Last 128 locations of bank 15 are the special function registers.

ECE 3724 Test #1 – Spring 2005/Reese Solution

Part I: (20 pts)

a. The PIC18 has something called the ACCESS bank. These locations are:

1. The locations in bank 0 and the locations of bank 15.
2. The last 128 locations of bank 0, and last 128 locations of bank 15.
3. ~~The first 128 locations of bank 0, and the first 128 locations of bank 15.~~
4. The first 128 locations of bank0, and the last 128 locations of bank 15.
5. ~~The last 128 locations of bank0, and the first 128 locations of bank 15.~~
6. Only the locations of bank 0.

b. Give the machine code in **HEX** for the instruction:

movff 0x150, 0x2A3

Requires two instruction words:
0xC150
0xF2A3

c. The machine code 0x9745 represents instruction? (use 'w' or 'f' for the destination, and 'ACCESS' or BANKED to represent the value of the *a* bit)

0x9745 → 1001 0111 0100 0101 , first FOUR bits indicate the bcf instruction
bcf 1001 bbba ffff ffff, → bcf 0x45,3, BANKED (a =1)

d. For a 25 MHz clock, how long does it take to execute the following instructions? (give the answer in microseconds)

movf 0x013,w
addwf 0x020,f

The movf and addwf each require 1 instruction cycle, so:
clocks = 2 instruction cycles * 4 clocks per instruction cycle
= 8 clocks.
1/25MHz = 1/(25e6) = 0.04 e –6 = 0.04 us (microseconds).
0.04 us * 8 clocks = 0.32 us (microseconds)

e. Circle the true statement (non-volatile means contents are retained when power is removed; volatile means contents are lost when power is removed).

1. On the PIC18, program memory is volatile and data memory(file registers) is volatile.
2. On the PIC18, program memory is non-volatile and data memory(file registers) is volatile
3. On the PIC18, program memory is volatile and data memory(file registers) is non-volatile.
4. On the PIC18, program memory is non-volatile and data memory(file registers) is non-volatile.

data registers are where temporary values are stored so volatile, program memory is non-volatile so that when power is removed, the program memory contents will be retained. When the PIC powers up, it needs something to execute, the program instructions are retained by the non-volatile program memory.

Location	Contents
0x048	0xAF
0x049	0x00
0x04A	0x4F
0x04B	0x1D

Assume the W register has the value 0xE3 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. bcf 0x04A, 3

```
                 7654 3210
[0x4A]  = 0x4F=  0100 1111  (clear bit 3)
new value [0x4A]=0100 0111 = 0x47,
```

Circle one: W dest. Reg. file dest.

New value (hex) 0x47__ C_flag :_0__ , Z flag:0_

b. iorlw 0x48

```
literal 0x48  =  0100 1000     OR operation
   W    = 0xE3 = 1110 0011
new W   =        1110 1011 = 0xEB,
```

Circle one: W dest. Reg. file dest.

New value (hex) _0xEB_ C_flag :_0__ , Z flag:_0_

c. rlcf 0x04B, f

```
[0x4B]  = 0x1D=  0001 1101  (← left shift)
new [0x4B]   =   0011 1010 = 0x3A,
MSB goes into C-flag, so C=0
```

Circle one: W dest. Reg. file dest.

New value (hex) _0x3A__ C_flag :_0__ , Z flag:_0_

Circle one: W dest. Reg. file dest.

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

d. addwf 0x04B,f

```
[0x4B]   =  0x1D  (1st digit is 13+3=16, so 0).
   W    = + 0xE3  (2nd digit is 14+1+1(carry)=16,0
new W   =   0x00
C = 1 because of carry out of 2nd digit, Z=1 because
 8-bit value is 0x00.
```

e. subwf 0x049, f

Circle one: W dest. Reg. file dest.

New value (hex) _0x1D__ C_flag :_0___ , Z flag:_0_

```
[0x49]        =  0x00
   W         = - 0xE3
new [0x49]   =   0x1D
C = 0 because of borrow out of MSB
```

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

Variable locations are: *i* is data location 0x000, *j* is data location 0x001, *k* is data location 002. Assume the BSR has a 0x0 value in it initially.

unsigned char i,j,k;

a. (6 pts)
k = (i >> 1) + 50;

```
; one solution
bcf   STATUS,C   ;C_flag=0
rrcf  i,w        ; w = i>>1
addlw 0x32       ;w = w+50
movwf k          ;k = w
```

```
; another solution, others exist
movff i,k        ; k = i
bcf   STATUS,C   ;C_flag=0
rrcf  k,f        ; k = k>>1
movlw 0x32       ;w = 50
addwf k,f          ;k = k+50
```

b. (10 pts)
if ((i == 0) && (j != 0) {
k--; j++;
}

```
  ; one solution
  movf  i,f       ;test i
  bnz   end_if    ;skip if i is non-zero
  movf  j,f       ;test j
  bz    end_if    ;skip if j is zero
  ;; only get here is i is zero and j is nonzero
  decf  k,f       ;k--
  incf  j,f       ;j++
end_if
```

....rest of code....

c. (7 pts)
k = i – j – 1;

```
; one solution
movf   j,w   ; w = j
subwf  i,w   ; w = i-j
movwf  k     ; k = w
decf   k,f   ; k = k -1
```

```
; a common error
movf   j,w   ; w = j
subwf  i,w   ; w = i-j
sublw  1
movwf  k     ; k = w
```

Some students used this thinking this does w – 1. It does NOT; instead, it computes 1 – w. So the computation implemented above is actually k = 1 – (i-j)

```
; an incorrect solution
decf   j,w   ; w = j - 1
subwf  i,w   ; w = i - w
movwf  k   ; k = w
```

This is incorrect.
i – j – 1 = i – (j + 1)
i – j – 1 is NOT equal to i – (j-1)
A correct solution using the above approach is:

```
;correct solution
incf   j,w   ; w = j + 1
subwf  i,w   ; w = i - w
movwf  k   ; k = w
```

d. (10 pts)

```
while ( i < k) {
  k = k << 2;
}
```

the comparison (i < k) is actually (k > i), so do i – k as test. The TRUE case is C = 0 (borrow as we subtract a larger number from smaller number); the FALSE case is C = 1 (no borrow).

Use the FALSE case for the branch as we want to skip around the loop body

```
loop_top
  movf   k,w
  subwf  i,w       ; i - k
  bc     loop_end  ; skip if C=1,k > i is FALSE
  bcf    STATUS,C
  rlcf   k,f
  bcf    STATUS,C
  rlcf   k,f
  bra    loop_top  ; this is a LOOP, jump back to top
loop_end
  ....rest of code....
```

e. (5 pts) Write a sequence of assembly code instructions that will clear the MSb (most significant bit) and LSb (least significant bit) of variable k.

```
; one solution
bcf    k,0   ; clear LSB
bcf    k,7   ; clear MSB
```

```
;another solution
movlw  0x7E   ; clear LSB
andwf  k,f    k = k & 0x7E, clears LSb, MSb
```

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

k = (i | j) & 0xA0

```
; one solution
movf   i,w   ; w = i
iorwf  j,w   ; w = w | j
andlw  0xA0  ; w = w & 0xA0
movwf  k     ; k = w
```

```
; another solution
movff   i,k   ; k = i
movf    j,w
iorwf   k,f   ; k = k | j
movlw   0xA0  ; w = 0xA0
andwf   k,f   ; k = k & w
```