

Circle one: JONES section / REESE section

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

Part I: (20 pts)

a. Give the machine code in HEX for the instruction *xorwf 0x3AF, f*

XORWF 0001 10da ffff ffff d = 1 because destination is 'f'
a = 1 because 0x3AF is not in access bank!
XORWF → 0001 1011 1010 1111 → 0x1BAF (note: ffff ffff is lower 8 bits of 0x3AF!!)

b. Circle the statement that is a true statement about a 4K x 16 memory

- 1. Has 16 address lines and 4K locations
 - 2. Has 12 address lines and 4 bits per location
 - 3. Has 4 K address lines and 16 bits per location
 - 4. Has 12 address lines and 16 bits per location
 - 5. Has 4 K locations and 12 bits per location
- 4K is number of locations**
 $4K = 4 * 1024 = 2^2 * 2^{10} = 2^{12}$, so
need 12 address lines.
16 is number of bits per location.

c. The machine code 0x313A 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.

0x313A → 0011 0001 0011 1010 , first size bits indicate the RRCF instruction
RRCF 0011 00da ffff ffff, → RRCF 0x3A,w, BANKED (a =1)

d. How many instruction cycles does it take to execute the following instructions? How many clock cycles? With a 5 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.

movff 0x1F0, 0x2A0
 addwf 0x3A0, f
 decf 0x020, w

movff = 2 inst cycle = 8 clocks
addwf = 1 inst cycle = 4 clocks
dec = 1 inst cycle = 4 clock
total 4 inst cycles = 16 clocks
clock period = 1/5MHz = 1/(5x10⁶) = 0.2 x 10⁻⁶ = 0.2 μs
Total time = 16 * 0.2 μs = 3.2 μs

e. Circle the statement that is a true statement about a *Program Counter* inside of a computer:

- 1. Counts the number of instructions in a program
- 2. Is connected to the data lines of Program Memory
- 3. Is connected to the address lines of Program Memory
- 4. Implements the counting function for a program
- 5. Is connected to the inputs of the ALU

Program counter contains the address of the instruction to be fetched from program memory, so connected to address lines.

Location	Contents
0x05A	0x02
0x05B	0xF0
0x05C	0x90
0x05D	0x00

Assume the W register has the value 0xA4 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. `addlw 0x5C`

literal = 0x5C (1st digit is 12+4=16, so 0).
W = + 0xA4 (2nd digit is 5+10+1 (carry)=16,0
new W = 0x00
C = 1 because of carry out of 2nd digit, Z=1 because 8-bit value is 0x00.

Circle one: W dest. Reg. file dest.

New value (hex) _0x00_ C_flag : _1_, Z flag: _1_

b. `subwf 0x05A, w`

[0x5A] = 0x02
W = - 0xA4
new [0x5A] = 0x5E
C = 0 because of borrow out of MSB

Circle one: W dest. Reg. file dest.

New value (hex) _0x5E_ C_flag : _0_, Z flag: _0_

c. `andwf 0x05B, f`

[0x05B] = 1111 0000 AND operation
W = 0xA4 = 1010 0100
new W = 1010 0000 = 0xA0,

Circle one: W dest. Reg. file dest.

New value (hex) _0xA0_ C_flag : _0_, Z flag: _0_

d. `btg 0x05B,4`

7654 3210
[0x05B] = 0xF0 = 1111 0000 (toggle bit 4)
new value [0x05B]=1110 0000 = 0xE0

Circle one: W dest. Reg. file dest.

New value (hex) _0xE0_ C_flag : _0_, Z flag: _0_

e. `rlcf 0x05C, w`

[0x05C] = 0x90 = 1001 0000 (← left shift)
new [0x05C] = 0010 0000 = 0x20,
MSb goes into C-flag, so C=1

Circle one: W dest. Reg. file dest.

New value (hex) _0x20_ 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.

