description:** The current data enabled onto the DATA PAD BUS HIGH MANTISSA (MANTISSA Bits \(00-11\)) are loaded into the 12 right-most bits of the LITES REGISTER. The transfer is via PANEL BUS.

**3**

**DBLLIT**  
TRANSFER DATA PAD BUS LOW MANTISSA to the LITES REGISTER, via PANEL BUS

**Assembler Format:**  
```
DBLLIT
```

**Effect:**  
```
(DB\_LMANT) + PNLBS + LITES
```

**Description:** The current data enabled onto the DATA PAD BUS LOW MANTISSA (MANTISSA Bits \(12-27\)) is loaded into the LITES REGISTER, via the PANEL BUS.
HOST/PANEL FIELD (HOSTPNL)

<table>
<thead>
<tr>
<th>#</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MANDATORY FIELDS
OPTIONAL FIELDS
DISABLED FIELDS

VALUE

4 through 7

NOP
No Operation

10

SWDB
TRANSFER SWITCHES to the DATA PAD BUS, via PANEL BUS

Assembler
Format:
Example:
Effect:

\[
\begin{align*}
\text{SWDB} & \\
\text{DPX(1) < DB; SWDB} & \\
\text{(SWR bits 06-15)} & \rightarrow \text{DB EXP Bits 02-11,} \\
\text{(SWR bits 04-15)} & \rightarrow \text{DB MANT Bits 00-11,} \\
\text{(SWR)} & \rightarrow \text{DB MANT Bits 12-27.} \\
\end{align*}
\]

via PNLBS

Description: The current contents of the 16-bit SWITCH Register (SWR) are enabled onto the DATA PAD BUS (DB) in the following manner:

- SWR (bits 06-15) are enabled onto the DATA PAD BUS (EXPONENT),
- SWR (bits 04-15) are enabled onto the DATA PAD BUS (HIGH MANTISSA), and
- SWR (bits 00-15) are enabled onto the DATA PAD BUS (LOW MANTISSA).

The transfer is via PNLBS.

This instruction is used concurrently with a write from DATA PAD BUS operation to transfer the contents of SWR into a designated memory location or Register. Use of this instruction disables any current DB or PNLBS source-enabling operation.
HOST/PANEL FIELD (HOSTPNL)

<table>
<thead>
<tr>
<th>B</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>tab</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>SH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tab</td>
<td></td>
<td>SOPL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tab</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MANDATORY FIELDS
OPTOINAL FIELDS
DISABLED FIELDS

VALUE

11

SWDBE

TRANSFER SWITCHES to the DATA PAD BUS EXPONENT
via PANEL BUS

Assembler
Format:
Example:  DFY(-2)<DB; SWDBE

Effect:
(SWR Bits 06-15) → DB EXP Bits 02-11, via PNLBS; WRTEXP⁺ is forced
(SWR Bits 04-15) → DB MANT Bits 00-11,
(SWR) → DB MANT Bits 12-27.

Description: The current contents of the 16-bit SWITCH Register (SWR) are enabled onto the DATA PAD BUS (DB) and a WRTEXP is forced.

A WRTEXP is forced as part of this instruction, restricting the writing to the EXPONENT portion of the designated memory location.

⁺ see, SOP1 for a detailed description of WRTEXP
HOST/PANEL FIELD (HOSTPNL)

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

VALUE

12

SWDBH

TRANSFER SWITCHES to DATA PAD BUS HIGH MANTISSA

Assembler Format: SWDBH

Example: SETMA; MI<DB; SWDBH

Effect:

\[
\begin{align*}
\text{SWR Bits } & \text{ } \sim6-15 \rightarrow \text{DB EXP Bits } \sim2-11, \\
\text{SWR Bits } & \text{ } \sim4-15 \rightarrow \text{DB MANT Bits } \sim0-11, \\
\text{SWR} & \rightarrow \text{DB MANT Bits 12-27. } \\
\end{align*}
\]

via PNLBS;WRTHMN \text{\ding{53}} is forced

Description: The current contents of the 16-bit SWITCH Register (SWR) are enabled onto the DATA PAD BUS (DB) in the following manner:

The transfer is via PNLBS.

A WRTHMN is forced as part of this instruction, restricting the writing to the HIGH-MANTISSA portion (MANTISSA[bits 00-11]) of the designated memory location, only.

\text{\ding{53}} See SOP1 for a detailed description of WRTHMN
HOST/PANEL FIELD (HOSTPNL)

<table>
<thead>
<tr>
<th>#</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>SH</td>
<td>SPS</td>
<td>SPF</td>
<td>SPF</td>
<td>SPF</td>
<td>SPF</td>
<td>SPF</td>
<td>SPF</td>
<td>SPF</td>
<td>SPF</td>
<td>SPF</td>
<td>SPF</td>
</tr>
</tbody>
</table>

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

VALUE

14 through
17

NOP

No Operation

13

SWDBL

TRANSFER SWITCHES to DATA PAD BUS LOW MANTISSA

Assembler Format:

```
SWDBL
```

Example:

```
SWDBL; DPY(1)<DB
```

Effect:

```
(SWR Bits 6-15) --> DB EXP Bits 2-11, via PNLBS; WRTLMN
(SWR Bits 4-15) --> DB MANT Bits 0-11, is forced
(SWR) ---> DB MANT Bits 12-27.
```

Description: The current contents of the 16-bit SWITCH Register (SWR) are enabled onto the DATA PAD BUS (DB) in the following manner:

The transfer is via PNLBS.

A WRTLMN is forced as part of this instruction, restricting the writing to the LOW-MANTISSA portion (MANTISSA[bits 12-27]) of the designated memory location only.

See SOP1 for a detailed description of WRTLMN
SET PROGRAM SOURCE ADDRESS (SETPSA)

Assembler Format: JMPA adr
Effect: (VALUE) → PSA

Description: JMPA forces the address contained in the least-significant 12 bits of the VALUE field (Instruction Word Bits 48-63) into the 12-bit PROGRAM SOURCE ADDRESS REGISTER (PSA). The effect is an ABSOLUTE JUMP to the location in program memory contained in the VALUE field.
### SET PROGRAM SOURCE ADDRESS (SETPSA)

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

**VALUE**

<table>
<thead>
<tr>
<th>JSRA</th>
</tr>
</thead>
</table>

**JUMP to SUB ROUTINE (ABSOLUTE)**

**Assembler Format:**

```
JSRA adr
```

**Effect:**

```
(SRA) + 1 + SRA; (PSA) + 1 + SRS_SRA; (VALUE) → PSA
```

**Description:**

SUB-ROUTINE JUMP. First, the contents of the current PROGRAM SOURCE ADDRESS (PSA), plus "1," are saved by incrementing the SUB-ROUTINE STACK ADDRESS REGISTER (SRA) and storing the current 

```
[(PSA) + 1]
```

into the "last-in" location of the SUB-ROUTINE RETURN STACK (SRS). The least-significant 12 bits of the VALUE field (Instruction Word[bits 48-63]) are then loaded into the 12-bit PROGRAM SOURCE ADDRESS REGISTER (PSA). The program then jumps to that location. (See also RETURN, BRANCH.)

**WARNINGS:** The JSR instructions have a timing problem that causes certain instruction sequences to execute improperly. The following sequences should be avoided by the programmer:

1) Instruction from PS, PSEVEN or PSODD followed by a JSR.

   ```
   example: LDPSL
   JSRA
   ```

2) HALT instruction or a Panel Breakpoint before a JSR.

   ```
   example: HALT
   JSRT
   ```

3) Execution of consecutive RETURN instructions. This appears in the coding as a JSR followed by a RETURN.

   ```
   example: JSR
   RETURN
   ```

These cases all result in a mismatch between the subroutine Return address pointers (SRA) and program execution, and thus will cause severe debugging problems if extreme caution is not taken to avoid them.
SET PROGRAM SOURCE ADDRESS (SETPSA)

```
| 0 0 1 | 1 0 0 0 |
```

**Mandatory Fields**

**Optional Fields**

**Disabled Fields**

**Value**

2

```
JMP
```

**Jump (Relative)**

Assembler Format: \texttt{JMP addr}

Effect: \((\text{VALUE}) + (\text{PSA}) \rightarrow \text{PSA}\)

Description: UNCONDITIONAL RELATIVE JUMP. The address contained on the least-significant 12 bits of the VALUE field (Instruction Words Bits 48-63) is added to the current contents of the PROGRAM SOURCE ADDRESS REGISTER (PSA). The address thus formed is loaded into PSA and the program jumps to that location.

3

```
JSR
```

**Jump to Subroutine (Relative)**

Assembler Format: \texttt{JSR adr}

Effect: \[
\begin{align*}
(\text{SRA}) + 1 & \rightarrow \text{SRA}; \\
(\text{PSA}) + 1 & \rightarrow \text{SRS}_{\text{SRA}}; \\
(\text{VALUE}) + (\text{PSA}) & \rightarrow \text{PSA}
\end{align*}
\]

Description: RELATIVE SUB-Routine JUMP. First, the current contents of the PROGRAM SOURCE ADDRESS (PSA) plus "1" are saved by incrementing the SUB-Routine ADDRESS REGISTER (SRA) and storing \([(\text{PSA}) + 1]\) into the "last-in" position of the SUB-Routine RETURN STACK (SRS_{SRA}).

The address contained in the least-significant 12 bits of the VALUE field (Instruction Word Bits 48-63) is added to the current contents of the PROGRAM SOURCE ADDRESS REGISTER (PSA). The address thus formed is loaded into PSA and the program jumps to that location, (see also, RETURN, BRANCH ).
SET PROGRAM SOURCE ADDRESS (SETPSA)

<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 10 11 12 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1</td>
</tr>
<tr>
<td>SH SPD</td>
</tr>
<tr>
<td>SPS SPD</td>
</tr>
<tr>
<td>MANDATORY FIELDS</td>
</tr>
<tr>
<td>OPTIONAL FIELDS</td>
</tr>
<tr>
<td>DISABLED FIELDS</td>
</tr>
</tbody>
</table>

VALUE

4 [JMPT]  JUMP to location specified by TABLE MEMORY ADDRESS (ABSOLUTE)

Assembler

Format: JMPT

Example: ADD 1,2; SETTMA

Effect: (TMA) + PSA

Description: UNCONDITIONAL ABSOLUTE JUMP. The contents of the PROGRAM SOURCE REGISTER (PSA) are replaced by the least-significant 12 bits of the current contents of the TABLE MEMORY ADDRESS REGISTER (TMA).

5 [JSRT]  JUMP to SUB-ROUTINE at location specified by TABLE MEMORY ADDRESS (ABSOLUTE)

Assembler

Format: JSRT

Example: ADD 1,2; SETTMA

JSRT

Effect: (SRA) + 1 + SRA; (PSA) + 1 + SRS SRA; (TMA Bits 04-15) + PSA

Description: UNCONDITIONAL ABSOLUTE SUB-ROUTINE JUMP. First, the current PROGRAM SOURCE ADDRESS plus "1" is saved by incrementing the SUB-ROUTINE ADDRESS POINTER (SRA) and storing [(PSA) + 1] into the "last-in" position of the SUB-ROUTINE RETURN STACK (SRS SRA).

The contents of PSA are then replaced by the least-significant 12 bits of the current contents of the TABLE MEMORY ADDRESS REGISTER (TMA). The program then jumps to the new PROGRAM SOURCE location. (See also, RETURN, BRANCH).
SET PROGRAM SOURCE ADDRESS (SETPSA)

<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 10 11 12 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>B 0 0 1 SH SPS SPD</td>
</tr>
<tr>
<td>MANDATORY FIELDS</td>
</tr>
<tr>
<td>OPTIONAL FIELDS</td>
</tr>
<tr>
<td>DISABLED FIELDS</td>
</tr>
</tbody>
</table>

Value

6

JMPP

Jump to location indicated by SWITCH REGISTER, via PANEL BUS (ABSOLUTE)

Assembler
Format: JMPP
Effect: (SRW Bits 6-15) →PNLBS + PSA

Description: UNCONDITIONAL ABSOLUTE JUMP. The current contents of the least-significant 12 bits of the SWITCH REGISTER (SWR) are loaded into the PROGRAM SOURCE ADDRESS REGISTER (PSA) via the PANEL BUS (PNLBS).

The program then jumps to the new PROGRAM SOURCE location.

WARNING: Propagation delays inherent when executing JMPP may disallow proper decoding of certain target instructions. The preferred alternative sequence is as follows:

\[ t_n \quad \text{SWDB; LDTMA} \]

\[ t_{n+1} \quad \text{JMPT} \]
SET PROGRAM SOURCE ADDRESS (SETPSA)

<table>
<thead>
<tr>
<th>1 2 3 4 5 6 7 8 9 10 11 12 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0 0 0 0 0 0 0 0 1 0 0</td>
</tr>
</tbody>
</table>

Mandatory Fields
Optional Fields
Disabled Fields

7

**JSRP**

JUMP to SUB-Routine pointed by the SWITCH REGISTER

Assembler
JSRP

Format:

Effect:
(SRA) + 1 → SRA; (PSA) + 1 → SRS

(SWR \(_{4-15}\) + PNLBS + PSA

Description: UNCONDITIONAL ABSOLUTE SUB-Routine JUMP. First, the current PROGRAM SOURCE ADDRESS (PSA) plus "1" is "saved" by incrementing the SUB-Routine ADDRESS POINTER (SRA) by "1" and storing \( [(PSA) + 1] \) into the "last-in" position of the SUB-Routine RETURN STACK (SRS

Then the contents of the PROGRAM SOURCE ADDRESS REGISTER are replaced by the current contents of the least significant 12 bits of the SWITCH REGISTER (via the PANEL BUS). The program then jumps to the new PROGRAM SOURCE location. (See also, RETURN, BRANCH ).

**WARNING:** Propagation delays inherent when executing JSRP may disallow proper decoding of certain target instructions. The preferred alternative sequence is as follows:

\[ t_n \text{ SWDB; LDTMA} \]
\[ t_{n+1} \text{ JSRT} \]

E - 50
The following instructions available in the PSEVEN and PSODD fields involve PROGRAM SOURCE partial-word transfers via the PANEL BUS. Addressing for a given instruction may be RELATIVE or ABSOLUTE. (See SPEC SUMMARY.) All instructions within these fields require two cycles to execute and instructions from other fields which reference PNLBS for addressing or use PNLBS as a transfer-conduit should not be concurrently specified.

Formats for the various PS partial-words is given below:

Please note the warnings in the HALT, BDBN, BDBZ and JSRA descriptions for WRT the PS, PSEVEN and PSODD instructions.

NOTES

1) When VALUE or TMA is used as an addressing subscript, the least-significant 12 bits are used.
   Example:
   \[ PS(Q0)(VALUE)=PS(Q0)(VALUE)(bits\ 52-63) \]
   \[ PS(Q0)(TMA)=PS(Q0)(TMA)(bits\ 04-15) \]

   TMA is TMA Register, not the Table Memory Address which may be modified by the FFT bit APSTATUS.

