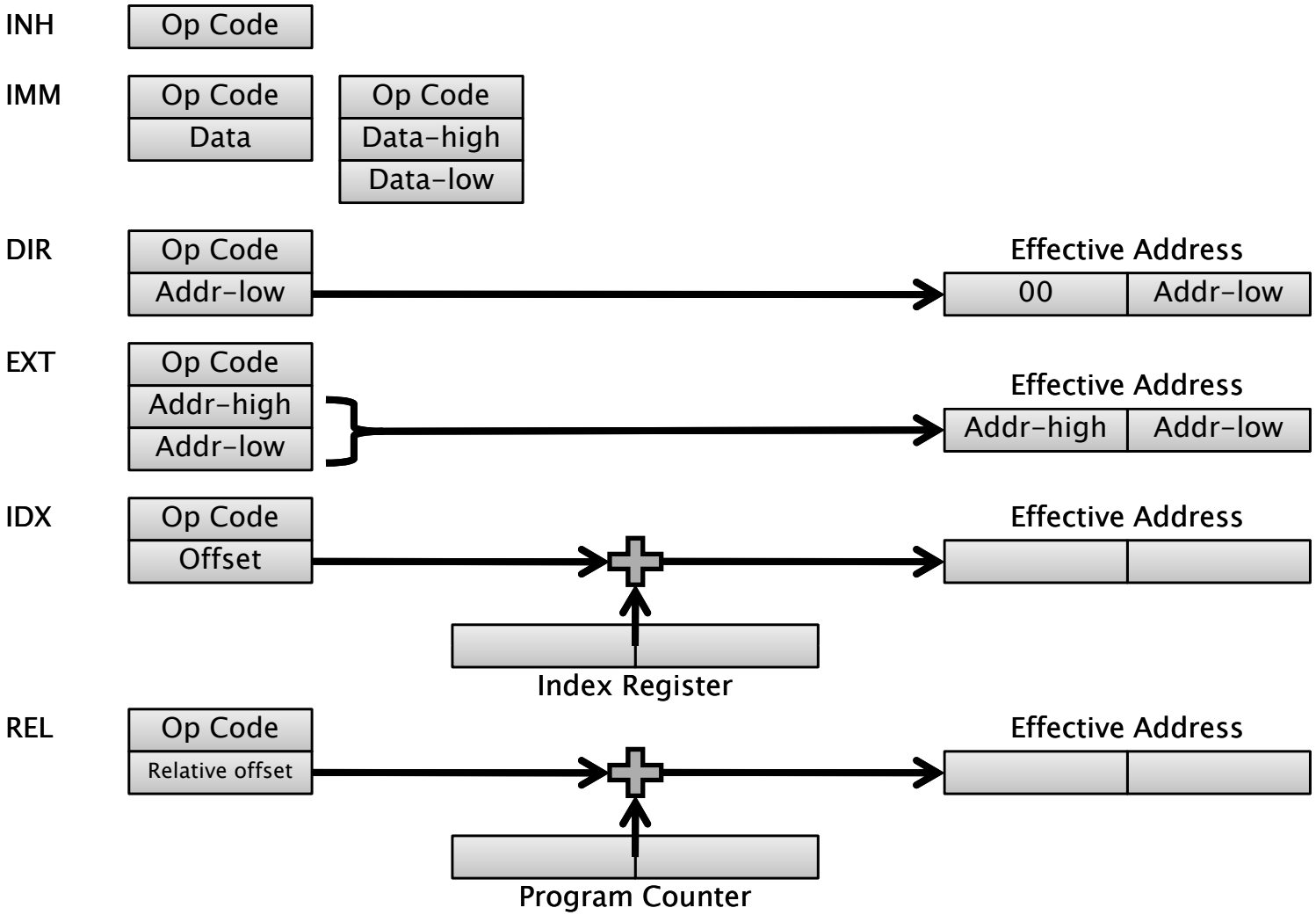
Lecture 7:
Comparison Branches

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



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

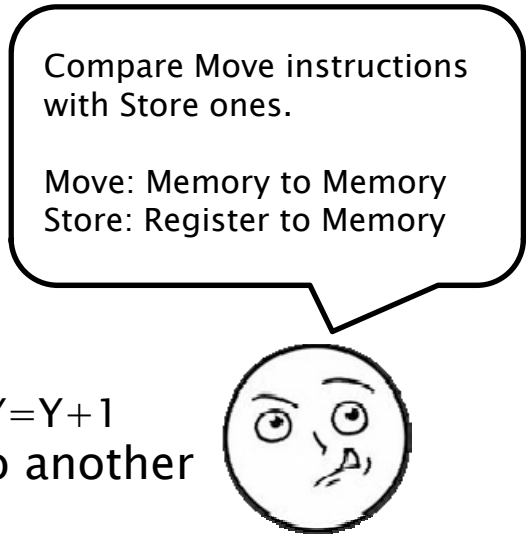
Basic Instructions

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 → (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
  - CLRB



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) → (B)
  - TBA: (B) → (A)
  - SEX: Sign EXtended transfer from 8 bit register to 16 bit register
    - SEX A D
  - TPA: (CCR) → (A)
  - TAP: (A) → (CCR)
  - TSX: (SP) → (X)
  - TXS: (X) → (SP)
  - TSY: (SP) → (Y)
  - TYS: (Y) → (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 is nothing but subtraction discarding the answer.

- 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 $E0 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.

But, actually comparison branch instructions only check CCR bits.

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

- Comparison branch instructions examine the CCR bits.

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.

Comparison	Unsigned	Signed
>	BHI	BGT
≥	BHS	BGE
<	BLO	BLT
≤	BLS	BLE
=	BEQ	BEQ
≠	BNE	BNE

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
```

...	
2000	40
2001	F0
2002	55
...	

```
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
```

…	
1000	
…	
2000	40
2001	F0
2002	55
…	

Trace	Line	PC	A	B	X	N	Z	V	C
1	1	1503	-	-	2000	0	0	0	-
2	2	1508	-	-	2000	0	0	0	-
3	3	150A	-	02	2000	0	0	0	-
4	4	150C	-	02	2000	0	0	0	-
5	5	150E	40	02	2000	0	0	0	-
6	6	1511	40	02	2000	0	0	0	1
7	7	1513	40	02	2000	0	0	0	1
8	8	1516	40	02	2000	0	0	0	1
9	9	1517	40	02	2001	0	0	0	1
10	10	1518	40	01	2001	0	0	0	1

Continued

```
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
```

…	
1000	
…	
2000	40
2001	F0
2002	55
…	

FF → 40

Trace	Line	PC	A	B	X	N	Z	V	C
11	11	150A	40	01	2001	0	0	0	1
12	4	150C	40	01	2001	0	0	0	1
13	5	150E	F0	01	2001	1	0	0	1
14	6	1511	F0	01	2001	1	0	0	0
15	7	1516	F0	01	2001	1	0	0	0
16	9	1517	F0	01	2002	1	0	0	0
17	10	1518	F0	00	2002	0	1	0	0
18	11	150A	F0	00	2002	0	1	0	0
19	4	151A	F0	00	2002	0	1	0	0
20	12	-	-	-	-	-	-	-	-

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