- **Some more simple assembly language programs**
- **Using 9S12 input and output ports**
- **Huang Sections 7.2 through 7.5**
    - Using a subroutine to wait for and respond to an event
    - Using an input port to chech the state of DIP switches
    - Using an output port to control LEDs
    - An assembly language program to display a pattern on a set of LEDs

```
; Subroutine to wait for 100 ms

delay:      psha              ; 2 cycles
            pshx              ; 2 cycles
            ldaa    #250      ; 1 cycle
loop2:      ldx     #3200     ; 2 cycles ------------
loop1:      dbne    x,loop1   ; 3 cycles inner loop | Outer loop
            dbne    a,loop2   ; 3 cycles ------------
            pulx              ; 3 cycls
            pula              ; 3 cycls
            rts               ; 5 cycls
```

• Inner loop takes 3 cycles; is executed **3200** (**X**) times
• Outer loop takes (2 + 3X + 3) cycles; is executed **250** (**A**) times
• Total number of cycles: 2+2+1+A*(2+3X+3)+3+3+5 = **2,401,266 cycles**
• This takes 100 ms with 24 MHz E-clock

Some C basics

• Every C program has a function **main()**
- The simplest C program is:

```
main()
{

}
```

– Our compiler ends a program by executing an infinite loop – the program never returns to DBug-12. In order to return to DBug-12, include the **swi** assembly language instruction. Here is how to do that:

```
main()
{
        asm(" swi");
}
```

• Every statement ends with a semicolon

```
x = a+b;
```

• Comment starts with /* ends with */

```
/* This is a comment */
```

or

```
// This is a comment too
```

• Simple program – increment Port A

```
#include "hcs12.h"
main()
{
        DDRA = 0xff;            /* Make PORTA output */
        PORTA = 0;              /* Start at 0 */
        while(1)                /* Repeat forever */
        {
                PORTA = PORTA + 1;
        }
}
```

• Data Types:

8-bit	16-bit
unsigned char	unsigned int
signed char	signed int

• Need to declare variable before using it:

```
signed char c;
unsigned int i;
```

• Can initialize variable when you define it:

```
signed char c = 0xaa;
signed int i = 1000;
```

– You tell compiler it you are using signed or unsiged numbers; the compiler will figure out whether to use BGT or BHI

• Arrays:

```
unsigned char table[10]; /* Set aside 10 bytes for table */
```

– Can refer to elements table[0] through table[9]
– Can initialize an array when you define it:
```
unsigned char table[] = {0xaa, 0x55, 0xa5, 0x5a};
```

• Arithmetic operators:

```
+ (add)         x = a+b;
- (subtract)    x = a-b;
* (multiply)    x = a*b;
/ (divide)      x = a/b;
% (modulo)      x = a%b;      (Remainder on divide)
```

• Logical operators

```
& (bitwise AND)                 y = x & 0xaa;
| (bitwise OR)                  y = x | 0xaa;
^ (bitwise XOR)                 y = x ^ 0xaa;
<< (shift left)                 y = x << 1;
>> (shift right)                y = x >> 2;
~ (1's complement)              y = ~x;
- (2's complement - negate)     y = -x;
```

Check for equality - use ==
```
if (x == 5)
```

Check if two conditions true:
```
if ((x==5) && (y==10))
```

Check if either of two conditions true:
```
if ((x==5) || (y==10))
```

• Assign a name to a number
```
#define COUNT 5
```

• Include a header file (such as **hcs12.h**):
```
#include "hcs12.h"
```

• Declare a function: Tell what parameters it uses, what type of number it returns:
```
int read_port(int port);
```

• If a function doesn't return a number, declare it to be type void

```
void delay(int num);
```

Hello, World!

• Here is the standard "hello, world" program:

```
#include <stdio.h>
main()
{
        printf("hello, world\r\n");
}
```

• To write the "hello, world" program, you need to use the printf() function.

• The printf() function is normally a library function

• Our compiler **does not have a library** which includes the **printf()** function.

• DBug-12 has a built-in printf, which you can access in the following way:

```
#include "DBug12.h"
main()
{
        DB12FNP->printf("hello, world\r\n");
        asm(" swi");
}
```

• The above program is about 40 bytes long.

• Note that the DBug-12 printf() does not work for floating point numbers.

• You can access a few other standard C functions through DBug-12. Look at the DBug12.h include file (on the EE 308 web page) to see which ones.

