**Instruction Glossary**

### LDAA

**Operation:** \[(M) \rightarrow A\]

**Description:** Loads the content of memory location \(M\) into accumulator \(A\). The condition codes are set according to the data.

**CCR Details:**

- N: Set if MSB of result is set; cleared otherwise
- Z: Set if result is \(0\); cleared otherwise
- V: C; cleared

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Address Mode</th>
<th>Object Code</th>
<th>Access Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAA [oprx16,xysp]</td>
<td>[IDX2]</td>
<td>A4 [ \text{oprx16}, xysp ]</td>
<td>AXx[ff]</td>
</tr>
</tbody>
</table>

**Glossary**

**LDAA**

Load Accumulator A

**Operation:** \[(M) \rightarrow A\]

**Description:** Loads the content of memory location \(M\) into accumulator \(A\). The content of \(B\) is unchanged.

**CCR Details:**

- N: Set if MSB of result is set; cleared otherwise
- Z: Set if result is \(0\); cleared otherwise
- V: C; cleared

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Address Mode</th>
<th>Object Code</th>
<th>Access Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAA [oprx16,xysp]</td>
<td>[IDX2]</td>
<td>A4 [ \text{oprx16}, xysp ]</td>
<td>AXx[ff]</td>
</tr>
</tbody>
</table>

### STAB

**Operation:** \[(B) \rightarrow M\]

**Description:** Stores the content of accumulator \(B\) in memory location \(M\). The content of \(B\) is unchanged.

**CCR Details:**

- N: Set if MSB of result is set; cleared otherwise
- Z: Set if result is \(0\); cleared otherwise
- V: C; cleared

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Address Mode</th>
<th>Object Code</th>
<th>Access Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAB [oprx16,xysp]</td>
<td>[IDX2]</td>
<td>A4 [ \text{oprx16}, xysp ]</td>
<td>AXx[ff]</td>
</tr>
</tbody>
</table>

### ADDA

**Operation:** \[(A) \rightarrow M\]

**Description:** Adds the content of accumulator \(B\) to accumulator \(A\) and places the result in \(A\). This instruction affects the \(H\) status bit, so it is suitable for use in BCD arithmetic operations. See DAA instruction for additional information.

**CCR Details:**

- N: Set if MSB of result is set; cleared otherwise
- Z: Set if result is \(0\); cleared otherwise
- V: C; cleared

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Address Mode</th>
<th>Object Code</th>
<th>Access Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDA [oprx16,xysp]</td>
<td>[IDX2]</td>
<td>A4 [ \text{oprx16}, xysp ]</td>
<td>AXx[ff]</td>
</tr>
</tbody>
</table>
### INCA - Increment A

**Operation:**

\[(A) + 01 \Rightarrow A\]

**Description:**
Add one to the content of accumulator A.

The N, Z, and V status bits are set or cleared according to the results of the operation. The C status bit is not affected by the operation, thus allowing the INC instruction to be used as a loop counter in multiple-precision computations.

When operating on unsigned values, only BEQ, BNE, LBEQ, and LBNE branches can be expected to perform consistently. When operating on two's complement values, all signed branches are available.

**CCR Details:**

<table>
<thead>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Δ</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Δ</td>
<td></td>
</tr>
</tbody>
</table>

- **N:** Set if MSB of result is set; cleared otherwise
- **Z:** Set if result is $00$; cleared otherwise
- **V:** Set if there is a two's complement overflow as a result of the operation; cleared otherwise. Two's complement overflow occurs if and only if \((A)\) was $7F$ before the operation.

### CLRB - Clear B

**Operation:**

\[0 \Rightarrow B\]

**Description:**
All bits in accumulator B are cleared to 0.

**CCR Details:**

<table>
<thead>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **N:** 0; cleared
- **Z:** 1; set
- **V:** 0; cleared
- **C:** 0; cleared

### DECB - Decrement B

**Operation:**

\[(B) - 01 \Rightarrow B\]

**Description:**
Subtract one from the content of accumulator B.

The N, Z, and V status bits are set or cleared according to the results of the operation. The C status bit is not affected by the operation, thus allowing the DEC instruction to be used as a loop counter in multiple-precision computations.

