### Figure 1. Programming Model

<table>
<thead>
<tr>
<th>Condition Code Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>CARRY</td>
</tr>
<tr>
<td>X</td>
<td>OVERFLOW</td>
</tr>
<tr>
<td>H</td>
<td>ZERO</td>
</tr>
<tr>
<td>I</td>
<td>NEGATIVE</td>
</tr>
<tr>
<td>N</td>
<td>MASK (DISABLE) IRQ INTERRUPTS</td>
</tr>
<tr>
<td>Z</td>
<td>STOP DISABLE (IGNORE STOP OPCODES)</td>
</tr>
<tr>
<td>V</td>
<td>RESET OR XIRQ SET X, INSTRUCTIONS MAY CLEAR X BUT CANNOT SET X</td>
</tr>
<tr>
<td>C</td>
<td>STOP DISABLE (IGNORE STOP OPCODES)</td>
</tr>
</tbody>
</table>

- **Condition Code Register**
  - Carry (C)
  - Overflow (O)
  - Zero (Z)
  - Negative (N)
  - Mask (Disable) IRQ INTerrupts (I)
  - Mask (Disable) XIRQ INTerrupts (X)
  - Reset or XIRQ set X, instructions may clear X but cannot set X
  - Stop disable (Ignore STOP OPCODES) (S)

- **Register Descriptions**
  - 8-BIT ACCUMULATORS A AND B OR 16-BIT DOUBLE ACCUMULATOR D
  - INDEX REGISTER X
  - INDEX REGISTER Y
  - STACK POINTER
  - PROGRAM COUNTER
### Stack and Memory Layout

**SP BEFORE INTERRUPT**

<table>
<thead>
<tr>
<th>Address</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP +8</td>
<td>CCR</td>
</tr>
<tr>
<td>SP +6</td>
<td>B</td>
</tr>
<tr>
<td>SP +4</td>
<td>X_HI</td>
</tr>
<tr>
<td>SP +2</td>
<td>Y_HI</td>
</tr>
<tr>
<td>SP</td>
<td>RTN_HI</td>
</tr>
</tbody>
</table>

**SP AFTER INTERRUPT**

<table>
<thead>
<tr>
<th>Address</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP +9</td>
<td>A</td>
</tr>
<tr>
<td>SP +7</td>
<td>X_LO</td>
</tr>
<tr>
<td>SP +5</td>
<td>Y_LO</td>
</tr>
<tr>
<td>SP +3</td>
<td>RTN_LO</td>
</tr>
<tr>
<td>SP +1</td>
<td>SP -1</td>
</tr>
</tbody>
</table>

### Interrupt Vector Locations

- \$FFFE, \$FFFF: Power-On (POR) or External Reset
- \$FFFFC, \$FFFD: Clock Monitor Reset
- \$FFFFA, \$FFF8: Computer Operating Properly (COP Watchdog Reset)
- \$FFFB, \$FF9: Unimplemented Opcode Trap
- \$FF6, \$FF7: Software Interrupt Instruction (SWI)
- \$FF4, \$FF5: XIRQ
- \$FF2, \$FF3: IRQ
- \$FC0–\$FF1: Device-Specific Interrupt Sources
Notation Used in Instruction Set Summary

Explanation of Italic Expressions in Source Form Column

abc — A or B or CCR
abcdxys — A or B or CCR or D or X or Y or SP. Some assemblers also allow T2 or T3.
abd — A or B or D
abdxys — A or B or D or X or Y or SP
dxys — D or X or Y or SP
msk8 — 8-bit mask, some assemblers require # symbol before value
opr8i — 8-bit immediate value
opr16i — 16-bit immediate value
opr8a — 8-bit address used with direct address mode
opr16a — 16-bit address value
opr0_xysp — Indexed addressing postbyte code:
   oprx3,–xys Predecrement X or Y or SP by 1 . . . 8
   oprx3,+xys Preincrement X or Y or SP by 1 . . . 8
   oprx3,xys– Postdecrement X or Y or SP by 1 . . . 8
   oprx3,xys+ Postincrement X or Y or SP by 1 . . . 8
   oprx5,xys 5-bit constant offset from X or Y or SP or PC
   abdxysp Accumulator A or B or D offset from X or Y or SP or PC
opr3 — Any positive integer 1 . . . 8 for pre/post increment/decrement
opr5 — Any value in the range −16 . . . +15
opr9 — Any value in the range −256 . . . +255
opr16 — Any value in the range −32,768 . . . 65,535
page — 8-bit value for PPAGE, some assemblers require # symbol before this value
rel8 — Label of branch destination within −256 to +255 locations
rel9 — Label of branch destination within −512 to +511 locations
rel16 — Any label within 64K memory space
trapnum — Any 8-bit value in the range $30-$39 or $40-$FF
xys — X or Y or SP
xysp — X or Y or SP or PC
Address Modes

IMM — Immediate
IDX — Indexed (no extension bytes) includes:
   5-bit constant offset
   Pre/post increment/decrement by 1 . . . 8
   Accumulator A, B, or D offset
IDX1 — 9-bit signed offset (1 extension byte)
IDX2 — 16-bit signed offset (2 extension bytes)
[D, IDX] — Indexed indirect (accumulator D offset)
[IDX2] — Indexed indirect (16-bit offset)
INH — Inherent (no operands in object code)
REL — 2’s complement relative offset (branches)

Machine Coding

dd — 8-bit direct address $0000 to $00FF. (High byte assumed to be $00).
eeb — High-order byte of a 16-bit constant offset for indexed addressing.
eeb — Exchange/Transfer post-byte. See Table 3 on page 23.
eff — Low-order eight bits of a 9-bit signed constant offset for indexed addressing,
or low-order byte of a 16-bit constant offset for indexed addressing.
hh — High-order byte of a 16-bit extended address.
il — 8-bit immediate data value.
jj — High-order byte of a 16-bit immediate data value.
kk — Low-order byte of a 16-bit immediate data value.
lb — Loop primitive (DBNE) post-byte. See Table 4 on page 24.
ll — Low-order byte of a 16-bit extended address.
mm — 8-bit immediate mask value for bit manipulation instructions.
   Set bits indicate bits to be affected.
pg — Program page (bank) number used in CALL instruction.
qq — High-order byte of a 16-bit relative offset for long branches.
tn — Trap number $30–$39 or $40–$FF.
rr — Signed relative offset $80 (−128) to $7F (+127).
   Offset relative to the byte following the relative offset byte, or
   low-order byte of a 16-bit relative offset for long branches.
xb — Indexed addressing post-byte. See Table 1 on page 21
   and Table 2 on page 22.
Access Detail

Each code letter equals one CPU cycle. Uppercase = 16-bit operation and lowercase = 8-bit operation. For complex sequences see the CPU12 Reference Manual (CPU12RM/AD).

- \( f \) — Free cycle, CPU doesn’t use bus
- \( g \) — Read PPAGE internally
- \( I \) — Read indirect pointer (indexed indirect)
- \( i \) — Read indirect PPAGE value (call indirect)
- \( n \) — Write PPAGE internally
- \( o \) — Optional program word fetch (P) if instruction is misaligned and has an odd number of bytes of object code — otherwise, appears as a free cycle (f)
- \( p \) — Program word fetch (always an aligned word read)
- \( r \) — 8-bit data read
- \( R \) — 16-bit data read
- \( s \) — 8-bit stack write
- \( S \) — 16-bit stack write
- \( w \) — 8-bit data write
- \( W \) — 16-bit data write
- \( u \) — 8-bit stack read
- \( U \) — 16-bit stack read
- \( V \) — 16-bit vector fetch
- \( t \) — 8-bit conditional read (or free cycle)
- \( T \) — 16-bit conditional read (or free cycle)
- \( x \) — 8-bit conditional write

Special Cases

- \( PPP/P \) — Short branch, PPP if branch taken, P if not
- \( OPPP/OPO \) — Long branch, OPPP if branch taken, OPO if not

Condition Codes Columns

- \( - \) — Status bit not affected by operation.
- \( 0 \) — Status bit cleared by operation.
- \( 1 \) — Status bit set by operation.
- \( \Delta \) — Status bit affected by operation.
- \( \downarrow \) — Status bit may be cleared or remain set, but is not set by operation.
- \( \uparrow \) — Status bit may be set or remain cleared, but is not cleared by operation.
- \( ? \) — Status bit may be changed by operation but the final state is not defined.
- \( ! \) — Status bit used for a special purpose.
### Instruction Set Summary

| Source Form | Operation | Addr. Mode | Machine Coding (hex) | Access Detail | S | X | H | I | N | Z | V | C |
|-------------|-----------|------------|----------------------|---------------|---|---|---|---|---|---|---|---|---|
| ABA         | (A) + (B) ⇒ A Add Accumulators A and B | NH | 18 06 00 | – – | – | – | – | – | – | – | – | – | – |
| ABX         | (B) + (X) ⇒ X Translates to LEAX B,X | IDX | 1A E5 PP | – – – – – – – – |
| ABY         | (B) + (Y) ⇒ Y Translates to LEAY B,Y | IDX | 19 ED PP | – – – – – – – – |
| ADCB (op16) | (M) + C ⇒ B Add with Carry to B | IMM | 89 li P | – – – – – – – – |
| ADCB (op16) | (M) + C ⇒ A Add with Carry to A | IMM | 90 li P | – – – – – – – – |
| ADDA (op16) | (M) ⇒ A Add without Carry to A | IMM | 8B li P | – – – – – – – – |
| ADDB (op16) | (M) ⇒ B Add without Carry to B | IMM | 8C li P | – – – – – – – – |
| ADDD (op16) | (A:B) + (M:M+1) Add 16-Bit to D (A:B) | IMM | C3 jj kk | – – – – – – – |
| ANDA (op16) | (M) ⇒ A Logical And A with Memory | IMM | 84 li P | – – – – – – – |
| ANDB (op16) | (M) ⇒ B Logical And B with Memory | IMM | 84 li P | – – – – – – – |

Note 1. Due to internal CPU requirements, the program word fetch is performed twice to the same address during this instruction.
### Instruction Set Summary (Continued)

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Operation</th>
<th>Addr. Mode</th>
<th>Machine Coding (hex)</th>
<th>Access Detail</th>
<th>S</th>
<th>X</th>
<th>H</th>
<th>I</th>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANDCC #opr8</td>
<td>(CCR) • (M) ⇒ CCR Logical And CCR with Memory</td>
<td>IMM</td>
<td>10 ii P</td>
<td>- - - - - - - - -</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASL opr6a</td>
<td>ASL opr0, xysp ASL opr0, xysp ASL opr16, xysp ASL [D, xysp] ASL [opr16, xysp] ASL</td>
<td>EXT IDX IDX1 IDX2 [IDX2]</td>
<td>78 hh ll rCPrw - - - - - - -</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASLR opr16a</td>
<td>ASLR opr0, xysp ASLR opr0, xysp ASLR opr16, xysp ASLR [D, xysp] ASLR [opr16, xysp] ASLR</td>
<td>EXT IDX IDX1 IDX2 [IDX2]</td>
<td>68 x x x x ff frPrw - - - - - - -</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCC rel8</td>
<td>Branch if Carry Clear (if C = 0) REL</td>
<td>24 rr PPP/P³ - - - - - - -</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCLR opr8a, msk8</td>
<td>BCLR opr0a, msk8 BCLR opr0, xysp BCLR opr16, xysp BCLR [D, xysp] BCLR [opr16, xysp] BCLR</td>
<td>DIR EXT IDX IDX1 IDX2 [IDX2]</td>
<td>40 dd mm rPw - - - - - - -</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BGND</td>
<td>Place CPU in Background Mode see CPU12 Reference Manual</td>
<td>REL</td>
<td>00 vPPP</td>
<td>- - - - - - -</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BGT rel8</td>
<td>Branch if Greater Than (if Z &amp; (N ⊕ V) = 0) (signed) REL</td>
<td>2E rr PPP/P³</td>
<td>- - - - - - -</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BHI rel8</td>
<td>Branch if Higher (if C &amp; (N ⊕ V) = 0) (unsigned) REL</td>
<td>22 rr PPP/P³</td>
<td>- - - - - - -</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BITA #opr8</td>
<td>BITA opr6a BITA opr16a BITA opr0, xysp BITA opr0, xysp BITA opr16, xysp BITA [D, xysp] BITA [opr16, xysp]</td>
<td>IMM DIR EXT IDX IDX1 IDX2 [IDX2]</td>
<td>85 li A5 x A5 x frPrfP - - - - - - -</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BITB #opr8</td>
<td>BITB opr6a BITB opr16a BITB opr0, xysp BITB opr0, xysp BITB opr16, xysp BITB [D, xysp] BITB [opr16, xysp]</td>
<td>IMM DIR EXT IDX IDX1 IDX2 [IDX2]</td>
<td>C5 li E5 x E5 x frPrfP - - - - - - -</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: 1. PPP/P indicates this instruction takes three cycles to refill the instruction queue if the branch is taken and one program fetch cycle if the branch is not taken.
## Instruction Set Summary (Continued)

