- **Decimal, Hexadecimal and Binary Numbers**
- **Writing an assembly language program**
  - Disassembly of MC9S12 op codes
  - Use flow charts to lay out structure of program
  - Use common flow structures
    - if-then
    - if-then-else
    - do-while
    - while
  - Do not use spaghetti code
  - Plan structure of data in memory
  - Plan overall structure of program
  - Work down to more detailed program structure
  - Implement structure with instructions
  - Optimize program to make use of instruction efficiencies
  - **Do not sacrifice clarity for efficiency**

Binary, Hex and Decimal Numbers (4-bit representation)

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
...	...	...
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

What does a number represent?

Binary numbers are a code, and represent what the programmer intends for the code.

0x72   Some possible meanings:
   'r' (ASCII)
   INC MEM (hh ll) (HC12 instruction)
   2.26V (Input from A/D converter)
   114_{10} (Unsigned number)
   $+114_{10}$ (Signed number)
   Set temperature in room to 69 °F
   Set cruise control speed to 120 mph

Binary to Unsigned Decimal:
Convert Binary to Unsigned Decimal
1111011_2
$1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
$1 \times 64 + 1 \times 32 + 1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$
123_{10}

Hex to Unsigned Decimal
Convert Hex to Unsigned Decimal
$82D6_{16}$
$8 \times 16^3 + 2 \times 16^2 + 13 \times 16^1 + 6 \times 16^0$
$8 \times 4096 + 2 \times 256 + 13 \times 16 + 6 \times 1$
33494_{10}

Unsigned Decimal to Hex

Convert Unsigned Decimal to Hex

Division	Q	R	
		Decimal	**Hex**
721/16	45	1	1
45/16	2	13	D
2/16	0	2	2

$$721\ _{10} =\ 2D1\ _{16}$$

Signed Number Representation in 2's Complement Form:

If the most significant bit (MSB) is 0 (most significant hex digit 0−7), then the number is positive.

Get decimal equivalent by converting number to decimal, and use the + sign.

Example for 8−bit number:

$3A\ _{16}$ −> + (3 x 16^1 + 10 x 16^0) $_{10}$
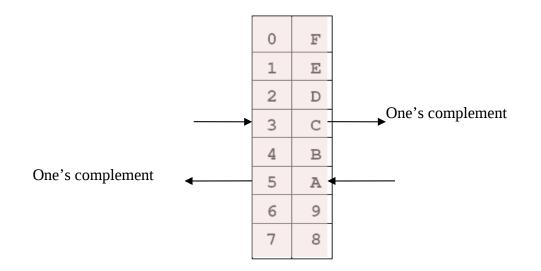   + (3 x 16 + 10 x 1) $_{10}$
        + 58 $_{10}$

If the most significant bit is 1 (most significant hex digit 8−F), then the number is negative.

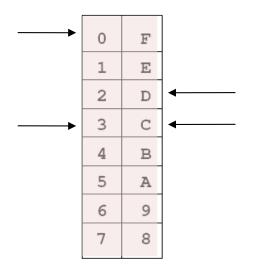Get decimal equivalent by taking 2's complement of number, converting to decimal, and using − sign.

Example for 8−bit number:

A3_{16} –> - $(5C+1)_{16}$

$\quad\quad$ - $(5 \times 16^1 + 13 \times 16^0)_{10}$

$\quad\quad$ - $(5 \times 16 + 13 \times 1)_{10}$

$\quad\quad$ - **93** $_{10}$

One's complement table makes it simple to finding 2's complements

0	F
1	E
2	D
3	C
4	B
5	A
6	9
7	8

One's complement

One's complement

One's complement table makes it simple to finding 2's complements

0	F
1	E
2	D
3	C
4	B
5	A
6	9
7	8

To take two's complement, add one to one's complement.

Take two's complement of **D0C3**:

$$2F3C + 1 = \textbf{2F3D}$$

Addition and Subtraction of Binary and Hexadecimal Numbers

Setting the C (Carry), V (Overflow), N (Negative) and Z (Zero) bits

How the C, V, N and Z bits of the CCR are changed?