2) PS Quarter 0 is Bits 00 to 15 = PS(Q0)
   1 16 to 31 = PS(Q1)
   2 32 to 47 = PS(Q2)
   3 48 to 64 = PS(Q3)
PROGRAM SOURCE - EVEN FIELD (PSEVEN)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

VALUE

0  RPS0A

READ PROGRAM SOURCE \(^{QUARTER \ 0}\) to LITES (ABSOLUTE)

Assembler Format:

Effect:

RPS0A adr

(\(PS0^0\ VALUE\)) \text{ } + \text{PNLBS} \text{ } + \text{LITES}

1  RPS2A

READ PROGRAM SOURCE \(^{QUARTER \ 2}\) from LITES (ABSOLUTE)

Assembler Format:

Effect:

RPS2A adr

(\(PS2^2\ VALUE\)) \text{ } + \text{PNLBS} \text{ } + \text{LITES}

2  RPS0

READ PROGRAM SOURCE \(^{QUARTER \ 0}\) to LITES (RELATIVE)

Assembler Format:

Effect:

RPS0 adr

(\(PS0^0\ VALUE + PSA\)) \text{ } + \text{PNLBS} \text{ } + \text{LITES}

3  RPS2

READ PROGRAM SOURCE \(^{QUARTER \ 2}\) to LITES (RELATIVE)

Assembler Format:

Effect:

RPS2 adr

(\(PS2^2\ VALUE + PSA\)) \text{ } + \text{PNLBS} \text{ } + \text{LITES}

4  RPS0T

READ PROGRAM SOURCE \(^{QUARTER \ 0}\) to LITES

from the location specified by TABLE MEMORY ADDRESS

Assembler Format:

Effect:

RPS0T

(\(PS0^0\ TMA\)) \text{ } + \text{PNLBS} \text{ } + \text{LITES}
PROGRAM SOURCE - EVEN FIELD (PSEVEN)

5  RPS2T

READ PROGRAM SOURCE QUARTER 2 to LITES from the location specified by TABLE MEMORY ADDRESS

Assembler Format: RPS2T Effecct: (PS₂₉) → PNLSB + LITES

6 and 7

NOP

No Operation

10  WPS₉A

WRITE PROGRAM SOURCE QUARTER 9 from the SWITCHES (ABSOLUTE)

Assembler Format: WPS₉A adr Effecct: (SWR) → PNLSB + PS₉₉ VALUE

11  WPS₂₉A

WRITE PROGRAM SOURCE QUARTER 2 from the SWITCHES (ABSOLUTE)

Assembler Format: WPS₂₉A adr Effecct: (SWR) → PNLSB + PS₂₂ VALUE
### PROGRAM SOURCE - EVEN FIELD (PSEVEN)

<p>| | | | | | | | | | | | | |</p>
<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>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>SH</td>
<td>SPS</td>
<td>SPD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### MANDATORY FIELDS

#### OPTIONAL FIELDS

#### DISABLED FIELDS

**VALUE**

<p>| | | | | | | | | | | | | |</p>
<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>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**WPSØ**

**Assembler Format:**  
WPSØ adr

**Effect:**  
(SWR) + PNLBS + P$S^Q$Ø VALUE + PSA

**WRITE PROGRAM SOURCE**  
QUARTER Ø from the SWITCHES (RELATIVE)

<p>| | | | | | | | | | | | | |</p>
<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>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**WPS2**

**Assembler Format:**  
WPS2 adr

**Effect:**  
(SWR) + PNLBS + P$S^Q$2 VALUE + PSA

**WRITE PROGRAM SOURCE**  
QUARTER 2 from the SWITCHES (RELATIVE)

<p>| | | | | | | | | | | | | |</p>
<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>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**WPSØT**

**Assembler Format:**  
WPSØT

**Effect:**  
(SWR) + PNLBS + P$S^Q$Ø TMA

**WRITE PROGRAM SOURCE**  
QUARTER Ø from the SWITCHES at the location specified by TABLE MEMORY ADDRESS

<p>| | | | | | | | | | | | | |</p>
<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>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**WPS2T**

**Assembler Format:**  
WPS2T

**Effect:**  
(SWR) + PNLBS + P$S^Q$2 TMA

**WRITE PROGRAM SOURCE**  
QUARTER 2 from the SWITCHES at the location specified by TABLE MEMORY ADDRESS
PROGRAM SOURCE - ODD FIELD (PSODD)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>SB</td>
<td>SPS</td>
<td>SPD</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MANDATORY FIELDS
OPTIONAL FIELDS
DISABLED FIELDS

VALUE

0  RPS1A READ PROGRAM SOURCE QUARTER 1 to LITES (ABSOLUTE)

Assembler Format: RPS1A adr
Effect: (PSQ1 VALUE) + PNLBS → LITES

1  RPS3A READ PROGRAM SOURCE QUARTER 3 from LITES (ABSOLUTE)

Assembler Format: RPS3A adr
Effect: (PSQ3 VALUE) + PNLBS → LITES

2  RPS1 READ PROGRAM SOURCE QUARTER 1 to LITES (RELATIVE)

Assembler Format: RPS1 adr
Effect: (PSQ1 VALUE + PSA) → PNLBS → LITES

3  RPS3 READ PROGRAM SOURCE QUARTER 3 to LITES (RELATIVE)

Assembler Format: RPS3 adr
Effect: (PSQ3 VALUE + PSA) → PNLBS → LITES
**PROGRAM SOURCE - ODD FIELD (PSODD)**

<table>
<thead>
<tr>
<th>Value</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>RPS1T</td>
</tr>
<tr>
<td></td>
<td>READ PROGRAM SOURCE QUARTER 1 to LITES from the location specified by TABLE MEMORY ADDRESS</td>
</tr>
<tr>
<td>Assembler Format:</td>
<td>RPS1T</td>
</tr>
<tr>
<td>Effect:</td>
<td>(PS(^{Q1})(^{TMA}) \rightarrow PNLBS \rightarrow LITES)</td>
</tr>
<tr>
<td>5</td>
<td>RPS3T</td>
</tr>
<tr>
<td></td>
<td>READ PROGRAM SOURCE QUARTER 3 to LITES from the location specified by TABLE MEMORY ADDRESS</td>
</tr>
<tr>
<td>Assembler Format:</td>
<td>RPS3T</td>
</tr>
<tr>
<td>Effect:</td>
<td>(PS(^{Q3})(^{TMA}) \rightarrow PNLBS \rightarrow LITES)</td>
</tr>
<tr>
<td>6</td>
<td>NOP</td>
</tr>
<tr>
<td></td>
<td>No Operation</td>
</tr>
<tr>
<td>7</td>
<td>WPS1A</td>
</tr>
<tr>
<td></td>
<td>WRITE PROGRAM SOURCE QUARTER 1 from the SWITCHES (ABSOLUTE)</td>
</tr>
<tr>
<td>Assembler Format:</td>
<td>WPS1A adr</td>
</tr>
<tr>
<td>Effect:</td>
<td>(SWR) \rightarrow PNLBS \rightarrow PS(^{Q1}) VALUE</td>
</tr>
<tr>
<td>10</td>
<td>WPS3A</td>
</tr>
<tr>
<td></td>
<td>WRITE PROGRAM SOURCE QUARTER 3 from the SWITCHES (ABSOLUTE)</td>
</tr>
<tr>
<td>Assembler Format:</td>
<td>WPS3A adr</td>
</tr>
<tr>
<td>Effect:</td>
<td>(SWR) \rightarrow PNLBS \rightarrow PS(^{Q3}) VALUE</td>
</tr>
</tbody>
</table>
WRITE PROGRAM SOURCE QUARTER 1 from the SWITCHES (RELATIVE)

Assembler Format: WPS1 adr
Effect: (SWR) + PNLBS + PSQ1 VALUE + PSA

WRITE PROGRAM SOURCE QUARTER 3 from the SWITCHES (RELATIVE)

Assembler Format: WPS3 adr
Effect: (SWR) + PNLBS + PSQ3 VALUE + PSA

WRITE PROGRAM SOURCE QUARTER 1 from the SWITCHES at the location specified by TABLE MEMORY ADDRESS

Assembler Format: WPS1T
Effect: (SWR) + PNLBS + PSQ1 TMA

WRITE PROGRAM SOURCE QUARTER 3 from the SWITCHES at the location specified by TABLE MEMORY ADDRESS

Assembler Format: WPS3T
Effect: (SWR) + PNLBS + PSQ3 TMA
$ \text{RPSLA} \quad \text{READ PROGRAM SOURCE LEFT HALF to DATA PAD BUS (ABSOLUTE)}$

**Assembler**

**Format:**

\[
\text{RPSLA \ addr}
\]

**Effect:**

\[
(\text{PS}^{\text{LH}} \text{VALUE}) \rightarrow \text{DB}
\]

**Description:**

PROGRAM SOURCE LEFT HALF \((\text{PS LH}=\text{PS bits 00-31})\), as addressed by the least-significant 12 bits contained in the VALUE field (Instruction Word bits 48-63), is enabled onto DATA PAD BUS (DB).

The transfer is executed in the following manner:

* ZEROS \(\rightarrow\) DB(02-07)
* PS(Bits 00-03) \(\rightarrow\) DB(08-11)
* PS(Bits 04-31) \(\rightarrow\) DB(MANT)Bits 00-27

This instruction requires two cycles to execute.
PROGRAM SOURCE FIELD (PS)

<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 10 11 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>B 0 0 1 SH SPS SPD</td>
</tr>
<tr>
<td>1 0 1 1</td>
</tr>
</tbody>
</table>

MANDATORY FIELDS
OPTIONAL FIELDS
DISABLED FIELDS

VALUE

1

RPSFA

READ PROGRAM SOURCE FLOATING-POINT LITERAL to DATA PAD BUS (ABSOLUTE)

Assembler Format: RPSFA adr
Effect: \((PS^{FPL}_{VALUE}) \rightarrow DB\)

Description: PROGRAM SOURCE (FLOATING-POINT LITERAL) \((PS^{FPL}_{VALUE})\), as addressed by the least-significant 12 bits contained in the VALUE field (Instruction Word bits 48-63), is enabled onto DATA PAD BUS (DB).

The transfer is executed in the following manner:

* PS(Bits 26-35) DB(EXP)Bits 02-11
* PS(Bits 36-63) DB(MANT)Bits 00-27

This instruction requires two cycles to execute.
2  

**RPSL**  

**READ PROGRAM SOURCE LEFT HALF** to DATA PAD BUS (RELATIVE)

**Assembler Format:**  

RPSL adr  

**Effect:**  

\( (PS^{LH}_{VALUE} + PSA) \rightarrow DB \)

3  

**RPSF**  

**READ PROGRAM SOURCE FLOATING-POINT LITERAL** to DATA PAD BUS (RELATIVE)

**Assembler Format:**  

RPSF addr  

**Effect:**  

\( (PS^{FPL}_{VALUE} + PSA) \rightarrow DB \)

This instruction requires two cycles to execute.

4  

**RPSLT**  

**READ PROGRAM SOURCE LEFT HALF** to DATA PAD BUS from the location specified by TABLE MEMORY ADDRESS (ABSOLUTE)

**Assembler Format:**  

RPSLT  

**Effect:**  

\( (PS^{LH}_{TMA}) \rightarrow DB \)

This instruction requires two cycles to execute.
5  

**RPSFT**

READ PROGRAM SOURCE \(^{\text{FLOATING-POINT LITERAL}}\) to DATA PAD BUS from the location specified by TABLE MEMORY ADDRESS (ABSOLUTE)

Assembler  
Format: \(\text{RPSFT}\)  
Effect: \((\text{PS}^{\text{FPL}} \text{TMA}) \rightarrow \text{DB}\)

This instruction requires two cycles to execute.

---

6  

**RPSLP**

READ PROGRAM SOURCE \(^{\text{LEFT HALF}}\) to DATA PAD BUS from the address contained on PANEL BUS (ABSOLUTE)

Assembler  
Format: \(\text{RPSLP}\)  
Effect: \((\text{PS}^{\text{LH}} \text{PNLBS'}) \rightarrow \text{DB}\)

This instruction requires two cycles to execute.

---

7  

**RPSFP**

READ PROGRAM SOURCE \(^{\text{FLOATING-POINT LITERAL}}\) to DATA PAD BUS from the address contained on PANEL BUS (ABSOLUTE)

Assembler  
Format: \(\text{RPSFP}\)  
Effect: \((\text{PS}^{\text{FPL}} \text{PNLBS}) \rightarrow \text{DB}\)

This instruction requires two cycles to execute.
VALUE

LPSLA

LOAD PROGRAM SOURCE LEFT HALF from DATA PAD BUS (ABSOLUTE)

Assembler
Format: LPSLA adr
Effect: (DB) \rightarrow PS^LH \text{ VALUE}

Description: The right-most 32 bits of the DATA currently enabled onto DATA PAD BUS (DB) are loaded into the PROGRAM SOURCE(LEFT HALF) (PS LH]=PS bits 00-31]).

The transfer is executed in the following manner:

* DB(EXP) Bits 08-11 \rightarrow PS(Bits 00-03)
* DB(MANT) Bits 00-27 \rightarrow PS(Bits 04-31)

This instruction requires two cycles to execute.
**LOAD PROGRAM SOURCE RIGHT HALF from DATA PAD BUS (ABSOLUTE)**

**Description:** The right-most 32 bits of data currently enabled onto the DATA PAD BUS (DB) are loaded into the PROGRAM SOURCE(RIGHT HALF) (PS [RH]=PS bits 48-63) as addressed by the least-significant 12 bits of the VALUE field (Instruction Word bits 48-63).

The transfer is executed in the following manner:

- **DB(EXP) Bits 08-11** → **PS(Bits 32-35)**
- **DB(MANT) Bits 00-27** → **PS(Bits 36-63)**

This instruction requires two cycles to execute.
PROGRAM SOURCE FIELD (PS)

<table>
<thead>
<tr>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPSL</td>
<td>LPSR</td>
<td>LPSLT</td>
</tr>
</tbody>
</table>

**12. LPSL**

LOAD PROGRAM SOURCE**LEFT HALF** from DATA PAD BUS (RELATIVE)

**Assembler Format:**

LPSL addr

**Effect:**

(DB) \( \rightarrow \) PS\(^LH\) VALUE + PSA

This instruction requires two cycles to execute.

**13. LPSR**

LOAD PROGRAM SOURCE**RIGHT HALF** from DATA PAD BUS (RELATIVE)

**Assembler Format:**

LPSR addr

**Effect:**

(DB) \( \rightarrow \) PS\(^RH\) VALUE + PSA

This instruction requires two cycles to execute.

**14. LPSLT**

LOAD PROGRAM SOURCE**LEFT HALF** from DATA PAD BUS as addressed by TABLE MEMORY ADDRESS (ABSOLUTE)

**Assembler Format:**

LPSLT

**Effect:**

(DB) \( \rightarrow \) PS\(^LH\) TMA

This instruction requires two cycles to execute.
PROGRAM SOURCE FIELD (PS)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

VALUE

15 LPSRT

LOAD PROGRAM SOURCE RIGHT HALF from DATA PAD BUS as
addressed by TABLE MEMORY ADDRESS (ABSOLUTE)

Assembler
Format: LPSRT
Effect: (DB) → PS RH TMA

This instruction requires two cycles to execute.

16 LPSLP

LOAD PROGRAM SOURCE LEFT HALF from DATA PAD BUS at the
address contained on PANEL BUS (ABSOLUTE)

Assembler
Format: LPSLP
Effect: (DB) → PS LH PNLBS

This instruction requires two cycles to execute.

17 LPSRP

LOAD PROGRAM SOURCE RIGHT HALF from DATA PAD BUS at the
address contained on PANEL BUS (ABSOLUTE)

Assembler
Format: LPSRP
Effect: (DB) → PS RH PNLBS

This instruction requires two cycles to execute.
SET EXIT FIELD (SETEXIT)

No Operation

SET EXIT ADDRESS (ABSOLUTE)

Assembler Format: SETEXA
Effect: \( (\text{VALUE}) \rightarrow \text{SRS}_{\text{SRA}} \)

Description: The contents of the current SUB-ROUTINE RETURN ADDRESS (SRS[SRA]), the "last-in" address, are replaced by the least-significant 12 bits of the VALUE field (Instruction Word[bits 48-63]). SRA not affected.