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Operation</th>
<th>Addr. Mode</th>
<th>Machine Coding (hex)</th>
<th>Access Detail</th>
<th>S</th>
<th>X</th>
<th>H</th>
<th>I</th>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLE rel8</td>
<td>Branch if Less Than or Equal (if (Z + (N \oplus V) = 1)) (signed)</td>
<td>REL</td>
<td>2F rr</td>
<td>P/D/P'</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BLO rel8</td>
<td>Branch if Lower (if (C = 1)) (unsigned)</td>
<td>REL</td>
<td>25 rr</td>
<td>P/D/P'</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BLS rel8</td>
<td>Branch if Lower or Same (if (C + Z = 1)) (unsigned)</td>
<td>REL</td>
<td>23 rr</td>
<td>P/D/P'</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BLT rel8</td>
<td>Branch if Less Than (if (N \oplus V = 1)) (signed)</td>
<td>REL</td>
<td>2B rr</td>
<td>P/D/P'</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BMI rel8</td>
<td>Branch if Minus (if (N = 1))</td>
<td>REL</td>
<td>26 rr</td>
<td>P/D/P'</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BNE rel8</td>
<td>Branch if Not Equal (if (Z = 0))</td>
<td>REL</td>
<td>2A rr</td>
<td>P/D/P'</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BPL rel8</td>
<td>Branch if Plus (if (N = 0))</td>
<td>REL</td>
<td>20 rr</td>
<td>P/D/P'</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BRA rel8</td>
<td>Branch Always (if (1 = 1))</td>
<td>REL</td>
<td>21 rr</td>
<td>P</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BRCLR opr8a, msk8, rel8</td>
<td>Branch if (M) (\bullet) (mm) = 0</td>
<td>DIR</td>
<td>4F dd mm rr</td>
<td>rPPP</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BRCLR opr16a, msk8, rel8</td>
<td>Subroutine address (\Rightarrow) PC</td>
<td>EXT</td>
<td>1F hh ll mm rr</td>
<td>rPPP</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BRCLR opr0_xysp, msk8, rel8</td>
<td>(SP) (-) 2 (\Rightarrow) SP; RTNH:RTNL (\Rightarrow) M(SP):M(SP+1)</td>
<td>IDX</td>
<td>0F xb mm rr</td>
<td>rPPP</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BRCLR opr9,xysp, msk8, rel8</td>
<td>(SP) (-) 1 (\Rightarrow) SP; (PPG) (\Rightarrow) M(SP); Program address (\Rightarrow) PC</td>
<td>IDX1</td>
<td>0b sb mm rr</td>
<td>rPPP</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BRCLR opr16,xysp, msk8, rel8</td>
<td>(SP) (-) 1 (\Rightarrow) SP; pg (\Rightarrow) PPAE register; Program address (\Rightarrow) PC</td>
<td>IDX2</td>
<td>0b sb ef mm rr</td>
<td>rPPP</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BRN rel8</td>
<td>Branch never (if (1 = 0))</td>
<td>REL</td>
<td>21 rr</td>
<td>P</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BRSET opr8, msk8, rel8</td>
<td>Branch if (M) (\bullet) (mm) = 0 (if all Selected Bit(s) Set)</td>
<td>DIR</td>
<td>4E dd mm rr</td>
<td>rPPP</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BRSET opr16a, msk8</td>
<td>Subroutine address (\Rightarrow) PC</td>
<td>EXT</td>
<td>1E hh ll mm rr</td>
<td>rPPP</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BRSET opr0_xysp, msk8</td>
<td>(SP) (-) 2 (\Rightarrow) SP; RTNH:RTNL (\Rightarrow) M(SP):M(SP+1)</td>
<td>IDX</td>
<td>0C xb mm rr</td>
<td>rPPP</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BRSET opr9,xysp, msk8</td>
<td>(SP) (-) 1 (\Rightarrow) SP; (PPG) (\Rightarrow) M(SP); Program address (\Rightarrow) PC</td>
<td>IDX1</td>
<td>0C xb ef mm rr</td>
<td>rPPP</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BRSET opr16,xysp, msk8</td>
<td>(SP) (-) 1 (\Rightarrow) SP; pg (\Rightarrow) PPAE register; Program address (\Rightarrow) PC</td>
<td>IDX2</td>
<td>0C xb ef mm rr</td>
<td>rPPP</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BSR rel8</td>
<td>Branch if Overflow Bit Clear (if (V = 0))</td>
<td>REL</td>
<td>28 rr</td>
<td>P/D/P'</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BVS rel8</td>
<td>Branch if Overflow Bit Set (if (V = 1))</td>
<td>REL</td>
<td>29 rr</td>
<td>P/D/P'</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>CALL opr16a, page</td>
<td>Call subroutine in extended memory (Program may be located on another expansion memory page.)</td>
<td>EXT</td>
<td>4A hh ll pg</td>
<td>gntSSPPP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>CALL opr0_xysp, page</td>
<td>Indirect modes get program address and new pg value based on pointer.</td>
<td>IDX</td>
<td>4b sb pg</td>
<td>gntSSPPP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>CALL opr9,xysp, page</td>
<td>Call in extended memory (Program is located on another expansion memory page.)</td>
<td>IDX1</td>
<td>4b sb ef pg</td>
<td>gntSSPPP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>CALL opr16,xysp</td>
<td>Call subroutine in extended memory (Program may be located on another expansion memory page.)</td>
<td>IDX2</td>
<td>4b sb ef</td>
<td>gntSSPPP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>CBA</td>
<td>Compare 8-Bit Accumulators</td>
<td>INH</td>
<td>18 17</td>
<td>00</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>CLI</td>
<td>Translates to ANDCC #$FE</td>
<td>IMM</td>
<td>10 FE</td>
<td>P</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Notes
1. PPP/P indicates this instruction takes three cycles to refill the instruction queue if the branch is taken and one program fetch cycle if the branch is not taken.
## Instruction Set Summary (Continued)

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Operation</th>
<th>Addr. Mode</th>
<th>Machine Coding (hex)</th>
<th>Access Detail</th>
<th>S</th>
<th>X</th>
<th>H</th>
<th>I</th>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR opr16a</td>
<td>0 (\rightarrow) M</td>
<td>Clear Memory Location</td>
<td>EXT 79 hh ll</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLR opr0,xysp</td>
<td>IDX 69 xb</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>CLR opr9,yxsp</td>
<td>IDX1 69 xb ff</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>CLR [D,yxsp]</td>
<td>IDX2 69 xb ee ff</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>CLR [opr16,xysp]</td>
<td>[D,IDX] 69 xb ee ff</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>CLR</td>
<td>0 (\Rightarrow) B</td>
<td>Clear Accumulator B</td>
<td>INH 87</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLRA</td>
<td>INH</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>CLRB</td>
<td>INH</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>CLV</td>
<td>0 (\Rightarrow) V</td>
<td>Translates to ANDCC #SFD</td>
<td>IMM 10 FD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMPA #opr8i</td>
<td>(A) (\rightarrow) (M)</td>
<td>Compare Accumulator A with Memory</td>
<td>IMM 81 li</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMPA opr8a</td>
<td>DIR 91 dd</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>CMPA opr16a</td>
<td>EXT 8b hh ll</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>CMPA opr0,xysp</td>
<td>IDX A1 xb</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>CMPA opr9,yxsp</td>
<td>IDX1 A1 xb ff</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>CMPA opr16,xysp</td>
<td>IDX2 A1 xb ee ff</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>CMPA [D,yxsp]</td>
<td>[D,IDX] A1 xb fiorP</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>CMPA [opr16,xysp]</td>
<td>[D,IDX2] A1 xb ee ff</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>CMPB #opr8i</td>
<td>(B) (\rightarrow) (M)</td>
<td>Compare Accumulator B with Memory</td>
<td>IMM C1 li</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMPB opr8a</td>
<td>DIR 91 dd</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>CMPB opr16a</td>
<td>EXT F1 hh ll</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>CMPB opr0,xysp</td>
<td>IDX E1 xb</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>CMPB opr9,yxsp</td>
<td>IDX1 E1 xb ff</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>CMPB opr16,xysp</td>
<td>IDX2 E1 xb ee ff</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>CMPB [D,yxsp]</td>
<td>[D,IDX] E1 xb fiorP</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>CMPB [opr16,xysp]</td>
<td>[D,IDX2] E1 xb ee ff</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>COM opr16a</td>
<td>(M) (\Rightarrow) M equivalent to SFF (\rightarrow) (M) (\Rightarrow) M</td>
<td>1's Complement Memory Location</td>
<td>EXT 71 hh ll</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COM opr0,xysp</td>
<td>IDX 61 xb</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>COM opr9,yxsp</td>
<td>IDX1 61 xb ff</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>COM opr16,xysp</td>
<td>IDX2 61 xb ee ff</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>COM [D,yxsp]</td>
<td>[D,IDX] 61 xb fiorPw</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>COM [opr16,xysp]</td>
<td>[D,IDX2] 61 xb ee ff</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>COMA</td>
<td>(A) (\Rightarrow) A</td>
<td>Complement Accumulator A</td>
<td>IMM 41</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMB</td>
<td>(B) (\Rightarrow) B</td>
<td>Complement Accumulator B</td>
<td>INH 51</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPD #opr16i</td>
<td>(A-B) (\rightarrow) (M:M+1)</td>
<td>Compare D to Memory (16-Bit)</td>
<td>IMM BC jj kk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPD opr8a</td>
<td>DIR 9C dd</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>CPD opr15a</td>
<td>EXT BC hh ll</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>CPD opr0,xysp</td>
<td>IDX AC xb</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>CPD opr9,yxsp</td>
<td>IDX1 AC xb ff</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>CPD opr16,xysp</td>
<td>IDX2 AC xb ee ff</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>CPD [D,yxsp]</td>
<td>[D,IDX] AC xb fiorP</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>CPD [opr16,xysp]</td>
<td>[D,IDX2] AC xb ee ff</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>CPS #opr16i</td>
<td>(SP) (\rightarrow) (M:M+1)</td>
<td>Compare SP to Memory (16-Bit)</td>
<td>IMM BF jj kk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPS opr8a</td>
<td>DIR 9F dd</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>CPS opr15a</td>
<td>EXT BF hh ll</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>CPS opr0,xysp</td>
<td>IDX AC xb</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>CPS opr9,yxsp</td>
<td>IDX1 AC xb ff</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>CPS opr16,xysp</td>
<td>IDX2 AC xb ee ff</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>CPS [D,yxsp]</td>
<td>[D,IDX] AC xb fiorP</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>CPS [opr16,xysp]</td>
<td>[D,IDX2] AC xb ee ff</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>CPX #opr16i</td>
<td>(X) (\rightarrow) (M:M+1)</td>
<td>Compare X to Memory (16-Bit)</td>
<td>IMM BE jj kk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPX opr8a</td>
<td>DIR 9E dd</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>CPX opr15a</td>
<td>EXT BE hh ll</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>CPX opr0,xysp</td>
<td>IDX AE xb</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>CPX opr9,yxsp</td>
<td>IDX1 AE xb ff</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>CPX opr16,xysp</td>
<td>IDX2 AE xb ee ff</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>CPX [D,yxsp]</td>
<td>[D,IDX] AE xb fiorP</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>CPX [opr16,xysp]</td>
<td>[D,IDX2] AE xb ee ff</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>CPHY #opr16i</td>
<td>(Y) (\rightarrow) (M:M+1)</td>
<td>Compare Y to Memory (16-Bit)</td>
<td>IMM BD jj kk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPHY opr8a</td>
<td>DIR 9D dd</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>CPHY opr15a</td>
<td>EXT BD hh ll</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>CPHY opr0,xysp</td>
<td>IDX AD xb</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>CPHY opr9,yxsp</td>
<td>IDX1 AD xb ff</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>CPHY opr16,xysp</td>
<td>IDX2 AD xb ee ff</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>CPHY [D,yxsp]</td>
<td>[D,IDX] AD xb fiorP</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>CPHY [opr16,xysp]</td>
<td>[D,IDX2] AD xb ee ff</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>

- NON-DISCLOSURE AGREEMENT REQUIRED
### Instruction Set Summary (Continued)

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Operation</th>
<th>Addr. Mode</th>
<th>Machine Coding (hex)</th>
<th>Access Detail</th>
<th>S X H I N Z V C</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAA</td>
<td>Adjust Sum to BCD Decimal Adjust Accumulator A</td>
<td>INH</td>
<td>18 07</td>
<td>OFO</td>
<td>--</td>
</tr>
<tr>
<td>DBEQ abdys, rel9</td>
<td>(cntr) – 1 =&gt; cntr if (cntr) = 0, then Branch else Continue to next instruction Decrement Counter and Branch if # 0 (cntr = A, B, D, X, Y, or SP)</td>
<td>REL (9-bit)</td>
<td>04 lb rr</td>
<td>PPP</td>
<td>--</td>
</tr>
<tr>
<td>DBNE abdys, rel9</td>
<td>(cntr) – 1 = cntr if (cntr) not = 0, then Branch; else Continue to next instruction Decrement Counter and Branch if # 0 (cntr = A, B, D, X, Y, or SP)</td>
<td>REL (9-bit)</td>
<td>04 lb rr</td>
<td>PPP</td>
<td>--</td>
</tr>
<tr>
<td>DECB</td>
<td>(A) – $01 =&gt; A Decrement A</td>
<td>INH</td>
<td>43</td>
<td></td>
<td>--</td>
</tr>
<tr>
<td>DEC op16a</td>
<td>(M) – $01 =&gt; M Decrement Memory Location</td>
<td>EXT</td>
<td>73 hh 11</td>
<td>eCPw</td>
<td>--</td>
</tr>
<tr>
<td>DEC op0,xysp</td>
<td>(M) – $01 =&gt; M Decrement Memory Location</td>
<td>EXT</td>
<td>63 xb</td>
<td>SPw</td>
<td>--</td>
</tr>
<tr>
<td>DEC op16,xysp</td>
<td>(M) – $01 =&gt; M Decrement Memory Location</td>
<td>EXT</td>
<td>63 xb ff</td>
<td>SPw</td>
<td>--</td>
</tr>
<tr>
<td>DEC [D,xysp]</td>
<td>(M) – $01 =&gt; M Decrement Memory Location</td>
<td>EXT</td>
<td>63 xb ee ff</td>
<td>SPPw</td>
<td>--</td>
</tr>
<tr>
<td>DECA</td>
<td>(A) – $01 =&gt; A Decrement A</td>
<td>INH</td>
<td>63</td>
<td></td>
<td>--</td>
</tr>
<tr>
<td>DECB</td>
<td>(B) – $01 =&gt; B Decrement B</td>
<td>INH</td>
<td>43</td>
<td></td>
<td>--</td>
</tr>
<tr>
<td>DEP</td>
<td>(M) – $01 =&gt; M Decrement Memory Location</td>
<td>EXT</td>
<td>63</td>
<td></td>
<td>--</td>
</tr>
<tr>
<td>DES</td>
<td>(SP) – $0001 =&gt; SP Translates to LEAS –1,SP</td>
<td>INH</td>
<td>19 9F</td>
<td>PP</td>
<td>--</td>
</tr>
<tr>
<td>DEX</td>
<td>(X) – $0001 =&gt; X Decrement Index Register X</td>
<td>INH</td>
<td>09</td>
<td></td>
<td>--</td>
</tr>
<tr>
<td>DEX</td>
<td>(Y) – $0001 =&gt; Y Decrement Index Register Y</td>
<td>INH</td>
<td>03</td>
<td></td>
<td>--</td>
</tr>
<tr>
<td>EDIV</td>
<td>(Y) + (X) =&gt; Y Remainder =&gt; D 32 × 16 Bit =&gt; 16 Bit Divide (unsigned)</td>
<td>INH</td>
<td>11</td>
<td></td>
<td>--</td>
</tr>
<tr>
<td>EDIVS</td>
<td>(Y) + (X) =&gt; Y Remainder =&gt; D 32 × 16 Bit =&gt; 16 Bit Divide (signed)</td>
<td>INH</td>
<td>18 14</td>
<td></td>
<td>--</td>
</tr>
<tr>
<td>EMACS op16a 2</td>
<td>(M)[0:3]M(x,y) × (M[y,x]) + (M[M+3]) =&gt; M+M+3 16 × 16 Bit =&gt; 32 Bit Multiply and Accumulate (signed)</td>
<td>Special</td>
<td>19 12 hh 11</td>
<td>OR1200000000</td>
<td>--</td>
</tr>
<tr>
<td>EMAXD op0,xysp</td>
<td>MAX(D), (M+M+1) =&gt; D MAX of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1A xb</td>
<td>ORP</td>
<td>--</td>
</tr>
<tr>
<td>EMAXD op9,xysp</td>
<td>MAX(D), (M+M+1) =&gt; D MAX of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1A xb ff</td>
<td>ORPO</td>
<td>--</td>
</tr>
<tr>
<td>EMAXD op16,xysp</td>
<td>MAX(D), (M+M+1) =&gt; D MAX of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1A xb ee ff</td>
<td>ORPWP</td>
<td>--</td>
</tr>
<tr>
<td>EMAXD [D,xysp]</td>
<td>MAX(D), (M+M+1) =&gt; D MAX of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1A xb</td>
<td>ORP</td>
<td>--</td>
</tr>
<tr>
<td>EMAXD [op16,xysp]</td>
<td>MAX(D), (M+M+1) =&gt; D MAX of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1A xb ee ff</td>
<td>ORPP</td>
<td>--</td>
</tr>
<tr>
<td>EMAXD op0,xysp</td>
<td>MAX(D), (M+M+1) =&gt; D MAX of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1A xb</td>
<td>ORP</td>
<td>--</td>
</tr>
<tr>
<td>EMAXD op9,xysp</td>
<td>MAX(D), (M+M+1) =&gt; D MAX of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1A xb ff</td>
<td>ORPO</td>
<td>--</td>
</tr>
<tr>
<td>EMAXD op16,xysp</td>
<td>MAX(D), (M+M+1) =&gt; D MAX of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1A xb ee ff</td>
<td>ORPWP</td>
<td>--</td>
</tr>
<tr>
<td>EMAXD [D,xysp]</td>
<td>MAX(D), (M+M+1) =&gt; D MAX of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1A xb</td>
<td>ORP</td>
<td>--</td>
</tr>
<tr>
<td>EMAXD [op16,xysp]</td>
<td>MAX(D), (M+M+1) =&gt; D MAX of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1A xb ee ff</td>
<td>ORPP</td>
<td>--</td>
</tr>
<tr>
<td>EMIND op0,xysp</td>
<td>MIN(D), (M+M+1) =&gt; D MIN of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1B xb</td>
<td>ORP</td>
<td>--</td>
</tr>
<tr>
<td>EMIND op9,xysp</td>
<td>MIN(D), (M+M+1) =&gt; D MIN of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1B xb ff</td>
<td>ORPO</td>
<td>--</td>
</tr>
<tr>
<td>EMIND op16,xysp</td>
<td>MIN(D), (M+M+1) =&gt; D MIN of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1B xb ee ff</td>
<td>ORPWP</td>
<td>--</td>
</tr>
<tr>
<td>EMIND [D,xysp]</td>
<td>MIN(D), (M+M+1) =&gt; D MIN of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1B xb</td>
<td>ORP</td>
<td>--</td>
</tr>
<tr>
<td>EMIND [op16,xysp]</td>
<td>MIN(D), (M+M+1) =&gt; D MIN of 2 Unsigned 16-Bit Values</td>
<td>IDX</td>
<td>18 1B xb ee ff</td>
<td>ORPP</td>
<td>--</td>
</tr>
<tr>
<td>EMUL</td>
<td>(D) × (Y) =&gt; Y.D 16 × 16 Bit Multiply (unsigned)</td>
<td>INH</td>
<td>13</td>
<td></td>
<td>--</td>
</tr>
</tbody>
</table>

