Russell Bjella
ASEN 5519 Lab 2
1 September 2016

QUESTIONS

1. MPASM directives
    a. The ‘R’ option in the list directive sets the radix to either hexadecimal, octal, or decimal. In lab2_bjella.asm, it is set to ‘DEC’ for decimal.
    b. The ‘#include’ directive reads in the specified file as source code, equivalent to if the text were inserted at the same location as the ‘#include’ directive.
    c. In the included ‘p18f87k22.inc’ file, PORTA is defined with its hexadecimal register address (0x0F80). This is a configuration file that specifies that instructions meant for PORTA go to that address.
    d. The ‘CONFIG’ directive sets the PIC18 MPU’s configuration bits, defining settings for the rest of the file.
    e. The ‘CBLOCK’ directive defines a list of sequential symbols stored in memory starting at the given address. The list ends with the ‘ENDC’ directive.
    f. The ‘ORG’ directive sets the program origin for subsequent code. It tells the MPU where to store and read program instructions.

2. The ‘CONFIG FOSC=HS1’ directive sets the PIC oscillator mode to high speed crystal resonator mode at medium power (4 MHz-16 MHz). If it were changed to ‘INITIO1’, it would switch to internal oscillator mode with Fosc/4 output on RA6 and I/O on RA7.

3. The ‘Mainline’ program is located at address 0x1C. The subroutine ‘Initial’ is located at 0xEC0.

4. The hexadecimal representation of the ‘MOVWF’ instruction on line 59 is 0x6E93, or 0110 1110 1001 0011 in binary. From the data sheet, this indicates that the OPCODE for this operation is 0110 111, the ‘a’ argument is 0, and the ‘f’ file register address is 1001 0011.

5. The hexadecimal representation of the ‘CLRF’ command on line 60 is 0x6A8A, or 0110 1010 1000 1010 in binary. From the datasheet, this indicates that the OPCODE is 0110 101, the ‘a’ argument is 0, and the file register address is 1000 1010.

6. Since the ‘ACCESS’ option is used (‘a’=0), both the ‘MOVWF’ and ‘CLRF’ commands use the access bank. Referencing the SFRs memory viewer, the address of TRISB is 0xF93 and the address of LATB is F8A.

7. The value of WREG when at 0x22 is 0x00. When I step to location 0x24, WREG changes to 0xC0. The ‘MOVLW’ command on line 58 moved the binary argument 0b11000000 into WREG. Pressing F7 again, the ‘MOVWF’ command on line 59 copied the byte that was just put in WREG into TRISB. Now WREG and TRISB both show a value of 0xC0.

8. When at program memory address 0x1E, WREG and TRISB both have a value of 0xC0, and LATB has a value of 0x00. When I press F7, LATB increments to 0x01. Pressing it again, it goes back to

0x00. Every time the ‘BTG’ command executes, it toggles the value of LATB between 0 and 1. This is because ‘BTG’ is the bit toggle command.

9. When the simulation is reset, the values in WREG and LATB don’t change (0xC0 and 0x01, respectively), but the value of TRISB changes from 0xC0 to 0xFF. This means that all 8 pins on PORTB are configured as inputs.

10. Prior to the ‘CLRF’ command in the ‘Initial’ subroutine, the STATUS register has a value of 0x00. When I step in the program, it changes to 0x04, or ‘0000 0100’ in binary. Because the result of the ‘CLRF’ command was all zeros, the ZERO flag (B2) in the status register was set to logic high.

11. Stepping the debugger again and executing the next ‘MOVLW’ command, the STATUS register value changes back to 0x00. This is because the result of the operation of putting the value 0x01 into WREG was nonzero, didn’t overflow, wasn’t negative, and didn’t require a carry, so none of the STATUS flags got set to 1.

12. Stopped at the ‘INCF’ command, the STATUS register shows a value of 0x04. Stepping with F7, the value of the STATUS register goes to 0x00. This is because the result of incrementing WREG is nonzero, non-negative, doesn’t overflow, and didn’t require a carry.