SET EXIT ADDRESS (RELATIVE)

Assembler Format: SETEX adr
Effect: \( (\text{VALUE} = \text{PSA}) \rightarrow \text{SRS}_{\text{SRA}} \)

Description: The contents of the current SUB-ROUTINE RETURN ADDRESS (SRS[SRA]), the "last-in" address, are replaced with the address formed by adding the current contents of the PROGRAM SOURCE ADDRESS (PSA) to the least-significant 12 bits contained on the VALUE field.
SET EXIT FIELD (SETEXIT)

Assembler Format:

Effect: (TMA) \rightarrow SRS_{SRA}

Description: The contents of the current SUB-Routine RETURN ADDRESS (SRS_{SRA}), the "last-in" address, are replaced by the least-significant 12 bits of the current contents of the TABLE MEMORY ADDRESS REGISTER (TMA).

SETEXIT

SET EXIT ADDRESS from TABLE MEMORY ADDRESS (ABSOLUTE)

Assembler Format: SETEXIT

Effect: (TMA) \rightarrow SRS_{SRA}

Description: The contents of the current SUB-Routine RETURN ADDRESS (SRS_{SRA}), the "last-in" address, are replaced by the least-significant 12 bits of the current contents of the TABLE MEMORY ADDRESS REGISTER (TMA).

SETEXP

SET EXIT ADDRESS from PROGRAM SOURCE ADDRESS plus ONE (ABSOLUTE)

Assembler Format: SETEXP

Effect: (PSA + 1) \rightarrow SRS_{SRA}

Description: The contents of the current SUB-Routine RETURN ADDRESS (SRS_{SRA}), the "last-in" address, are replaced by the current contents of the PROGRAM SOURCE ADDRESS REGISTER (PSA) plus "1"
BRANCH GROUP (BRANCH)

VALUE

Ø                No Operation

1    #                S-PAD DESTINATION REGISTER NO-LOAD

Assembler
Format:          < # >

(Example: ADD# 4,5)

Effect:          (SPFN) → SP_{SPD} is inhibited.

Description: This instruction inhibits the normal loading of the current
S-PAD OPERATION result (SPFN) back into the S-PAD DESTINATION REGISTER
(SP_{SPD}) specified during the S-PAD Operation. (See S-PAD summary)

NOTE: Brackets indicate optional use with S-PAD operations.
UNCONDITIONAL BRANCH (RELATIVE)

Assembler Format:  
BR disp  
(Examples: BR LOOP)

Effect:  
(PSA) + (DISP † - BIAS) + PSA  
(Where BIAS = 20₈).

Description: UNCONDITIONAL RELATIVE BRANCH

The program will branch to the target location "disp" (Assembler Format) by adding the current PSA to the BIASED value contained in the DISPlacement field of the instruction word.

NOTE: The BRANCH TARGET ADDRESS must be within a range of -20(8) to +17(8) locations relative to the current PROGRAM SOURCE ADDRESS (PSA).

DISP = bits 27-31 of the current instruction word and is completed as follows:  
† DISP = disp - PSA + BIAS.
BRANCH GROUP (BRANCH)

<table>
<thead>
<tr>
<th>23 24 25 26 27 28 29 30 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE</td>
</tr>
</tbody>
</table>

VALUE

3  

**BINTRQ**  
BRANCH ON INTERRUPT REQUEST FLAG NON-ZERO

Assembler Format:  
BINTRQ disp

Effect:  
If INTRQ = 1, then (PSA) + (DISP - BIAS) → PSA  
(Where BIAS = 208).

Description: CONDITIONAL RELATIVE BRANCH if Interrupt Request.

If the INTERRUPT REQUEST FLAG (INTRQ) equals "1," then the program will branch to the target location "disp."

This instruction can be used in conjunction with a succeeding INTA instruction (see I/O group) to identify the interrupting I/O device.

4  

**BION**  
BRANCH ON I/O DATA READY FLAG NON-ZERO

Assembler Format:  
BION disp

Effect:  
If IODRDY (DA) = 1, then (PSA) + (DISP - BIAS) → PSA  
(Where BIAS = 208).

Description: CONDITIONAL RELATIVE BRANCH is I/O Device Ready.

If the I/O DATA READY FLAG (IODRDY) of the I/O device specified by the I/O DEVICE ADDRESS REGISTER (DA) is "1," then the program will branch to the target location "disp."
VALUE

5

<table>
<thead>
<tr>
<th>23 24 25 26 27 28 29 30 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>COND</td>
</tr>
</tbody>
</table>

BIOZ

BRANCH ON I/O DATA READY FLAG ZERO

Assembler Format: BIOZ disp
Effect: If (IODRDY) = 0, then (PSA) + (DISP - BIAS) + PSA
(Where BIAS = 208).

Description: CONDITIONAL RELATIVE BRANCH if I/O Device not ready.

If the I/O DATA READY FLAG (IODRDY) of the I/O DEVICE specified by the I/O DEVICE ADDRESS REGISTER (DA) equals "0," then the program will branch to the target location "disp."

6

BFPE

BRANCH ON FLOATING POINT ERROR

Assembler Format: BFPE disp
Effect: If OVF, UNF, or DIVZ = "1", then (PSA) + (DISP - BIAS) + PSA
(Where BIAS = 208).

Description: CONDITIONAL RELATIVE BRANCH if Floating Point Error.

The OVERFLOW (OVF), UNDERFLOW (UNF), and DIVIDE BY ZERO (DIVZ), FLAGS (bits 0, 1, 2 of the APSTATUS REGISTER) are tested. If any of the three flags = "1," then a branch will occur to the target location "disp."
BRANCH GROUP (BRANCH)

VALUE

7

RETURN

RETURN FROM SUB-ROUTINE

Assembler
Format: RETURN

Effect: 

\[(SRS_{\text{SRA}}) + \text{PSA}; (\text{SRA}) - 1 + \text{SRA}\]

Description: UNCONDITIONAL RETURN JUMP.

The address contained in the "last-in" position of the SUB-ROUTINE RETURN STACK (SRS) is forced into the PROGRAM SOURCE ADDRESS REGISTER (PSA) and the program branches to that program location.

The SUB-ROUTINE ADDRESS POINTER REGISTER (SRA) is then decremented by "1," and will point to the next "last-in" SUB-ROUTINE ADDRESS in event of another RETURN instruction. RETURN effects a "RETURN" from the last SUB-ROUTINE call.

NOTE: Two or more RETURNS may not be executed in time sequential instructions. There must be at least one instruction cycle between the last RETURN instruction and the next one, e.g., the following coding example is illegal in that it results in having the visible RETURN instruction execute immediately after the RETURN instruction in "sub" that brings the processor back to this level.

illegal code: JSR sub RETURN
BRANCH GROUP (BRANCH)

<table>
<thead>
<tr>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>COND</td>
<td>DISP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VALUE

10

BFEQ

BRANCH on FLOATING ADDER EQUAL ZERO

Assembler Format: BFEQ disp

Example: BFEQ .-3

Effect: If FA = 0.0, then (PSA) + (DISP - BIAS) → PSA

(Where BIAS = 208).

Description: CONDITIONAL RELATIVE BRANCH if FA equal to zero.

BFEQ will cause a PROGRAM BRANCH if the FLOATING ADDER Result (FA) available during the previous instruction was equal to 0.0. This instruction tests the FZ FLAG (bit 3 of APSTATUS) as set by the previous instruction. If FZ is equal to "1," (i.e., FA during the last instruction was equal to 0.0), then the program will branch to the target location "disp."
VALUE

11

BFNE

BRANCH on FLOATING ADDER NOT EQUAL to ZERO

Assembler
Format: BFNE disp
Example: BFNE HELP+6

Effect: If FA ≠ 0.0, then (PSA) + (DISP - BIAS) → PSA
(Where BIAS = 20g).

Description: CONDITIONAL RELATIVE BRANCH if FA not equal to zero.

BFNE will cause a PROGRAM BRANCH if the FLOATING ADDER Result for the
previous instruction was not equal to 0.0. This instruction tests the
FZ flag (bit 3 of APSTATUS) as set by the previous instruction. If FZ
equals "0" (i.e., FA for the last instruction was not equal to 0.0),
the branch will occur to the target location "disp."
BFGE

BRANCH on FLOATING ADDER GREATER or EQUAL to ZERO

Assembler
Format: BFGE disp

Effect: If FA > 0.0, then (PSA) + (DISP - BIAS) -> PSA
(Where BIAS = 20_8).

Description: CONDITIONAL RELATIVE BRANCH if FA greater than or equal to zero.

BFGE will cause a PROGRAM BRANCH if the FLOATING ADDER Result (FA) for the previous instruction was greater than or equal to 0.0.

This instruction tests the condition of the FLOATING ADDER NEGATIVE (FN) FLAG (bit 4 of APSTATUS) as set by the previous instruction.

If FN equals "0," (indicating that FA was not negative, i.e., greater than or equal to zero, during the last instruction cycle), a branch will occur to the TARGET ADDRESS "disp."
VALUE

13  BFGT  BRANCH on FLOATING ADDER GREATER THAN ZERO

Assembler Format:  BFGT  disp

Effect:  If FA > 0.0, then (PSA) + (DISP - BIAS)* PSA  
         (Where BIAS = 208)

Description:  CONDITIONAL RELATIVE BRANCH if FA greater than zero.

BFGT will cause a PROGRAM BRANCH to occur if the last FLOATING ADDER RESULT (FA) for the previous instruction was greater than 0.0.

The instruction tests the FLOATING ADDER ZERO (FZ) and FLOATING ADDER NEGATIVE (FN) flags (bits 3 and 4 of APSTATUS) as set by the previous instruction.

If both flags equal "0" (indicating that FA during the last instruction was greater than zero), then the program will branch to the target location "disp."
BRANCH GROUP (BRANCH)

<table>
<thead>
<tr>
<th>COND</th>
<th>DISP</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
</tr>
<tr>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>

VALUE

14

BEQ

BRANCH on S-PAD RESULT EQUALS ZERO

Assembler Format: BEQ disp

Effect: If SPFN = Ø, then (PSA) + (DISP - BIAS) = PSA
(Where BIAS = 2^8).

Description: CONDITIONAL RELATIVE BRANCH if SPFN equals zero.

BEQ will cause a PROGRAM BRANCH if the result of the last S-PAD operation (SPFN) was equal to zero.

This instruction tests the S-PAD ZERO FLAG (Z) (bit 5 of APSTATUS) as set by the previous instruction. If Z equals "1" (indicating that SPFN of the last S-PAD operation was equal to zero), then a branch will occur to the target location "disp."
BRANCH GROUP (BRANCH)

23 24 25 26 27 28 29 30 31
\[\begin{array}{c|c}
\text{COND} & \text{DISP} \\
\end{array}\]

VALUE

15 BNE BRANCH on S-PAD RESULT NON-ZERO

Assembler Format: BNE disp

Effect: If SPFN ≠ ∅, then \((\text{PSA}) + (\text{DISP} + \text{BIAS})\) + PSA
(Where BIAS = 20).\(^{\text{a}}\)

Description: CONDITIONAL RELATIVE BRANCH if SPFN not equal to zero.

BNE will cause a PROGRAM RELATIVE BRANCH if the result of the last previous S-PAD operation (SPFN) was not equal to 0.

The instruction tests the S-PAD ZERO FLAG (Z) (bit 5 of APSTATUS) as set by the previous instruction. If Z equals "0" (indicating that SPFN of the last S-PAD operation was not equal to zero), then a branch will occur to the target location "disp."
BRANCH GROUP (BRANCH)

<table>
<thead>
<tr>
<th>23 24 25 26 27 28 29 30 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>COND</td>
</tr>
</tbody>
</table>

VALUE

16  BGE  BRANCH on S-PAD RESULT GREATER or EQUAL to ZERO

Assembler Format:
BGE  disp

Effect:
If SPFN > 0, then (PSA) + (DISP - BIAS) → PSA
(Where BIAS = 2^{08}).

Description: CONDITIONAL RELATIVE BRANCH if SPFN greater than or equal to zero.

BGE will cause a PROGRAM BRANCH if the result of the last S-PAD operation (SPFN) was greater than or equal to zero.

The instruction tests the S-PAD NEGATIVE FLAG (N) (bit 6 of APSTATUS) as set by the previous instruction. If N is equal to "0" (indicating SPFN of the last operation was zero or greater), a branch will occur to the target location "disp."
BGT  \hspace{0.5cm} \text{BRANCH on S-PAD RESULT GREATER THAN ZERO}

Assembler Format: \hspace{0.5cm} \text{BGT \ disp}

Effect: \hspace{0.5cm} \text{If SPFN > 0, then (PSA) + (DISP - BIAS) \rightarrow PSA}
\hspace{0.5cm} \text{(Where BIAS = 20\text{e}).}

Description: \text{CONDITIONAL RELATIVE BRANCH if SPFN greater than zero.}

\text{BGT} \hspace{0.2cm} \text{will cause a PROGRAM BRANCH if the result of the last S-PAD operation (SPFN) was greater than zero.}

The instruction tests the S-PAD ZERO (Z) and S-PAD NEGATIVE (N) flags (bits 5, 6 or APSTATUS) as set by the previous instruction. If both flags equal "0" (indicating SPFN of last S-PAD operation was greater than zero), then a branch will occur to the target location "disp."
FLOATING ADDER GROUP (FADD)

<table>
<thead>
<tr>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>FADD</td>
<td>A1</td>
<td>A2</td>
<td></td>
<td></td>
<td></td>
<td>MANDATORY FIELDS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OPTIONAL FIELDS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DISABLED FIELDS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VALUE
0    See FADD1 field.

1    FSUBR

FLOATING-POINT SUBTRACT REVERSE;
A2 minus A1.

Assembler Format:  FSUBR < A1, A2 > (< > indicates optional fields)

Effect:           (A2) - (A1)
FSUBR  TM,MD

Description: FSUBR reverses the order of operands in a FLOATING-POINT SUBTRACTION. The contents of A1 REGISTER (A1) undergo a FLOATING-POINT SUBTRACTION from the contents of A2 REGISTER (A2).

The NORMALIZED, CONVERGENTLY-ROUNDED RESULT is available as the FLOATING-ADDER OUTPUT (FA) one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)

2    FSUB

FLOATING-POINT SUBTRACT; A1 minus A2.

Assembler Format:  FSUB < A1, A2 >

Effect:           (A1) - (A2)

Description: The contents of A2 REGISTER (A2) undergo a FLOATING-POINT SUBTRACTION from the contents of A1 REGISTER (A1).

The NORMALIZED, CONVERGENTLY-ROUNDED RESULT becomes available as the FLOATING ADDER OUTPUT (FA) one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)
FLOATING ADDER GROUP (FADD)

Assembler Format:  
FADD < A1, A2 >

Effect:  
(A1) + (A2)

Description: The contents of A1 REGISTER undergo a FLOATING-POINT ADDITION with the contents of A2 REGISTER.

The NORMALIZED, CONVERGENTLY-ROUNDED RESULT becomes available as the FLOATING ADDER OUTPUT (FA) one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)
FLOATING ADDER GROUP (FADD)

Mandatory Fields

Optional Fields

Disabled Fields

VALUE

FLOATING-POINT LOGICAL EQUIVALENCE:
A1 with A2.