Notes:
1. Due to internal CPU requirements, the program word fetch is performed twice to the same address during this instruction.
2. op16a is an extended address specification. Both X and Y point to source operands.
### Instruction Set Summary (Continued)

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Operation</th>
<th>Addr. Mode</th>
<th>Machine Coding (hex)</th>
<th>Access Detail</th>
<th>S</th>
<th>X</th>
<th>H</th>
<th>I</th>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMULS</td>
<td>(D) × (Y) ⇒ Y.D 16 × 16 Bit Multiply (signed)</td>
<td>INH</td>
<td>18 13</td>
<td>OFF</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>EORA opr8l</td>
<td>(A) @ (M) ⇒ A Exclusive-OR A with Memory</td>
<td>IMM</td>
<td>88 ii</td>
<td>P</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>0</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DIR</td>
<td>98 dd</td>
<td>sFP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXT</td>
<td>88 hh ll</td>
<td>sFP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDX</td>
<td>A8 xb</td>
<td>sFP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDX1</td>
<td>A8 xb ff</td>
<td>sFP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDX2</td>
<td>A8 xb ee ff</td>
<td>sFP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[IDX2]</td>
<td>A8 xb ff</td>
<td>sFP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[IDX2]</td>
<td>A8 xb ee ff</td>
<td>sFP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>EORB opr8l</td>
<td>(B) @ (M) ⇒ B Exclusive-OR B with Memory</td>
<td>IMM</td>
<td>C8 ii</td>
<td>P</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DIR</td>
<td>D8 dd</td>
<td>sFP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXT</td>
<td>88 hh ll</td>
<td>sFP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDX</td>
<td>E8 xb</td>
<td>sFP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDX1</td>
<td>E8 xb ff</td>
<td>sFP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDX2</td>
<td>E8 xb ee ff</td>
<td>sFP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[IDX2]</td>
<td>E8 xb ff</td>
<td>sFP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[IDX2]</td>
<td>E8 xb ee ff</td>
<td>sFP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>ETBL opr0_xysp</td>
<td>(M(M+1)+ (B)(X)(M+2:M+3) − (M:M+1)) ⇒ D 16-Bit Table Lookup and Interpolate</td>
<td>IDX</td>
<td>18 3F xb</td>
<td>ORRffffffffffP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Normalize B, and index before ETBL. &lt;ea&gt; points at first table entry (M:M+1) and B is fractional part of lookup value (no indirect addr. modes or extensions allowed)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXG abcdys,abcdys</td>
<td>(r1) ⇒ (r2) (if r1 and r2 same size) or 000:00 (if r1=8-bit; r2=16-bit) or (r1) ⇒ (r2) (if r1=16-bit; r2=8-bit) r1 and r2 may be A, B, CCR, D, X, Y, or SP</td>
<td>INH</td>
<td>B7 sb</td>
<td>P</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(9-bit)</td>
<td>04 lb rr</td>
<td>PPP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>FDIV</td>
<td>(D) ÷ (X) ⇒ X, Remainder ⇒ D 16 × 16 Bit Fractional Divide</td>
<td>INH</td>
<td>18 11</td>
<td>0fffffff cc</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>IBEQ abdxys, rel9</td>
<td>(cntr) + 1 ⇒ cntr If (cntr) = 0, then Branch else Continue to next instruction Increment Counter and Branch if = 0 (cntr = A, B, D, X, Y, or SP)</td>
<td>REL</td>
<td>04 lb rr</td>
<td>PPP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>IBNE abdxys, rel9</td>
<td>(cntr) + 1 ⇒ cntr If (cntr) not = 0, then Branch; else Continue to next instruction Increment Counter and Branch if not = 0 (cntr = A, B, D, X, Y, or SP)</td>
<td>REL</td>
<td>04 lb rr</td>
<td>PPP</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>IDIV</td>
<td>(D) ÷ (X) ⇒ X, Remainder ⇒ D 16 × 16 Bit Integer Divide (unsigned)</td>
<td>INH</td>
<td>18 10</td>
<td>0fffffff cc</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>0</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>IDIVS</td>
<td>(D) ÷ (X) ⇒ X, Remainder ⇒ D 16 × 16 Bit Integer Divide (signed)</td>
<td>INH</td>
<td>18 15</td>
<td>0fffffff cc</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>INC opr16a</td>
<td>(M) + $01 ⇒ M Increment Memory Byte</td>
<td>EXT</td>
<td>72 hh ll</td>
<td>sCPw</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>INC opr0_xysp</td>
<td>(M) + $01 ⇒ M Increment Memory Byte</td>
<td>IDX</td>
<td>62 xb</td>
<td>sPw</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>INC opr9_xysp</td>
<td>(M) + $01 ⇒ M Increment Memory Byte</td>
<td>IDX1</td>
<td>62 xb ff</td>
<td>sFPw</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>INC opr16_xysp</td>
<td>(M) + $01 ⇒ M Increment Memory Byte</td>
<td>IDX2</td>
<td>62 xb ee ff</td>
<td>sFPw</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>INC [X:ysp]</td>
<td>(M) + $01 ⇒ M Increment Memory Byte</td>
<td>[IDX2]</td>
<td>62 xb ff</td>
<td>sFPw</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>INCA</td>
<td>(A) + $01 ⇒ A Increment Acc. A</td>
<td>INH</td>
<td>42</td>
<td>O</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>INCB</td>
<td>(B) + $01 ⇒ B Increment Acc. B</td>
<td>INH</td>
<td>52</td>
<td>O</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>INS</td>
<td>(SP) + $0001 ⇒ SP Translates to LEAS 1,SP</td>
<td>IDX</td>
<td>1B 81</td>
<td>PP*</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>INX</td>
<td>(X) + $0001 ⇒ X Increment Index Register X</td>
<td>INH</td>
<td>08</td>
<td>O</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>INY</td>
<td>(Y) + $0001 ⇒ Y Increment Index Register Y</td>
<td>INH</td>
<td>02</td>
<td>O</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Note 1. Due to internal CPU requirements, the program word fetch is performed twice to the same address during this instruction.
### Instruction Set Summary (Continued)

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Operation</th>
<th>Addr. Mode</th>
<th>Machine Coding (hex)</th>
<th>Access Detail</th>
<th>S</th>
<th>X</th>
<th>H</th>
<th>I</th>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMP opr16a</td>
<td>Subroutine address ⇒ PC</td>
<td>EXT</td>
<td>06 hh 11</td>
<td>PPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JMP opr0,xysp</td>
<td>Jump</td>
<td>IDX</td>
<td>05 xb</td>
<td>PPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JMP opr16,xysp</td>
<td></td>
<td>IDX2</td>
<td>05 xb ff</td>
<td>PPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JMP [D,xysp]</td>
<td></td>
<td>[IDX2]</td>
<td>05 xb se ff</td>
<td>fPPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JMP [opr16,xysp]</td>
<td></td>
<td>[IDX2]</td>
<td>05 xb se ff</td>
<td>fPPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSR opr8a</td>
<td>[SPI] – 2 ⇒ SP; RTN0,RTN1 ⇒ M_sp,M_sp+1; Jump to Subroutine</td>
<td>DIR</td>
<td>17 dd</td>
<td>PPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSR opr16a</td>
<td></td>
<td>EXT</td>
<td>16 hh 11</td>
<td>PPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSR opr0,xysp</td>
<td></td>
<td>IDX</td>
<td>15 xb</td>
<td>PPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSR opr16,xysp</td>
<td></td>
<td>IDX1</td>
<td>15 xb ff</td>
<td>PPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSR [D,xysp]</td>
<td></td>
<td>IDX2</td>
<td>15 xb se ff</td>
<td>fPPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSR [opr16,xysp]</td>
<td></td>
<td>[IDX2]</td>
<td>15 xb se ff</td>
<td>fPPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBCC rel6</td>
<td>Long Branch if Carry Clear (if C = 0)</td>
<td>REL</td>
<td>18 24 qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBCS rel6</td>
<td>Long Branch if Carry Set (if C = 1)</td>
<td>REL</td>
<td>18 25 qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBEQ rel6</td>
<td>Long Branch if Equal (if Z = 1)</td>
<td>REL</td>
<td>18 27 qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBGE rel6</td>
<td>Long Branch Greater Than or Equal (if N ⊕ V = 0) (signed)</td>
<td>REL</td>
<td>18 2C qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBGT rel6</td>
<td>Long Branch if Greater Than (if Z + (N ⊕ V) = 0) (signed)</td>
<td>REL</td>
<td>18 2K qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBHI rel6</td>
<td>Long Branch if Higher (if C ⊕ Z = 0) (unsigned)</td>
<td>REL</td>
<td>18 22 qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBHS rel6</td>
<td>Long Branch if Higher or Same (if C = 0) (unsigned) same function as LBCC</td>
<td>REL</td>
<td>18 24 qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBLE rel6</td>
<td>Long Branch if Less Than or Equal (if Z ⊕ (N ⊗ V) = 1) (signed)</td>
<td>REL</td>
<td>18 2F qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBO rel6</td>
<td>Long Branch if Lower (if C ⊗ Z = 0) (unsigned) same function as LBCS</td>
<td>REL</td>
<td>18 25 qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBLS rel6</td>
<td>Long Branch if Lower or Same (if C ⊕ Z = 1) (unsigned)</td>
<td>REL</td>
<td>18 23 qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBLT rel6</td>
<td>Long Branch if Less Than (if N ⊗ V = 1) (signed)</td>
<td>REL</td>
<td>18 2D qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBMI rel6</td>
<td>Long Branch if Minus (if N = 1)</td>
<td>REL</td>
<td>18 2B qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBNE rel6</td>
<td>Long Branch if Not Equal (if Z = 0)</td>
<td>REL</td>
<td>18 26 qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBPL rel6</td>
<td>Long Branch if Plus (if N = 0)</td>
<td>REL</td>
<td>18 2A qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBRA rel6</td>
<td>Long Branch Always (if 1=1)</td>
<td>REL</td>
<td>18 20 qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBRN rel6</td>
<td>Long Branch Never (if 1=0)</td>
<td>REL</td>
<td>18 21 qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBVC rel6</td>
<td>Long Branch if Overflow Bit Clear (if V=0)</td>
<td>REL</td>
<td>18 28 qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LBVS rel6</td>
<td>Long Branch if Overflow Bit Set (if V=1)</td>
<td>REL</td>
<td>18 29 qq rr</td>
<td>OPPP/OPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA A #opr8i</td>
<td>Load Accumulator A</td>
<td>IMM</td>
<td>06 11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Δ</td>
<td>Δ</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA A opr8a</td>
<td></td>
<td>DIR</td>
<td>06 dd</td>
<td>rFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA opr16a</td>
<td></td>
<td>EXT</td>
<td>06 hh 11</td>
<td>sOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA opr0,xysp</td>
<td></td>
<td>IDX</td>
<td>A6 xb</td>
<td>rFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA opr16,xysp</td>
<td></td>
<td>IDX1</td>
<td>A6 xb ff</td>
<td>rPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA opr0,xysp</td>
<td></td>
<td>IDX2</td>
<td>A6 xb se ff</td>
<td>fPPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA [D,xysp]</td>
<td></td>
<td>[IDX2]</td>
<td>A6 xb se ff</td>
<td>fPPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA [opr16,xysp]</td>
<td></td>
<td>[IDX2]</td>
<td>A6 xb se ff</td>
<td>fPPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA A #opr8i</td>
<td>Load Accumulator B</td>
<td>IMM</td>
<td>06 11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Δ</td>
<td>Δ</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA B #opr8i</td>
<td></td>
<td>DIR</td>
<td>06 dd</td>
<td>rFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA B opr8a</td>
<td></td>
<td>EXT</td>
<td>06 hh 11</td>
<td>sOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA B opr16a</td>
<td></td>
<td>IDX</td>
<td>A6 xb</td>
<td>rFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA B opr0,xysp</td>
<td></td>
<td>IDX1</td>
<td>A6 xb ff</td>
<td>rPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA B opr16,xysp</td>
<td></td>
<td>IDX2</td>
<td>A6 xb se ff</td>
<td>fPPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA B [D,xysp]</td>
<td></td>
<td>[IDX2]</td>
<td>A6 xb se ff</td>
<td>fPPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA B [opr16,xysp]</td>
<td></td>
<td>[IDX2]</td>
<td>A6 xb se ff</td>
<td>fPPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note 1. OPPP/OPO indicates this instruction takes four cycles to refill the instruction queue if the branch is taken and three cycles if the branch is not taken.
**Instruction Set Summary (Continued)**