Programming the HC12 in C

• A comparison of some assembly language and C constructs

```
Assembly                                      |              C
----------------------------------------------|------------------------------------------------
; Use a name instead of a num                 |        /* Use a name instead of a num */
COUNT: EQU 5                                  |        #define COUNT 5

;---------------------------------------------|        /*----------------------------*/
;start a program                              |        /* To start a program */
        org     $1000                         |        main()
        lds     #0x3C00                       |        {
                                              |        }
;---------------------------------------------|        /*----------------------------*/
```

• Note that in C, the starting location of the program is defined when you compile the program, not in the program itself.
• Note that C always uses the stack, so C automatically loads the stack pointer for you.

```
Assembly                                      |              C
----------------------------------------------|----------------------------------------------
;allocate two bytes for                       |        /* Allocate two bytes for
;a signed number                              |        * a signed number */
                                              |
        org     $2000                         |
i:      ds.w    1                             |        int i;
j:      dc.w    $1A00                         |        int j = 0x1a00;
;---------------------------------------------|        /*----------------------------*/
;allocate two bytes for                       |        /* Allocate two bytes for
;an unsigned number                           |        * an unsigned number */
i:      ds.w    1                             |        unsigned int i;
j:      dc.w    $1A00                         |        unsigned int j = 0x1a00;
;---------------------------------------------|        /*----------------------------*/
;allocate one byte for                        |        /* Allocate one byte for
;an signed number                             |        * an signed number */
                                              |
i:      ds.b    1                             |        signed char i;
j:      dc.b    $1F                           |        signed char j = 0x1f;
```

```
                                               |
Assembly                                       |          C
;----------------------------------------------|       /*----------------------------*/
;Get a value from an address                  |       /* Get a value from an address */
; Put contents of address                      |       /* Put contents of address */
; $E000 into variable i                        |       /* 0xE000 into variable i */
                                               |
i:        ds.b    1                            |       unsigned char i;
                                               |
          ldaa    $E000                        |       i = * (unsigned char *) 0xE000;
          staa    i                            |
                                               |       /*----------------------------------*/
                                               |       /* Use a variable as a pointer
                                               |       (address) */
                                               |
                                               |       unsigned char *ptr, i;
                                               |
                                               |       ptr = (unsigned char *) 0xE000;
                                               |       i = *ptr;
                                               |       *ptr = 0x55;
                                               |
;----------------------------------------------|       /*----------------------------*/
```

• In C, the construct *(num) says to treat num as an address, and to work with the contents of that address.

• Because C does not know how many bytes from that address you want to work with, you need to tell C how many bytes you want to work with. You also have to tell C whether you want to treat the data as signed or unsigned.

– i = * (unsigned char *) 0xE000; tells C to take one byte from address 0xE000, treat it as unsigned, and store that value in variable i.

– j = * (int *) 0xE000; tells C to take two bytes from address 0xE000, treat it as signed, and store that value in variable j.

– * (char *) 0xE000 = 0xaa; tells C to write the number 0xaa to a single byte at addess 0xE000.

– * (int *) 0xE000 = 0xaa; tells C to write the number 0x00aa to two bytes starting at addess 0xE000.

```
Assembly                                     |            C
;--------------------------------------------|        /*-----------------------------*/
;To call a subroutine                        |        /* To call a function */
          ldaa      i                        |        sqrt(i);
          jsr       sqrt                     |
;--------------------------------------------|        /*-----------------------------*/
;To return from a subroutine                 |        /* To return from a function */
          ldaa      j                        |        return j;
          rts                                |
;--------------------------------------------|        /*-----------------------------*/
;Flow control                                |        /* Flow control */
          blo                                |        if (i < j)
          blt                                |        if (i < j)
                                             |
          bhs                                |        if (i >= j)
          bge                                |        if (i >= j)
;--------------------------------------------|        /*-----------------------------*/
                                             |
```

• Here is a simple program written in C and assembly. It simply divides 16 by 2. It does the division in a function.

```
Assembly                                     |            C
---------------------------------------------|-----------------------------------------------
                                             |
          org       $2000                    |        unsigned char i;
i:        ds.b      1                        |
                                             |
                                             |        unsigned char div(unsigned char j);
          org       $1000                    |        main()
          lds       #$3C00                   |        {
          ldaa      #16                      |                i = div(16);
          jsr       div                      |        }
          staa      i                        |
          swi                                |
                                             |
          div:      asra                     |        unsigned char div(unsigned char j)
          rts                                |        {
                                             |                return j >> 1;
                                             |        }
```