Lecture 7: Comparison Branches
Today’s Goals

- Review addressing modes
- Use basic instructions
- Use the Unsigned and Signed Comparison Branches to control the flow of programs
Addressing Mode Summary

How to Get an Effective Address

- **INH**:
  - Op Code

- **IMM**:
  - Op Code
  - Data
  - Data-high
  - Data-low

- **DIR**:
  - Op Code
  - Addr-low
  - Effective Address
    - 00
    - Addr-low

- **EXT**:
  - Op Code
  - Addr-high
  - Addr-low
  - Effective Address
  - Addr-high
  - Addr-low

- **IDX**:
  - Op Code
  - Offset
  - Effective Address
  - Index Register

- **REL**:
  - Op Code
  - Relative offset
  - Effective Address
  - Program Counter
Basic Instructions

Load and store instruction

• 8 Bit accumulator load
  ▪ LDAA: load a value from the specified memory to accumulator A
  ▪ LDAB: load a value from the specified memory to accumulator B

• 8 bit accumulator store
  ▪ STAA: store a value in accumulator A into the specified memory
  ▪ STAB: store a value in accumulator B into the specified memory

• 16 bit register load and store
  ▪ LDD, LDX, LDY, LDS
  ▪ STD, STX, STY, STS

• Examples:
  ▪ Tell the difference between
    • LDAA #$10 and LDAA $10
    • LDD $1000 and LDD #$1000
Basic Instructions

Exchange, Move, and Clear

- Exchange instructions
  - EXG: exchange register contents
    - EXG X Y
    - EXG A B
    - EXG X B
    - EXG B X
  - XGDX: exchange register D and X
  - XGDY: exchange register D and Y

- Move
  - MOVB: move a byte from a memory to another
    - MOVB #32 $0811
    - MOVB 1,X+1,Y+ ; (X)+1 \rightarrow (Y)+1 and X=X+1, Y=Y+1
  - MOVW: move a word (2 bytes) from a memory to another

- Clear
  - CLR: clear a byte in the specified memory
    - CLR $0800 ; set the content at $0800 to 0
  - CLRA
  - CLR B

Compare Move instructions with Store ones.

Move: Memory to Memory
Store: Register to Memory
Basic Instructions

Register to register transfer

• Copy a value from one register to another
  • TFR: Transfer a content of one register to another
    • TFR A B
  • TAB: (A) \(\rightarrow\) (B)
  • TBA: (B) \(\rightarrow\) (A)
  • SEX: Sign EXtended transfer from 8 bit register to 16 bit register
    • SEX A D
  • TPA: (CCR) \(\rightarrow\) (A)
  • TAP: (A) \(\rightarrow\) (CCR)
  • TSX: (SP) \(\rightarrow\) (X)
  • TXS: (X) \(\rightarrow\) (SP)
  • TSY: (SP) \(\rightarrow\) (Y)
  • TYS: (Y) \(\rightarrow\) (SP)
Basic Instructions

Increments, Decrements, and Negate

• Increments
  ▪ INC: (M) + 1 → M
  ▪ INCA: (A) + 1 → A
  ▪ INCB
  ▪ INS
  ▪ INX
  ▪ INY

• Decrements
  ▪ DEC
  ▪ DECA
  ▪ DECB
  ▪ DES
  ▪ DEX
  ▪ DEY

• Negate
  ▪ NEG: negate a memory byte
  ▪ NEGA
  ▪ NEGB
Basic Instructions

Comparison

• Comparison instructions
  ▪ Actually, they are subtractions.
  ▪ Discard the answer
  ▪ No change in the registers and the memories
  ▪ CCR bits are affected instead.

• CBA: Compare B to A:
  ▪ Subtract the B accumulator from the A accumulator
  ▪ (A) – (B)

• CMPA, CMPB: Compare accumulator to memory:
  ▪ Subtract the content of a memory from the accumulator
  ▪ (A) – (M), (B) – (M)

The order is important!
Need to know which one is minuend or which subtrahend to interpret CCR bits.
Comparison Instruction

Example

• Let register A have 10h, register B have 20h
  ▪ \((A) = 10h, (B) = 20h\)

• CBA
  ▪ \((A) - (B) = E0h\)
  ▪ Instead of saving the result, the result $E0h$ affects CCR bits.
  ▪ N: 1, Z: 0, V: 0, C: 1

• CMPA, CMPB
  ▪ Assume FFh at address $1000$
  ▪ CMPA $1000$
    ▪ \((A) - ($1000) = 10h - FFh = 11h\)
    ▪ N: 0, Z: 0, V: 0, C: 1

• Therefore,
  ▪ CBA does not mean that I want to compare B and A.
  ▪ Rather, CBA means that I want to know what happens in CCR bits after \((A) - (B)\) operation.
Comparison Branches

• Comparison branches are based on comparing two numbers.
• Comparing is done by subtraction (see the previous slides).
• The subtraction instruction set CCR bits.
  ▪ Three categories of subtraction
    ▪ Actual subtraction
      ▪ Perform operation and keep the result.
    ▪ Comparison*
      ▪ Perform subtraction and discard the answer.
    ▪ Test
      ▪ Perform subtraction using 0.

• Comparison branch instructions examine the CCR bits.

  But, actually comparison branch instructions only check CCR bits.

  Therefore, any instructions that can affect CCR bits can be placed before comparison branch instructions.

