Today's Goals

Two major goals

- Understand addressing modes so figure out how to use them.
  - If you don’t get addressing modes, you will have a serious problem to complete this course.

- Learn how to use a program trace.
Addressing Modes

How to get effective addresses

- The operand of an instruction can use different methods for specifying data in the memory (=addressing modes).
  - If the data number is in registers (inside the microprocessor), a memory address is not needed.

- The HCS12 has six addressing modes
  - Extended (EXT)
  - Direct (DIR)
  - Inherent (INH)
  - Immediate (IMM)
  - Index (IDX)
  - Relative (REL) : Used only with branch instructions.

- Effective Address
  - The effective address is the location that holds the data to be used by the operation.
  - The operand is often used to construct the effective address.
  - An addressing mode tells the microprocessor the way of calculation to get the effective address.

Extended Addressing (EXT)

Also called Absolute Addressing

- Effective address:
  - No operation needed.
  - Extended addressing tells the full memory address.

- Format:
  - Two-byte hexadecimal number (4-digit) preceded with a $. Actually ‘$’ simply means that the number is a hexadecimal number. (A number could be followed by ‘h’ excluding ‘’).

- Example:
  - (Assuming the instruction is stored at $2000)
  - LDAA $3000

    ```
    2000 6B 00 30 00
    3000 98
    ```

    - Load a byte value stored at address $3000 into the register A.
    - LDAA opr16a (M) ➔ A
    - 98 ➔ A
Direct Addressing (DIR)

Also called Zero-Paging Addressing

- Effective address:
  - This addressing mode only supplies the lower byte of the address.
  - Extend the one byte address to two-bytes by concatenating $00$ to the beginning of the operand.

- Format:
  - One byte hexadecimal number (2-digit) preceded with a $\$$.

- Example:
  - (Assuming the instruction is stored at $2000$)
  - LDAA $80$
    - Load a byte value stored at address $0080$ into the register A.
  - LDAA $opr8a$ (M) $A$ DIR $96$ dd
  - $98 \rightarrow A$

Inherent Addressing (INH)

Also called Implied Addressing

- Effective address:
  - No operation.

- Format:
  - No operand.

- Example:
  - (Assuming the instruction is stored at $2000$)
  - INCA
    - Increase register A by 1
  - INCA (A) $+ 01 \rightarrow A$ INH $42$
Immediate Addressing (IMM)

- Effective address:
  - No operation. The data itself is supplied as the operand.

- Format:
  - Number preceded with a #. '#' is followed by a number that is a value instead of an address.

- Example:
  - (Assuming the instruction is stored at $2000)
  - LDAA #$80
    - Load a byte value (the operand itself) into the register A.
  - LDD #1000
    - 1000 is 03E816 \(\rightarrow\) D (meaning 03 \(\rightarrow\) A and E8 \(\rightarrow\) B)

- The size of an operand
  - Register A and B have one-byte immediate operands.
  - Register D, X, Y, SP, and PC have two-byte ones.

Index Addressing (IDX, IDX1, IDX2)

- Effective Address
  - Add the operand as a signed number to the value in the X, Y, PC, or S registers.

- Format
  - Signed number, Register (X, Y, PC, or S)

- Example:
  - LDAA 0,X
    - The effective address is the value (=address) in register X. (=X + 0)
  - LDD -100,Y
    - The effective address is 100 lower than the value in Y. (=Y - 100)
  - LDX 1000, Y
    - The effective address is 1000 higher than the value in Y. (=Y + 1000)

- Notes:
  - The value in the specified register is not changed.
  - The smallest number of bits will be used to represent the address.
Index Addressing Postbytes

- An operand in the index addressing are called a **postbyte**.
- The postbyte tells the processor which two-byte register to be used as the base address, the size of the offset.

<table>
<thead>
<tr>
<th>Register</th>
<th>rr</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>00</td>
</tr>
<tr>
<td>Y</td>
<td>01</td>
</tr>
<tr>
<td>SP</td>
<td>10</td>
</tr>
<tr>
<td>PC</td>
<td>11</td>
</tr>
</tbody>
</table>

Postbyte for 5-bit Offset: **rr0nnnn**
Postbytes for 9-bit Offset: **111rr00n nnnnnnnn**
Postbytes for 16-bit Offset: **111rr010 nnnnnnn nnnnnnnn**

