Lecture 5

9/16/16

A simple program

With the mov and add instructions we can we can write simple programs. The following is a simple program which adds two numbers.

```
Disassembly of section .text:

00000000 <_main>:
   0:    06 00 fa      lnk        #0x6
   2:    50 f8 2f      mov.w      #0xff85, w0
   4:    00 0f 78      mov.w      w0, [w14]
   6:    e0 02 20      mov.w      #0x2e, w0
   8:    10 07 98      mov.w      w0, [w14+2]
   a:    1e 00 90      mov.w      [w14+2], w0
   c:    1e 00 40      add.w      w0, [w14], w0
   e:    20 07 98      mov.w      w0, [w14+4]
  10:    00 00 eb      clr.w      w0
  12:    00 80 fa      ulnk
  14:    00 00 06      return
```

Here each instruction is on a line that begins with a hex number of the words the instruction is offset from the first instruction, then the 24-bit instruction written in hex, then the assembly mnemonic, and lastly any operands. For example, the 3rd line

$$\underbrace{\texttt{4:}}_{\text{relative address}} \quad \underbrace{\texttt{000f78}}_{\text{24-bit instruction}} \quad \underbrace{\texttt{mov.w}}_{\text{mnemonic}} \quad \underbrace{\texttt{w0,[w14]}}_{\text{operands}}$$

Here the lnk and ulnk commands can be ignored, they relate to the structure of the C program we used to generate this code. The return instruction should be clear, it returns program execution to wherever it originated, which we will discuss later.

Lines 2 and 3 contain two move instructions, Line 2 copies the 16-bit literal into register w0. (In fact this literal represents a negative integer in 2c representation, -123 decimal)

Line 3 then moves this to an address given by the value in w14.

If you reference the programmers you will see W14 is used to store the frame pointer, the frame pointer is the starting address for the region in data memory allocated to this program.

The next two lines do a similar operation, storing 0x2e (46 decimal in the following word of data memory

After these 4 instructions we have the following

w0 $\boxed{0x002e}$

frame pointer is in w14

	byte addresses
0x85	w14
0xff	w14+1
0x2e	w14+2
0x00	w14+3
?	w14+4
?	w14+4

frame

The add instruction now executes and adds the Two 16-bit operands

$$\text{add } \underbrace{w0, [w14]}_{\text{source registers}}, \underbrace{w0}_{\text{destination register}}$$

(w0)	0xff85	-123
([w14])	+ 0x002e	46
w0	0xffb3	-77

This instruction overwrites the contents of w0 with the result, and then moves it to the address w14 + 4, and then uses the clear word instruction to clear (set to 0x0000) w0.

Upon completion of these instructions we have

0x0000

a	0x85	w14
	0xff	w14+1
b	0x2e	w14+2
	0x00	w14+3
c	0xb3	w14+4
	0xff	w14+5

As an aside this assembly was generated by the XC16 C compiler for the C code

```
int main(void) {

    int a,b,c;

    a = -123;
    b = 46;

    c = a + b;

    return 0;
}
```

>xc16-objdump -d newmainXC16.o

```
Disassembly of section .text:

00000000 <_main>:
   0:     06 00 fa      lnk       #0x6
   2:     50 f8 2f      mov.w     #0xff85, w0
   4:     00 0f 78      mov.w     w0, [w14]
   6:     e0 02 20      mov.w     #0x2e, w0
   8:     10 07 98      mov.w     w0, [w14+2]
   a:     1e 00 90      mov.w     [w14+2], w0
   c:     1e 00 40      add.w     w0, [w14], w0
   e:     20 07 98      mov.w     w0, [w14+4]
  10:     00 00 eb      clr.w     w0
  12:     00 80 fa      ulnk
  14:     00 00 06      return
```

```
int main(void) {

    int a,b,c;

    a = -123;
    b = 46;

    c = a + b;

    return 0;
}
```

We can further investigate this program.

There are 11 instructions, each instruction is 3 bytes in size, so we expect this program to require 33 bytes or $2/3 \times 33$ words of program memory

For the PIC24FJ64GA002 program flash memory is located from 0x000200 to 0x00abfe. The XC16 compiler will place this code at 0x0002aC (additional code generated by the XC16 C compiler occupies the program memory below this, there is 258 bytes for this code)

The PIC24FJ64GA002 has Data RAM from 0x0800 to , The XC16 compiler will put W14=0x0806, place the value of a at this address, b at 0x0808, and c at 0x80a. These are local variables for this program.

How long does it take this program code to execute on a PIC24FJ64GA002 with a 16 MHz instruction clock frequency?

From the Programmer's Reference Manual we know that each instruction in this code, except RETURN, takes 1 instruction cycle to execute. The RETURN instruction normally takes 3 instruction cycles.

With the 16 MHz instruction clock frequency, $F_{cyc} = 16\ MHz$, the instruction clock period is $T_{cyc} = 1/16MHz = 62.5\ ns$.

So for the program execution time

$$(10 \times 1 + 1 \times 3) \cdot 62.5\ ns = 812.5\ ns$$