

Circle one:      JONES section      /      REESE section

ECE 3724 Test #1 – Fall 2006 – Jones/Reese -- there 5 pages (3 pages front/back)

Net 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. Give the machine code in HEX for the instruction *incf 0x3AF,w*

f = 0xAF. The BSR is assume to hold 3. d = 0 (store in the W register) and a = 1 (banked). From the data sheet, *incf* is 0010 10da ffff ffff, which becomes 0100 1001 1010 1111. In hex, this is **0x29AF**.

b. Circle the statement that is a true statement about a 1K x 8 memory

1. Has 8 address lines and 1K locations
2. Has 10 address lines and 8 bits per location
3. Has 3 address lines and 10 bits per location
4. Has 8 address lines and 1 bit per location
5. Has 1K locations and 10 bits per location

c. The machine code 0x8A57 represents instruction? (use 'w' or 'f' for the destination, and 'ACCESS' or BANKED to represent the value of the a bit). You need to give the complete instruction including all of the operands.

From the data sheet, *bsf* = 1000 bbba ffff ffff. In binary, the given instruction is 1000 1010 0101 0111. Therefore, bbb = 101, a = 1, and ffff ffff = 0101 0111. From bbb = 101 we know that bit 5 will be set. Since a = 1, we are in access RAM. The ffff ffff gives a file register of 0x57. Putting this together, we have: **bsf 0x5F, 5, ACCESS**

d. How many instruction cycles does it take to execute the following instructions? How many clock cycles? With a 25 MHz clock, how long does it take to execute the following instructions? (give the answer in **microseconds, do NOT round to the nearest microsecond!**). For reference, 1 MHz has a period = 1 us = 1000 ns.

```
goto next_line
next_line movff 0x3A0, 0x020
          movwf 0x020,w
```

Referring to the data sheet, *goto* takes two cycles, *movff* takes 2 cycles, and *movwf* takes 1 cycle. The total number of instruction cycles is therefore 2 + 2 + 1 = 5 instruction cycles. Each

instruction cycle takes 4 clock cycles. Therefore, the time is  
5 instruction cycles  $\cdot \frac{4 \text{ clock cycles}}{1 \text{ instruction cycle}} \cdot \frac{1 \mu\text{s}}{25 \text{ clock cycles}} = \mathbf{0.8 \mu\text{s}}$ .

e. Circle the statement that is a true statement about finite-state machine:

1. Executes slower than a comparable microprocessor
2. Stores instructions in D flip-flops
3. Is easier to reprogram than a comparable microprocessor
4. Uses D flip-flops to store the program counter
5. Uses fewer logic gates than a comparable microprocessor

Location	Contents
0x071	0x3A
0x072	0x10
0x073	0xA5
0x074	0x01

Assume the W register has the value 0x2C 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. xorlw 0x35

Circle one:  W dest.     Reg. file dest.  
 New value (hex) 0x19    C\_flag : 0 , Z flag: 0

b. subwf 0x071, w

Circle one:  W dest.     Reg. file dest.  
 New value (hex) 0x0E    C\_flag : 1 , Z flag: 0

c. movf 0x073, f

Circle one:     W dest.     Reg. file dest.  
 New value (hex) 0xA5    C\_flag : 0 , Z flag: 0

d. btg 0x072, 4

Circle one:     W dest.     Reg. file dest.  
 New value (hex) 0x00    C\_flag : 0 , Z flag: 0

e. rrcf 0x073,w

Circle one:  W dest.     Reg. file dest.  
 New value (hex) 0x52    C\_flag : 1 , Z flag: 0

(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.

**You may use temporary variables named tmp1, tmp2, tmp3, ... if necessary.**

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

a. (7 pts)

```
t = q & (p >> 2);
```

```
bcf    STATUS, C  
rrcf   p, w  
bcf    STATUS, C  
rrcf   WREG, w  
andwf  q, w  
movwft
```

b. (8 pts)

```
if (5 >= t) {  
    //if-body – just write a placeholder here  
} else {  
    //else-body – just write a placeholder here  
}
```

```
movf   t, w    ; w = t  
sublw  5       ; do 5 – t (this instruction does literal – w!!!  
bnc    if_else ; If no carry, 5 < t so do else  
; if body code here – 5 >= t  
bra    end_if  ; After if body, skip to end of if statement
```

```
if_else:
```

```
; else body code here
```

```
end_if:
```

- c. (7 pts) Write the following in assembly language  
 $j = (p + 5) | t ;$

```
movf  p,w      ;w = p
addlw 5        ;w = p+5
iorwf t,w      ;w = w | t
movwf j        ;j = w
```

- d. (8 pts)

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

```
loop_top
    movf  s,w      ;test s
    bnz   loop_body ;do loop body if non-zero
    movf  k,w      ;w=k
    subwf j,w      ;j-k
    bc    loop_end ;skip if k>= j
loop_body
    ....loop_body...
    bra   loop_top
loop_end
    ....rest of code....
```



- e. (7 pts) Assume that  $r$  is in bank 2,  $s$  is in bank 4, and  $t$  is in bank 7. Implement the following code in assembly language:

```

r = s - (t << 1);

    movlb    7
    bcf     STATUS,C
    rlc     t,w      ;w = t<< 1
    movlb    4
    subwf   s,w      ;w = s - (t<<1)
    movlb    2
    movwf   r

```

- f. (8 pts) Write a PIC18 instruction sequence that does the following.

```

do {
//loop-body – just write a placeholder here
} while ((k >= j) && (p != q))

```

```

loop_top
    ....loop_body...
    movf   j,w      ;w=j
    subwf  k,w      ;k-j
    bnc    loop_exit ;exit if j > k
    movf   q,w      ;w = q
    subwf  p,w      ; p-q
    bnz    loop_top ;back to loop top if p != q
loop_exit
    ....rest of code....

```