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Operation</th>
<th>Addr. Mode</th>
<th>Machine Coding (hex)</th>
<th>Access Detail</th>
<th>S</th>
<th>X</th>
<th>H</th>
<th>I</th>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDD opr16i</td>
<td>(MM+1) ⇒ A:B Load Double Accumulator D (A:B)</td>
<td>IMM cc jj kk</td>
<td>OP</td>
<td>– – – Δ Δ 0 –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDD opr8a</td>
<td></td>
<td>DIR dc dd</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDD opr16a</td>
<td></td>
<td>EXT fc hh 11</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDD opr0,xsp</td>
<td></td>
<td>IDX ec xb</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDD opr9,xsp</td>
<td></td>
<td>IDX1 ec xb ff</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDD opr16,xsp</td>
<td></td>
<td>IDX2 ec xb ee ff</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDD [D,xsp]</td>
<td></td>
<td>[IDX2]</td>
<td>–</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDD [opr16,xsp]</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>LDS #opr16i</td>
<td>(MM+1) ⇒ SP Load Stack Pointer</td>
<td>IMM cf jj kk</td>
<td>OP</td>
<td>– – – Δ Δ 0 –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDS opr8a</td>
<td></td>
<td>DIR df dd</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDS opr15a</td>
<td></td>
<td>EXT ff hh 11</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDS opr0,xsp</td>
<td></td>
<td>IDX ef xb</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDS opr9,xsp</td>
<td></td>
<td>IDX1 ef xb ff</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDS opr16,xsp</td>
<td></td>
<td>IDX2 ef xb ee ff</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDS [D,xsp]</td>
<td></td>
<td>[IDX2]</td>
<td>–</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDS [opr16,xsp]</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>LDX #opr16i</td>
<td>(MM+1) ⇒ X Load Index Register X</td>
<td>IMM ce jj kk</td>
<td>OP</td>
<td>– – – Δ Δ 0 –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDX opr8a</td>
<td></td>
<td>DIR dd dd</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDX opr15a</td>
<td></td>
<td>EXT fe hh 11</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDX opr0,xsp</td>
<td></td>
<td>IDX ee xb</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDX opr9,xsp</td>
<td></td>
<td>IDX1 ee xb ff</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDX opr16,xsp</td>
<td></td>
<td>IDX2 ee xb ee ff</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDX [D,xsp]</td>
<td></td>
<td>[IDX2]</td>
<td>–</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDX [opr16,xsp]</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>LDY #opr16i</td>
<td>(MM+1) ⇒ Y Load Index Register Y</td>
<td>IMM cd jj kk</td>
<td>OP</td>
<td>– – – Δ Δ 0 –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDY opr8a</td>
<td></td>
<td>DIR dd dd</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDY opr15a</td>
<td></td>
<td>EXT fd hh 11</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDY opr0,xsp</td>
<td></td>
<td>IDX ed xb</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDY opr9,xsp</td>
<td></td>
<td>IDX1 ed xb ff</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDY opr16,xsp</td>
<td></td>
<td>IDX2 ed xb ee ff</td>
<td>RP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDY [D,xsp]</td>
<td></td>
<td>[IDX2]</td>
<td>–</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDY [opr16,xsp]</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>LEAS opr0,xsp</td>
<td>Effective Address ⇒ SP Load Effective Address into SP</td>
<td>IDX ib xb</td>
<td>PP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEAS opr16,xsp</td>
<td></td>
<td>IDX1 ib xb ff</td>
<td>PO</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEAS opr0,xsp</td>
<td></td>
<td>IDX2 ib xb ee ff</td>
<td>PP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEAX opr0,xsp</td>
<td>Effective Address ⇒ X Load Effective Address into X</td>
<td>IDX ia xb</td>
<td>PP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEAX opr0,xsp</td>
<td></td>
<td>IDX1 ia xb ff</td>
<td>PO</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEAX opr0,xsp</td>
<td></td>
<td>IDX2 ia xb ee ff</td>
<td>PP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEAY opr0,xsp</td>
<td>Effective Address ⇒ Y Load Effective Address into Y</td>
<td>IDX ia xb</td>
<td>PP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEAY opr0,xsp</td>
<td></td>
<td>IDX1 ia xb ff</td>
<td>PO</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEAY opr0,xsp</td>
<td></td>
<td>IDX2 ia xb ee ff</td>
<td>PP</td>
<td>– – – – – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSL opr16a</td>
<td>Logical Shift Left same function as ASL</td>
<td>EXT 78 hh 11</td>
<td>rCWr</td>
<td>– – – Δ Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSL opr0,xsp</td>
<td></td>
<td>IDX 68 hh</td>
<td>rPw</td>
<td>– – – Δ Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSL opr0,xsp</td>
<td></td>
<td>IDX1 68 hh ff</td>
<td>rCWr</td>
<td>– – – Δ Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSL opr16,xsp</td>
<td></td>
<td>IDX2 68 hh ee ff</td>
<td>rPw</td>
<td>– – – Δ Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSLD</td>
<td>Logical Shift Left D Accumulator same function as ASLD</td>
<td>IDX 48</td>
<td>0</td>
<td>– – – Δ Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSR opr15a</td>
<td>Logical Shift Right same function as ASLDRD</td>
<td>EXT 74 hh 11</td>
<td>rCWr</td>
<td>– – – 0 Δ Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSR opr0,xsp</td>
<td></td>
<td>IDX 64 hh</td>
<td>rPw</td>
<td>– – – Δ Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSR opr0,xsp</td>
<td></td>
<td>IDX1 64 hh ff</td>
<td>rCWr</td>
<td>– – – Δ Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSR opr16,xsp</td>
<td></td>
<td>IDX2 64 hh ee ff</td>
<td>rPw</td>
<td>– – – Δ Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSR [D,xsp]</td>
<td></td>
<td>[IDX2] 64 hh ee ff</td>
<td>rPw</td>
<td>– – – Δ Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSR [opr16,xsp]</td>
<td></td>
<td>64 hh ee ff</td>
<td>rPw</td>
<td>– – – Δ Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSRB</td>
<td>Logical Shift Accumulator B to Right</td>
<td>IDX 44</td>
<td>0</td>
<td>– – – Δ Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSRB</td>
<td>Logical Shift Accumulator A to Right</td>
<td>IDX 54</td>
<td>0</td>
<td>– – – Δ Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note 1. Due to internal CPU requirements, the program word fetch is performed twice to the same address during this instruction.
Instruction Set Summary (Continued)

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Operation</th>
<th>Addr. Mode</th>
<th>Machine Coding (hex)</th>
<th>Access Detail</th>
<th>S</th>
<th>X</th>
<th>H</th>
<th>I</th>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSRD</td>
<td>Logical Shift Right D Accumulator</td>
<td>INH</td>
<td>49</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td></td>
</tr>
<tr>
<td>MAXA opr10_xysp</td>
<td>MAX(IA), (MI) =&gt; A</td>
<td>IDX</td>
<td>18 18 ab</td>
<td>0x8F</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td></td>
</tr>
<tr>
<td>MAXA opr9_xysp</td>
<td>MAX of 2 Unsigned 8-Bit Values</td>
<td>IDX1</td>
<td>18 18 ab ff</td>
<td>0x8P</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td></td>
</tr>
<tr>
<td>MAXA opr16_xysp</td>
<td>N, Z, V, and C status bits reflect result of internal compare (IA) – (MI)</td>
<td>IDX2</td>
<td>18 18 ab ee ff</td>
<td>0xFFP</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td></td>
</tr>
<tr>
<td>MAXX opr16_xysp</td>
<td>MAX(IA), (MI) =&gt; M</td>
<td>IDX</td>
<td>18 18 ab ff</td>
<td>0xFFP</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td></td>
</tr>
<tr>
<td>MAXX opr16.xysp</td>
<td>MAX of 2 Unsigned 8-Bit Values</td>
<td>IDX1</td>
<td>18 18 ab ee ff</td>
<td>0xFFP</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td></td>
</tr>
<tr>
<td>MAXX opr16.xysp</td>
<td>N, Z, V, and C status bits reflect result of internal compare (IA) – (MI)</td>
<td>IDX2</td>
<td>18 18 ab ee ff</td>
<td>0xFFP</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td>Δ</td>
<td></td>
</tr>
</tbody>
</table>
| MEM | μ (grade) ⇒ M<sub>y</sub> 
(Y) + 4 ⇒ X; (Y) + 1 ⇒ Y; A unchanged | Special | 01 | SXXOw | – | – | – | ? | ? | ? | ? |
| MINA opr10_xysp | MIN(IA), (MI) ⇒ A | IDX | 18 19 ab | 0x8F | – | – | – | Δ | Δ | Δ | Δ |
| MINA opr9_xysp | MIN of 2 Unsigned 8-Bit Values | IDX1 | 18 19 ab ff | 0x8P | – | – | – | Δ | Δ | Δ | Δ |
| MINA opr16_xysp | N, Z, V, and C status bits reflect result of internal compare (IA) – (MI) | IDX2 | 18 19 ab ee ff | 0xFFP | – | – | – | Δ | Δ | Δ | Δ |
| MINM opr10_xysp | MIN(IA), (MI) ⇒ M | IDX | 18 10 ab | 0x8F | – | – | – | Δ | Δ | Δ | Δ |
| MINM opr9.xysp | MIN of 2 Unsigned 8-Bit Values | IDX1 | 18 10 ab ff | 0x8P | – | – | – | Δ | Δ | Δ | Δ |
| MINM opr16.xysp | N, Z, V, and C status bits reflect result of internal compare (IA) – (MI) | IDX2 | 18 10 ab ee ff | 0xFFP | – | – | – | Δ | Δ | Δ | Δ |
| MOV B #opr8, opr16a<sup>1</sup> | (M<sub>y</sub>) ⇒ M<sub>y</sub> 
Memory to Memory Byte-Move (8-Bit) | IMM-EXT | 18 08 li hh ll | 0xFFP | – | – | – | – | – | – | – | – | – |
| MOV B #opr8, opr16a<sup>1</sup> | | IMM-IDX | 18 08 ab li | 0xFFP | – | – | – | – | – | – | – | – | – |
| MOV B #opr8, opr16a<sup>1</sup> | | EXT-EXT | 18 0C hh ll hh ll | 0xFFP | – | – | – | – | – | – | – | – | – |
| MOV B #opr8, opr16a<sup>1</sup> | | EXT-IDX | 18 09 ab hh ll | 0xFFP | – | – | – | – | – | – | – | – | – |
| MOV B #opr8, opr16a<sup>1</sup> | | IDX-IDX | 18 0D ab hh ll | 0xFFP | – | – | – | – | – | – | – | – | – |
| MOV B #opr8, opr16a<sup>1</sup> | | IM1-MIDX | 18 0A hh ll hh ll | 0xFFP | – | – | – | – | – | – | – | – | – |
| MOV B #opr8, opr16a<sup>1</sup> | | IMM-IDX | 18 02 hh ll jj | 0xFFP | – | – | – | – | – | – | – | – | – |
| MOV B #opr8, opr16a<sup>1</sup> | | EXT-EXT | 18 04 hh ll hh ll | 0xFFP | – | – | – | – | – | – | – | – | – |
| MOV B #opr8, opr16a<sup>1</sup> | | EXT-IDX | 18 05 ab hh ll | 0xFFP | – | – | – | – | – | – | – | – | – |
| MOV B #opr8, opr16a<sup>1</sup> | | IDX-IDX | 18 02 ab hh ll | 0xFFP | – | – | – | – | – | – | – | – | – |
| MUL | (A) × (B) ⇒ A:B | NH | 12 | fFG | – | – | – | – | – | – | – | Δ |
| NEGB | No Operation | NH | 47 | D | – | – | – | – | – | – | – | – |
| NEG opr16a | 0 – (A) ⇒ A equivalent to (X) + 1 ⇒ A 
Negate Accumulator A | EXT | 70 hh ll | 0xFFP | – | – | – | Δ | Δ | Δ | Δ |
| NEG opr10.xysp | Two’s Complement Negate | IDX | 60 ab | 0xFF | – | – | – | Δ | Δ | Δ | Δ |
| NEG opr9.xysp | | IDX1 | 60 ab ff | 0xFF | – | – | – | Δ | Δ | Δ | Δ |
| NEG opr16.xysp | | IDX2 | 60 ab ee ff | 0xFF | – | – | – | Δ | Δ | Δ | Δ |
| NEG opr16.xysp | | D[IDX] | 60 ab | 0xFF | – | – | – | Δ | Δ | Δ | Δ |
| NEG opr16.xysp | | [IDX2] | 60 ab ee ff | 0xFF | – | – | – | Δ | Δ | Δ | Δ |
| NEG | 0 – (A) ⇒ A equivalent to (X) + 1 ⇒ A 
Negate Accumulator A | INH | 50 | D | – | – | – | – | – | – | – | – |
| NEGB | 0 – (B) ⇒ B equivalent to (X) + 1 ⇒ B 
Negate Accumulator B | INH | 50 | D | – | – | – | – | – | – | – | – |