Logically, comparison instructions are needed before we are using comparison branches.
Comparison Branches

Instructions

• Two sets of comparison branches: unsigned and signed

• Unsigned:
  ▪ Higher, Higher or Same, Lower, Lower or Same

• Signed
  ▪ Greater Than, Greater or Equal, Less Than, Less or Equal

• HCS12 instructions for comparison branches.

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Unsigned</th>
<th>Signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>BHI</td>
<td>BGT</td>
</tr>
<tr>
<td>≥</td>
<td>BHS</td>
<td>BGE</td>
</tr>
<tr>
<td>&lt;</td>
<td>BLO</td>
<td>BLT</td>
</tr>
<tr>
<td>≤</td>
<td>BLS</td>
<td>BLE</td>
</tr>
<tr>
<td>=</td>
<td>BEQ</td>
<td>BEQ</td>
</tr>
<tr>
<td>≠</td>
<td>BNE</td>
<td>BNE</td>
</tr>
</tbody>
</table>
Comparison Branches

Example Program

• Trace the program below. Assume the memory locations $2000$, $2001$, and $2002$ are already set to $40$, $F0$, and $55$ respectively.

```
1:  1500  CE 2000  LDX #$2000
2:  1503  180B FF 1000  MOVB #$FF,$1000
3:  1508  C6 02  LDAB #2
4:  150A  27 0E  BEQ 14
5:  150C  A6 00  LDAA 0,X
6:  150E  B1 1000  CMPA $1000
7:  1511  24 03  BHS 3
8:  1513  7A 1000  STAA $1000
9:  1516  08  INX
10: 1517  53  DECB
11: 1518  20 F0  BRA -16
12: 151A  3F  SWI
```

<table>
<thead>
<tr>
<th>...</th>
<th>2000</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>2001</td>
<td>F0</td>
</tr>
<tr>
<td>2002</td>
<td>2001</td>
<td>F0</td>
</tr>
<tr>
<td>...</td>
<td>2002</td>
<td>55</td>
</tr>
</tbody>
</table>
1: 1500 CE 2000 LDX #$2000
2: 1503 180B FF 1000 MOVB #$FF,$1000
3: 1508 C6 02 LDAB #2
4: 150A 27 0E BEQ 14
5: 150C A6 00 LDAA 0,X
6: 150E B1 1000 CMPA $1000
7: 1511 24 03 BHS 3
8: 1513 7A 1000 STAA $1000
9: 1516 08 INX
10: 1517 53 DECB
11: 1518 20 F0 BRA -16
12: 151A 3F SWI

<table>
<thead>
<tr>
<th>Trace</th>
<th>Line</th>
<th>PC</th>
<th>A</th>
<th>B</th>
<th>X</th>
<th>N</th>
<th>Z</th>
<th>V</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1503</td>
<td>-</td>
<td>-</td>
<td>2000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1508</td>
<td>-</td>
<td>-</td>
<td>2000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>150A</td>
<td>-</td>
<td>02</td>
<td>2000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>150C</td>
<td>-</td>
<td>02</td>
<td>2000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>150E</td>
<td>40</td>
<td>02</td>
<td>2000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1511</td>
<td>40</td>
<td>02</td>
<td>2000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>1513</td>
<td>40</td>
<td>02</td>
<td>2000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>1516</td>
<td>40</td>
<td>02</td>
<td>2000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>1517</td>
<td>40</td>
<td>02</td>
<td>2001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>1518</td>
<td>40</td>
<td>01</td>
<td>2001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Trace</td>
<td>Line</td>
<td>PC</td>
<td>A</td>
<td>B</td>
<td>X</td>
<td>N</td>
<td>Z</td>
<td>V</td>
<td>C</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>-----</td>
<td>------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>150A</td>
<td>40</td>
<td>01</td>
<td>2001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>150C</td>
<td>40</td>
<td>01</td>
<td>2001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>150E</td>
<td>F0</td>
<td>01</td>
<td>2001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>1511</td>
<td>F0</td>
<td>01</td>
<td>2001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>7</td>
<td>1516</td>
<td>F0</td>
<td>01</td>
<td>2001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>9</td>
<td>1517</td>
<td>F0</td>
<td>01</td>
<td>2002</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>10</td>
<td>1518</td>
<td>F0</td>
<td>00</td>
<td>2002</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>11</td>
<td>150A</td>
<td>F0</td>
<td>00</td>
<td>2002</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>4</td>
<td>151A</td>
<td>F0</td>
<td>00</td>
<td>2002</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>12</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Questions

• What does this program do?
  ▪ Get a minimum value from the values from $2000 to ($2000 + the initial content in register B)

• What changes are needed to process 200 bytes?
  ▪ Line 3: LDAB #2 → LDAB #200 (or #$C8 or #C8h)

• What changes are needed to process signed numbers?
  ▪ Line 7: BHS → BGT
  ▪ Line 2: #$FF → #$7F (or #7Fh)

• What changes are needed if the list of data begins at $3000?
  ▪ Line 1: #$2000 → #$3000h (or #3000h)

• What changes are needed if the answer must be stored to location $3FFF?
  ▪ Line 2, 6, and 8: $1000 → $3FFF (or 3FFFh)
Questions?
Wrap-up

What we’ve learned

• Quick tour of basic instructions

• Comparison branches
  ▪ Unsigned
    • BHI, BGT, BHS, BGE
  ▪ Signed
    • BLO, BLT, BLS, BLE
  ▪ Either signed or unsigned
    • BEQ, BNE
What to Come

- Assembly language
- Flowchart