Using X and Y as Pointers

- Registers X and Y are often used to point to data.
- To initialize pointer use

```
ldx     #table
```

not

```
ldx     table
```

- For example, the following loads the address of `table` (`$0900`) into X; i.e., X will point to `table`:

```
ldx     #table    ;  Address of table => X
```

The following puts the first two bytes of `table` (`$0C7A`) into X. X will **not** point to `table`:

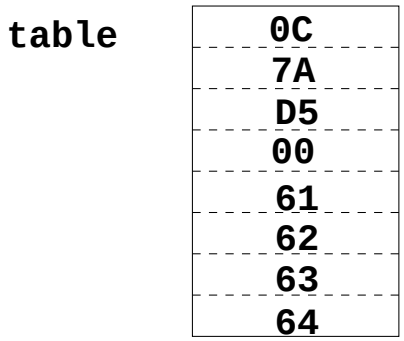
```
ldx     table     ;  First two bytes of table => X
```

- To step through table, need to increment pointer after use

```
ldaa    0,x
inx
```

or

```
ldaa    1,x+
```




```
        org   $900
table:  dc.b  12,122,-43,0
        dc.b  'a','b','c','d'
```

Which branch instruction should you use?

Branch if A > B

Is 0xFF > 0x00?

If unsigned, 0xFF = 255 and 0x00 = 0,

so 0xFF > 0x00

If signed, 0xFF = −1 and 0x00 = 0,

so 0xFF < 0x00

Using unsigned numbers: BHI (checks C bit of CCR)

Using signed numbers: BGT (checks V bit of CCR)

For unsigned numbers, use branch instructions which check C bit

For signed numbers, use branch instructions which check V bit

Hand Assembling a Program

To hand-assemble a program, do the following:

1. Start with the `org` statement, which shows where the first byte of the program will go into memory.
   (E.g., `org $0800` will put the first instruction at address `$0800`.)
2. Look at the first instruction. Determine the addressing mode used.
   (E.g., `ldab #10` uses `IMM` mode.)
3. Look up the instruction in the `CPU12 Reference Manual`, find the appropriate Addressing Mode, and the Object Code for that addressing mode.
   (E.g., `ldab IMM` has object code `C6 ii`.)
4. Put in the object code for the instruction, and put in the appropriate operand. Be careful to convert decimal operands to hex operands if necessary.
   (E.g., `ldab #10` becomes `C6 0A`.)
5. Add the number of bytes of this instruction to the address of the instruction to determine the address of the next instruction.
   (E.g., `$0800 + 2 = $0802` will be the starting addrss of the next instruction.)

```
        org    $0800
        ldab   #10
loop:   clra
        decb
        bne    loop
        swi
```

68HC12 Cycles

- 68HC12 works on 16 MHz clock
- A processor cycle takes 2 clock cycles – P clock is 8 MHz
- Each processor cycle takes 125 ns (1/8 μs) to execute
- An instruction takes from 1 to 12 processor cycles to execute

```
0800                  org  $0800    ; Inst   Mode   Cycles
0800 C6 0A            ldab #10      ; LDAB   (IMM)    1
0802 87       loop:   clra          ; CLRA   (INH)    1
0803 53               decb          ; DECB   (INH)    1
0804 26 FC            bne  loop     ; BNE    (REL)  3/1   (branch/no branch)
0806 3F               swi           ; SWI             9
```

Program goes through loop 10 times. First 9 times it branches to loop, last time it does not.

Total number of cycles:

$1 + 9 \times (1 + 1 + 3) + 1 \times (1 + 1 + 1) + 9 = 58$

58 cycles = 58 $\times$ 0.125 μs/cycle = 7.25 μs