Note 1. The first operand in the source code statement specifies the source for the move.
### Instruction Set Summary (Continued)

| Source Form | Operation | Addr. Mode | Machine Coding (hex) | Access Detail | S | X | H | I | N | Z | V | C |
|-------------|-----------|------------|----------------------|---------------|---|---|---|---|---|---|---|---|---|
| ORAA opr8i | (A) + (M) ⇒ A | IMM | 8A ii | P | – | – | – | Δ | Δ | 0 | – |
| ORAA opr8a | Logical OR A with Memory | 9A dd | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAA opr16a | AA bb | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAA opr16,xyisp | AA xB | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAA opr16,xyisp | AA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAA opr16,xyisp | AA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAA opr16,xyisp | AA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAA opr16,xyisp | AA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAA opr16,xyisp | AA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAA opr16,xyisp | AA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAB opr8i | (B) + (M) ⇒ B | IMM | CA ii | P | – | – | – | Δ | Δ | 0 | – |
| ORAB opr8a | Logical OR B with Memory | DA dd | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAB opr16a | FA bb | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAB opr16,xyisp | EA xB | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAB opr16,xyisp | EA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAB opr16,xyisp | EA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAB opr16,xyisp | EA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAB opr16,xyisp | EA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAB opr16,xyisp | EA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAB opr16,xyisp | EA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAB opr16,xyisp | EA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAB opr16,xyisp | EA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAB opr16,xyisp | EA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORAB opr16,xyisp | EA xB ff | rFP | – | – | – | Δ | Δ | 0 | – |
| ORCC opr8i | (CCR) + M ⇒ CCR | IMM | 14 ii | P | – | – | – | – | – | – | – | – | – |
| ORCC opr8a | Logical OR CCR with Memory | – | – | – | – | – | – | – | – | – | – | – | – |
| PSHA | (SP) – 1 ⇒ SP; (A) ⇒ M<sub>SP</sub> | INH | 36 | 0a | – | – | – | – | – | – | – | – | – |
| PSHB | Push Accumulator A onto Stack | 37 | 0a | – | – | – | – | – | – | – | – | – | – |
| PSHC | (SP) – 1 ⇒ SP; (CCR) ⇒ M<sub>SP</sub> | INH | 39 | 0a | – | – | – | – | – | – | – | – | – |
| PSHD | Push CCR onto Stack | – | – | – | – | – | – | – | – | – | – | – | – |
| PSHX | (SP) – 2 ⇒ SP; (XH,XL) ⇒ M<sub>SP</sub>:M<sub>SP+1</sub> | INH | 38 | 0a | – | – | – | – | – | – | – | – | – |
| PSY | Push Index Register X onto Stack | – | – | – | – | – | – | – | – | – | – | – | – |
| PSHA | Pull Accumulator A from Stack | 32 | ufo | – | – | – | – | – | – | – | – | – | – |
| PSHB | Pull Accumulator B from Stack | 33 | ufo | – | – | – | – | – | – | – | – | – | – |
| PSHC | (SP) + 1 ⇒ SP | INH | 38 | ufo | – | – | – | – | – | – | – | – | – |
| PSHD | Pull CCR from Stack | – | – | – | – | – | – | – | – | – | – | – | – |
| PSHX | (SP) + 1 ⇒ SP; (XH,XL) ⇒ M<sub>SP</sub>:M<sub>SP+1</sub> | INH | 39 | ufo | – | – | – | – | – | – | – | – | – |
| PSHY | Push Index Register Y onto Stack | – | – | – | – | – | – | – | – | – | – | – | – |
| PSHA | Pull Accumulator A from Stack | 32 | ufo | – | – | – | – | – | – | – | – | – | – |
| PSHB | Pull Accumulator B from Stack | 33 | ufo | – | – | – | – | – | – | – | – | – | – |
| PSHC | (SP) + 1 ⇒ SP | INH | 38 | ufo | – | – | – | – | – | – | – | – | – |
| PSHD | Pull D from Stack | – | – | – | – | – | – | – | – | – | – | – | – |
| PSHX | (SP) + 1 ⇒ SP; (XH,XL) ⇒ M<sub>SP</sub>:M<sub>SP+1</sub> | INH | 39 | ufo | – | – | – | – | – | – | – | – | – |
| PSHY | Push Index Register Y onto Stack | – | – | – | – | – | – | – | – | – | – | – | – |
| PSHA | Pull Accumulator A from Stack | 32 | ufo | – | – | – | – | – | – | – | – | – | – |
| PSHB | Pull Accumulator B from Stack | 33 | ufo | – | – | – | – | – | – | – | – | – | – |
| PSHC | (SP) + 1 ⇒ SP | INH | 38 | ufo | – | – | – | – | – | – | – | – | – |
| PSHD | Pull D from Stack | – | – | – | – | – | – | – | – | – | – | – | – |
| PSHX | (SP) + 1 ⇒ SP; (XH,XL) ⇒ M<sub>SP</sub>:M<sub>SP+1</sub> | INH | 39 | ufo | – | – | – | – | – | – | – | – | – |
| PSHY | Push Index Register Y onto Stack | – | – | – | – | – | – | – | – | – | – | – | – |
### Instruction Set Summary (Continued)

| Source Form | Operation | Addr. Mode | Machine Coding (hex) | Access Detail | S | X | H | I | N | Z | V | C |
|-------------|-----------|------------|----------------------|---------------|---|---|---|---|---|---|---|---|---|
| REV (add if interrupted) | MIN-MAX rule evaluation | Special | 18 3A | Orf(txa)0² \( ff + Orf \) | - | - | - | - | - | - | - | - | - |
| REVW (add 2 at end of ins if wts) (add if interrupted) | MIN-MAX rule evaluation | Special | 18 3B | Orf(txa)0² \( sffrft² \) \( ff + Orf(t) \) | - | - | - | - | - | - | - | - | - |
| ROL opr16a | Rotate Memory Left through Carry | EXT | 75 bh 1l | xDPw | - | - | - | - | - | - | - | - | - |
| ROL opr0, xyp | Rotate A Left through Carry | IDX | 65 xab | rFw | - | - | - | - | - | - | - | - | - |
| ROL opr9, xyp | Rotate B Left through Carry | IDX1 | 65 xab ff | rDFw | - | - | - | - | - | - | - | - | - |
| ROL [D, xyp] | Rotate A Left through Carry | IDX2 | 65 xab ee ff | rDFPrW | - | - | - | - | - | - | - | - | - |
| ROL [opr9, xyp] | Rotate B Left through Carry | INH | 45 | o | - | - | - | - | - | - | - | - | - |
| ROLB | Rotate B Right through Carry | INH | 55 | o | - | - | - | - | - | - | - | - | - |
| ROR opr16a | Rotate Memory Right through Carry | EXT | 76 bh 1l | xDPw | - | - | - | - | - | - | - | - | - |
| ROR opr0, xyp | Rotate A Right through Carry | IDX | 66 xab | rFw | - | - | - | - | - | - | - | - | - |
| ROR opr9, xyp | Rotate B Right through Carry | IDX1 | 66 xab ff | rDFw | - | - | - | - | - | - | - | - | - |
| ROR [D, xyp] | Rotate A Right through Carry | IDX2 | 66 xab ee ff | rDFPrW | - | - | - | - | - | - | - | - | - |
| ROR [opr9, xyp] | Rotate B Right through Carry | INH | 46 | o | - | - | - | - | - | - | - | - | - |
| RORB | Rotate B Right through Carry | INH | 56 | o | - | - | - | - | - | - | - | - | - |
| RTC | Return from Call | NH | 0A | uUnPPP | - | - | - | - | - | - | - | - | - |
| RTI (if interrupt pending) | Return from Interrupt | NH | 0B | uUnUnPPP | - | - | - | - | - | - | - | - | - |
| RTS | Return from Subroutine | NH | 3D | uRFPPP | - | - | - | - | - | - | - | - | - |
| SBA | Subtract B from A | INH | 18 16 | oO | - | - | - | - | - | - | - | - |

**Notes:**

1. The 3-cycle loop in parentheses is executed once for each element in the rule list. When an interrupt occurs, there is a 2-cycle exit sequence, a 4-cycle re-entry sequence, then execution resumes with a prefetch of the last antecedent or consequent being processed at the time of the interrupt.
2. The 3-cycle loop in parentheses expands to 5 cycles for separators when weighting is enabled. The loop is executed once for each element in the rule list. When an interrupt occurs, there is a 2-cycle exit sequence, a 4-cycle re-entry sequence, then execution resumes with a prefetch of the last antecedent or consequent being processed at the time of the interrupt.
### Instruction Set Summary (Continued)

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Operation</th>
<th>Addr. Mode</th>
<th>Machine Coding (hex)</th>
<th>Access Detail</th>
<th>S</th>
<th>X</th>
<th>H</th>
<th>I</th>
<th>N</th>
<th>Z</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBCA #opr8i</td>
<td>(A) – (M) – C ⇒ A</td>
<td>IMM</td>
<td>82 li * P</td>
<td>– – – Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBCA opr8i</td>
<td>Subtract with Borrow from A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBCA opr16a</td>
<td></td>
<td>DIR</td>
<td>92 dd * rFP</td>
<td>– – – Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBCA opr0,xysp</td>
<td></td>
<td>EXT</td>
<td>82 hh ll * rFP</td>
<td>– – – Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBCA oprx16,xysp</td>
<td></td>
<td>IDX</td>
<td>A2 xh * rFP</td>
<td>– – – Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBCA [D,xysp]</td>
<td></td>
<td>IDX1</td>
<td>A2 xh ff * rFP</td>
<td>– – – Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBCA [oprx16,xysp]</td>
<td></td>
<td>IDX2</td>
<td>A2 xh eq ff * frFP</td>
<td>– – – Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBCB #opr8i</td>
<td>(B) – (M) – C ⇒ B</td>
<td>IMM</td>
<td>C2 li * P</td>
<td>– – – Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBCB opr8i</td>
<td>Subtract with Borrow from B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBCB opr16a</td>
<td></td>
<td>DIR</td>
<td>D2 dd * rFP</td>
<td>– – – Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBCB oprx0,xysp</td>
<td></td>
<td>EXT</td>
<td>F2 hh ll * rFP</td>
<td>– – – Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBCB oprx16,xysp</td>
<td></td>
<td>IDX</td>
<td>E2 xh * rFP</td>
<td>– – – Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBCB [D,xysp]</td>
<td></td>
<td>IDX1</td>
<td>E2 xh ff * rFP</td>
<td>– – – Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBCB [oprx16,xysp]</td>
<td></td>
<td>IDX2</td>
<td>E2 xh eq ff * frFP</td>
<td>– – – Δ Δ Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEC</td>
<td>1 ⇒ C</td>
<td>IMM</td>
<td>14 01 * P</td>
<td>– – – – – –</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEI</td>
<td>1 ⇒ I, (inhibit interrupts)</td>
<td>IMM</td>
<td>14 10 * P</td>
<td>– – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEV</td>
<td>1 ⇒ V</td>
<td>IMM</td>
<td>14 02 * P</td>
<td>– – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEX abc,xys</td>
<td>$00/(r1) ⇒ r2 if r1, bit 7 is 0 or</td>
<td>NH</td>
<td>67 xh * P</td>
<td>– – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$FF/(r1) ⇒ r2 if r1, bit 7 is 1</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>Sign Extend 8-bit r1 to 16-bit r2</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>r1 may be A, B, or CCR</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>r2 may be D, X, Y, or SP</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>Alternate mnemonic for TFR r1, r2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAA opr8i</td>
<td>(A) ⇒ M</td>
<td>IMM</td>
<td>5a dd * Pw</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAA opr16a</td>
<td>Store Accumulator A to Memory</td>
<td></td>
<td>7a hh ll * wOP</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAA opr0,xysp</td>
<td></td>
<td>EXT</td>
<td>6a xh * Pw</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAA oprx16,xysp</td>
<td></td>
<td>IDX</td>
<td>6a xh eq * frFP</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAA [D,xysp]</td>
<td></td>
<td>IDX1</td>
<td>6a xh ff * PwO</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAA [oprx16,xysp]</td>
<td></td>
<td>IDX2</td>
<td>6a xh eq ff * frFP</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAB opr8i</td>
<td>(B) ⇒ M</td>
<td>IMM</td>
<td>6b dd * Pw</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAB opr16a</td>
<td>Store Accumulator B to Memory</td>
<td></td>
<td>7b hh ll * wOP</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAB opr0,xysp</td>
<td></td>
<td>EXT</td>
<td>6b xh * Pw</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAB oprx16,xysp</td>
<td></td>
<td>IDX</td>
<td>6b xh eq * frFP</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAB [D,xysp]</td>
<td></td>
<td>IDX1</td>
<td>6b xh ff * PwO</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAB [oprx16,xysp]</td>
<td></td>
<td>IDX2</td>
<td>6b xh eq ff * frFP</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STD opr8i</td>
<td>(A) ⇒ M, (B) ⇒ M+1</td>
<td>IMM</td>
<td>5c dd * Pw</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STD opr16a</td>
<td>Store Double Accumulator</td>
<td></td>
<td>7c hh ll * wOP</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STD opr0,xysp</td>
<td></td>
<td>EXT</td>
<td>6c xh * Pw</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STD oprx16,xysp</td>
<td></td>
<td>IDX</td>
<td>6c xh eq * frFP</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STD [D,xysp]</td>
<td></td>
<td>IDX1</td>
<td>6c xh ff * PwO</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STD [oprx16,xysp]</td>
<td></td>
<td>IDX2</td>
<td>6c xh eq ff * frFP</td>
<td>– – – Δ Δ 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STOP (entering STOP) (continuing STOP)</td>
<td>(SP) – 2 ⇒ SP;</td>
<td>NH</td>
<td>18 3E * 0000SSSx</td>
<td>– – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RTN * RTN * ⇒ M_{SP}; M_{SP+1};</td>
<td></td>
<td>F4FFFF</td>
<td>– – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(SP) – 2 ⇒ SP; (Y_{x}; Y_{y}); ⇒ M_{SP}; M_{SP+1};</td>
<td></td>
<td>P</td>
<td>– – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(SP) – 2 ⇒ SP; (B_{x}); ⇒ M_{SP}; M_{SP+1};</td>
<td></td>
<td>E</td>
<td>– – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(SP) – 1 ⇒ SP; (CCR) ⇒ M_{SP};</td>
<td></td>
<td>K</td>
<td>– – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>STOP All Clocks</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>If S control bit = 1, the STOP instruction is disabled and acts</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>like a two-cycle NOP. Registers stacked to allow quicker recovery by interrupt.</td>
<td></td>
<td></td>
<td>– – – – – –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Instruction Set Summary (Continued)