Assembler Format:
FEQV < A1, A2 >

Effect:
(AL) XOR (A2)

Description: The MANTISSAS of A1 and A2 are compared for EQUIVALENCE in the following manner:

Following the arithmetic right-shift of the MANTISSA corresponding to the smaller EXPONENT, the MANTISSAS of A1 and A2, including the three bits of residue, undergo a "bit by bit" comparison. When corresponding bits of A1 and A2 are equal; (i.e., both '0's or both '1's), a '1' is written into the corresponding bit of the RESULT. All other combinations result in a '0' being written.

The NORMALIZED, CONVERGENTLY-ROUNDED RESULT of this logical operation becomes available as the FLOATING ADDER OUTPUT (FA) one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)
FLOATING ADDER GROUP (FADD)

<table>
<thead>
<tr>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>FADD</td>
<td>Al</td>
<td>A2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

5

FAND

FLOATING-POINT AND; A1, A2.

Assembler Format: FAND < A1, A2 >

Effect: (A1) and (A2)

Description: The MANTISSAS of A1 and A2 are logically ANDED in the following manner:

Following the arithmetic right-shift of the MANTISSA corresponding to the smaller EXPONENT, the MANTISSAS of A1 and A2 undergo a "bit by bit" comparison. When corresponding bits of A1 and A2 both equal "1," then a "1" is written into the corresponding bit position of the RESULT. All other combinations result in a "0" being written.

The NORMALIZED, CONVERGENTLY-ROUNDED RESULT of this logical operation becomes available as the FLOATING ADDER OUTPUT (FA) one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)
FLOATING ADDER GROUP (FADD)

Description: The contents of A1 and A2 are ORed in the following manner:

Following the arithmetic right-shift of the MANTISSA corresponding to the smaller EXPONENT, the MANTISSAS of A1 and A2, including the three bits of residue, undergo a "bit by bit" comparison. When either or both corresponding bits of A1 and A2 equal "1," a "1" is written into the corresponding bit position of the RESULT. When neither corresponding bit is equal to "1," a "0" is written.

The NORMALIZED and CONVERGENTLY-ROUNDED RESULT of this logical operation becomes available as the FLOATING ADDER OUTPUT (FA) one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)
FLOATING ADDER GROUP (FADD)

No-operation.

**FIX**

FIX A2 to an INTEGER (result rounded)

Assembler Format: FIX <A2>  (< > indicates optional field)

Effect: Convert (A2) to a 28-Bit Two's Complement integer.

Example: FIX MD

Description: The contents of A2 are FIXED to an integer in the following manner:

1) An exponent of 28 (apparent value = 1034[octal]) is forced into A1(EXPONENT).
   A1(MANTISSA) = 0. A2 contains the selected argument to be FIXED.†

2) The EXPONENTS of the operands are compared and the MANTISSA corresponding to the smaller EXPONENT is arithmetically right-shifted the number of positions that reflect the difference in the two EXPONENTS. The aligned MANTISSAS are then algebraically added producing a PRELIMINARY-RESULT along with the larger of the two input exponents.

3) The PRELIMINARY-RESULT(EXPONENT) is decremented by "1" and the PRELIMINARY-RESULT(MANTISSA) is correspondingly left-shifted one position while preserving the MANTISSA-SIGN.

4) The PRELIMINARY-RESULT is then CONVERGENTLY-ROUNDED and becomes available as FA one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)

If the TRUE-VALUE of A2(EXPONENT) was in a range of 1 to 27, then the TRUE-VALUE of the RESULT(EXPONENT) will be 27 (APPARENT-VALUE=1033[octal] or 539). If RESULT(MANTISSA) is \leq 2(-28), then a FLOATING-POINT ZERO is forced as the result.

†The TRUE-VALUE of A2(EXPONENT) must not exceed +27.

APPARENT-VALUE\leq 1033[octal]. If the TRUE-VALUE of A2(EXPONENT)\geq 28, the following RESULT will be obtained:

RESULT(EXPONENT)=A2(EXPONENT) minus "1;"  RESULT(MANTISSA-SIGN)=A2(MANTISSA-SIGN),

RESULT(MANTISSA Bits 01-26)=A2(MANTISSA Bits 02-27); RESULT (MANTISSA Bit 27)=0.
FLOATING ADDER GROUP (FADD)

VALUE

<table>
<thead>
<tr>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>A1</td>
<td>A2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

FIXT

FIX and TRUNCATE A2 to an INTEGER

Assembler Format: FIXT <A2>

Effect: Convert (A2) to a 28-Bit Two's Complement INTEGER; TRUNCATE (sign magnitude)

Example: FIXT DPX(3)

Description: The contents of A2(MANTISSA) are FIXED to an integer and the RESULT is TRUNCATED in the following manner:

1) Current SPFN(Bits 06-15) plus BIAS are forced into A1(EXPONENT). A1(MANTISSA)=0 . A2 contains the selected argument to be FIXED.†

2) The EXPONENTS of the operands are compared and the MANTISSA corresponding to the smaller EXPONENT is arithmetically right-shifted the number of positions that reflect the difference in the two EXPONENTS. The aligned MANTISSAS are then algebraically added producing a PRELIMINARY-RESULT.

3) The PRELIMINARY-RESULT(EXPONENT) is decremented by "1" and the PRELIMINARY-RESULT(MANTISSA) is correspondingly left-shifted. (This operation preserves the MANTISSA-SIGN following an internal sign-extension operation).

4) The Truncation truth table logic is enabled for this operation (see Floating Point Arithmetic theory) and a TRUNCATED RESULT becomes available as FA one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)

If the TRUE-VALUE of A2(EXPONENT) was in a range of 1 to 27, then the TRUE-VALUE of the RESULT(EXPONENT) will be 27 (APPARENT-VALUE=1033[octal] or 539). If RESULT(MANTISSA) is<2(-27), then a FLOATING-POINT ZERO is forced as the result.

†TRUE-VALUE of A2(EXPONENT) must not exceed +27. (APPARENT-VALUE 539). If TRUE-VALUE of A2(EXPONENT) 28, the following RESULT will be obtained:

RESULT(EXPONENT)=A2(EXPONENT)minus "1"; RESULT(MANTISSA-SIGN)=A2(MANTISSA-SIGN), RESULT(MANTISSA Bits 01-26)=A2(MANTISSA Bits 02-27);RESULT(MANTISSA Bit 27)=0.
FLOATING ADDER GROUP (FADD)

<table>
<thead>
<tr>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A1</td>
<td>A2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**VALUE**

3

**FSCLT**

**FLOATING-POINT SCALE of A2; TRUNCATE**

**Assembler Format:**

FSCLT A2

**Effect:**

Shift A2 MANTISSA right and increment

\[ (A2^{\text{EXPONENT}}) \text{ until } A2^E = \text{(SPFN + BIAS)} - 1; \]

result TRUNCATED. Converts an FPN to a 28-Bit Two's Complement integer within a dynamic range of \(2^{127}\).

**Description:** The contents of A2(MANTISSA) are SCALED in the following manner:

1) Current SPFN(Bits 06-15) plus BIAS(=512) are forced into Al(EXPONENT). Al(MANTISSA)=0. A2 contains the selected argument to be SCALED.††

2) The EXPONENTS of the operands are compared and the MANTISSA corresponding to the smaller EXPONENT is arithmetically right-shifted the number of positions that reflect the difference in the two EXPONENTS. The aligned MANTISSAS are then algebraically added producing a PRELIMINARY-RESULT.

3) The PRELIMINARY-RESULT(EXPONENT) is decremented by "1" and the PRELIMINARY-RESULT(MANTISSA) is correspondingly left-shifted. (This operation preserves the MANTISSA-SIGN following an internal sign-extension operation).

4) The CONVERGENT-ROUNDING logic is inhibited for this operation and a TRUNCATED RESULT becomes available as FA one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY).

If the TRUE-VALUE OF A2(EXPONENT) was in a range of 1 to 27, then the TRUE-VALUE of the RESULT(EXPONENT) will be 27 (APPARENT-VALUE=1033 [octal] or 539). If RESULT(MANTISSA) is \(<2(-28)\), then a FLOATING-POINT ZERO is forced as the result.

†Current SPFN( Bits 06-15) must equal maximum A2 Exponent plus "1" in order to obtain a correct result from this operation.

†† TRUE-VALUE of A2(EXPONENT) must not exceed the value of current SPFN minus "1." If it does, the result obtained will be the same as in the case of FIX or FIXT when A2(EXP) \(\geq 28\).
**FLOATING ADDER 1 FIELD (FADD)**

<table>
<thead>
<tr>
<th>14 15 16 17 18 19 20 21 22</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0          A1 A2</td>
</tr>
<tr>
<td>MANDATORY FIELDS</td>
</tr>
<tr>
<td>OPTIONAL FIELDS</td>
</tr>
<tr>
<td>DISABLED FIELDS</td>
</tr>
</tbody>
</table>

**VALUE**

<table>
<thead>
<tr>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSM2C</td>
</tr>
</tbody>
</table>

**FORMAT-CONVERSION; A2 from SIGNED-MAGNITUDE to TWO'S COMPLEMENT**

**Assembler Format:**

FSM2C A2

**Effect:**

Converts (A2) from SIGNED-MAGNITUDE to TWOS-COMPLEMENT.

**Description:** The contents of A2 REGISTER are converted from SIGNED-MAGNITUDE to TWOS-COMPLEMENT format in the following manner:

1) If A2(MANTISSA) is negative, then A2(MANTISSA bit 00) (MANTISSA-SIGN) remains unchanged while A2(MANTISSA Bits 01-27) undergo a TWOS-COMPLEMENT conversion.

2) If A2(MANTISSA) is positive, A2(MANTISSA) is unchanged.

The normalized RESULT becomes available as the FLOATING ADDER OUTPUT (FA) one cycle after the next FADDR group operation. This operation can result in Floating Point underflow if the exponent is large and negative and the mantissa unnormalized. (See FADDR SUMMARY).

†See FLOATING-POINT THEORY, Types of notation.

This operation can result in FLOATING-POINT underflow if the exponent is large and negative and the mantissa unnormalized.
FLOATING ADDER 1 FIELD (FADD)

<table>
<thead>
<tr>
<th>0 0 0</th>
<th>A2</th>
</tr>
</thead>
</table>

MANDATORY FIELDS
OPTIONAL FIELDS
DISABLED FIELDS

F2CSM

FORMAT CONVERSION; A2 from TWOS-COMPLEMENT to SIGNED-MAGNITUDE.

Assembler
Format: F2CSM A2

Effect: Convert (A2) from TWOS-COMPLEMENT to SIGNED-MAGNITUDE.

Description: The contents of A2 REGISTER are converted to SIGN-MAGNITUDE format in the following manner:

1) If A2(MANTISSA) is positive, A2(MANTISSA) is unchanged.

2) If A2(MANTISSA) is negative, then A2(MANTISSA Bit 00) (MANTISSA-SIGN) remains unchanged while A2(MANTISSA Bits 01-27) undergo a TWOS-COMPLEMENT conversion.

The normalized and convergently rounded RESULT becomes available as the FLOATING ADDER OUTPUT (FA) one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY). Both underflow and overflow are possible as a result of this operation.

†See FLOATING-POINT SUMMARY, Types of Notation
## Floating Adder 1 Field (FADD)

### Floating-Point Scale of A2 (rounded result)

<table>
<thead>
<tr>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>A1</td>
<td>A2</td>
<td>I/O</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Assembler Format:**

- Shift (A2 MANTISSA) right and increment
- (A2 EXPONENT) until (A2 EXPONENT) = (SPFN) + BIAS -1

**Description:** The contents of A2 are scaled in the following manner:

1. Current SPFN (Bits 06-15) plus BIAS (=512) are forced into A1(EXPONENT). A1(MANTISSA) = 0†. A2 contains the selected argument to be scaled. ‡‡

2. The EXPONENTS of the operands are compared and the MANTISSA corresponding to the smaller EXPONENT is arithmetically right-shifted the number of positions that reflect the difference in the two EXPONENTS. The aligned MANTISSAS are then algebraically added producing a PRELIMINARY-RESULT.

3. The PRELIMINARY-RESULT(EXPONENT) is decremented by "1" and the PRELIMINARY-RESULT(MANTISSA) is correspondingly left-shifted. (This operation preserves the MANTISSA-SIGN following an internal sign-extension operation).

4. The PRELIMINARY-RESULT is then CONVERGENTLY-ROUNDED and becomes available as FA one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY.)

If RESULT(MANTISSA) ≤ 2(-28), then a FLOATING-POINT ZERO is forced as the result.

† Current SPFN (Bits 06-15) must equal maximum plus "1" in order to obtain a correct result from this operation.

‡‡ True-value of A2(EXPONENT) must not exceed the value of current SPFN minus "1."
**Assembler Format:**
FABS A2

**Effect:**
(A2) → ABSOLUTE VALUE

**Description:** The contents of A2 are converted to ABSOLUTE VALUE format in the following manner:

1) If A2(MANTISSA) is positive, A2(MANTISSA) is unchanged. A2 becomes the PRELIMINARY RESULT.

2) If A2(MANTISSA) is negative, then A2(MANTISSA Bits 00-27) undergo a TWOS-COMPLEMENT conversion. (See FLOATING-POINT SUMMARY -- Types of Notation).

The PRELIMINARY RESULT is then normalized and CONVERGENTLY-ROUNDED and becomes available as FA one cycle after the next FADDR group instruction is initiated. (See FADDR SUMMARY). Both overflow and underflow are possible as a result of this operation.
FLOATING-ADDER GROUP (A1)

<table>
<thead>
<tr>
<th>14 15 16 17</th>
<th>18 19 20 21 22</th>
</tr>
</thead>
<tbody>
<tr>
<td>FADD</td>
<td>A1</td>
</tr>
<tr>
<td>FADD1</td>
<td></td>
</tr>
</tbody>
</table>

MANDATORY FIELDS
OPTIONAL FIELDS
DISABLED FIELDS

FLOATING ADDER REGISTER INPUT SUMMARY

A1

VALUE

Ø  

Assembler Format:

Effect:

NO CHANGE: The contents of A1 during the last FADD operation are used as the A1 REGISTER OPERAND.

FADD NC, <A2> Note: If no A1 operand is specified, NC is implied. A1 is unaltered from previous operation.

1  

FM

Assembler Format:

Effect:

The current FLOATING-MULTIPLIER OUTPUT (FM) is the A1 REGISTER OPERAND.

FSUB FM, <A2> (FM) → A1

2  

DPX (idx)

Assembler Format:

Effect:

DATA PAD X (DPA + XR -4) is the A1 REGISTER OPERAND.

FSUBR DPX (idx), <A2> [DPX(DPA + XR -4)] → A1

3  

DPY (idx)

Assembler Format:

Effect:

DATA PAD Y (DPA + YR -4) is A1 REGISTER OPERAND.

FOR DPY (idx), <A2> [DPY(DPA + YR -4)] → A1

E - 93
The current contents of the TABLE MEMORY REGISTER (TMREG) is the Al REGISTER OPERAND (See MEMORY GROUP SUMMARY - TM)

Assembler Format:

```
FAND TM, <A2>
```

Effect:

 LâmTMREG) → Al

FLOATING-POINT ZERO (0.0) is the Al REGISTER OPERAND.