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

- a. (7 pts)
k = i - (k << 1);

```
; one solution
movf  k,w      ; w = j
bcf   STATUS,C ; C_flag=0
rlcf  WREG,w   ; w = w << 1
subwf i, w     ; w = i - w
movwf k       ; k = w
```

- b. (9 pts)
if ((i == 0) || (j != k)) {
 //if-body – just write a placeholder here
} else {
 //else-body – just write a placeholder here
}

```
;;; OR condition, can execute if body if one test is true
movf  i,f      ; i = i, test i
bz    if_body  ;if zero, test true, execute if body
movf  j,w      ; w = j
subwf k,w      ; does k - j
bz    else_body ; test false, do else_body
if_body
...some code   ;only reach here if both tests are true
...some code
bra  end_if    ;DO NOT FORGET to skip else_body!!
else_body
...some code
...some code

end_if
..rest of code
```

- c. (6 pts) Write the following in assembly language
 $i = (k \& 0xF0) - 1$;

```

; one solution
movf  k,w      ;w = k
andlw 0xF0     ;w = k & 0xF0
decf  WREG,w   ;w = w - 1
movwf i       ;i = w

```

- d. (10 pts)

```

do {
//loop-body – just write a placeholder here
}
while ( ( i <= k || j ) )

```

For $k \geq i$, do “ $k - i$ ”.

	Operation	True case	False Case
$k \geq i$	$k - i$	no Borrow, $C = 1$	borrow, $C = 0$

Use the TRUE case because of logical OR (\parallel) – if one of the tests is TRUE, then can branch back up to the top of the loop

```

; one solution
loop_top
...loop body...
...loop body...
movf  i,w
subwf k,w      ; w = k-i
bc    loop_top ; back to top if k >= i
movf  j,f      ; test j
bnz   loop_top ; back to top if j is not zero
loop_exit
....rest of code....

```

- 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 >> 1) + s;
```

```
; one solution
movlb 1          ; BSR = 1
movf   r,w       ; w = r
bcf    STATUS,C
rrcf   r,w       ; w = w >> 1
movlb 2          ; BSR = 2
addwf  s,w       ; w = w + 2
movlb 3          ; BSR = 3
movwf  t         ; t = w
```

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

```
while (k > (j+5)) {
    //loop body, place holder
}
```

For $k > (j+5)$, do “ $(j+5) - k$ ”.

	Operation	True case	False Case
$k > (j+5)$	$(j+5) - k$	Borrow, $C = 0$	No borrow, $C = 1$

Use the FALSE case so can skip loop body.

```
; most straight forward solution is to use a temp location
loop_top
movf   j,w       ;w = j
addlw  0x05      ;w = j + 5
movwf  temp      ;w = temp
movf   k,w       ;w = k
subwf  temp,w    ; w = temp - k
bc     end_loop  ; if C=1, no borrow, k <= j+5, skip loop
...loop body...
...loop body...
bra    loop_top  ; DO NOT FORGET to loop back to top!
end_loop
....rest of code....
```

For $k > (j+5)$, could do " $k - (j+5)$ " if you did not want to use a temp location. But have to check both C, Z flags

	Operation	True case	False Case
$k > (j+5)$	$k - (j+5)$	(no Borrow) $C = 0$ and $Z=0$	(borrow) $C = 0$ or $Z=1$

Use the FALSE case so can skip loop body.

```
    ; do k - (j-5), test both C and Z flags
loop_top
  movf    j,w          ;w = j
  addlw   0x05         ;w = j + 5
  subwf   k,w          ; w = temp - k
  bz      end_loop     ; if Z=1, they are k == j+5, skip loop
  bnc     end_loop     ; if C=0, borrow, k < j+5, skip loop
  ...loop body...
  ...loop body...
  bra    loop_top      ; DO NOT FORGET to loop back to top!
end_loop
  ....rest of code....
```

If you tried to do $(j+5) - k$ as $j + (5-k)$ this could work, as long as you used an ADD WITH CARRY so that the carry flag from the 5-K operation affected the add operation.