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Operation</th>
<th>Addr. Mode</th>
<th>Machine Coding (hex)</th>
<th>Access Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>STS opr8a</td>
<td>(SPH,SPL) ⇒ MM+1</td>
<td>DIR 5F dd</td>
<td>PW</td>
<td>– – – – Δ Δ 0 –</td>
</tr>
<tr>
<td>STS opr16a</td>
<td></td>
<td>EXT 7F hh 1l</td>
<td>WOP</td>
<td></td>
</tr>
<tr>
<td>STS opr0/4,xysp</td>
<td></td>
<td>IDX 6F xb</td>
<td>PW</td>
<td></td>
</tr>
<tr>
<td>STS opr16,xysp</td>
<td></td>
<td>IDX1 6F xb ff</td>
<td>PW</td>
<td></td>
</tr>
<tr>
<td>STS [D,xysp]</td>
<td></td>
<td>IDX2 6F xb ee ff</td>
<td>PW</td>
<td></td>
</tr>
<tr>
<td>STS [oprx16,xysp]</td>
<td></td>
<td>IDX2 6F xb ee ff</td>
<td>PW</td>
<td></td>
</tr>
<tr>
<td>STS opr8a</td>
<td>(XH,XL) ⇒ MM+1</td>
<td>DIR 5E dd</td>
<td>PW</td>
<td>– – – – Δ Δ 0 –</td>
</tr>
<tr>
<td>STX opr8a</td>
<td></td>
<td>EXT 7E hh 1l</td>
<td>WOP</td>
<td></td>
</tr>
<tr>
<td>STX opr16a</td>
<td></td>
<td>IDX 6E xb</td>
<td>PW</td>
<td></td>
</tr>
<tr>
<td>STX opr0/4,xysp</td>
<td></td>
<td>IDX1 6E xb ff</td>
<td>PW</td>
<td></td>
</tr>
<tr>
<td>STX opr16,xysp</td>
<td></td>
<td>IDX2 6E xb ee ff</td>
<td>PW</td>
<td></td>
</tr>
<tr>
<td>STX [D,xysp]</td>
<td></td>
<td>IDX2 6E xb ee ff</td>
<td>PW</td>
<td></td>
</tr>
<tr>
<td>STX [oprx16,xysp]</td>
<td></td>
<td>IDX2 6E xb ee ff</td>
<td>PW</td>
<td></td>
</tr>
<tr>
<td>STY opr8a</td>
<td>(YH,YL) ⇒ MM+1</td>
<td>DIR 5D dd</td>
<td>PW</td>
<td>– – – – Δ Δ 0 –</td>
</tr>
<tr>
<td>STY opr16a</td>
<td></td>
<td>EXT 7D hh 1l</td>
<td>WOP</td>
<td></td>
</tr>
<tr>
<td>STY opr0/4,xysp</td>
<td></td>
<td>IDX 6D xb</td>
<td>PW</td>
<td></td>
</tr>
<tr>
<td>STY opr16,xysp</td>
<td></td>
<td>IDX1 6D xb ff</td>
<td>PW</td>
<td></td>
</tr>
<tr>
<td>STY [D,xysp]</td>
<td></td>
<td>IDX2 6D xb ee ff</td>
<td>PW</td>
<td></td>
</tr>
<tr>
<td>STY [oprx16,xysp]</td>
<td></td>
<td>IDX2 6D xb ee ff</td>
<td>PW</td>
<td></td>
</tr>
<tr>
<td>SUBA #opr8i</td>
<td>(A) – (M) ⇒ A</td>
<td>IMM 80 ii</td>
<td>P</td>
<td>– – – – Δ Δ Δ</td>
</tr>
<tr>
<td>SUBA opr8a</td>
<td></td>
<td>DIR 90 dd</td>
<td>rfp</td>
<td></td>
</tr>
<tr>
<td>SUBA opr16a</td>
<td></td>
<td>EXT B0 hh 1l</td>
<td>cOP</td>
<td></td>
</tr>
<tr>
<td>SUBA opr0/4,xysp</td>
<td></td>
<td>IDX A0 xb</td>
<td>rfp</td>
<td></td>
</tr>
<tr>
<td>SUBA opr16,xysp</td>
<td></td>
<td>IDX1 A0 xb ff</td>
<td>rfp</td>
<td></td>
</tr>
<tr>
<td>SUBA [D,xysp]</td>
<td></td>
<td>IDX2 A0 xb ee ff</td>
<td>rfp</td>
<td></td>
</tr>
<tr>
<td>SUBA [oprx16,xysp]</td>
<td></td>
<td>IDX2 A0 xb ee ff</td>
<td>rfp</td>
<td></td>
</tr>
<tr>
<td>SUBB #opr8i</td>
<td>(B) – (M) ⇒ B</td>
<td>IMM C0 ii</td>
<td>P</td>
<td>– – – – Δ Δ Δ</td>
</tr>
<tr>
<td>SUBB opr8a</td>
<td></td>
<td>DIR D0 dd</td>
<td>rfp</td>
<td></td>
</tr>
<tr>
<td>SUBB opr16a</td>
<td></td>
<td>EXT F0 hh 1l</td>
<td>cOP</td>
<td></td>
</tr>
<tr>
<td>SUBB opr0/4,xysp</td>
<td></td>
<td>IDX E0 xb</td>
<td>rfp</td>
<td></td>
</tr>
<tr>
<td>SUBB opr16,xysp</td>
<td></td>
<td>IDX1 E0 xb ff</td>
<td>rfp</td>
<td></td>
</tr>
<tr>
<td>SUBB [D,xysp]</td>
<td></td>
<td>IDX2 E0 xb ee ff</td>
<td>rfp</td>
<td></td>
</tr>
<tr>
<td>SUBB [oprx16,xysp]</td>
<td></td>
<td>IDX2 E0 xb ee ff</td>
<td>rfp</td>
<td></td>
</tr>
<tr>
<td>SUBD #opr16i</td>
<td>(D) – (MM+1) ⇒ D</td>
<td>IMM 83 jj kk</td>
<td>OP</td>
<td>– – – – Δ Δ Δ</td>
</tr>
<tr>
<td>SUBD opr16a</td>
<td></td>
<td>DIR 93 dd</td>
<td>rfp</td>
<td></td>
</tr>
<tr>
<td>SUBD opr0/4,xysp</td>
<td></td>
<td>EXT B3 hh 1l</td>
<td>cOP</td>
<td></td>
</tr>
<tr>
<td>SUBD opr16,xysp</td>
<td></td>
<td>IDX A3 xb</td>
<td>rfp</td>
<td></td>
</tr>
<tr>
<td>SUBD [D,xysp]</td>
<td></td>
<td>IDX1 A3 xb ff</td>
<td>rfp</td>
<td></td>
</tr>
<tr>
<td>SUBD [oprx16,xysp]</td>
<td></td>
<td>IDX2 A3 xb ee ff</td>
<td>rfp</td>
<td></td>
</tr>
<tr>
<td>SWI</td>
<td></td>
<td>IMM 3F</td>
<td>V8PSSPSs*1</td>
<td>– – – 1 – – –</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAB</td>
<td>(A) ⇒ B</td>
<td>NH 18 08</td>
<td>00</td>
<td>– – – Δ Δ 0 –</td>
</tr>
<tr>
<td>TAP</td>
<td>(A) ⇒ CCR</td>
<td>NH B7 02</td>
<td>P</td>
<td>Δ Δ Δ Δ Δ Δ Δ</td>
</tr>
<tr>
<td>TBA</td>
<td>(B) ⇒ A</td>
<td>NH 18 0F</td>
<td>00</td>
<td>– – – Δ Δ 0 –</td>
</tr>
</tbody>
</table>

Note 1. The CPU also uses the SWI processing sequence for hardware interrupts and unimplemented opcode traps. A variation of the sequence (V8PSSPSs*1) is used for resets.
## Instruction Set Summary (Continued)

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Operation</th>
<th>Addr. Mode</th>
<th>Machine Coding (hex)</th>
<th>Access Detail</th>
<th>S</th>
<th>X</th>
<th>H</th>
<th>I</th>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBEQ abdxys,rel9</td>
<td>If (cntr) = 0, then Branch; else Continue to next instruction</td>
<td>REL</td>
<td>04 lb cr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test Counter and Branch if Zero (cntr = A, B, D, X, Y, or SP)</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>TBL oprz0_zysp</td>
<td>(M) + [(B) × (M+1) – (M))] ⇒ A 8-Bit Table Lookup and Interpolate</td>
<td>IDX</td>
<td>18 3d sb</td>
<td>OrsfffP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Δ</td>
<td>Δ</td>
<td>-</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>Initialize B, and index before TBL. &lt;ea&gt; points at first 8-bit table entry (M) and B is fractional part of lookup value. (no indirect addressing modes or extensions allowed)</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>TBNE abdxys,rel9</td>
<td>If (cntr) not = 0, then Branch; else Continue to next instruction</td>
<td>REL</td>
<td>04 lb cr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test Counter and Branch if Not Zero (cntr = A, B, D, X, Y, or SP)</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>TFR abcdxys,abcdxys</td>
<td>(r1) ⇒ r2 or (SP) – 0 ⇒ r2 or (1) ⇒ r2</td>
<td>INH</td>
<td>B7 sb</td>
<td>P</td>
<td></td>
<td></td>
<td>Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transfer Register to Register r1 and r2 may be A, B, CCR, D, X, Y, or SP</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>TPA</td>
<td>(CCR) ⇒ A Translates to TFR CCR, A</td>
<td>INH</td>
<td>B7 20</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRAP trapnum</td>
<td>(SP) – 2 ⇒ SP; RTN(S); RTN(SP) ⇒ M_{SP}+1;</td>
<td>INH</td>
<td>18 tn</td>
<td>O4VSPSEPSPap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(SP) – 2 ⇒ SP; (Y_{r1}) ⇒ M_{SP}+1; (SP) – 2 ⇒ SP; (X_{r1}) ⇒ M_{SP}+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>
<tr>
<td></td>
<td>(SP) – 2 ⇒ SP; (BA) ⇒ M_{SP}+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>
<tr>
<td></td>
<td>(SP) – 1 ⇒ SP; (CCR) ⇒ M_{SP}; 1 ⇒ i; (TRAP Vector) ⇒ PC</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></td>
<td>Unimplemented opcode trap</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>TST opr16a</td>
<td>(M) – 0 Test Memory for Zero or Minus</td>
<td>EXT</td>
<td>F7 hh 11</td>
<td>rCP</td>
<td></td>
<td></td>
<td>Δ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TST opr32_zysp</td>
<td></td>
<td></td>
<td>E7 sb</td>
<td>rFP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TST oprx0_zysp</td>
<td></td>
<td></td>
<td>E7 sb ff</td>
<td>rPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TST oprx16,zysp</td>
<td></td>
<td></td>
<td>E7 sb ee ff</td>
<td>fPPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TST [opxx16,zysp]</td>
<td></td>
<td></td>
<td>E7 xb</td>
<td>ffrfP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSTA</td>
<td>(A) – 0 Test A for Zero or Minus</td>
<td>INH</td>
<td>97</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSTB</td>
<td>(B) – 0 Test B for Zero or Minus</td>
<td>INH</td>
<td>D7</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSX</td>
<td>(SP) ⇒ X Translates to TFR SP, X</td>
<td>INH</td>
<td>B7 75</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSY</td>
<td>(SP) ⇒ Y Translates to TFR SP, Y</td>
<td>INH</td>
<td>B7 76</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXS</td>
<td>(X) ⇒ SP Translates to TFR X,SP</td>
<td>INH</td>
<td>B7 57</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYS</td>
<td>(Y) ⇒ SP Translates to TFR Y,SP</td>
<td>INH</td>
<td>B7 67</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WA1 [before interrupt]</td>
<td></td>
<td></td>
<td>E0</td>
<td>O3SFSaSf</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>VDPP</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></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></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- S: Stack Pointer
- X: Address
- H: Hold Bit
- I: Interrupt Enable
- N: Negative
- Z: Zero
- V: Overflow
- C: Carry
## Instruction Set Summary (Continued)

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Operation</th>
<th>Addr. Mode</th>
<th>Machine Coding (hex)</th>
<th>Access Detail</th>
<th>S</th>
<th>X</th>
<th>H</th>
<th>I</th>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAV (add if interrupt)</td>
<td>[ \sum_{i=1}^{B} S_i F_i \Rightarrow Y:D ] [ \sum_{i=1}^{B} F_i \Rightarrow X ]</td>
<td>Special</td>
<td>18 3C</td>
<td>\text{Off}(rrrrrrrr)\text{O} \text{SSS} + \text{UUrr}</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>?</td>
<td>?</td>
<td>Δ</td>
<td>?</td>
</tr>
<tr>
<td>wavr</td>
<td>see WAV</td>
<td>Special</td>
<td>3C</td>
<td>–</td>
<td>–</td>
<td>?</td>
<td>–</td>
<td>?</td>
<td>Δ</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>pseudo-instruction</td>
<td>Resume executing an interrupted WAV instruction (recover intermediate results from stack rather than initializing them to zero)</td>
<td>Special</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>XGDX</td>
<td>(D) \text{=} (X) \text{ Translates to EXG D, X}</td>
<td>INH</td>
<td>B7 C5</td>
<td>P</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>XGDY</td>
<td>(D) \text{=} (Y) \text{ Translates to EXG D, Y}</td>
<td>INH</td>
<td>B7 C6</td>
<td>P</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
### Table 1. Indexed Addressing Mode Postbyte Encoding (xb)

<table>
<thead>
<tr>
<th>Postbyte (hex)</th>
<th>Type Offset Used</th>
<th>Source Code Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0, #REG</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Key to Table 1

- **0x**: 5b const
- **1x**: pre-inc
- **2x**: post-inc
- **3x**: 5b constant
- **4x**: 16b constant
- **5x**: 16b indirect
- **6x**: 16b indirect
- **7x**: 16b indirect
- **8x**: 16b indirect
- **9x**: 16b indirect
- **Ax**: A offset
- **Bx**: B offset
- **Cx**: C offset
- **Dx**: D offset
- **Ex**: E offset
- **Fx**: F offset
- **[n,X]**: n, X
- **[n,SP]**: n, SP
- **[D,X]**: D, X
- **[n,Y]**: n, Y
- **[n,PC]**: n, PC
- **A offset**: A offset
- **B offset**: B offset
- **C offset**: C offset
- **D offset**: D offset
- **E offset**: E offset
- **F offset**: F offset
- **[A,P]**: A, P
- **[B,P]**: B, P
- **[C,P]**: C, P
- **[D,P]**: D, P
- **[E,P]**: E, P
- **[F,P]**: F, P