Assembler Format:

```
FEQV ZERO, <A2>
```

Effect:

0.0 → Al
FLOATING ADDER GROUP

FLOATING ADDER REGISTER INPUT SUMMARY

A2

OCTAL
VALUE

<table>
<thead>
<tr>
<th>OCTAL VALUE</th>
<th>NC</th>
<th>FA</th>
<th>DPX (idx)</th>
<th>DPY (idx)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FADD &lt;A1&gt;, NC</td>
<td>(A2) is unaltered from previous operation.</td>
<td>DATA PAD X (DPA + XR -4) is the A2 REGISTER operand.</td>
<td>DATA PAD Y (DPA + YR -4) is the A2 REGISTER operand.</td>
</tr>
<tr>
<td>1</td>
<td>FIX FA</td>
<td>(FA) \rightarrow A2</td>
<td>FSCLT DPX(idx) \rightarrow A2</td>
<td>FABS DPY(idx) \rightarrow A2</td>
</tr>
</tbody>
</table>

The contents of A2 during the last FADDR operation are used as the A2 REGISTER operand.
The current contents of the MAIN DATA MEMORY OUTPUT REGISTER (MDREG) are used as the A2 REGISTER operand. (See MEMORY GROUP SUMMARY - MD).

Assembler Format:  
F2CSM MD  
Effect:  
(MDREG) → A2

FLOATING POINT ZERO (0.0) is the A2 REGISTER operand.

Assembler Format:  
PADD <A1>, ZERO  
Effect:  
0.0 → A2

"Split-word" transfer to A2. (mantissa of DPX)

(1) The SPAD FUNCTION (SPFN) plus the BIAS-VALUE (512) forms the EXPONENT portion of the A2 OPERAND.

(2) The MANTISSA portion of DATA PAD X(DPA + XR -4) forms the MANTISSA portion of the A2 OPERAND.

Assembler Format:  
FSUB <A1>, MDPX (idx)  
Effect:  
(SPFN) + 512 → A2^EXPONENT;  
[DPX (DPA + XR -4)] → A2^MANTISSA

"Split-word" transfer to A2. (exponent of DPX)

(1) The EXPONENT of DATA PAD X(DPA + XR -4) forms the EXPONENT portion of the A2 OPERAND.

(2) The 2 least-significant bits of SPAD FUNCTION (SPFN bits 14,15) are put into A2^MANTISSA bits ØØ, 01. The remainder of A2^MANTISSA is zeroed.
EDPX (idx) is used to generate either a
\( \pm 1 \) or \(-1\) MANTISSA value.

**Assembler Format:**

FSUBR <A1>, EDPX (idx)

**Effect:**

\[
\begin{align*}
\text{[DPX (DPAXR -4)]} & \rightarrow A_2^\text{EXPONENT} \\
\text{(SPPN Bits 14, 15)} & \rightarrow A_2^\text{MANTISSA \ bits } 00, 01; \\
\text{Øs} & \rightarrow A_2^\text{MANTISSA \ bits 01-27}. 
\end{align*}
\]
FLOATING MULTIPLIER GROUP

VALUE
Ø No Operation

1

FMUL

FLOATING POINT MULTIPLY

Assembler
Format: FMUL <M1,M2>

Effect: (M1) * (M2)

Description: A FLOATING POINT MULTIPLY (FMUL) is initiated using the operands selected by the M1 and M2 fields.

The CONVERGENTLY-ROUNDED result becomes available as FM 1 cycle after it has been "pushed" through the 3 stage pipeline by two subsequent FMUL operations. (See, FMULR SUMMARY).

† M1 and M2 operands need not be specified if a "dummy" is desired.
M1 REGISTER SUMMARY

VALUE

 communauté

0

<table>
<thead>
<tr>
<th>FM</th>
</tr>
</thead>
</table>

Assembler
Format:  
Effect:  

FMUL  FM< M2>

(FM) → M1

1

<table>
<thead>
<tr>
<th>DPX(idx)</th>
</tr>
</thead>
</table>

Assembler
Format:  
Effect:  

FMUL DPX(idx)< M2>

(DPX(DPA + idx)) → M1

2

<table>
<thead>
<tr>
<th>DPY(idx)</th>
</tr>
</thead>
</table>

Assembler
Format:  
Effect:  

FMUL DPY(idx) < M2>

(DPY(DPA + idx)) → M1

3

<table>
<thead>
<tr>
<th>TM</th>
</tr>
</thead>
</table>

Assembler
Format:  
Effect:  

FMUL TM,< M2>

(TMREG) → M1

DATA PAD X is the M1 REGISTER OPERAND

DATA PAD Y is the M1 REGISTER OPERAND

TABLE MEMORY OUTPUT REGISTER is the M1 REGISTER OPERAND
M2 REGISTER SUMMARY

MANDATORY FIELDS
OPTIONAL FIELDS
DISABLED FIELDS

VALUE

F.1. DATING ADDER OUTPUT is the M2 REGISTER INPUT

Assembler
Format:  FMUL<MD>,FA
Effect:   (FA) \rightarrow M2

DATA PAD X as M2 REGISTER INPUT

Assembler
Format:  FMUL<MD>,DPX(idx)
Effect:   (DPX(DPA+idx)) \rightarrow M2

DATA PAD Y is the M2 REGISTER INPUT

Assembler
Format:  FMUL<MD>,DPY(idx)
Effect:   (DPY(DPA+idx)) \rightarrow M2

MAIN DATA MEMORY REGISTER is the M2 REGISTER INPUT

Assembler
Format:  FMUL<MD>,MD
Effect:   (MDREG) \rightarrow M2
LOAD REGISTER FIELD (LDREG)

<table>
<thead>
<tr>
<th>14 15 16 17 18 19 20 21 22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
</tr>
<tr>
<td>[ ]</td>
</tr>
<tr>
<td>[ ]</td>
</tr>
<tr>
<td>[ ]</td>
</tr>
<tr>
<td>[ ]</td>
</tr>
<tr>
<td>[ ]</td>
</tr>
<tr>
<td>[ ]</td>
</tr>
<tr>
<td>[ ]</td>
</tr>
</tbody>
</table>

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

VALUE

Ø No Operation

1

LDSPD

LOAD S-PAD DESTINATION ADDRESS

Assembler

Format: LDSPD

Example: LDSPD; DB=DPX(-3)

Effect:

\[(DPBS^{MANTISSA} \text{ bits } 24-27) \rightarrow SPD\]

Description: DPBS(MANTISSA) bits 24-27 replace the contents of the four-bit S-PAD DESTINATION ADDRESS REGISTER (SPD) as of the next instruction cycle.

NOTE: A current S-PAD operation is unaffected by this instruction. However, if an S-PAD operation is executed on the next instruction cycle, the assembled S-PAD DESTINATION ADDRESS (SPD) to be used in the S-PAD operation will be replaced with the contents of SPD produced as a result of this instruction.

2

LDMA

LOAD MEMORY ADDRESS from the DATA PAD BUS;

INITIATE A MEMORY CYCLE

Assembler

Format: LDMA

Example: LDMA; DB=DPX(-1)

Effect:

\[(DPBS^{MANTISSA} \text{ bits } 12-27) \rightarrow MA\]

Description: DPBS(LOW MANTISSA) bits 12-27 are loaded into the MAIN DATA MEMORY ADDRESS REGISTER (MA) effective as of the next instruction cycle. A MAIN DATA (MD) MEMORY cycle is initiated using the new contents of MA. (See MEMORY GROUP SUMMARY.)

NOTE: This op-code supersedes INCMA and DECMA in the same instruction. It makes SETMA redundant since it would now load from DB instead of SPFN due to the use of the LDREG field.

E - 101
LOAD REGISTER FIELD (LDREG)

MANDATORY FIELDS

OPTIONAL FIELDS

DISABLED FIELDS

VALUE

3

LDMA LOAD TABLE MEMORY ADDRESS from DATA PAD BUS

Assembler Format:

LDTMA

Example: LDTMA; DB = DPY(-4)

Effect:

(DPBS\(^{\text{MANTISSA}}\) bits 12-27) \rightarrow \text{TMA}

Description: The DATA PAD BUS (LOW MANTISSA) (DPBS[MANTISSA]bits 12-27) is loaded into the TABLE MEMORY ADDRESS REGISTER (TMA) effective as of the next instruction cycle.

Two cycles later the contents of the TABLE MEMORY location specified by the two new contents of TMA will become available as the contents of TABLE MEMORY OUTPUT REGISTER (TMREG).

NOTE: This op-code supersedes INCTMA and DECTMA in the same instruction. It makes SETTMA redundant.

4

LDDPA LOAD DATA PAD ADDRESS from DATA PAD BUS

Assembler Format:

LDDPA

Example: LDDPA; DB = DPX(3)

Effect:

(DPBS\(^{\text{MANTISSA}}\) bits 21-27) \rightarrow \text{DPA}

Description: The contents enabled onto the DATA PAD BUS (MANTISSA)bits 12-27 are loaded into the DATA PAD ADDRESS REGISTER (DPA). The change in DPA is effective as of the next instruction cycle.

NOTE: LDDPA supersedes INCDPA and DECDPA. It makes SETDPA redundant.
LOAD REGISTER FIELD (LDREG)

**Mandatory Fields**

```
14 15 16 17 18 19 20 21 22
1 1 1 1 1 1 0 0 0
```

**Optional Fields**

**Disabled Fields**

**LOAD S-PAD DESTINATION REGISTER from DATA PAD BUS**

**Assembler**

**Format:**

```
LDSP
```

**Example:**

```
LDSP; DB=VALUE
```

**Effect:**

```
(DPBS MANTISSA bits 12-27) → SP SPD
```

**Description:** The data currently enabled onto the DATA PAD BUS (MANTISSA) bits 12-27 are loaded into SP (SPD).

SPD is selected either via current S-PAD operation or a preceding LDSP.

**NOTE:** LDSP supersedes LDSPNL, LDSPE and LDSPT. It makes LDSP redundant. However, one of these op-codes could be used to select SPD if the immediately preceding instruction was not an LDSPD.

When combined with an S-PAD operation, LDSP results in the inclusive OR of SPFN and DPBS[MANTISSA](12-27) being written into SP(SPD).

**LOAD APSTATUS REGISTER from DATA PAD BUS**

**Assembler**

**Format:**

```
LDAPS
```

**Example:**

```
LDAPS; DB=DPY(2)
```

**Effect:**

```
(DPBS MANTISSA bits 12-27) -- APSTATUS
```

**Description:** The data currently enabled onto the DATA PAD BUS (MANTISSA) bits 12-27 are loaded into the APSTATUS REGISTER (APSTATUS). The new contents of APSTATUS may be tested. Two cycles later, i.e., at least one cycle must intervene between the LDAPS and a related test. Refer to the I/O Group Summary for a complete description of the effects of LDAPS.
LOAD REGISTER FIELD (LDREG)

 Mandalory Fields

Optional Fields

Disabled Fields

OCTAL VALUE

7

LDDA

LOAD DEVICE ADDRESS from DATA PAD BUS.

Assembler Format: LDDA
Example: LDDA;DB=VALUE

Effect: The least significant 8 bits of the data currently enabled onto the DATA PAD BUS (DPBS) (DATA PAD BUS MANTISSA bits 20-27) are loaded into the 8 bit DEVICE ADDRESS REGISTER (DA); effective as of the next instruction cycle.
READ REGISTER FIELD (RDREG)

<table>
<thead>
<tr>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>A</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PADD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VALUE

0

RPSA

READ PROGRAM SOURCE ADDRESS onto the PANEL BUS.

Assembler Format: RPSA

Effect: (PSA) → PNLBS

Description: The contents of the PROGRAM SOURCE ADDRESS REGISTER (PSA) are enabled onto the PANEL BUS (bits 04-15). The value of PSA READ is the absolute address of this instruction.

1

RSPD

READ SPAD DESTINATION ADDRESS REGISTER onto the PANEL BUS.

Assembler Format: RSPD

Effect: (SPD) → PNLBS

Description: The contents of the currently designated S-PAD DESTINATION ADDRESS (SPD) are enabled onto the PANEL BUS (PNLBS) (bits 12-15).
READ REGISTER FIELD (RDREG)

<table>
<thead>
<tr>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>A1</td>
<td>A2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

MANDATORY FIELDS
OPTIONAL FIELDS
DISABLED FIELDS

VALUE

2

RMA

READ MEMORY ADDRESS REGISTER onto the PANEL BUS

Assembler Format: RMA
Effect: (MA)→PNLBS

Description: The contents of the 16-bit MEMORY ADDRESS REGISTER (MA) are enabled onto the PANEL BUS (PNLBS) (bits 0-15).

3

RTMA

READ TABLE MEMORY ADDRESS onto the PANEL BUS

Assembler Format: RTMA
Effect: (TMA)→PNLBS

Description: The contents of the 16-bit TABLE MEMORY ADDRESS (TMA) are enabled onto the PANEL BUS (PNLBS) (bits 00-15).

The value read by this instruction is the contents of TMA as modified by the FFT and IFFT Bits in the APSTATUS Register (APSTATUS Bits 11 and 12). If APSTATUS Bit 12 is a "0," then this instruction reads the unmodified contents of TMA. Thus, this instruction always reads the actual address being presented to TABLE MEMORY. This is distinct from the JMPT, JSRT, LPSLT, LPSRT, etc. op-codes in the SPEC group that always use the unmodified contents of TMA Register.
READ REGISTER FIELD (RDREG)

<table>
<thead>
<tr>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>A1</td>
<td>A2</td>
<td>PADD1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

MANDATORY FIELDS
OPTIONAL FIELDS
DISABLED FIELDS

VALUE

4

RDPA
READ DATA PAD REGISTER onto the PANEL BUS

Assembler Format: RDPA
Effect: (DPA) → PNLBS

Description: The contents of the six-bit DATA PAD ADDRESS REGISTER (DPA) are enabled onto the PANEL BUS (PNLBS) (bits 10-15).

NOTE: DPA appears to be six bits wide for the purposes of LDDPA, SETDPA, INCMDPA, DECDPA and RDPA. Only the least significant five bits, however, are effective in addressing DATA PAD.

5

RSPFN
READ S-PAD FUNCTION onto the PANEL BUS

Assembler Format: RPSFN
Effect: (SPFN) → PNLBS

Description: The result of the current S-PAD operation (SPFN) is enabled onto the PANEL BUS (bits 00-15) for this instruction cycle.
**READ REGISTER FIELD (RDREG)**

<table>
<thead>
<tr>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>A3</td>
<td>FADD</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **MANDATORY FIELDS**
- **OPTIONAL FIELDS**
- **DISABLED FIELDS**

**VALUE**

### 6

**RAPS**

**READ APSTATUS REGISTER onto the PANEL BUS**

**Assembler**

Format: RAPS

**Effect:** (APSTATUS) → PNLBS

**Description:** The contents of the 16-bit APSTATUS REGISTER (APSTATUS) are enabled onto the PANEL BUS (PNLBS) (bits 00-15) during the current instruction cycle.

### 7

**RDA**

**READ DEVICE ADDRESS onto the PANEL BUS**

**Assembler**

Format: RDA

**Effect:** (DA) → PNLBS bits 08-15; Æ's → PNLBS bits 00-07

**Description:** The current contents of the eight-bit DEVICE ADDRESS REGISTER (DA) are enabled onto the PANEL BUS (PNLBS), bits 08-15. Zeros are enabled onto the remaining left-most bits of PNLBS, bits 00-07.
INPUT/OUTPUT FIELD (IN/OUT)

14 15 16 17 18 19 20 21 22
1 1 1
Mandatory Fields
PADDI
1 0 0
Optional Fields
Disabled Fields

VALUE

Ø
OUT
OUTPUT to I/O DEVICE

Assembler
Format: OUT
Example: DB=DPX;OUT
Effect: (DPBS)→INBS→I/O DEVICE (DA)

Description: The data enabled onto the DATA PAD BUS (DPBS) during this instruction is enabled onto the INPUT/OUTPUT BUS (INBS) where it becomes available for output to the I/O DEVICE specified by the DEVICE ADDRESS REGISTER (DA).

