ECE 3724/CS 3124 Test #1 Solution – Summer 2004/Reese -there 4 pages (2 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. The PIC18 program counter has 21 bits. Based on this, what is the maximum number of program words that any PIC18 processor can reference (if your answer is > 512, then use 1K, 2K, 4K, 8K, 16K...1M, 2M, 4M, 8M, 16M)etc. as an answer, where 1K = 1024 = 2^{10} , 1M = 2^{20}).

$2^{21} = 2^{20} * 2^{1}$ = 1M * 2 = 2 M (bytes) or 1 M instruction words

b. Give the machine code as a **4-digit hex value** for the instruction:
andwf 0x2AC,f

0001 01da ffff ffff → d=1 since destination is f, a = 1 as 0x2AC not in access ram, so
0001 0111 1010 1100 → 0x 17AC

c. The machine code 0x5F59 represents instruction? (use 'w' or 'f' for the destination, and 'ACCESS' or BANKED to represent the value of the *a* bit)

0x5F59 → 0101 1111 0101 1001 , first six bits indicate the subwf instruction
subwf 0101 11da ffff ffff, → subwf 0x59, f, BANKED

d. What is ACCESS RAM on the PIC18?

1. Data memory locations 0x100-0x17F, 0xF80-0xFFF.
2. Data memory locations 0x000-0x07F, 0x180-0x1FF.
3. Data memory locations 0x100-0x07F, 0x200-0x27F.
4. Data memory locations 0x080-0x0FF, 0x100-0x1FF
5. Data memory locations 0x000-0x07F, 0xF80-0xFFF.
6. Data memory locations 0xF00-0xF80, 0xF80-0xFFF

e. What is function of the BSR on the PIC18?

1. Specifies the address of the next instruction
2. Always holds one of the operands for an ALU operation
3. Specifies the upper 4-bits of a 12-bit data memory address.
4. Holds the current instruction that is being executed.
5. Boolean operation selection register.
6. Contains the 4-bit shift value for all shift operations.

Location	Contents
0x038	0xA9
0x039	0x00
0x03A	0x1F
0x03B	0x27

Assume the W register has the value 0x48 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. rrcf 0x038, w

```
[0x38]  = 0xA9=  1010 1001  (→ right shift)
new W   =        0101 0100 = 0x54,
LSB goes into C-flag, so C=1
```

Circle one: (W dest.) Reg. file dest.

New value (hex) __0x54__ C_flag :_1___ , Z flag:_0_

b. incf 0x03A, f

```
[0x3A] = [0x3a] + 1
[0x3A] = 0x1F + 1 = 0x20
```

Circle one: W dest. (Reg. file dest.)

New value (hex) _0x20__ C_flag :__0__ , Z flag:_0__

c. iorwf 0x03B, f

```
W       =        0100 1000
[0x3B]  = 0x27 = 0010 0111 =  (OR op)
[0x3B](new)      0110 1111
```

Circle one: W dest. (Reg. file dest.)

New value (hex) _0x6F__ C_flag :_0___ , Z flag:_0__

d. movff 0x039,0x03B

```
Copies contents of 0x039 to 0x03B
New value of location 0x03B is 0x00
The movff instruction affects NO FLAGS!!!!!
so Z flag is still a 0 despite the fact the
destination location now has 0 in it.
```

Circle one: W dest. (Reg. file dest.)

New value (hex) _0x00___ C_flag :_0__ , Z flag:_0__

e. subwf 0x03A, w

```
W = [0x3A]-W = 0x1F – 0x48
LSD = 0xF-0x8 = 15 -8 = 0x7
MSD = 0x1 – 0x4 = (16 +1)-4 = 17-4 = 13 = 0xD
    Cflag = 0 as a borrow was required out of
the MSD.
```

Circle one: (W dest.) Reg. file dest.

New value (hex) __0xD7_ C_flag :_0____ , Z flag:_0__

BE CAREFUL!!! Variables are in different banks!!!!

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

Variable locations are: *i* is data location 0x100, *j* is data location 0x130, *k* is data location 0x2A0. Assume the BSR has a 0x0 value in it initially.

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;

a. (10 pts)

```
i = j << 2;
```

Common mistakes:
rlcf j, f This changes j!!!!!

Also, forget to clear carry flag.

```
movff j, i      ; copy i to j, as i is changed
movlb 1         ;select bank 1
bcf STATUS,C    ; clear carry
rlcf i, f       ; i = i << 1
bcf STATUS,C
rlcf i, f       ; i = i << 1
```

b. (10 pts)

```
while ( k >= j) {
    k = k + i;
}
```

For k >= j, must do k-j. If k >= j is true, then k – j produces no borrow, which is C = 1.

```
top
  movlb 1       ;bank1
  movf  j,w     ; w = j
  movlb 2       ;bank 2
  subwf k,w     ; k - j
  bnc  while_end ; skip if C=0, borrow
  movlb 1       ;bank 1
  movf  i,w     ; w = i
  movlb 2       ;bank 2
  addwf k,f     ; k = k + i
  bra top
while_end
  ...rest of code..
```

execute if C=1

c. (10 pts)

k = j - 0xA0 ;

```
movlw 0xA0    ; w = 0xA0
movlb 1          ;select bank 1
subwf j,w     ; w = j – 0xA0
movlb 2       ;bank 2
movwf k       ;save answer in k
```

You may be tempted to use the 'sublw' instruction, which is 'subtract W from literal'. This is fairly useless, will do '0xA0 – j'.

d. (15 pts)

```
if (k > i) {
  i++;
} else {
   j = k – i;
}
```

For k > i, must do i -k. If k > i is true, then i - k produces borrow, which is C = 0.

```
top
  movlb 2       ;bank 2
  movf  k,w     ; w = k
  movlb 1       ;bank 1
  subwf i,w     ;i - k
  bc   do_else  ; do_else if C=1, no borrow
  incf i,f      ;i++, BSR already 1
  bra  end_if
do_else
  movf  i,w     ;w = i
  movlb 2       ;bank 2
  subwf k,w     ;w = k –i
  movlb 1       ;bank 1
  movwf j       ; save result in j
end_if
  ...rest of code..
```

do if C=0, borrow.

do if C=1, no borrow.