**Note:** NON-DISCLOSURE AGREEMENT REQUIRED
## Table 2. Indexed Addressing Mode Summary

<table>
<thead>
<tr>
<th>Postbyte Code (xb)</th>
<th>Operand Syntax</th>
<th>Comments</th>
</tr>
</thead>
</table>
| rr0nnnnn           | ,r             | 5-bit constant offset  
|                    | n,r            | n = -16 to +15  
|                    | -n,r           | r can specify X, Y, SP, or PC  |
| 111rr0zs           | n,r            | Constant offset (9- or 16-bit signed)  
|                    | –n,r           | z- 0 = 9-bit with sign in LSB of postbyte (s)  
|                    |                | 1 = 16-bit  
|                    |                | if z = s = 1, 16-bit offset indexed-indirect (see below)  
|                    |                | r can specify X, Y, SP, or PC  |
| rr1pnnnn           | n,–r           | Auto pre-decrement /increment or Auto post-decrement/increment;  
|                    | n,+r           | p = pre-(0) or post-(1), n = –8 to –1, +1 to +8  
|                    | n,r–           | r can specify X, Y, or SP (PC not a valid choice)  
|                    | n,r+           |  |
| 111rr1aa           | A,r            | Accumulator offset (unsigned 8-bit or 16-bit)  
|                    | B,r            | 00 = A  
|                    | D,r            | 01 = B  
|                    |                | 10 = D (16-bit)  
|                    |                | 11 = see accumulator D offset indexed-indirect  
|                    |                | r can specify X, Y, SP, or PC  |
| 111rr011           | [n,r]          | 16-bit offset indexed-indirect  
|                    |                | r can specify X, Y, SP, or PC  |
| 111rr111           | [D,r]          | Accumulator D offset indexed-indirect  
|                    |                | r can specify X, Y, SP, or PC  |
## Table 3. Transfer and Exchange Postbyte Encoding

### TRANSFERS

<table>
<thead>
<tr>
<th>↓ LS</th>
<th>MS⇒</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>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A ⇒ A</td>
<td>B ⇒ A</td>
<td>CCR ⇒ A</td>
<td>TMP3_L ⇒ A</td>
<td>B ⇒ A</td>
<td>X_L ⇒ A</td>
<td>Y_L ⇒ A</td>
<td>SP_L ⇒ A</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>A ⇒ B</td>
<td>B ⇒ B</td>
<td>CCR ⇒ B</td>
<td>TMP3_L ⇒ B</td>
<td>B ⇒ B</td>
<td>X_L ⇒ B</td>
<td>Y_L ⇒ B</td>
<td>SP_L ⇒ B</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A ⇒ CCR</td>
<td>B ⇒ CCR</td>
<td>CCR ⇒ CCR</td>
<td>TMP3_L ⇒ CCR</td>
<td>B ⇒ CCR</td>
<td>X_L ⇒ CCR</td>
<td>Y_L ⇒ CCR</td>
<td>SP_L ⇒ CCR</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>sex:A ⇒ TMP2</td>
<td>sex:B ⇒ TMP2</td>
<td>sex:CCR ⇒ TMP2</td>
<td>TMP3 ⇒ TMP2</td>
<td>D ⇒ TMP2</td>
<td>X ⇒ TMP2</td>
<td>Y ⇒ TMP2</td>
<td>SP ⇒ TMP2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>sex:A ⇒ D</td>
<td>sex:B ⇒ D</td>
<td>sex:CCR ⇒ D</td>
<td>SEX B,D</td>
<td>SEX CCR,D</td>
<td>TMP3 ⇒ D</td>
<td>D ⇒ D</td>
<td>X ⇒ D</td>
<td>Y ⇒ D</td>
</tr>
<tr>
<td>5</td>
<td>sex:A ⇒ X</td>
<td>sex:B ⇒ X</td>
<td>sex:CCR ⇒ X</td>
<td>SEX A,X</td>
<td>SEX CCR,X</td>
<td>TMP3 ⇒ X</td>
<td>D ⇒ X</td>
<td>X ⇒ X</td>
<td>Y ⇒ X</td>
</tr>
<tr>
<td>6</td>
<td>sex:A ⇒ Y</td>
<td>sex:B ⇒ Y</td>
<td>sex:CCR ⇒ Y</td>
<td>SEX A,Y</td>
<td>SEX CCR,Y</td>
<td>TMP3 ⇒ Y</td>
<td>D ⇒ Y</td>
<td>X ⇒ Y</td>
<td>Y ⇒ Y</td>
</tr>
<tr>
<td>7</td>
<td>sex:A ⇒ SP</td>
<td>sex:B ⇒ SP</td>
<td>sex:CCR ⇒ SP</td>
<td>SEX A,SP</td>
<td>SEX CCR,SP</td>
<td>TMP3 ⇒ SP</td>
<td>D ⇒ SP</td>
<td>X ⇒ SP</td>
<td>Y ⇒ SP</td>
</tr>
</tbody>
</table>

### EXCHANGES

<table>
<thead>
<tr>
<th>↓ LS</th>
<th>MS⇒</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A ⇒ A</td>
<td>B ⇒ A</td>
<td>CCR ⇒ A</td>
<td>TMP3_L ⇒ A</td>
<td>B ⇒ A</td>
<td>X_L ⇒ A</td>
<td>$00:A ⇒ X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>A ⇒ B</td>
<td>B ⇒ B</td>
<td>CCR ⇒ B</td>
<td>TMP3_L ⇒ B</td>
<td>B ⇒ B</td>
<td>X_L ⇒ B</td>
<td>$00:A ⇒ $FF:B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>A ⇒ CCR</td>
<td>B ⇒ CCR</td>
<td>CCR ⇒ CCR</td>
<td>TMP3_L ⇒ CCR</td>
<td>B ⇒ CCR</td>
<td>X_L ⇒ CCR</td>
<td>$FF:CPR ⇒ X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$00:A ⇒ TMP2</td>
<td>$00:B ⇒ TMP2</td>
<td>$00:CCR ⇒ TMP2</td>
<td>$00:CPR ⇒ TMP2</td>
<td>$00:CCR ⇒ CPR</td>
<td>TMP3 ⇒ TMP2</td>
<td>D ⇒ TMP2</td>
<td>X ⇒ TMP2</td>
<td>Y ⇒ TMP2</td>
</tr>
<tr>
<td>4</td>
<td>$00:A ⇒ D</td>
<td>$00:B ⇒ D</td>
<td>$00:CCR ⇒ D</td>
<td>B ⇒ CPR</td>
<td>B ⇒ CPR</td>
<td>TMP3 ⇒ D</td>
<td>D ⇒ D</td>
<td>X ⇒ D</td>
<td>Y ⇒ D</td>
</tr>
<tr>
<td>5</td>
<td>$00:A ⇒ X</td>
<td>$00:B ⇒ X</td>
<td>$00:CCR ⇒ X</td>
<td>X_L ⇒ B</td>
<td>X_L ⇒ B</td>
<td>TMP3 ⇒ X</td>
<td>D ⇒ X</td>
<td>X ⇒ X</td>
<td>Y ⇒ X</td>
</tr>
<tr>
<td>6</td>
<td>$00:A ⇒ Y</td>
<td>$00:B ⇒ Y</td>
<td>$00:CCR ⇒ Y</td>
<td>Y_L ⇒ B</td>
<td>Y_L ⇒ B</td>
<td>TMP3 ⇒ Y</td>
<td>D ⇒ Y</td>
<td>X ⇒ Y</td>
<td>Y ⇒ Y</td>
</tr>
<tr>
<td>7</td>
<td>$00:A ⇒ SP</td>
<td>$00:B ⇒ SP</td>
<td>$00:CCR ⇒ SP</td>
<td>SP_L ⇒ A</td>
<td>SP_L ⇒ A</td>
<td>TMP3 ⇒ SP</td>
<td>D ⇒ SP</td>
<td>X ⇒ SP</td>
<td>Y ⇒ SP</td>
</tr>
</tbody>
</table>

TMP2 and TMP3 registers are for factory use only.
Table 4. Loop Primitive Postbyte Encoding (lb)

<table>
<thead>
<tr>
<th>Postbyte (hex)</th>
<th>Counter Used</th>
<th>Sign of 9-bit Relative Branch Offset</th>
<th>Lower Eight Bits Are an Extension Byte Following Postbyte</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 A</td>
<td>DBEQ (+)</td>
<td>Z = (N ⊕ V) = 0</td>
<td></td>
</tr>
<tr>
<td>01 B</td>
<td>DBEQ (+)</td>
<td>Z = 1</td>
<td></td>
</tr>
<tr>
<td>02 F</td>
<td>DBEQ (+)</td>
<td>Z + (N ⊕ V) = 1</td>
<td></td>
</tr>
<tr>
<td>03 G</td>
<td>DBEQ (+)</td>
<td>C = 0</td>
<td></td>
</tr>
<tr>
<td>04 H</td>
<td>DBEQ (+)</td>
<td>C + Z = 0</td>
<td></td>
</tr>
<tr>
<td>05 I</td>
<td>DBEQ (+)</td>
<td>C = 1</td>
<td></td>
</tr>
<tr>
<td>06 J</td>
<td>DBEQ (+)</td>
<td>No Carry</td>
<td></td>
</tr>
<tr>
<td>07 K</td>
<td>DBEQ (+)</td>
<td>BCC</td>
<td></td>
</tr>
<tr>
<td>08 L</td>
<td>DBEQ (+)</td>
<td>BHI</td>
<td></td>
</tr>
<tr>
<td>09 M</td>
<td>DBEQ (+)</td>
<td>BLS</td>
<td></td>
</tr>
<tr>
<td>0A N</td>
<td>DBEQ (+)</td>
<td>BLO/BCS</td>
<td></td>
</tr>
<tr>
<td>0B O</td>
<td>DBEQ (+)</td>
<td>BRA</td>
<td></td>
</tr>
<tr>
<td>0C P</td>
<td>DBEQ (+)</td>
<td>BRN</td>
<td></td>
</tr>
<tr>
<td>0D Q</td>
<td>DBEQ (+)</td>
<td>Always</td>
<td></td>
</tr>
<tr>
<td>0E R</td>
<td>DBEQ (+)</td>
<td>Never</td>
<td></td>
</tr>
</tbody>
</table>

Key to Table 4

- postbyte (hex)
- counter used
- branch condition
- sign of 9-bit relative branch offset
- (lower eight bits are an extension byte following postbyte)

Table 5. Branch/Complementary Branch

<table>
<thead>
<tr>
<th>Test</th>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Boolean</th>
<th>Test</th>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>r&gt;m</td>
<td>BGT</td>
<td>2E</td>
<td>Z + (N ⊕ V) = 0</td>
<td>r&lt;m</td>
<td>BLE</td>
<td>2F</td>
<td>Signed</td>
</tr>
<tr>
<td>r&lt;m</td>
<td>BGE</td>
<td>2C</td>
<td>N ⊕ V = 0</td>
<td>r&gt;m</td>
<td>BLT</td>
<td>2D</td>
<td>Signed</td>
</tr>
<tr>
<td>r&gt;m</td>
<td>BEQ</td>
<td>27</td>
<td>Z = 1</td>
<td>r&lt;m</td>
<td>BNE</td>
<td>26</td>
<td>Signed</td>
</tr>
<tr>
<td>r&lt;m</td>
<td>BLE</td>
<td>2F</td>
<td>Z + (N ⊕ V) = 1</td>
<td>r&gt;m</td>
<td>BGT</td>
<td>2E</td>
<td>Signed</td>
</tr>
<tr>
<td>r&gt;m</td>
<td>BLT</td>
<td>2D</td>
<td>N ⊕ V = 1</td>
<td>r&lt;m</td>
<td>BGE</td>
<td>2C</td>
<td>Signed</td>
</tr>
<tr>
<td>r&gt;m</td>
<td>BHI</td>
<td>22</td>
<td>C + Z = 0</td>
<td>r&lt;m</td>
<td>BLS</td>
<td>23</td>
<td>Signed</td>
</tr>
<tr>
<td>r&gt;m</td>
<td>BHS/BCC</td>
<td>24</td>
<td>C = 0</td>
<td>r&lt;m</td>
<td>BLO/BCS</td>
<td>25</td>
<td>Signed</td>
</tr>
<tr>
<td>r&lt;m</td>
<td>BLO/BCS</td>
<td>25</td>
<td>C + Z = 1</td>
<td>r&gt;m</td>
<td>BHI</td>
<td>22</td>
<td>Signed</td>
</tr>
<tr>
<td>r&lt;m</td>
<td>BLS</td>
<td>23</td>
<td>C = 1</td>
<td>r&gt;m</td>
<td>BHS/BCC</td>
<td>24</td>
<td>Signed</td>
</tr>
<tr>
<td>r&lt;m</td>
<td>BCS</td>
<td>25</td>
<td>No Carry</td>
<td>BCC</td>
<td>24</td>
<td>Simple</td>
<td></td>
</tr>
<tr>
<td>Carry</td>
<td>BCS</td>
<td>25</td>
<td>C = 1</td>
<td>BCC</td>
<td>24</td>
<td>Simple</td>
<td></td>
</tr>
<tr>
<td>Negative</td>
<td>BMI</td>
<td>2B</td>
<td>N = 1</td>
<td>BPL</td>
<td>2A</td>
<td>Simple</td>
<td></td>
</tr>
<tr>
<td>Overflow</td>
<td>BVS</td>
<td>29</td>
<td>V = 1</td>
<td>BVC</td>
<td>28</td>
<td>Simple</td>
<td></td>
</tr>
<tr>
<td>r=0</td>
<td>BEQ</td>
<td>27</td>
<td>Z = 1</td>
<td>BNE</td>
<td>26</td>
<td>Simple</td>
<td></td>
</tr>
<tr>
<td>Always</td>
<td>BRA</td>
<td>20</td>
<td>—</td>
<td>BRN</td>
<td>21</td>
<td>Unconditional</td>
<td></td>
</tr>
</tbody>
</table>

For 16-bit offset long branches precede opcode with a $18 page prebyte.
Memory Expansion

Some M68HC12 derivatives support >4 megabytes of program memory.

Memory precedence
— Highest —
On-chip registers (usually $0000 or $1000)
BDM ROM (only when BDM active)
On-chip RAM
On-chip EEPROM
On-chip program memory (FLASH or ROM)
Expansion windows (on MCUs with expanded memory)
Other external memory
— Lowest —

CPU sees 64 Kbytes of address space (CPU_ADDR [15:0])
PPAGE 8-bit register to select 1 of 256 — 16 Kbyte program pages
DPAGE 8-bit register to select 1 of 256 — 4 Kbyte data pages
EPAGE 8-bit register to select 1 of 256 — 1 Kbyte extra pages

Extended address is 22 bits (EXT_ADDR [21:0])
Program expansion window works with CALL and RTC instructions to simplify program access to extended memory space. Data and extra expansion windows (when present) use traditional banked expansion memory techniques.

Program window
If CPU_ADDR [15:0] = $8000–BFFF and PWEN = 1
Then EXT_ADDR [21:0] = PPAGE [7:0]:CPU_ADDR [13:0]
Program window works with CALL/RTC to automate bank switching. 256 pages (banks) of 16 Kbytes each = 4 M.

