Circle one: JONES section / REESE section

ECE 3724 Test #1 – Fall 2005 – Jones/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. What is the maximum size of PIC18 data memory in bytes given that a 12-bit address is used to address it? (give your answer in Kbytes, ie, 1 Kbytes, 2 Kbytes, 4 Kbytes, 8 Kbytes, etc).

b. What data memory locations comprise the ACCESS bank in the PIC18 architecture? (circle one)

1. 0x000 – 0x0FF
2. 0x000 – 0x07F and 0xF00 - 0xF7F
3. 0x080 – 0x0FF and 0xF80 – 0xFFF
4. 0x000 – 0x07F and 0xF80 – 0xFFF
5. 0xF00 – 0xFFF

c. What is the distinguishing feature between a Finite State Machine approach to implementing a digital system and a stored program machine approach?

d. How many instruction cycles does it take to execute the following instructions? How many clock cycles? With a 20 MHz clock, how long does it take to execute the following instructions? (give the answer in **nanoseconds**). For reference, 1 MHz has a period = 1 us = 1000 ns.

```
incf     0x045,f
movff  0x1F0, 0x2A0
goto    0x01030
```

e. The Number Sequencing Computer in Lab #2 had a 16 x 6 memory for a maximum program size of 16 instructions. You then modified it to have a maximum program size of 32 instructions. What would the memory size be (stated as K x N), if the maximum program size is increased to 128 instructions?

Location	Contents
0x048	0x01
0x049	0xFB
0x04A	0x90
0x04B	0xFF

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. incf 0x04B, f

Circle one: W dest. Reg. file dest.

New value (hex) ____ C_flag :____ , Z flag:__

b. subwf 0x04A, w

Circle one: W dest. Reg. file dest.

New value (hex) ____ C_flag :____ , Z flag:__

c. xorwf 0x04B, f

Circle one: W dest. Reg. file dest.

New value (hex) ____ C_flag :____ , Z flag:__

d. bsf 0x48,7

Circle one: W dest. Reg. file dest.

New value (hex) ____ C_flag :____ , Z flag:__

e. addlw 0x48

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 in 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 =  (j >> 2) + i;
```

b. (9 pts)

```
if ( (i != 0) && (j == 5) {
     //if-body – just write a placeholder here
} else {
   //else-body – just write a placeholder here
 }
```

c. (6 pts) Write the following in assembly language

```
i =  (k ^ j)  | 0x80;
```

d. (10 pts)

```
while  (  (i > 0x20) || (j == k) )
    //loop-body – just write a placeholder here
};
```

e. (6 pts) Assume that ***r*** is in bank 1, ***s*** is in bank 2, and ***t*** is in bank 3. Implement the following code in assembly language:

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

f. (7 pts) Write a PIC18 instruction sequence that does

```
do {
  //loop body, place holder
} while ( k < q);
```