13. After pressing F5 15 more times, WREG has incremented to 0x10 and the STATUS register value changed to 0x02. This means that the Digit Carry (DC) bit is set, caused by bit B3 generating a carry. The carry was generated because the binary value of WREG changed from 0b1111 to 0b10000, requiring an additional hexadecimal digit. Hitting F5 again, the STATUS register returned to 0x00 because the DC flag wasn’t set as a result of incrementing WREG again.

14. Every time the upper nibble of WREG is incremented (every 16 loops), the STATUS register goes to 0x02 due to bit B3 generating a carry, as explained above.

15. With WREG set to 0x7F and the STATUS register set to 0x00, pressing F5 increments WREG to 0x80 and the STATUS register goes to 0x1A, corresponding to binary 0b00011010. This means that the SIGN, OVERFLOW, and DC status flags were set by incrementing WREG again. This is because incrementing WREG exceeded the 8 bit signed integer limit, and since the value is signed the rollover resulted in a negative number. The DC flag was also set because bit B3 again generated a carry.

16. Pressing F5 again, the value in WREG (0x81) is still negative so the SIGN flag is set in the STATUS register, but there was no overflow, no carry generated by bit B3, and the result is nonzero so the full value of the STATUS register is 0x10.

17. When WREG is 0xFF and STATUS is 0x10 and F5 is pressed, WREG goes to 0x00 and STATUS goes to 0x07, meaning the CARRY, DC, and ZERO flags are all set because the previous operation resulted in a zero answer and required a borrow from the most significant bit. In addition, bit B3 generated a carry so the DC flag was set.

18. After executing the ‘INCF’ command and waiting at the new ‘ADDWF’ command, WREG is 0x02. Pressing F7, WREG changes to 0x04. This is because the ‘ADDWF’ command added the value of WREG to itself.

19. Pressing F5 to loop back to the ‘INCF’ command, WREG is still 0x04. Pressing F7 increments it to 0x05. The next line, ‘ADDWF’, adds the value in WREG to itself (multiplying by two) resulting in 0x0A. Finally, the ‘DAW’ command converts the value of WREG to decimal, that is, 0x0A=10, represented (incorrectly) as 0x10 in the watch menu.

20. The ‘NEGF’ command changed WREG from 0x04 to 0xFC and the STATUS register from 0x00 to 0x10 (meaning the SIGN flag is set, indicating that the result of the previous operation is negative). The ‘NEGF’ command “negated” WREG, which means it took the two’s complement of it.

21. Executing the ‘RLNCF’ command changed WREG from 0xFC to 0xF9, and the STATUS register remained at 0x10 because the result is negative. ‘RLNCF’ rotates all the bits in WREG to the left without carrying: so bit 0 takes the original value of bit 7, bit 1 takes the value of bit 0, bit 2 takes the value of bit 1, etc.

22. Replacing the ‘RLNCF’ command with ‘RLCF’ and running through it again, the value of WREG changed from 0xFC to 0xF8. In addition, the STATUS register now has the value of 0x11, meaning that both the SIGN and CARRY flags are set. ‘RLCF’ did the same thing as ‘RLNCF’, except bit 0 didn’t take the value of bit 7: it simply went to zero.

23. Adding the command ‘movff WREG,count’ copied the value in WREG to the variable ‘count’, provided the variable was properly initialized in the CBLOCK section.

24. From the disassembly listing, the ‘MOVFF’ command is copying the data in WREG to the RAM memory location 0x0.

25. From the Program Memory View, the hex representation of the opcode and operand for the command ‘NEGF WREG’ is 0x6CE8.

26. From the Program Memory View, the hex representation of the opcode and operand for the command ‘BTG LATB,0’ is 0x708A.

Computational Questions

4519

1) -35

35 = 0010 0011
invert 1101 1100
add one
$\boxed{-35 = \text{0b1101 1101}}$

-28

28 = 0001 1100
invert 1110 0011
add one
$\boxed{-28 = \text{0b1110 0100}}$

2) a) 35 + -10

35 = 0010 0011
10 = 0000 1010 → -10 = 1111 0110

```
  0010 0011
+ 1111 0110
  0001 1001
```

↳ 35 + -10 = 25

b) 100 + -84

100 = 0110 0100
84 = 0101 0100 → -84 = 1010 1100

```
  0110 0100
+ 1010 1100
  0001 0000
```

100 + -84 = 16

Gaonkar

2.12) The BSR is a 4-bit register used for direct addressing of data memory.