Data window
If CPU_ADDR [15:0] = $7000–7FFF and DWEN = 1
Then EXT_ADDR [21:0] = 1:1:DPAGE [7:0]:CPU_ADDR [11:0]
User program controls DPAGE value
Extra window

If \( \text{CPU_ADDR}[15:0] = 0000–03FF \) and \( \text{EWDIR} = 1 \)
and \( \text{EWEN} = 1 \)

or \( \text{CPU_ADDR}[15:0] = 0400–07FF \) and \( \text{EWDIR} = 0 \)
and \( \text{EWEN} = 1 \)

Then \( \text{EXT_ADDR}[21:0] = 1:1:1:1:1:1:\text{EPAGE}[7:0]:\text{CPU_ADDR}[9:0] \)

User program controls \( \text{EPAGE} \) value

**CPU address not in any enabled window**

256 pages of 1 Kbyte each viewed through E-window

256 pages of 4 Kbytes each viewed through D-window

256 pages of 16 Kbytes each viewed through P-window

CPU Address

00 0000
04 0000
07 FF
1 FFF
2 000
3 E FFFF
3 F 0000
3 F 7FFF
3 F 8000
3 F BFFF
3 F C000
D FFFF
E 0000
F FFFF
3 F FFFF

E window
(EWDIR = 1)

E window
(EWDIR = 0)

PPAGE ($FC) 252

PPAGE ($FD) 253

PPAGE ($FE) 254

PPAGE ($FF) 255

DPAGE ($F0) 224

DPAGE ($F1) 225

DPAGE ($F2) 226

DPAGE ($F3) 227

DPAGE ($F4) 228

DPAGE ($F5) 229

DPAGE ($F6) 230

DPAGE ($F7) 231

DPAGE ($F8) 232

DPAGE ($F9) 233

DPAGE ($FA) 234

DPAGE ($FB) 235

DPAGE ($FC) 236

DPAGE ($FD) 237

DPAGE ($FE) 238

DPAGE ($FF) 239

DPAGE ($00) 240

DPAGE ($01) 241

DPAGE ($02) 242

DPAGE ($03) 243

DPAGE ($04) 244

DPAGE ($05) 245

DPAGE ($06) 246

DPAGE ($07) 247

DPAGE ($08) 248

DPAGE ($09) 249

DPAGE ($0A) 250

DPAGE ($0B) 251

DPAGE ($0C) 252

DPAGE ($0D) 253

DPAGE ($0E) 254

DPAGE ($0F) 255

CPU12 Reference Guide
Table 6. CPU12 Opcode Map (Sheet 1 of 2)

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>BGN</td>
</tr>
<tr>
<td>01</td>
<td>MEM</td>
</tr>
<tr>
<td>02</td>
<td>INY</td>
</tr>
<tr>
<td>03</td>
<td>DEY</td>
</tr>
<tr>
<td>04</td>
<td>loop†</td>
</tr>
<tr>
<td>05</td>
<td>JMP</td>
</tr>
<tr>
<td>06</td>
<td>JMP</td>
</tr>
<tr>
<td>07</td>
<td>BSR</td>
</tr>
<tr>
<td>08</td>
<td>INX</td>
</tr>
<tr>
<td>09</td>
<td>DEX</td>
</tr>
<tr>
<td>0A</td>
<td>RTC</td>
</tr>
<tr>
<td>0B</td>
<td>RTI</td>
</tr>
<tr>
<td>0C</td>
<td>BSET</td>
</tr>
<tr>
<td>0D</td>
<td>BCLR</td>
</tr>
<tr>
<td>0E</td>
<td>BRSET</td>
</tr>
<tr>
<td>0F</td>
<td>BROCLR</td>
</tr>
</tbody>
</table>

Note: The table continues with similar entries.
Table 6. CPU12 Opcode Map (Sheet 2 of 2)

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOVW</td>
<td>Immediate</td>
</tr>
<tr>
<td>IM-ID</td>
<td>Immediate</td>
</tr>
<tr>
<td>EH</td>
<td>Immediate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDIV</td>
<td>Immediate</td>
</tr>
<tr>
<td>LBRAl</td>
<td>Immediate</td>
</tr>
<tr>
<td>TRAP</td>
<td>Immediate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBEQ</td>
<td>Immediate</td>
</tr>
<tr>
<td>TBNE</td>
<td>Immediate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA</td>
<td>Immediate</td>
</tr>
<tr>
<td>EMINM</td>
<td>Immediate</td>
</tr>
</tbody>
</table>

* Refer to instruction summary for more information.
‡ The opcode $04$ corresponds to one of the loop primitive instructions DBEQ, DBNE, IBEQ, IBNE, TBEQ, or TBNE.
Table 7. Hexadecimal to ASCII Conversion

<table>
<thead>
<tr>
<th>Hex</th>
<th>ASCII</th>
<th>Hex</th>
<th>ASCII</th>
<th>Hex</th>
<th>ASCII</th>
<th>Hex</th>
<th>ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td>$00</td>
<td>NUL</td>
<td>$20</td>
<td>SP</td>
<td>$40</td>
<td>@</td>
<td>$60</td>
<td>grave</td>
</tr>
<tr>
<td>$01</td>
<td>SOH</td>
<td>$21</td>
<td>!</td>
<td>$41</td>
<td>A</td>
<td>$61</td>
<td>a</td>
</tr>
<tr>
<td>$02</td>
<td>STX</td>
<td>$22</td>
<td>“quote</td>
<td>$42</td>
<td>B</td>
<td>$62</td>
<td>b</td>
</tr>
<tr>
<td>$03</td>
<td>ETX</td>
<td>$23</td>
<td>#</td>
<td>$43</td>
<td>C</td>
<td>$63</td>
<td>c</td>
</tr>
<tr>
<td>$04</td>
<td>EOT</td>
<td>$24</td>
<td>$</td>
<td>$44</td>
<td>D</td>
<td>$64</td>
<td>d</td>
</tr>
<tr>
<td>$05</td>
<td>ENQ</td>
<td>$25</td>
<td>%</td>
<td>$45</td>
<td>E</td>
<td>$65</td>
<td>e</td>
</tr>
<tr>
<td>$06</td>
<td>ACK</td>
<td>$26</td>
<td>&amp;</td>
<td>$46</td>
<td>F</td>
<td>$66</td>
<td>f</td>
</tr>
<tr>
<td>$07</td>
<td>BEL</td>
<td>$27</td>
<td>’ ‘ape</td>
<td>$47</td>
<td>G</td>
<td>$67</td>
<td>g</td>
</tr>
<tr>
<td>$08</td>
<td>BS</td>
<td>$28</td>
<td>(</td>
<td>$48</td>
<td>H</td>
<td>$68</td>
<td>h</td>
</tr>
<tr>
<td>$09</td>
<td>HT</td>
<td>$29</td>
<td>)</td>
<td>$49</td>
<td>I</td>
<td>$69</td>
<td>i</td>
</tr>
<tr>
<td>$0A</td>
<td>LF</td>
<td>$2A</td>
<td>*</td>
<td>$4A</td>
<td>J</td>
<td>$6A</td>
<td>j</td>
</tr>
<tr>
<td>$0B</td>
<td>VT</td>
<td>$2B</td>
<td>+</td>
<td>$4B</td>
<td>K</td>
<td>$6B</td>
<td>k</td>
</tr>
<tr>
<td>$0C</td>
<td>FF</td>
<td>$2C</td>
<td>,</td>
<td>$4C</td>
<td>L</td>
<td>$6C</td>
<td>l</td>
</tr>
<tr>
<td>$0D</td>
<td>CR</td>
<td>$2D</td>
<td>- dash</td>
<td>$4D</td>
<td>M</td>
<td>$6D</td>
<td>m</td>
</tr>
<tr>
<td>$0E</td>
<td>SO</td>
<td>$2E</td>
<td>. period</td>
<td>$4E</td>
<td>N</td>
<td>$6E</td>
<td>n</td>
</tr>
<tr>
<td>$0F</td>
<td>SI</td>
<td>$2F</td>
<td>/</td>
<td>$4F</td>
<td>O</td>
<td>$6F</td>
<td>o</td>
</tr>
<tr>
<td>$10</td>
<td>DLE</td>
<td>$30</td>
<td>0</td>
<td>$50</td>
<td>P</td>
<td>$70</td>
<td>p</td>
</tr>
<tr>
<td>$11</td>
<td>DC1</td>
<td>$31</td>
<td>1</td>
<td>$51</td>
<td>Q</td>
<td>$71</td>
<td>q</td>
</tr>
<tr>
<td>$12</td>
<td>DC2</td>
<td>$32</td>
<td>2</td>
<td>$52</td>
<td>R</td>
<td>$72</td>
<td>r</td>
</tr>
<tr>
<td>$13</td>
<td>DC3</td>
<td>$33</td>
<td>3</td>
<td>$53</td>
<td>S</td>
<td>$73</td>
<td>s</td>
</tr>
<tr>
<td>$14</td>
<td>DC4</td>
<td>$34</td>
<td>4</td>
<td>$54</td>
<td>T</td>
<td>$74</td>
<td>t</td>
</tr>
<tr>
<td>$15</td>
<td>NAK</td>
<td>$35</td>
<td>5</td>
<td>$55</td>
<td>U</td>
<td>$75</td>
<td>u</td>
</tr>
<tr>
<td>$16</td>
<td>SYN</td>
<td>$36</td>
<td>6</td>
<td>$56</td>
<td>V</td>
<td>$76</td>
<td>v</td>
</tr>
<tr>
<td>$17</td>
<td>ETB</td>
<td>$37</td>
<td>7</td>
<td>$57</td>
<td>W</td>
<td>$77</td>
<td>w</td>
</tr>
<tr>
<td>$18</td>
<td>CAN</td>
<td>$38</td>
<td>8</td>
<td>$58</td>
<td>X</td>
<td>$78</td>
<td>x</td>
</tr>
<tr>
<td>$19</td>
<td>EM</td>
<td>$39</td>
<td>9</td>
<td>$59</td>
<td>Y</td>
<td>$79</td>
<td>y</td>
</tr>
<tr>
<td>$1A</td>
<td>SUB</td>
<td>$3A</td>
<td>:</td>
<td>$5A</td>
<td>Z</td>
<td>$7A</td>
<td>z</td>
</tr>
<tr>
<td>$1B</td>
<td>ESCAPE</td>
<td>$3B</td>
<td>;</td>
<td>$5B</td>
<td>[</td>
<td>$7B</td>
<td>{</td>
</tr>
<tr>
<td>$1C</td>
<td>FS</td>
<td>$3C</td>
<td>&lt;</td>
<td>$5C</td>
<td>\</td>
<td>$7C</td>
<td></td>
</tr>
<tr>
<td>$1D</td>
<td>GS</td>
<td>$3D</td>
<td>=</td>
<td>$5D</td>
<td>]</td>
<td>$7D</td>
<td>}</td>
</tr>
<tr>
<td>$1E</td>
<td>RS</td>
<td>$3E</td>
<td>&gt;</td>
<td>$5E</td>
<td>^</td>
<td>$7E</td>
<td>~</td>
</tr>
<tr>
<td>$1F</td>
<td>US</td>
<td>$3F</td>
<td>?</td>
<td>$5F</td>
<td>_ under</td>
<td>$7F</td>
<td>DEL delete</td>
</tr>
</tbody>
</table>
Hexadecimal to Decimal Conversion

To convert a hexadecimal number (up to four hexadecimal digits) to decimal, look up the decimal equivalent of each hexadecimal digit in Table 8. The decimal equivalent of the original hexadecimal number is the sum of the weights found in the table for all hexadecimal digits.

Table 8. Hexadecimal to/from Decimal Conversion

<table>
<thead>
<tr>
<th>15 Bit</th>
<th>8</th>
<th>7 Bit</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>12</td>
<td>11</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4th Hex Digit</th>
<th>3rd Hex Digit</th>
<th>2nd Hex Digit</th>
<th>1st Hex Digit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex</td>
<td>Decimal</td>
<td>Hex</td>
<td>Decimal</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4,096</td>
<td>1</td>
<td>256</td>
</tr>
<tr>
<td>2</td>
<td>8,192</td>
<td>2</td>
<td>512</td>
</tr>
<tr>
<td>3</td>
<td>12,288</td>
<td>3</td>
<td>768</td>
</tr>
<tr>
<td>4</td>
<td>16,384</td>
<td>4</td>
<td>1,024</td>
</tr>
<tr>
<td>5</td>
<td>20,480</td>
<td>5</td>
<td>1,280</td>
</tr>
<tr>
<td>6</td>
<td>24,576</td>
<td>6</td>
<td>1,536</td>
</tr>
<tr>
<td>7</td>
<td>28,672</td>
<td>7</td>
<td>1,792</td>
</tr>
<tr>
<td>8</td>
<td>32,768</td>
<td>8</td>
<td>2,048</td>
</tr>
<tr>
<td>9</td>
<td>36,864</td>
<td>9</td>
<td>2,304</td>
</tr>
<tr>
<td>A</td>
<td>40,960</td>
<td>A</td>
<td>2,560</td>
</tr>
<tr>
<td>B</td>
<td>45,056</td>
<td>B</td>
<td>2,816</td>
</tr>
<tr>
<td>C</td>
<td>49,152</td>
<td>C</td>
<td>3,072</td>
</tr>
<tr>
<td>D</td>
<td>53,248</td>
<td>D</td>
<td>3,328</td>
</tr>
<tr>
<td>E</td>
<td>57,344</td>
<td>E</td>
<td>3,484</td>
</tr>
<tr>
<td>F</td>
<td>61,440</td>
<td>F</td>
<td>3,840</td>
</tr>
</tbody>
</table>

Decimal to Hexadecimal Conversion

To convert a decimal number (up to 65,535₁₀) to hexadecimal, find the largest decimal number in Table 8 that is less than or equal to the number you are converting. The corresponding hexadecimal digit is the most significant hexadecimal digit of the result. Subtract the decimal number found from the original decimal number to get the remaining decimal value. Repeat the procedure using the remaining decimal value for each subsequent hexadecimal digit.
Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts.

Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution, P.O. Box 5405, Denver, Colorado 80217, 1-800-441-2447 or 1-303-675-2140. Customer Focus Center, 1-800-521-6274

JAPAN: Nippon Motorola Ltd.: SPD, Strategic Planning Office, 141, 4-32-1 Nishi-Gotanda, Shinigawa-Ku, Tokyo, Japan. 03-5487-8488

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd., 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

Mfax™, Motorola Fax Back System: RMFAX0@email.sps.mot.com; http://sps.motorola.com/mfax/;

TOUCHTONE, 1-602-244-6609; US and Canada ONLY, 1-800-774-1848

HOME PAGE: http://motorola.com/sps/

Mfax is a trademark of Motorola, Inc.

© Motorola, Inc., 1998

CPU12RG/D