Index Addressing

**Examples**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Machine Code</th>
<th>[Image]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAA 4,Y</td>
<td>A6 44</td>
<td>91 000100</td>
</tr>
<tr>
<td>LDD -100,X</td>
<td>EC E1 9C</td>
<td>111 00 1 10011100</td>
</tr>
<tr>
<td>LDX -1000,Y</td>
<td>EE EA FC 18</td>
<td>111 010 1111 1100 0001 1000</td>
</tr>
</tbody>
</table>

Register | rr |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>00</td>
</tr>
<tr>
<td>Y</td>
<td>01</td>
</tr>
<tr>
<td>SP</td>
<td>10</td>
</tr>
<tr>
<td>PC</td>
<td>11</td>
</tr>
</tbody>
</table>
Instruction Set

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Operation</th>
<th>Addr. Mode</th>
<th>Machine Coding</th>
<th>Access Detail</th>
<th>S X H I</th>
<th>N Z V C</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAA #opr8b</td>
<td>(M) ⇒ A</td>
<td>IMM</td>
<td>86 ii</td>
<td>P</td>
<td>- - -</td>
<td>Δ Δ 1 0</td>
</tr>
<tr>
<td>LDAA opr8a</td>
<td>Load Acc. A</td>
<td>DIR</td>
<td>96 dd</td>
<td>rPF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDAA opr16a</td>
<td></td>
<td>EXT</td>
<td>86 hh ii</td>
<td>rPO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDAA oprx0_xysp</td>
<td></td>
<td>IDX</td>
<td>A6 xb</td>
<td>rPF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDAA oprx9_xysp</td>
<td></td>
<td>IDX1</td>
<td>A6 xb ff</td>
<td>rPO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDAA oprx16,xysp</td>
<td></td>
<td>IDX2</td>
<td>A6 xe ee ff</td>
<td>frPP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Above is a portion of the entry for the LDAA instruction.
- Now, we can better understand information in the HCS12 instruction sets.

Addressing Mode Summary

How to Get an Effective Address

INH

IMM

DIR

EXT

IDX

REL

Op Code
Data
Effective Address
00
Addr-low

Op Code
Data
Effective Address
Addr-high
Addr-low

Op Code
Offset
Effective Address

Op Code
Relative offset
Effective Address

Program Counter

Index Register
Program Trace

- A diagram showing the contents of the HCS12 memory which contains a program.
- A program trace shows the contents of the processor’s registers as the program is executed.
- Very useful for debugging programs

<table>
<thead>
<tr>
<th>Trace Line</th>
<th>Address</th>
<th>Instruction</th>
<th>PC</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2000</td>
<td>LDAA 3000h</td>
<td>2003</td>
<td>19</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>2003</td>
<td>LDAB #2</td>
<td>2005</td>
<td>19</td>
<td>02</td>
</tr>
<tr>
<td>3</td>
<td>2005</td>
<td>ABA</td>
<td>2007</td>
<td>1B</td>
<td>02</td>
</tr>
<tr>
<td>4</td>
<td>2007</td>
<td>STAA 3001h</td>
<td>200A</td>
<td>1B</td>
<td>02</td>
</tr>
<tr>
<td>5</td>
<td>200A</td>
<td>“Stop”</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### Program Trace

#### Another Example

<table>
<thead>
<tr>
<th>Trace Line</th>
<th>Address</th>
<th>Instruction</th>
<th>PC</th>
<th>X</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2000</td>
<td>LDX #3001h</td>
<td>2003</td>
<td>3001</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>2003</td>
<td>LDD 1,X</td>
<td>2005</td>
<td>3001</td>
<td>45</td>
<td>99</td>
</tr>
<tr>
<td>3</td>
<td>2005</td>
<td>INC 1,X</td>
<td>2006</td>
<td>3001</td>
<td>45</td>
<td>9A</td>
</tr>
<tr>
<td>4</td>
<td>2007</td>
<td>STD -1,X</td>
<td>2009</td>
<td>3001</td>
<td>00</td>
<td>9A</td>
</tr>
<tr>
<td>5</td>
<td>2009</td>
<td>&quot;STOP&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- X requires a 2-byte operand with immediate addressing since it is a 2-byte register.
- Note that using indexed addressing to load/store register D does not change the value in register X.
- What are the values in memory locations from 3000h to 3003h after the program is done executing? **45-9A-45-99**

### Questions?
Wrap-up

What we’ve learned

- Five addressing modes
- Program trace

What to Come

- Unconditional branches
- Relative addressing mode