N bit is set if result of operation is negative (MSB = 1)

Z bit is set if result of operation is zero (All bits = 0)

V bit is set if operation produced an overflow

C bit is set if operation produced a carry (borrow on subtraction)

Note: Not all instructions change these bits of the CCR

Addition of Hexadecimal Numbers

ADDITION:

C bit set when result does not fit in word

V bit set when P + P = N or
$\qquad$ N + N = P

N bit set when MSB of result is 1

Z bit set when result is 0

```
   7A           2A           AC           AC
 +52          +52          +8A          +72
 -----        -----        ------       ------
   CC           7C           36           1E
```

C: 0   C: 0   C: 1   C: 1

V: 1   V: 0   V: 1   V: 0

N: 1   N: 0   N: 0   N: 0

Z: 0   Z: 0   Z: 0   Z: 0

Subtraction of Hexadecimal Numbers

SUBTRACTION:

C bit set on borrow (when the magnitude of the subtrahend is greater than the minuend

V bit set when N - P = P or
$\qquad$ P - N = N

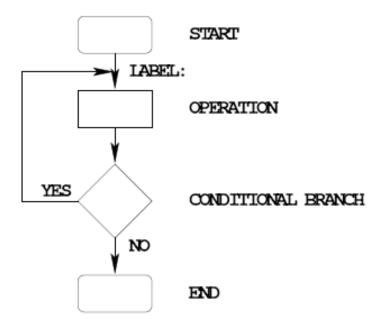N bit set when MSB is 1

Z bit set when result is 0

```
   7A            8A            5C            2C
  -5C           -5C           -8A           -72
  -----         -----         ------        ------
   1E            2E            D2            BA
```

C: 0   C: 0   C: 1   C: 1

V: 0   V: 1   V: 1   V: 0

N: 0   N: 0   N: 1   N: 1

Z: 0   Z: 0   Z: 0   Z: 0
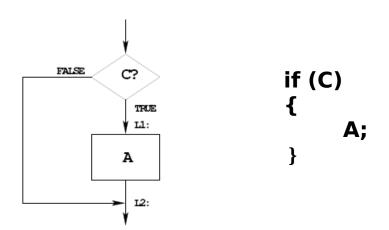
Writing Assembly Language Programs

Use Flowcharts to Help Plan Program Structure

Flow chart symbols:
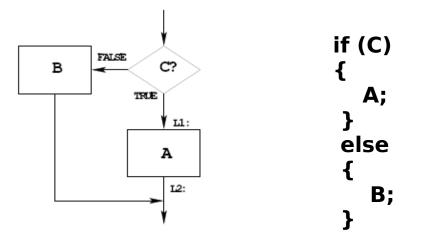
IF-THEN Flow Structure

if (C)
{
    A;
}

EXAMPLE:

if (A<10)	CMPA #10 ; if (A<10)
{	BLT L1 ; signed numbers
var = 5;	BRA L2
}	L1: LDAB #5 ; var=5
	STAB var
	L2: next instruction

OR:

```
        CMPA    #10 ; if(A<10)
        BGE     L2  ; signed numbers
        LDAB    #5   ; var=5
        STAB    var
L2:   next instruction
```

IF-THEN-ELSE Flow Structure


```
if (C)
{
    A;
}
else
{
    B;
}
```

```
if(A < 10)              CMPA      #10 ; if(A<10)
{                       BLT        L1  ; signed numbers
    var = 5;            CLR       var  ; var=0
}                       BRA        L2
else          L1:       LDAB      #5   ; var=5
{                       STAB       var
    var = 0;   L2:   next instruction
}
```
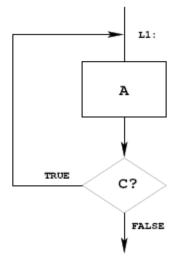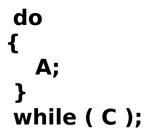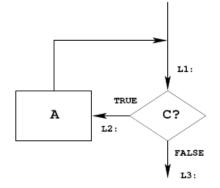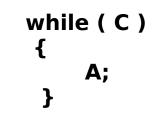
DO WHILE Flow Structure


```
do
{
   A;
}
while ( C );
```

