

ECE 3724 Summer 2006 Test #2 –Reese

Net ID: _____ (no names, please)

You may NOT use a calculator. You may use only the provided reference materials. If a binary result is required, give the value in HEX. Assume all variables are in the first 128 locations of bank 0 (access bank) unless stated otherwise. *For any signed right shifts, assume that the sign bit is preserved.*

Part I: (82 points)

a. (4 points) Write a PIC18 assembly language code fragment to implement the following.

```
signed int i;  
  
i = i << 1;
```

b. (6 points) Write a PIC18 assembly code fragment to implement the following. The code of the loop body has been left intentionally blank; I am only interested in the comparison test. For the loop body code, just use a couple of dummy instructions so I can see the start/begin of the loop body.

```
unsigned long i; //THIS IS A LONG!!!!!!!!!!!!!!!!!!!!  
  
do {  
    ...operation 1...  
    ...operation 2...  
}while (i != 0);
```

- c. (8 points) Write a PIC18 assembly code fragment to implement the following. The code of the loop body has been left intentionally blank; I am only interested in the comparison test. For the loop body code, just use a couple of dummy instructions so I can see the start/begin of the loop body.

```

signed char i, k;

while (i > k) {
    ...operation 1...
    ...operation 2...
}

```

```

loop_top:
    movf    _____, w
    _____, w
    b_____ L1
    b_____ loop_body ; if true, loop body
    b_____ loop_exit ; exit
L1:
    b_____ loop_exit ; exit

loop_body:
    ...code for operation 1...
    ...code for operation 2...

    bra    loop_top
loop_exit:
    ...rest of code...

```

- d. (8 points) Implement the *strswap()* function given below. Assume FSR0 already contains the pointer value for *char *strA* on function entry but that the pointer value for *char *strB* is passed in the CBLOCK. In the subroutine, you can use either FSR1 or FSR2 to implement the pointer operations for *char *strB*.

```

void strswap(unsigned char* strA, unsigned char* strB, unsigned char length)
{
    char tmp;
    while (length)
    {
        tmp = *strB; //save strB value
        *strB = *strA; //replace strB value with strA value
        *strA = tmp; //replace strA value with saved strB value
        strA++; //next strA location
        strB++ //next strB location
        length--;
    }
}

```

```

; Parameter block for the strswap function
CBLOCK 0x040
    length, strB:2, tmp ; Space for parameters
ENDC

```

- e. (8 points) Implement the main() code below in PIC assembly. Pass the value for “char *strA” directly in FSR0. Pass the value for “char *strB” and “char length” using the CBLOCK space for “strswap”.

```
void strswap(unsigned char* strA, unsigned char* strB, unsigned char length)
{
    // some code
}

char *s1[]="Hello!";
char *s2[]="olleh!"

main()
{

    strswap(&s1[0], &s2[0], 6);

}
```

```
; Parameter block for the strswap function
CBLOCK 0x040
    length, strB:2, tmp    ; Space for parameters
ENDC
```

```
CBLOCK 0x000    ;space for main
s1:7, s2:7
ENDC
```

- f. (6 points) Write a PIC18 assembly code fragment to implement the following. The code of the if{} body has been left intentionally blank; I am only interested in the comparison test. For the if{} body code, just use a couple of dummy instructions so I can see the start/begin of the if{} body.

```
signed int i, j;

if (i == j)
{
    ...operation 1...
    ...operation 2...
}
```

g. (6 pts) Starting at instruction "Start:", fill in the table with the order in which instructions are executed (give the label and instruction as shown, the first instruction is filled in).

Start: **call subA**
Start1: **nop**
Start2: **nop**

subA: **nop** ;
subA1: **goto subB**
subA2: **return**

subB: **nop**
subB1: **return**

Label	Instruction
1: Start	Start1: call subA
2:	
3:	
4:	
5:	
6:	
7:	

h. (20 points) After the execution of ALL of the C code below, fill in the memory location values. Assume little-endian order for multi-byte values.

```
long *ptra;
int *ptrb;
signed int b;
signed long a;
char c[4];
char *ptrc;
```

```
CBLOCK 0x020
    ptra:2, ptrb:2, b:2, a:4, c:4, ptrc:2
ENDC
```

```
a = -10;           // Note: value given in decimal
b = a >> 1;
ptrb = &a;
ptrb = &b;
ptrb = ptrb + 2;
ptrc = &c[3];
```

Location	Contents (<u>MUST</u> be given in hex)
0x0020	_____
0x0021	_____
0x0022	_____
0x0023	_____
0x0024	_____
0x0025	_____
0x0026	_____
0x0027	_____
0x0028	_____
0x0029	_____
0x002A	_____
0x002B	_____
0x002C	_____
0x002D	_____
0x002E	_____
0x002F	_____

For each of the following problems, give the FINAL contents of changed registers or memory locations. Give me the actual ADDRESSES for a changed memory location (e.g. Location 0x0100 = 0x??). Assume these memory/register contents at the **BEGINNING** of **EACH** problem.

W register = 0x02

Memory:

0x0100	0x03
0x0101	0x01
0x0102	0xB2
0x0103	0xA5
0x0104	0xF2

i. (4 points)

```
lfsr FSR1, 0x0102
movff PLUSW1, 0x0100
```

FSR1 = _____

Location _____ = _____

j. (4 points)

```
lfsr FSR1, 0x0100
movff PREINC1, 0x0104
```

FSR1 = _____

Location _____ = _____

k. (4 points)

```
lfsr FSR1, 0x0104
movff 0x0100, POSTDEC1
```

FSR1 = _____

Location _____ = _____

l. (4 points)

```
movff 0x100, FSR1L
movff 0x101, FSR1H
movff POSTINC1, 0x0102
```

FSR1 = _____

Location _____ = _____

Part II: (18 points) Answer 6 of the next 8 questions. Cross out the 2 question you do not want graded. Each question is worth 3 points.

a. Why are the FSR0, FSR1, FSR2 registers 12-bits long? Be explicit.

b. What return address is pushed on the stack for the following code?
`0x0204 call 0x100`

c. Write an addition of two 2's complement 8-bit numbers that will produce the following flag conditions: $V = 1$, $N = 1$, $C = 0$, $Z = 0$.

d. Give the machine code for the following instruction:

`here: bra here`

e. Write assembly code for the following:

```
long a, b;

a = a - b;
```

f. When would I have to use a *goto* instead of a *bra*?

g. When does return address stack overflow occur on the PIC18?

h. In the code below, the comparison $k > p$ is tested by doing $k - p$, and using the **false** case of $C=0 \parallel Z=1$ (borrow (k is less than p) or zero (k is equal to p)). However, this test does not work in the code below. Why? Be EXPLICIT, describe cases that it will work and cases that it won't work.

```
unsigned int k, p
```

```
while (k > p) {
//loop body
}
```

```
loop_top:
    movf    p,w
    subwf  k,w    ;k-p LSByte
    movf    p+1,w
    subwfb k+1,w  ;k-p MSByte
    bnc    loop_exit ;c=0 exit
    bz     loop_exit ;z=1 exit
loop_body
    instr1...
    bra    loop_top
loop_exit:
```