1
SPNOUT
SPIN until I/O DEVICE READY; then OUTPUT DATA

Assembler
Format: SPNOUT
Example: DA=DPY(3);SPNOUT
Effect: SPIN UNTIL IODRDY(DA) = 1; then, (DPBS)→INBS→I/O DEVICE(DA)

Description: First, the condition of the I/O DATA READY FLAG (IODRDY) is tested. If IODRDY = 0, then a SPIN will occur. When IODRDY = 1, the data currently enabled onto the DATA PAD BUS (DPBS) is enabled onto the INPUT/OUTPUT BUS (INBS) where it becomes available for output to the I/O DEVICE specified by the I/O DEVICE ADDRESS REGISTER (DA).

NOTE: If IODRDY(DA) never equals "1", an infinite SPIN condition will occur. This can only be cleared by an Interface Reset.
INPUT/OUTPUT FIELD (IN/OUT)

Assembler Format:

Example: MOV 5,5;DB=DPX(-2);OUTDA

Effect: (DPBS)→INBS+IODEVICE(DA); then, (SPFN)→DA

Description: First, the data currently enabled onto the DATA PAD BUS (DPBS) is enabled onto the INPUT/OUTPUT BUS (INBS), where it becomes available for output to the I/O DEVICE specified by the current contents of the I/O DEVICE ADDRESS REGISTER (DA).

Then, bits Ø8-15 of the current SPAD operation result (SPFN) are loaded into the 8 bit I/O DEVICE ADDRESS REGISTER; effective as of the next instruction cycle.
SPIN until I/O DEVICE is READY; then OUTPUT DATA; then SET DEVICE ADDRESS from SPFN.

Assembler Format: SPOTDA

Example: MOV 3,3;DB=MD;SPOTDA

Effect: SPIN until IODRDY (DA)=1; then (DPBS)+INBS + I/O DEVICE (DA); then (SPFN) + DA

Description: First, the condition of the I/O DATA READY FLAG (IODRDY[DA]) is tested. If IODRDY(DA)=0, then a SPIN will occur until IODRDY(DA)=1. When IODRDY(DA)=1, the data currently enabled onto the DATA PAD BUS (DPBS) is enabled onto the INPUT/OUTPUT BUS (INBS) for output to the I/O DEVICE specified by the current contents of the I/O DEVICE ADDRESS REGISTER (DA).

Then, the eight right-most bits (bits 08-15) of the current S-PAD operation result (SPFN) are loaded into the I/O DEVICE ADDRESS REGISTER (DA); effective as of the next instruction cycle.

NOTE: If IODRDY(DA) never equals "1", then an infinite SPIN will occur.
INPUT/OUTPUT FIELD (IN/OUT)

VALUE

4

IN

INPUT DATA from I/O DEVICE (DA)

Assembler Format: IN

Example: IN; DPX(2)<DB; DB = INBS

Effect: I/O DEVICE (DA) -> INBS

Description: The data input from the I/O DEVICE specified by the I/O DEVICE ADDRESS REGISTER (DA) is enabled onto the INPUT/OUTPUT BUS (INBS).

NOTE: In order to be used internally by the AP-120B, the data input onto the INBS must be enabled onto the DPBS. This is achieved by a concurrent software instruction to that effect, such as in the assembler format example above. (Note that the assembler would equate the form DPX(2)<INBS to the two instructions shown above).

5

SPININ

SPIN UNTIL I/O DEVICE is READY; then INPUT DATA.

Assembler Format: SPININ

Example: SPININ; DPX<INBS

Effect: SPIN until IODRDY (DA) = 1; then (I/O DEVICE (DA)) -> INBS

Description: First, a SPIN is executed until the I/O DATA READY FLAG (IODRDY), for the I/O DEVICE SPECIFIED by the I/O DEVICE ADDRESS REGISTER (DA), is equal to "1."

Then, (when IODRDY[DA]=1), the data from the I/O DEVICE (DA) is enabled onto the INPUT/OUTPUT BUS (INBS).

NOTE: If IODRDY (DA) never equals "1," an infinite SPIN loop will occur.
Assembler Format: INDA
Example: MOV 5,5;INDA;DPY<INBS
Effect: (I/O DEVICE(DA)) → INBS; then (SPFN) → DA.

Description: First, the data input from the I/O DEVICE specified by the current contents of the I/O DEVICE REGISTER (DA), is enabled onto the INPUT/OUTPUT BUS (INBS).

Then, bits 08-15 of the current S-PAD operation result (SPFN) is set into the I/O DEVICE ADDRESS REGISTER (DA); effective as of the next instruction cycle.
INPUT/OUTPUT FIELD (IN/OUT)

<table>
<thead>
<tr>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>A2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mandatory Fields: 
Optional Fields: 
Disabled Fields: 

**VALUE**

7

**SPINDA**

SPIN until I/O DEVICE READY; then INPUT DATA; then SET DEVICE ADDRESS from SPFN.

**Assembler**

Format:

Example: MOV 6,6;SPINDA;DB=INBS

Effect:

SPIN until IODRDY=1, then (I/O DEVICE(DA))→INBS; then (SPFN)→DA

Description: First, a SPIN is executed until the I/O DATA READY FLAG(IODRDY) of the I/O DEVICE specified by the contents of the I/O DEVICE ADDRESS REGISTER (DA) is equal to "1."

When IODRDY(DA)=1, the data from the I/O DEVICE(DA) is input onto the INPUT/OUTPUT BUS (INBS).

Then the right-most bits of the current S-PAD operation result (SPFN) are loaded into the I/O DEVICE ADDRESS REGISTER (DA); effective as of the next instruction cycle.

If IODRDY (DA) never equals "1," an infinite SPIN loop will be incurred.
I/O SENSE FIELD (SENSE)

<table>
<thead>
<tr>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VALUE

Ø SNSA

SENSE "A"; set IODRDY(DA) from "A"

Assembler Format: SNSA

Effect: (A(DA)) → IODRDY(DA)

Description: CONDITION "A" of the I/O DEVICE specified by the DEVICE ADDRESS REGISTER (DA) will be "sensed" and the I/O DATA READY FLAG (IODRDY[DA]) will be set to reflect the state of "A" (either "1" or "0"). These conditions are device dependent and may not necessarily be implemented for all I/O devices. Refer to the appropriate device manual for further information. Note that some devices (e.g., PIOP) use the SNSA instruction to perform control functions.

1 SPINA

SPIN UNTIL "A" NON-ZERO

Assembler Format: SPINA

Effect: (A(DA)) → IODRDY(DA); if IODRDY(DA) = Ø SPIN UNTIL IODRDY(DA) = 1

Description: First, CONDITION "A" of I/O DEVICE(DA) is sensed and the I/O DATA READY FLAG (IODRDY) is set accordingly (to either "1" or "0"). If IODRDY(DA) equals "0," a SPIN condition will occur and will continue until condition "A" is equal to "1."

NOTE: If "A" never equals "1," an infinite SPIN will occur.
I/O SENSE FIELD (SENSE)

<table>
<thead>
<tr>
<th>14 15 16 17 18 19 20 21 22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 A1 A2</td>
</tr>
<tr>
<td>(PAD)</td>
</tr>
<tr>
<td>1 0 1</td>
</tr>
</tbody>
</table>

MANDATORY FIELDS
OPTIONAL FIELDS
DISABLED FIELDS

VALUE

SENSE "A" SET to I/O READY FLAG; then set DEVICE ADDRESS from SPFN.

Assembler Format: SNSADA

Example: MOV 5,5;SNSADA

Effect: \((A_{(DA)}) + IODRDY_{(DA)}\); then \((SPFN) + DA\)

Description: First, CONDITION (A) for the I/O DEVICE specified by the I/O DEVICE ADDRESS REGISTER (DA) is sensed and its content (either "1" or "0") is set into the I/O DEVICE DATA READY FLAG (IODRDY_{(DA)}).

Then, the right-most bits (bits \(08-15\)) of the current S-PAD operation result (SPFN) are loaded into the I/O DEVICE ADDRESS REGISTER (DA); effective as of the next instruction cycle.
I/O SENSE FIELD (SENSE)

<table>
<thead>
<tr>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VALUE

3 | SPNADA |

SPIN UNTIL "A" NON-ZERO; then SET DEVICE ADDRESS from SPFN

Assembler Format: SPNADA

Effect:

\[(A_{(DA)}) \rightarrow \text{IODRDY}_{(DA)}; \text{if IODRY}_{(DA)} = \emptyset\]

SPIN until IODRDY = 1; then (SPFN) \rightarrow DA

Description: First, CONDITION "A" of I/O DEVICE\(^{(DA)}\) is sensed and set into the I/O DATA READY FLAG (IODRDY\(^{(DA)}\)). If IODRY then equals \(\emptyset\), a SPIN condition will occur until IODRDY\(^{(DA)}\) = 1. At that time, the 8 right-most bits (bits 8-15) of the current S-PAD operation result (SPFN) are loaded into the I/O DEVICE ADDRESS REGISTER \(^{(DA)}\); effective next instruction cycle.

**NOTE:** An infinite "SPIN" will occur if "A" never equals "1".
I/O SENSE FIELD (SENSE)

<table>
<thead>
<tr>
<th>14 15 16 17 18 19 20 21 22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
</tr>
<tr>
<td>MANDATORY FIELDS</td>
</tr>
<tr>
<td>OPTIONAL FIELDS</td>
</tr>
<tr>
<td>DISABLED FIELDS</td>
</tr>
<tr>
<td>VALUE</td>
</tr>
</tbody>
</table>

SENSE I/O DEVICE CONDITION "B", SET IODRDY(DA) from "B"

Assembler Format: SNSB
Effect: \((B(DA)) + IODRDY(DA)\)

Description: CONDITION "B" (B) of the I/O DEVICE specified by the DEVICE ADDRESS REGISTER (DA) will be "sensed" and the I/O DATA READY FLAG (IODRDY) for that I/O DEVICE will be set to reflect the contents of "B" (either "1" or "~").

SPIN UNTIL "B" is NON-ZERO

Assembler Format: SPINB
Effect: \((B(DA)) + IODRDY(DA)\)if IODRDY(DA) = ~ SPIN until IODRDY(DA) = 1.

Description: First, CONDITION "B" of I/O DEVICE(DA) is sensed and the I/O DATA READY FLAG (IODRDY) is set accordingly, (to either "1" or "~"). If IODRDY(DA) then equals "~", a SPIN condition will occur until "B" equals "1", at which time the IODRDY(DA) is set accordingly.

NOTE: If "B" never equals "1", an infinite SPIN condition will occur.
I/O SENSE FIELD (SENSE)

<table>
<thead>
<tr>
<th>14 15 16 17 18 19 20 21 22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>MANDATORY FIELDS</td>
</tr>
<tr>
<td>1 0 1</td>
</tr>
<tr>
<td>OPTIONAL FIELDS</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>DISABLED FIELDS</td>
</tr>
</tbody>
</table>

VALUE

<table>
<thead>
<tr>
<th>SNSBDA</th>
<th>SENSE &quot;B&quot; set to I/O DATA READY FLAG, then SET DEVICE ADDRESS from SPFN.</th>
</tr>
</thead>
</table>

Assembler Format: SNSBDA

Effect: \( (B(DA)) \rightarrow IODRDY(DA); \) then \( (SPFN) \rightarrow DA \)

Description: First, condition "B" of the I/O DEVICE specified by the I/O DEVICE ADDRESS REGISTER \((DA)\) is sensed and its content (either "1" or "0") set into the I/O DEVICE DATA READY FLAG \((IODRDY(DA))\).

Then, the right-most bits (bits \(0-15\)) of the current S-PAD operation result \((SPFN)\) are loaded into the I/O DEVICE ADDRESS REGISTER \((DA)\); effective next instruction cycle.
I/O SENSE FIELD (SENSE)

<table>
<thead>
<tr>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>A1</td>
<td>A2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FAD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SPNBDA SPIN UNTIL "B" is NON-ZERO; SET DEVICE ADDRESS from SPFN

Assembler Format: SPNBDA

Effect: 

\[(B(\text{DA})) \rightarrow \text{IODRDY(\text{DA})}; \text{if IODRDY(\text{DA})} = \emptyset, \text{SPIN until} \]

\[\text{IODRDY} = 1; \text{then (SPFN)} \rightarrow \text{DA}\]

Description: First, CONDITION "B" of I/O DEVICE(\text{DA}) is sensed and set into the I/O DATA READY FLAG (IODRDY(\text{DA})). If IODRDY(\text{DA}) then equals \"\emptyset\", a SPIN condition will occur until "B" equals "1", at which time the IODRDY(\text{DA}) is set accordingly.

Then the 8 right-most bits (bits 8-15) of the current S-PAD operation result (SPFN) are loaded into the I/O DEVICE ADDRESS REGISTER (DA); effective next instruction cycle.

NOTE: If "B" never equals "1", an infinite SPIN condition will occur.
The following instructions set one of four GENERAL FLAGS (0,1,2,3) to a "1". These flags may be tested and branched upon accordingly by software instructions (see STEST SPEC).

\[
\begin{align*}
\emptyset & \quad \text{SFL}\emptyset \quad \text{Set GENERAL FLAG } \emptyset \text{ to } "1" \\
1 & \quad \text{SFL1} \quad \text{Set GENERAL FLAG 1 to } "1" \\
2 & \quad \text{SFL2} \quad \text{Set GENERAL FLAG 2 to } "1" \\
3 & \quad \text{SFL3} \quad \text{Set GENERAL FLAG 3 to } "1"
\end{align*}
\]

Assembler Format: SFLn
Effect: \(1+\ \text{FLAG}_n\)

NOTE: Effective two instruction cycles later. At least one cycle must intervene between an SFLn and a Related Branch instruction.
The following instructions clear to "0" one of the four GENERAL FLAGS (0, 1, 2, 3) available for use in the AP. These flags may be tested and branched upon by software instructions (see STEST SPEC).

4  CLF0  Clear GENERAL FLAG 0 to "0"
5  CLF1  Clear GENERAL FLAG 1 to "0"
6  CLF2  Clear GENERAL FLAG 2 to "0"
7  CLF3  Clear GENERAL FLAG 3 to "0"

Assembler Format:  CLFn
Effect:  \( \emptyset + \text{FLAG}_n \)

NOTE: Effective two instruction cycles later.
DATA PAD X (DPX)

0

No Operation

1  

DPX(idx) + < DB

STORE DATA PAD BUS into DATA PAD X

Assembler Format:

DPX(idx) → DB

Example:

DPX(-3) < DB; DB=MD

Effect:

(DPBS) → DPX(DPA) + idx

Description: The data currently enabled onto the DATA PAD BUS (DB) is written into DATA PAD X (DPX) at the location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPA), plus the contents of the XWRITE FIELD (XW) minus 4 (the BIAS value).

Normally, a DPBS enable instruction is used concurrently with this instruction. If so, the two instructions can be expressed in shorthand notation (e.g., the example above can be expressed as follows: DPX[-3]<MD).

Note: All bits are written unless WRTEXT, WRTHMN, or WRTLIMN is set.