EXAMPLE:

i = 0;		LDX	#table
do		CLRA	; i=0
{	L1:	ASR	1,X+ ; table[i] /=2
table[i]=table[i]/2;		INCA	; i=i+1
i=i+1;		CMPA	#LEN ; while(i<=10)
}		BLE L1	; unsigned
while (i <= LEN);			; numbers

WHILE Flow Structure


```
while ( C )
 {
      A;
 }
```

EXAMPLE:

```
i = 0;                              LDX        #table
while( i <= LEN)                    CLRA
{                           L1:    CMPA       #LEN
    table[i]=table[i]*2;           BLT        L2
    i=i+1;                         BRA        L3
}                           L2:    ASL        1,X+
                                   INCA
                                   BRA        L1
                            L3:    next instruction
```
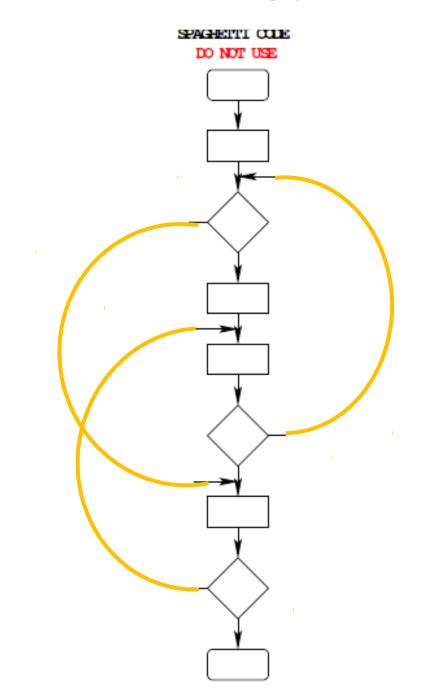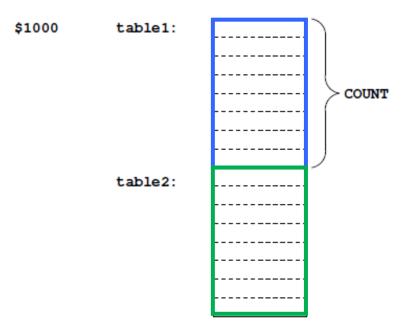
Use Good Structure When Writing Programs

— Do Not Use Spaghetti Code

Example Program: Divide a table of data by 2

Problem: Start with a table of data. The table consists of 5 values. Each value is between 0 and 255. Create a new table whose contents are the original table divided by 2.

1. Determine where code and data will go in memory.
   Code at $2000, data at $1000.

2. Determine type of variables to use.
   Because data will be between 0 and 255, can use unsigned 8-bit numbers.

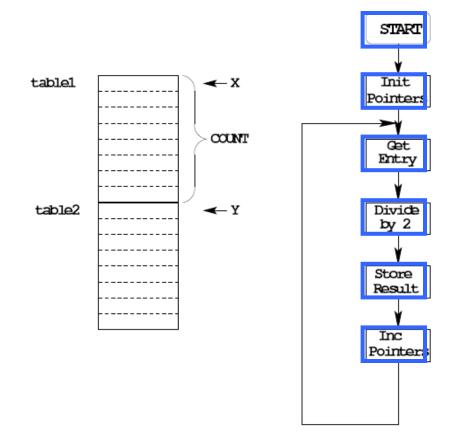**3.** Draw a picture of the data structures in memory:

4. Strategy: Because we are using a table of data, <u>we will need pointers to each table</u> so we can keep track of which table element we are working on.