3.5) MOVWF 0x7F, 1 with BSR = 0x04 will move the contents of WREG to the file register at address 0x47F.

3.10) MOVLW 0xFA
WREG = 0xFA
N = OV = Z = C = 0

ADDLW 0x38
WREG = 0xFA + 0x38 = 0x32
N = OV = Z = 0, C = 1

4.11) a) WREG = 0x9A

b) Would set SIGN and OVERFLOW Flags

c) STATUS = 0x18

4.13) 0xA7 + 0x92 ~~= 0xA7 + 0x58 + 0x~~

~~= 0x3 0 9~~

~~0xA7 = 1010 0111~~ ~~0x92 = 1001 0010~~
~~= 167~~ = 146

167
+146
3

~~343~~

~~167~~
~~+92~~
1 3 9

0xA7
+0x92
0x139 = $\boxed{0x39 = WREG}$

$\boxed{\text{Will set the OV and C Flags}}$

4.1SE) a) 0x34 + 0x56 = $\boxed{0x8A}$

b) STATUS = 0x18 (OV and N Flags set)

OV Flag because result ~~falls outside of range 0x00-0x7F range 0x80→0x?? (-128→?) and range 0x00-0x7F~~ of adding two positive numbers (MSB=0) was ~~[illegible]~~ negative (MSB=1, also set N Flag).

c) REG10 is cleared: 0b0000 0000

d) ✓

4.2SE) a) 0x35 + 0x2A = $\boxed{0x5F}$

b) STATUS = 0x0

c) REG10 = 0x5F

d) ✓

5.9) 0x3F + 0x47 → REG3

```
BYTE1    EQU    0x3F
BYTE2    EQU    0x56
REG3     EQU    0x03
MOVLW    BYTE1
ADDLW    BYTE2
MOVWF    REG3,0
```

Z Flag <u>not</u> set because result is nonzero

C Flag <u>not</u> set because MSB did not create a carry

5519 | 1) a) −437   12 bit +437 = ~~0b0000 110~~
0b0001 1011 0101

$\boxed{-437 = 1110\ 0100\ 1011}$

−299   12 bit +~~299~~ = 0b0001 0010 1011

$\boxed{-299 = 0b1110\ 1101\ 0111}$

2) a) (−34) + (−27)

34 = 0010 0010
27 = 0001 1011
−34 = 1101 1110
−27 = ~~1110 0101~~ 1110 010

```
  1 1 1 1   1
  1101 1110
+ 1110 0101
-----------
1 1100 0011  → ~~...~~
```

reverse 2's comp = 0011 1101 = 61

$\boxed{= -61}$

b) −76 + 23

76 = 0100 1100
23 = 0001 0111
−76 = 1011 0100

```
   1 1    1
  1011 0100
+ 0001 0111
-----------
  1100 1011  → reverse 2's comp → 0011 0101
```

$\boxed{= -53}$

Gaonkar | 3.20) ~~Branching requires the processor to first check if the condition is met~~ ~~then~~ ~~and if so, change the PC~~
The two cycle branch commands takes two cycles because it takes a 12 bit signed number which is too large for one cycle.

4.4SE) a) 0x 47 + 0x 88 = $\boxed{\text{0x CF}}$

b) STATUS = 0x0

c) REG10 = 0xCF

d) ✓

4.5SE) 0x 3$\overset{1}{2}$67 ~~+ 0x 42F2~~

$$\begin{array}{r} \text{0x3267} \\ +\ \text{0x42F2} \\ \hline \boxed{\text{0x7559}} \end{array}$$

5.10) 0x3F + 0x47 = 0x86

N flag set because MSB=1 (two's complement, it's < 0)

OV flag set because 0x3F and 0x47 are both positive but the result was negative