+idx: An integer in a range from -4 to +3.

XW = idx + 4

E - 123
**DATA PAD X (DPX)**

**Assembler Format:**

\[ \text{DPX}(\text{idx}) < \text{FA} \]

**Effect:**

\[(\text{FA}) \rightarrow \text{DPX} (\text{DPA}) + \text{idx} - 4\]

**Description:** The current FLOATING ADDER result (FA) is written into DATA PAD X (DPX) at the location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPA), plus idx.
DATA PAD X (DPX)

<table>
<thead>
<tr>
<th>32</th>
<th>33</th>
<th>34</th>
<th>35</th>
<th>36</th>
<th>37</th>
<th>38</th>
<th>39</th>
<th>40</th>
<th>41</th>
<th>42</th>
<th>43</th>
<th>44</th>
<th>45</th>
<th>46</th>
<th>47</th>
<th>48</th>
<th>49</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPX</td>
<td>DPY</td>
<td>DPZS</td>
<td>XR</td>
<td>XR</td>
<td>XW</td>
<td>FM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DESCRIPTION:**

The current FLOATING MULTIPLIER OUTPUT (FM) is written into DATA PAD X (DPX) at the location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPA), plus idx.

**Assembler Format:**

DPX(idx) < FM

**Effect:**

(FM) → DPX (DPA) + idx
DATA PAD Y (DPY)

**Assembler**

Format: 

```
DPX(idx) < DB
```

Example: 

```
DPY(+3) < DB; DB=MD
```

**Effect:**

```
(DB) → DPY (DPA) + idx
```

**Description:** The data currently enabled onto the DATA PAD BUS (DB) is written into DATA PAD Y (DPY) at the location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPA), plus the contents of the Y WRITE FIELD (YW) minus 4 (the BIAS value).

Normally, a DPBS enable instruction is used concurrently with this instruction. If so, the two instructions can be expressed in shorthand notation (e.g., the example above can be expressed as follows: DPY[+3] < MD).

**Note:** All bits written unless WRTEXP, WRTHMN, or WRTLMN is set.

\[†\] idx: An integer in a range from -4 to +3.  
YW = idx+4.

\[‡\] If VALUE field is used during this instruction, the XW field is referenced instead. And if a DPX (idx) write instruction is used concurrently, the indices specified by the programmer must be equal. Errors in this regard are flagged by the assembler.
DATA PAD Y (DPY)

STORE FLOATING ADDER OUTPUT into DATA PAD Y

Assembler Format: DPY(idx) < FA

Effect: (FA) → DPY (DPA) + idx

Description: The current FLOATING ADDER OUTPUT (FA) is written into DATA PAD Y at the location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPA), plus idx.
DATA PAD Y (DPY)

Assembler Format: 

\[ \text{DPY}(\text{idx}) < \text{FM} \]

Effect: 

\[ (\text{FM}) \rightarrow \text{DPY} (\text{DPA}) + \text{idx} \]

Description: The current contents of the FLOATINGmultiplier output (FM) are written into DATA PAD Y (DPY) at the location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPA), plus idx.
DATA PAD BUS (DPBS)

Assembler Format: DB = ZERO (see Data Pad summary)

Effect: 0.0 -> DPBS

Description: FLOATING POINT ZERO (0.0) is enabled onto the DATA PAD BUS (DPBS) during the current instruction cycle.

Note that this is the default condition for the DPBS field. Thus, if DPBS is not specified via a "DB = " mnemonic, and no other DATA PAD BUS enable field (e.g. RPSF, see SPEC) is utilized, then there will be a zero on the DATA PAD BUS.
DATA PAD BUS (DPBS)

| DB = INBS |

ENABLE INBUS onto the DATA PAD BUS

Assembler Format:
DB = INBS (see Data Pad summary)

Example: IN: DPX(-3) DB; DB=INBS

Effect: (INBS)→DPBS

Description: The contents of the 38-bit INPUT/OUTPUT BUS (INBS) are enabled onto the DATA PAD BUS (DPBS) for the current instruction cycle.

This instruction is used to transfer data from INBS to DPBS during a current input operation (See example, above).

It must be accompanied by an I/O group INPUT instruction (IN, SPININ, INDA or SPINDA).
**VALUE**

### DB = VALUE

**Assembler Format:**

`DB = val` (where `val` is a 16-bit value contained in the Value field).

**Example:**

- `DB = 25`
- `DB = -1`

**Effect:**

- `(VALUE bits 54-63) -> DPBS^EXPONENT`
- `(VALUE bits 49-63) -> DPBS^MANTISSA` bits 13-27
- `(VALUE bit 48 (sign) -> DPBS^MANTISSA` bits 00-12

**Description:**

The 16-bit value contained in the VALUE field of the instruction word is enabled onto the DATA PAD BUS (DPBS) during the current instruction cycle, in the following manner:

1. The right-most 15 bits of the VALUE field are enabled onto DPBS (MANTISSA) bits 13-27. The left-most remaining bit (the sign bit) is extended and enabled onto DPBS (MANTISSA) bits 00-12.

2. The right-most 10 bits of the VALUE field are enabled onto the DPBS (EXPONENT) bits 02-11.

^YW field is disabled for the current instruction cycle.
**DATA PAD BUS (DPBS)**

<table>
<thead>
<tr>
<th>32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50</th>
<th>63</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPX</td>
<td>DPF</td>
</tr>
</tbody>
</table>

**VALUE**

3

\[ DB = DPX(idx)^{+} \]

**ENABLE DATA PAD X onto the DATA PAD BUS**

**Assembler Format:**

\[ DB = DPX(idx) \]

**Effect:**

\[ (DPX (DPA) + idx) \rightarrow DPBS \]

**Description:** The contents of the DATA PAD X (DPX) location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPS), plus the contents of the X READ FIELD (XR) minus 4, are enabled onto the DATA PAD BUS (DPBS) for the current instruction cycle.

\[ ^{+} \text{idx: An integer in a range from } -4 \text{ to } +3. \]

\[ XR = idx + 4 \]
DATA PAD BUS (DPBS)

4

\[
\text{DB} = \text{DPY}(\text{idx})
\]

\*ENABLE DATA PAD Y onto the DATA PAD BUS*

Assembler Format:

\[ \text{DB} = \text{DPY}(\text{idx}) \]

Effect:

\[ (\text{DPY} (\text{DPA}) + \text{idx}) \rightarrow \text{DPBS} \]

Description: The contents of the DATA PAD Y (DPY) location specified by the current contents of the DATA PAD ADDRESS REGISTER (DPA), plus the contents of the Y READ FIELD (YR) minus 4, are enabled onto the DATA PAD BUS (DPBS) for the current instruction cycle.

\* idx: An integer in a range from -4 to +3.

\[ \text{YR} = \text{idx} + 4 \]
**Assembler**

**Format:**

```plaintext
DB = MD
```

**Effect:**

```
(MDREG) → DPBS
```

**Description:** The contents of the MAIN DATA MEMORY (MD) location entered into the MEMORY OUTPUT REGISTER (MDREG) during this instruction cycle are enabled onto the DATA PAD BUS for the current instruction cycle. (See MEMORY GROUP SUMMARY for the set-up requirements for a MAIN DATA MEMORY (MD) READ operation.)
DATA PAD BUS (DPBS)

6

**DB = SPFN**

**ENABLE SPAD FUNCTION onto the DATA PAD BUS**

**Assembler Format:**

\[ DB = SPFN \]

**Effect:**

- \((SPFN_\text{bits } 7-15) + DPBS^{\text{EXponent}} \text{ bits } 03-11\)
- \((SPFN_\text{bit } 6) \text{ INVERTED}\) + DPBS^{\text{EXponent}} \text{ bit } 02
- \((SPFN_\text{bits } 01-15) + DPBS^{\text{Mantissa}} \text{ bits } 13-27\)
- \((SPFN_\text{bit } 0 \text{ (sign)}) + DPBS^{\text{Mantissa}} \text{ bits } 00-12\)

**Description:** The 10 right-most bits of the S-PAD FUNCTION (SPFN bits [6-15]) plus BIAS, are enabled onto the DATA PAD BUS (EXponent) (DPBS [EXP] bits 02-11). The 15 right-most bits of SPFN (bits 01-15) are concurrently enabled onto the DPBS (Mantissa bits) 13-27 and the left-most remaining bit (SPFN [bit 0]) is extended and enabled onto the left-most remaining bits of the DPBS (Mantissa) (bits 00-12). SPFN is enabled onto DPBS during the current instruction only.

\[ \dagger \text{BIAS: For this operation, a BIAS of } 512(10) \text{ is added by inverting SPFN(bit)06.} \]
DATA PAD BUS (DPBS)

Value

```
DB = TM
ENABLE TABLE MEMORY onto the DATA PAD BUS
```

Assembler

Format: `DB=TM`

Effect: `(TMREG) → DPBS`

Description: The contents of the TABLE MEMORY (TM) location currently entered into the TABLE MEMORY REGISTER (TM REG) is enabled onto the DATA PAD BUS (DPBS) for the current instruction cycle. (See MEMORY GROUP SUMMARY-TM for the set-up requirements for a TABLE MEMORY [TM] READ operation).
MEMORY INPUT GROUP (MI)

<table>
<thead>
<tr>
<th>56</th>
<th>57</th>
<th>58</th>
<th>59</th>
<th>60</th>
<th>61</th>
<th>62</th>
<th>63</th>
</tr>
</thead>
<tbody>
<tr>
<td>MI</td>
<td>MA</td>
<td>FPA</td>
<td>FPA</td>
<td>FPA</td>
<td>FPA</td>
<td>FPA</td>
<td>FPA</td>
</tr>
</tbody>
</table>

### VALUE

<table>
<thead>
<tr>
<th>0</th>
<th>No Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WRITE FLOATING ADDER RESULT to MAIN DATA MEMORY; VIA MEMORY INPUT REGISTER</td>
</tr>
</tbody>
</table>

**Assembler Format:**

```
MOV 5,5; SETMA; MI<FA
```

**Effect:**

```
(FA) \rightarrow MI \rightarrow MD (MA)
```

**Description:** The FLOATING ADDER RESULT (FA) currently available is loaded into the MEMORY INPUT REGISTER (MI) and written into the MAIN DATA MEMORY (MD) location specified by the current contents of the MEMORY ADDRESS REGISTER (MA). (See: MEMORY GROUP SUMMARY - MD for the necessary timing implications).

**NOTE:** The contents of the MEMORY ADDRESS REGISTER (MA), which specifies the MD location to be written, are determined by the mandatory and simultaneous SETMA, INCMA, DECMA or LDMA instruction being processed.
MEMORY INPUT GROUP (MI)

MI<FM

WRITE FLOATING MULTIPLY RESULT to MAIN DATA MEMORY; VIA MEMORY INPUT REGISTER

Assembler Format: MI<FM
Example: INCMA; MI<FM
Effect: (FM)→MI→MD(MA)

Description: The FLOATING MULTIPLIER RESULT (FM) currently available is loaded into the MEMORY INPUT REGISTER (MI) and written into MAIN DATA MEMORY (MD) at the location specified by the current contents of the MEMORY ADDRESS REGISTER (MA).
MEMORY INPUT GROUP (MI)

VALUE

3

MI<DB

WRITE DATA PAD BUS to MAIN DATA MEMORY;
VIA MEMORY INPUT REGISTER

Assembler Format: MI<DB
Example: DECMA; MI<DB; DB=ZERO
or: DECMA; MI<ZERO

Effect: (DPBS)→MI→MD (MA)

Description: The current data enabled onto the DATA PAD BUS (DB) is loaded into the MEMORY INPUT REGISTER (MI) and written into MAIN DATA MEMORY (MD) at the location specified by the current contents of the MEMORY ADDRESS REGISTER (MA).

This instruction should be used in conjunction with a DPBS ENABLE instruction unless a zero is desired.
MEMORY ADDRESS GROUP (MA)

VALUE

0  No Operation

1  INCREMENT MEMORY ADDRESS REGISTER;
   INITIATE MAIN DATA MEMORY

Assembler
Format:  INCMA
Example:  INCMA; MI<FA

Effect:  (MA) + 1−→MA; initiate MD cycle

Description: The contents of the MEMORY ADDRESS REGISTER (MA) are incremented by "1" and a MAIN DATA MEMORY (MD) cycle is initiated.

If used concurrently with an MI group instruction, an MD WRITE is initiated using the incremented value of MA as the MD location specifier.

If used without a concurrent MI group instruction, then an MD READ is initiated using the incremented value of MA as the MD location specifier.

(See MEMORY GROUP SUMMARY - MD, for the necessary memory timing implications involved in the use of this field.)
DECMA  
DECREMENT MEMORY ADDRESS REGISTER;  
INITIATE MAIN DATA MEMORY  

Assembler Format: DECMA 
Example: DECMA; MI<FM 

Effect: (MA)-1+MA; initiate MD cycle  

Description: The contents of MEMORY ADDRESS REGISTER (MA) are decremented by "1" and a MAIN DATA MEMORY (MD) cycle is initiated. 

If used with an MI group instruction, an MD WRITE is initiated using the decremented contents of MA as the MD location specifier. 

If used without a concurrent MI group instruction, then an MD READ is initiated using the decremented contents of MA as the MD location specifier.
### MEMORY ADDRESS GROUP (MA)

<table>
<thead>
<tr>
<th>56 57 58</th>
<th>59 60 61 62 63</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALUE</td>
<td>VALUE</td>
</tr>
</tbody>
</table>

#### MANDATORY FIELDS

#### OPTIONAL FIELDS

#### DISABLED FIELDS

---

### SETMA

**SET MEMORY ADDRESS REGISTER FROM THE S-PAD FUNCTION: INITIATE MAIN DATA MEMORY**

**Assembler Format:**

```
SETMA
```

**Example:**

```
MOVE 0,0; SETMA; MI<TM
```

**Effect:**

```
(SPFFN)+MA; or, if LDREG field is being used, then
(DPBS)+MA, instead.
```

**Description:** The contents of the MEMORY ADDRESS REGISTER (MA) are replaced by the S-PAD OUTPUT (SPFN) of the current S-PAD operation.

However, if an LDREG field instruction (see I/O) is used during the same instruction, then the contents currently enabled onto the DATA PAD BUS (DPBS), and not SPFN, are loaded into the MEMORY ADDRESS REGISTER (MA).

If used with an MI group instruction, an MD WRITE will be initiated. If used without an MI group instruction, then an MD READ will be initiated. In either case, the newly-formed value of MA will be used as the MD location specifier.
### DATA PAD ADDRESS GROUP (DPA)

<table>
<thead>
<tr>
<th>56 57 58 59 60 61 62 63</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ø</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

**Assembeher:**

**Format:** INCDPA

**Effect:** (DPA) + 1 + DPA

**Description:** The contents of the DATA PAD ADDRESS REGISTER (DPA) are decremented by "1."

DPA will not be affected during the current instruction cycle but will be available as of the next instruction cycle.

(See DATA PAD GROUP SUMMARY - DPA.)
DATA PAD ADDRESS GROUP (DPA)

THE CONTENTS OF THE DATA PAD ADDRESS REGISTER (DPA) ARE INCREMENTED BY "1."

DPA WILL NOT BE AFFECTED DURING THE CURRENT INSTRUCTION CYCLE BUT WILL BE AVAILABLE AS OF THE NEXT INSTRUCTION CYCLE.