Use the X and Y registers as pointers to the tables.
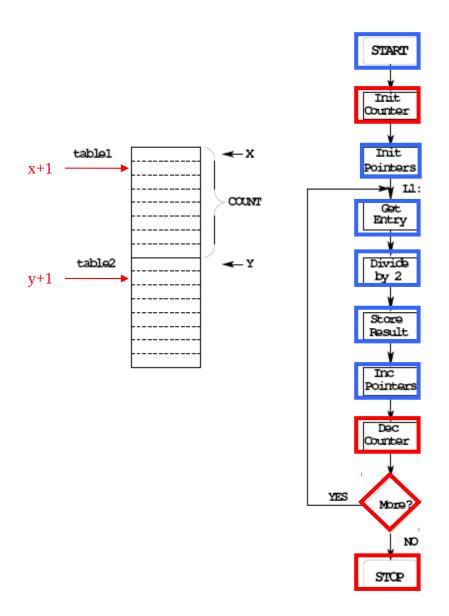
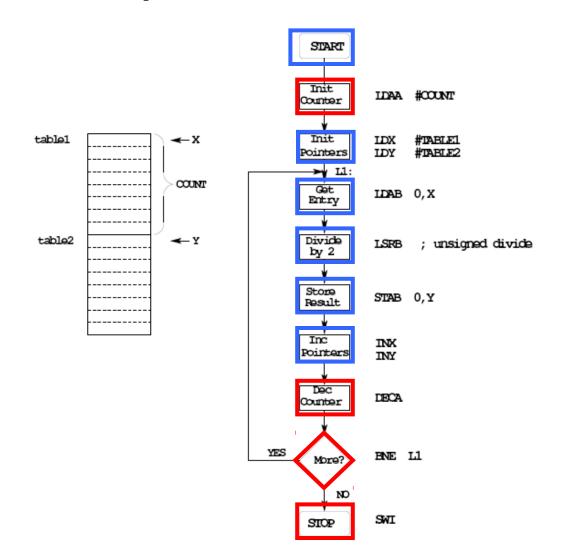**5.** Use a simple flow chart to plan structure of program.

6. Need a way to determine when we reach the end of the table.

One way: Use a counter (say, register A) to keep track of how many Elements we have processed.

7. Add code to implement blocks:

8. Write the program:

```
; Program to divide a table by two
; and store the results in memory

prog:  equ   $2000
data:  equ   $1000
count:       equ   5

              org    prog  ; Set program counter to 0x2000
              ldaa  #count        ; Use A as counter
              ldx    #table1       ; Use X as data pointer to table1
              ldy    #table2       ; Use Y as data pointer to table2
l1:           ldab   0,x           ; Get entry from table1
              lsrb                  ; Divide by two (unsigned)
              stab    0,y           ; Save in table2
              inx                   ; Increment table1 pointer
              iny                   ; Increment table2 pointer
              deca                  ; Decrement counter
              bne    l1            ; Counter != 0 => more entries
                                    ; to divide
              swi                   ; Done


              org data
table1:       dc.b   $07,$c2,$3a,$68,$f3
table2:       ds.b    count
```

9. Advanced: Optimize program to make use of instructions set efficiencies:

```
; Program to divide a table by two
; and store the results in memory

prog:  equ   $1000
data:        equ    $2000

count:       equ   5

             org    prog        ; Set program counter to 0x1000
             ldaa   #count      ; Use A as counter
             ldx    #table1     ; Use X as data pointer to table1
             ldy    #table2     ; Use Y as data pointer to table2
l1:          ldab   1,x+        ; Get entry from table1; then inc ptr.
             lsrb               ; Divide by two (unsigned)
             stab   1,y+        ; Save in table2; then inc ptr.
             dbne   a,l1        ; Decrement counter; if not 0,
                                ;  more to do
             swi                ; Done


             org    data
table1:      dc.b   $07,$c2,$3a,$68,$f3
table2:      ds.b   count
```

TOP-DOWN PROGRAM DESIGN

• PLAN DATA STRUCTURES IN MEMORY

• START WITH A LARGE PICTURE OF THE PROGRAM STRUCTURE

• WORK DOWN TO MORE DETAILED STRUCTURE

• TRANSLATE STRUCTURE INTO CODE

• OPTIMIZE FOR EFFICIENCY

DO NOT SACRIFICE CLARITY FOR EFFICIENCY