**CCR Details:**

<table>
<thead>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Δ</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Δ</td>
<td></td>
</tr>
</tbody>
</table>

- **N:** Set if MSB of result is set; cleared otherwise
- **Z:** Set if result is $00$; cleared otherwise
- **V:** Set if there was a two's complement overflow as a result of the operation; cleared otherwise. Two's complement overflow occurs if and only if \((B)\) was $80$ before the operation.

### LSRA - Logical Shift Right A

**Operation:**

\[\begin{array}{c}
\text{C} \\
\text{b0} \\
\text{b7} \\
\text{LSRA} \\
\end{array}\]

**Description:**
Shifts all bits of accumulator A one place to the right. Bit 7 is loaded with 0. The C status bit is loaded from the least significant bit of A.

**CCR Details:**

<table>
<thead>
<tr>
<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **N:** 0; cleared
- **Z:** Set if result is $00$; cleared otherwise
- **V:** \(N \oplus C = [N \cdot C] + [R \cdot C]\) (for \(N\) and \(C\) after the shift)
  - Set if \((N\) is set and \(C\) is cleared) or \((N\) is cleared and \(C\) is set); cleared otherwise (for values of \(N\) and \(C\) after the shift)
- **C:** \(A0\)
  - Set if the LSB of A was set before the shift; cleared otherwise
ASRA
Arithmetic Shift Right A

Operation:

Operation: \( C \)

Description: Shifts all bits of accumulator A one place to the right. Bit 7 is held constant. Bit 0 is loaded into the C status bit. This operation effectively divides a two’s complement value by two without changing its sign. The carry bit can be used to round the result.

CCR Details:

\[
\begin{array}{cccccccc}
S & X & H & I & N & Z & V & C \\
\hline
& & & & & & & \\
\end{array}
\]

N: Set if MSB of result is set; cleared otherwise
Z: Set if result is 00; cleared otherwise
V: N • C = [N + C] + [N + C] (for N and C after the shift)
Set if (N is set and C is cleared) or (N is cleared and C is set); cleared otherwise (for values of N and C after the shift)
C: A0
Set if the LSB of A was set before the shift; cleared otherwise

Source Form | Address Mode | Object Code | Access Detail | HCS12 | M68HC12
--- | --- | --- | --- | --- | ---
ASRA | SP+1 | – | – | – | –

SWI
Software Interrupt

Operation:

\[
\begin{array}{c}
\text{SP} - \$0002 \rightarrow \text{SP}; \text{RTN} \rightarrow (M_{\text{SP}} : M_{\text{SP}+1}) \\
\text{SP} - \$0002 \rightarrow \text{SP}; \text{RTN} \rightarrow (M_{\text{SP}} : M_{\text{SP}+1}) \\
\text{SP} - \$0002 \rightarrow \text{SP}; \text{RTN} \rightarrow (M_{\text{SP}} : M_{\text{SP}+1}) \\
\text{SP} - \$0001 \rightarrow \text{SP}; \text{CCR} \rightarrow (M_{\text{SP}}) \\
1 \rightarrow I \\
(\text{SWI Vector}) \rightarrow \text{PC}
\end{array}
\]

Description: Causes an interrupt without an external interrupt service request. Uses the address of the next instruction after SWI as a return address. Stacks the return address, index registers Y and X, accumulators B and A, and the CCR, decrementing the SP before each item is stacked. The I mask bit is then set, the PC is loaded with the SWI vector, and instruction execution resumes at that location. SWI is not affected by the I mask bit. Refer to Section 7. Exception Processing for more information.

CCR Details:

\[
\begin{array}{cccccccc}
S & X & H & I & N & Z & V & C \\
\hline
& & & & & & & \\
\end{array}
\]

I: 1; set

Source Form | Address Mode | Object Code | Access Detail | HCS12 | M68HC12
--- | --- | --- | --- | --- | ---
SWI | 3F | VSPSSPSsP | – | – |

1. The CPU also uses the SWI processing sequence for hardware interrupts and unimplemented opcode traps. A variation of the sequence (VSPSSPSsP) is used for resets.