(SEE DATA PAD GROUP SUMMARY - DPA.)
DATA PAD ADDRESS GROUP (DPA)

Assembler
Format: SETDPA
Effect: (SPFN)→DPA; or, if LDREG instruction is being used, then (DPBS)→DPA, instead.

Description: The current S-PAD (SPFN) is loaded into the DATA PAD ADDRESS REGISTER (DPA). However, if an LDREG field instruction (see I/O) is used during the same instruction cycle, the data enabled onto the DATA PAD BUS (DPBS) during this cycle, and not SPFN, will be loaded into the DPA.

DPA will not be affected during the current instruction cycle but will be available as of the next instruction cycle.

(See DATA PAD GROUP SUMMARY – DPA.)
TABLE MEMORY ADDRESS GROUP (TMA)

VALUE

Ø  No Operation

l  INCTMA  INCREMENT TABLE MEMORY ADDRESS REGISTER

Assembler  Format:  INCTMA

Effect:  (TMA)+1+ TMA

Description: The contents of the TABLE MEMORY ADDRESS REGISTER (TMA) are incremented by one.

The contents of TM (TMA) are available in TMREG two cycles after execution of this instruction. The modified contents of TMA are available on the next instruction cycle.
TABLE MEMORY ADDRESS GROUP (TMA)

Decrement Table Memory Address Register

Assembler Format: DECTMA (See, MEMORY GROUP SUMMARY - TMA)

Effect: (TMA) - 1 → TMA

Description: The contents of the TABLE MEMORY ADDRESS REGISTER (TMA) are decremented by one.

The contents of TM (TMA) are available in TMREG two cycles after execution of this instruction.
**TABLE MEMORY ADDRESS GROUP (TMA)**

VALUE

<table>
<thead>
<tr>
<th>56</th>
<th>57</th>
<th>58</th>
<th>59</th>
<th>60</th>
<th>61</th>
<th>62</th>
<th>63</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>M2</td>
<td>M3</td>
<td>M4</td>
<td>DPA</td>
<td>TMA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**MANUALY FIELDS**

**OPTIONAL FIELDS**

**DISABLED FIELDS**

**VALUE**

3

**SETTMA**

SET TABLE MEMORY ADDRESS REGISTER from S-PAD FUNCTION

Assembler

Format: SETTMA

Effect: (SPFN) → TMA; or, if LDREG field is being used, then (DPBS) → TMA, instead.

Description: The contents of the result of the current S-PAD operation (SPFN) are loaded into the TABLE MEMORY ADDRESS REGISTER (TMA). However, if an LDREG field instruction is used concurrently (see I/O), then the data enabled onto the DATA PAD BUS (DPBS), and not SPFN, is loaded into TMA.

The contents of TMA are available as TM two cycles after execution of this instruction.
INDEX

&.................E-13
A1.................E-93
A2.................E-95
ADD.................E-17
AND.................E-20
B.................E-13
BDBN.................E-35
BDBZ.................E-36
BEQ.................E-77
BFCT.................E-76
BFL0.................E-38
BFL1.................E-38
BFL2.................E-38
BFL3.................E-38
BFLT.................E-31
BFNE.................E-74
BFPE.................E-71
BGE.................E-79
BGT.................E-80
BIFN.................E-37
BIFZ.................E-37
BINTRQ.................E-70
BION.................E-70
BIOZ.................E-71
BLT.................E-32
BNC.................E-33
BNE.................E-78
BR.................E-69
BZC.................E-34
CFL0.................E-122
CFL1.................E-122
CFL2.................E-122
CFL3.................E-122
CLR.................E-25
COM.................E-26
COND.................E-68
CONTROL.................E-8
DB.................E-123, E-129, E-139
DBELIT.................E-39
DBHLIT.................E-40
DBLIT.................E-40
DEC.................E-26
DECDPA.................E-144
DECMA.................E-141
DECTMA.................E-147
DISP.................E-31, E-69
DPA.................E-143
DPBS.................E-129
DPX.................E-93, E-95, E-99
DPS.................E-100, E-123, E-132
DPY.................E-93, E-95, E-99
E-100, E-126, E-133
EDPX.................E-96
EQV.................E-22
FZCM.................E-90
FA.................E-95, E-100, E-124
E-127, E-137
FABS.................E-92
FADD.................E-81
FADD1.................E-81
FAND.................E-84
FEQV.................E-83
FIX.................E-86
FIXT.................E-87
FLAG.................E-121
FM.................E-93, E-99, E-125
E-128, E-138
FMUL.................E-98
FOR.................E-85
FSCALE.................E-91
FSCLT.................E-88
FMM2C.................E-89
FSUB.................E-81
FSUBR.................E-81
HALT.................E-8
HOSTPNL.................E-39
IN.................E-112
INBS.................E-130
INC.................E-25
INCDPA.................E-143
INCM.................E-140
INCTMA.................E-146
INDA.................E-113
INOUT.................E-109
INTA.................E-10
INTEN.................E-9
IO.................E-7, E-101
IORST.................E-9
JMP.................E-47
JMPA.................E-45
JMPF.................E-49
JMPT.................E-48
JSR.................E-47
JSRA.................E-46
JSRP.................E-50
JSTRT.................E-48
L.................E-14
LDAPS.................E-103
LDDA.................E-104
LDDMA.................E-102
LDMA.................E-101
LDREG.................E-101
LDSP.................E-103
LDSPE.................E-28
LDSPI.................E-29
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDSPNL</td>
<td>E-27</td>
<td>SETEX</td>
<td>E-66</td>
</tr>
<tr>
<td>LDSPT</td>
<td>E-30</td>
<td>SETEXIT</td>
<td>E-66</td>
</tr>
<tr>
<td>LDTMA</td>
<td>E-102</td>
<td>SETEXP</td>
<td>E-67</td>
</tr>
<tr>
<td>LPSL</td>
<td>E-64</td>
<td>SETEXT</td>
<td>E-67</td>
</tr>
<tr>
<td>LPSLA</td>
<td>E-62</td>
<td>SETMA</td>
<td>E-142</td>
</tr>
<tr>
<td>LPSLF</td>
<td>E-63</td>
<td>SETSPA</td>
<td>E-45</td>
</tr>
<tr>
<td>LPSLT</td>
<td>E-64</td>
<td>SETTM</td>
<td>E-148</td>
</tr>
<tr>
<td>LPSR</td>
<td>E-64</td>
<td>SFO0</td>
<td>E-121</td>
</tr>
<tr>
<td>LPSRA</td>
<td>E-63</td>
<td>SFL1</td>
<td>E-121</td>
</tr>
<tr>
<td>LPSRP</td>
<td>E-65</td>
<td>SFL2</td>
<td>E-121</td>
</tr>
<tr>
<td>LPSRT</td>
<td>E-65</td>
<td>SFL3</td>
<td>E-121</td>
</tr>
<tr>
<td>M1</td>
<td>E-99</td>
<td>SH</td>
<td>E-14</td>
</tr>
<tr>
<td>M2</td>
<td>E-100</td>
<td>SNSA</td>
<td>E-115</td>
</tr>
<tr>
<td>MA</td>
<td>E-140</td>
<td>SNSADA</td>
<td>E-116</td>
</tr>
<tr>
<td>MD</td>
<td>E-96</td>
<td>SNSB</td>
<td>E-118</td>
</tr>
<tr>
<td>MDPX</td>
<td>E-96</td>
<td>SNSBDA</td>
<td>E-119</td>
</tr>
<tr>
<td>MI</td>
<td>E-137</td>
<td>SOP</td>
<td>E-6</td>
</tr>
<tr>
<td>MOV</td>
<td>E-19</td>
<td>SOP1</td>
<td>E-23</td>
</tr>
<tr>
<td>NC</td>
<td>E-95</td>
<td>SPD</td>
<td>E-17</td>
</tr>
<tr>
<td>NOP</td>
<td>E-5</td>
<td>SPEC</td>
<td>E-6</td>
</tr>
<tr>
<td>OR</td>
<td>E-21</td>
<td>SFIN</td>
<td>E-135</td>
</tr>
<tr>
<td>OUT</td>
<td>E-109</td>
<td>SPINA</td>
<td>E-115</td>
</tr>
<tr>
<td>OUTDA</td>
<td>E-110</td>
<td>SPINB</td>
<td>E-118</td>
</tr>
<tr>
<td>PNLLIT</td>
<td>E-39</td>
<td>SPINDA</td>
<td>E-114</td>
</tr>
<tr>
<td>PS</td>
<td>E-58</td>
<td>SPININ</td>
<td>E-112</td>
</tr>
<tr>
<td>PSEVEN</td>
<td>E-52</td>
<td>SPMDA</td>
<td>E-6</td>
</tr>
<tr>
<td>PSODD</td>
<td>E-55</td>
<td>SPMDAV</td>
<td>E-7</td>
</tr>
<tr>
<td>R</td>
<td>E-15</td>
<td>SPNADA</td>
<td>E-117</td>
</tr>
<tr>
<td>RAPS</td>
<td>E-108</td>
<td>SNBDA</td>
<td>E-120</td>
</tr>
<tr>
<td>RDA</td>
<td>E-108</td>
<td>SPNOUT</td>
<td>E-109</td>
</tr>
<tr>
<td>RDA PA</td>
<td>E-107</td>
<td>SPOTDA</td>
<td>E-11</td>
</tr>
<tr>
<td>RDREG</td>
<td>E-105</td>
<td>SPS</td>
<td>E-17</td>
</tr>
<tr>
<td>REFR</td>
<td>E-10</td>
<td>STEST</td>
<td>E-31</td>
</tr>
<tr>
<td>RETURN</td>
<td>E-72</td>
<td>SUB</td>
<td>E-18</td>
</tr>
<tr>
<td>RETURN</td>
<td>E-7</td>
<td>SWDB</td>
<td>E-41</td>
</tr>
<tr>
<td>RMA</td>
<td>E-106</td>
<td>SWDBE</td>
<td>E-42</td>
</tr>
<tr>
<td>RPS</td>
<td>E-652</td>
<td>SWDBH</td>
<td>E-43</td>
</tr>
<tr>
<td>RPSOA</td>
<td>E-52</td>
<td>SWDBL</td>
<td>E-44</td>
</tr>
<tr>
<td>RPSOT</td>
<td>E-52</td>
<td>TN</td>
<td>E-94</td>
</tr>
<tr>
<td>RPS1</td>
<td>E-55</td>
<td>TMA</td>
<td>E-146</td>
</tr>
<tr>
<td>RPS1A</td>
<td>E-55</td>
<td>VALUE</td>
<td>E-121</td>
</tr>
<tr>
<td>RPS1T</td>
<td>E-56</td>
<td></td>
<td>E-131</td>
</tr>
<tr>
<td>RPS2</td>
<td>E-52</td>
<td>WPS0</td>
<td>E-54</td>
</tr>
<tr>
<td>RPS2A</td>
<td>E-52</td>
<td>WPSOA</td>
<td>E-53</td>
</tr>
<tr>
<td>RPS2T</td>
<td>E-53</td>
<td>WPSOT</td>
<td>E-54</td>
</tr>
<tr>
<td>RPS3</td>
<td>E-55</td>
<td>WPS1</td>
<td>E-57</td>
</tr>
<tr>
<td>RPS3A</td>
<td>E-55</td>
<td>WPS1A</td>
<td>E-56</td>
</tr>
<tr>
<td>RPS3T</td>
<td>E-56</td>
<td>WPS1T</td>
<td>E-57</td>
</tr>
<tr>
<td>RPSA</td>
<td>E-105</td>
<td>WPS2</td>
<td>E-54</td>
</tr>
<tr>
<td>RPSF</td>
<td>E-60</td>
<td>WPS2A</td>
<td>E-53</td>
</tr>
<tr>
<td>RPSFA</td>
<td>E-59</td>
<td>WPS2T</td>
<td>E-54</td>
</tr>
<tr>
<td>RPSFP</td>
<td>E-61</td>
<td>WPS3</td>
<td>E-57</td>
</tr>
<tr>
<td>RPSFT</td>
<td>E-61</td>
<td>WPS3A</td>
<td>E-56</td>
</tr>
<tr>
<td>RPSL</td>
<td>E-60</td>
<td>WPS3T</td>
<td>E-57</td>
</tr>
<tr>
<td>RPSLA</td>
<td>E-58</td>
<td>WRTEX</td>
<td>E-11</td>
</tr>
<tr>
<td>RPSLP</td>
<td>E-61</td>
<td>WRTEXP</td>
<td>E-23</td>
</tr>
<tr>
<td>RPSLT</td>
<td>E-60</td>
<td>WRTHM</td>
<td>E-24</td>
</tr>
<tr>
<td>RR</td>
<td>E-16</td>
<td>WRTLMN</td>
<td>E-24</td>
</tr>
<tr>
<td>RSPD</td>
<td>E-105</td>
<td>WRTMAN</td>
<td>E-11</td>
</tr>
<tr>
<td>RSPFN</td>
<td>E-107</td>
<td>XR</td>
<td>E-132</td>
</tr>
<tr>
<td>RTMA</td>
<td>E-106</td>
<td>XM</td>
<td>E-123</td>
</tr>
<tr>
<td>SENSE</td>
<td>E-115</td>
<td>YR</td>
<td>E-133</td>
</tr>
<tr>
<td>SETDPA</td>
<td>E-145</td>
<td>YW</td>
<td>E-126</td>
</tr>
<tr>
<td>SETEX</td>
<td>E-145</td>
<td>ZERO</td>
<td>E-94</td>
</tr>
<tr>
<td>SETEX</td>
<td>E-66</td>
<td></td>
<td>E-96</td>
</tr>
<tr>
<td>SETEXP</td>
<td>E-66</td>
<td></td>
<td>E-129</td>
</tr>
<tr>
<td>SETEX</td>
<td>E-66</td>
<td></td>
<td>E-68</td>
</tr>
</tbody>
</table>
Notice to the Reader

- Help us improve the quality and usefulness of this manual.
- Your comments and answers to the following READERS COMMENT form would be appreciated.

To mail: fold the form in three parts so that FLOATING POINT SYSTEMS' mailing address is visible; seal.

Thank you
READERS COMMENT FORM

Document Title __________________________

Your comments and answers will help us improve the quality and usefulness of our publications. If your answers require qualification or additional explanation, please comment in the space provided below.

How did you use this manual?

( ) AS AN INTRODUCTION TO THE SUBJECT
( ) AS AN AID FOR ADVANCED TRAINING
( ) TO LEARN OF OPERATING PROCEDURES
( ) TO INSTRUCT A CLASS
( ) AS A STUDENT IN A CLASS
( ) AS A REFERENCE MANUAL
( ) OTHER __________________________

Did you find this material . . .

YES NO

• USEFUL? ( ) ( )
• COMPLETE? ( ) ( )
• ACCURATE? ( ) ( )
• WELL ORGANIZED? ( ) ( )
• WELL ILLUSTRATED? ( ) ( )
• WELL INDEXED? ( ) ( )
• EASY TO READ? ( ) ( )
• EASY TO UNDERSTAND? ( ) ( )

Please indicate below whether your comment pertains to an addition, deletion, change or error; and, where applicable, please refer to specific page numbers.

<table>
<thead>
<tr>
<th>Page</th>
<th>Description of error or deficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From:

Name __________________________ Title __________________________
Firm __________________________ Department __________________________
Address __________________________ City, State __________________________
Telephone __________________________ Date __________________________
BUSINESS REPLY
No postage stamp necessary if mailed in the United States

Postage will be paid by:
FLOATING POINT SYSTEMS, INC.
P.O. Box 23489
Portland, Oregon 97223

Attention: Technical Publications