

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. Name the register on the PIC18 that holds the address of the instruction being fetched from memory.

PROGRAM COUNTER (PC)

b. Give the machine code in **HEX** for the instruction, use the assumptions for the access bit value ('a' bit) discussed in class:

XORWF 0x23A, f

**XORWF 0001 10da ffff ffff d = 1 because destination is 'f'
a = 1 because 0x23A is not in access bank!
XORWF → 0001 1011 0011 1010 → 0x1B3A (note: ffff ffff is lower 8 bits of 0x23A!!)**

c. The machine code 0x0FEA represents instruction? (*if necessary*, use 'w' or 'f' for the destination, and 'ACCESS' or BANKED to represent the value of the *a* bit)

0x0FEA → 0000 1111 1110 1010 , first eight bits indicate the ADDLW instruction, last 8 bits is the operand, so the instruction is ADDLW 0xEA

d. For a 40 MHz clock, how long does it take to execute the following instructions? (give the answer in **nanoseconds**)

bcf 0xF80,2
goto 0x01030

The BCF instruction is 1 instruction cycle, 'goto' is 2 instruction cycles (3 total instr. cycles). Each instruction cycle is four clock cycles. One clock cycle is $1/40\text{MHz} = 1/40\text{e-6} = 0.025 \text{ e-6 s}$. One instruction cycle is

$4 * 0.025\text{e-6 s} = 0.100 \text{ e-6s}$. Three instruction cycles is $3 * 0.100 \text{ e-6s}$ is 0.300 e-6 s (.3 us)

To convert to nanoseconds (1e-9), $0.300 \text{ e-6 s} * 1\text{e9 ns/ (1 s)} = 0.3 \text{ e3 ns} = 300 \text{ ns}$

e. After power up, what is the location of the first instruction fetched from memory?

The first instruction is always fetched from location 0x00000.

(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 + 10) << 2;

```
; one solution
movf  i,w      ; w = i
addlw 0x0A     ; i = i + 10
bcf   STATUS,C ; C_flag=0
rlcf  WREG,w   ; w = w << 1
bcf   STATUS,C ; C flag = 0
rlcf  WREG,w   ; w = w << 1
movwf k       ; k = w
```

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

```
; one solution
movf  i,f      ; i = i, test i
bz    if_body  ; if zero, execute if body
movf  j,f      ; j = j, test j
bz    end_if   ; if zero, skip if_body
if_body
    ...some code
    ...some code
end_if
    ..rest of code
```

```
; an incorrect tsolution
movf  i,f      ; i = i, test i
bnz   end_if   ; WRONG!!! Just because i is nonzero, does not mean that
                    ; you can skip if body, still must test j because of
                    ; the logical OR condition (||)
movf  j,f      ; j = j, test j
bz    end_if   ; if zero, skip if_body
if_body
    ...some code
    ...some code
end_if
    ..rest of code
```

- c. (6 pts) Write the following in assembly language
 $i = k + j - p;$

```
    ; one solution
    movf  p,w      ;w = p
    subwf j,w      ;w = j - w
    addwf k,w      ;w = k + w
    movwf i        ;i = w
```

- d. (7 pts)

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

```
    ; one solution
loop_top
    movf  p,w      ;w = p
    subwf k,w      ;w = k - w (do k - p)
    bc   loop_exit ; if C=1, no borrow, k >= p, exit loop
    ...loop body...
    ...loop body...
    bra  loop_top
loop_exit
    ....rest of code....
```

- e. (6 pts) Assume that r is in bank 0, s is in bank 1, and t is in bank 2. Implement the following code in assembly language:

$t = (r \& s) - 1;$

```
; one solution
movlb 0      ; BSR = 0, 'r' might be in last 128 bytes of bank 0
movf  r,w    ; w = r
movlb 1      ; BSR = 1, bank 1
andwf s,w    ; w = s & w
decf WREG,w  ; w = w - 1
movlb 2      ; BSR = 2, bank 2
movwf t     ; t = w
```

- f. (10 pts) Write a PIC18 instruction sequence that does

```
do {
    //loop body, place holder
} while ( (k >= 10) && (p != q));
```

```
; one solution
loop_top
    ...loop body...
    ...loop body...
    movlw 0x0A      ;w = 10
    subwf k,w      ;w = k - 10
    bnc  loop_exit ; if no carry, then borrow, k < 10, so exit
    movf p,w
    subwf q,w      ; w = p-q
    bnz  loop_top  ; back to top if p != q
loop_exit
    ....rest of code....
```