

ECE 3724 Test #1 – Summer 2005/Reese -- there 5 pages (3 pages front/back)

Student ID: \_\_\_\_\_ (no names please)

You may NOT use a calculator. You may use only the provided reference materials. For problems in part II, assume initial memory contents as shown below at the start of EACH instruction. Explain what register/memory location is modified, and give the FINAL HEX value of the modified register/memory location, and the final status of the C, Z flags. Recall that the 'd' bit in a machine word indicating a destination is '0' if the destination is W, is '1' if the destination is the file register. For the 'a' bit, use the assumptions we have used in class ('a' bit is '0' if address in ACCESS RAM, 'a' bit is '1' if not in ACCESS RAM).

Part I: (20 pts)

a. Name the register on the PIC18 that holds the address of the instruction being fetched from memory.

b. Give the machine code in **HEX** for the instruction, use the assumptions for the access bit value ('a' bit) discussed in class:

```
XORWF 0x23A, f
```

c. The machine code 0x0FEA represents instruction? (*if necessary*, use 'w' or 'f' for the destination, and 'ACCESS' or BANKED to represent the value of the *a* bit)

d. For a 40 MHz clock, how long does it take to execute the following instructions? (give the answer in **nanoseconds**)

```
bcf 0xF80,2  
goto 0x01030
```

e. After power up, what is the location of the first instruction fetched from memory?

Location	Contents
0x029	0xA2
0x02A	0x80
0x02B	0x4F
0x02C	0x01

Assume the W register has the value 0x5E in it, and that initial values of C, Z are both '0'.

Part II. (35 pts) Assume the above memory contents, W register value, initial C,Z values at the START of each instruction.

a. `btg 0x02A, 7`

Circle one:    W dest.        Reg. file dest.  
 New value (hex) \_\_\_\_ C\_flag : \_\_\_\_, Z flag: \_\_

b. `subwf 0x029, f`

Circle one:    W dest.        Reg. file dest.  
 New value (hex) \_\_\_\_ C\_flag : \_\_\_\_, Z flag: \_\_

c. `rrcf 0x02C, w`

Circle one:    W dest.        Reg. file dest.  
 New value (hex) \_\_\_\_ C\_flag : \_\_\_\_, Z flag: \_\_

d. `andlw 0x2A`

Circle one:    W dest.        Reg. file dest.  
 New value (hex) \_\_\_\_ C\_flag : \_\_\_\_, Z flag: \_\_

e. `clrf 0x29, f`

Circle one:    W dest.        Reg. file dest.  
 New value (hex) \_\_\_\_ C\_flag : \_\_\_\_, Z flag: \_\_

(45 pts) PART III. Convert the following C code fragments to PIC18 assembly.

UNLESS otherwise stated in a particular problem, assume all variables are in locations 0x000 to 0x07F.

If you use a temporary memory location, use temp and assume it is in bank 0. When writing code, you **must use** symbolic names for variable names, register names, and bit names for (i.e, use: bsf STATUS, C instead of bsf 0xFD8, 0x0). You do not have to show the CBLOCK declaration for variables.

Hint: A common mistake in these problems is to write code that modifies variables to the right of the '=' sign (i.e, for 'a = b - c;' the code you write somehow modifies *b*, or *c*, as well as *a*). This is incorrect; make sure that your code only modifies variables to the left of the '=' sign.

Also, recall that 'k++' is the same as 'k=k+1;', 'j- -' is the same as 'j = j - 1', that "i = j" is true if *i* is equal to *j*, that "i != j" is true if *i* is not equal to *j*, "<<" is a left shift, ">>" is a right shift, '|' is bitwise logical OR, '&' is a bitwise logic AND, '^' is a bitwise logical XOR.

unsigned char i,j,k,p,q,r,s,t;

a. (7 pts)

```
k = (i + 10) << 2;
```

b. (9 pts)

```
if ( (i == 0) || (j != 0) ) {  
    //if-body – just write a placeholder here  
}
```

c. (6 pts) Write the following in assembly language  
 $i = k + j - p$  ;

d. (7 pts)

```
while ( p > k) {  
    //loop-body – just write a placeholder here  
}
```

- e. (6 pts) Assume that  $r$  is in bank 0,  $s$  is in bank 1, and  $t$  is in bank 2. Implement the following code in assembly language:

```
t = (r & s) - 1;
```

- f. (10 pts) Write a PIC18 instruction sequence that does

```
do {  
    //loop body, place holder  
} while ( (k >= 10) && (p != q));
```