

ECE 3724 Fall 2005 Test #2 – Jones / 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.

Part I: (82 points)

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

```
signed int i;  
  
i++;
```

b. (8 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.

```
int i, k;  
  
if (i || !k) {  
    ...operation 1...  
    ...operation 2...  
}
```

- c. (8 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 char i, k;

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

```
loop_top:
    ...code for operation 1...
    ...code for operation 2...

    movf    _____, w
           _____, w
    b_____ L1
    b_____ loop_top    ; if true, loop top
    b_____ loop_exit   ; exit
L1:
    b_____ loop_top    ; if true, loop top
loop_exit:
    ...rest of code...
```

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

```
void strrev(unsigned char* in, unsigned char* out, unsigned char length)
{
    out = out + length;
    while (length)
    {
        *out = *in;
        out--;
        in++;
        length--;
    }
}
```

```
; Parameter block for the strrev function
CBLOCK 0x010
    out:2, length    ; Space for "char *out",
                    ; unsigned char length
                    ; parameters.
ENDC
```

- e. (8 points) Implement the main() code below in PIC assembly. Pass the value for “int *ptrb” directly in FSR0. Pass the value for “long *ptrb” and “char c” using the CBLOCK space for “a_sub”.

```

a_sub(int* ptrb, long *ptrb, char c)
{
    // some code
}

main()
{
    char n;
    int i;
    long k;

    // Some code that initializes
    // n, i, k

    a_sub(&i, &k, n);
}

```

```

CBLOCK 0x20          ; param. block
    ptrb:2, c        ; for a_sub
ENDC

CBLOCK 0x30          ; param. Block for main()
    n:1, i:2, k:4
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. CAREFUL: the variables are LONG data type!!!!!!!!!!!!

```

signed long i, j;

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

```

- g. (4 points) Write a PIC18 assembly language code fragment to implement the following.
CAREFUL: the variables are LONG data type!!!!!!!!!!!!

```
signed long a, b;
```

```
b = b - a;
```

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.

```
unsigned char a;  
unsigned char* ptra;  
signed long b;  
signed long *ptrb;  
signed int c;
```

```
CBLOCK 0x060  
    a, ptra:2, b:4, ptrb:2, c:2  
ENDC
```

```
a = 192;           // Note: value given in decimal  
b = a - 195;      // Note: value given in decimal  
c = b << 2;       // Note: value given in decimal
```

```
ptra = &a;  
ptrb = &b;  
ptrb = ptrb + 2;
```

Location	Contents (<u>MUST</u> be given in hex)
0x0060	_____
0x0061	_____
0x0062	_____
0x0063	_____
0x0064	_____
0x0065	_____
0x0066	_____
0x0067	_____
0x0068	_____
0x0069	_____
0x006A	_____

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 = 0x04

Memory:

0x0150	0x93
0x0151	0xD9
0x0152	0x3F
0x0153	0x88
0x0154	0xE1

i. (4 points)

```
lfsr FSR1, 0x0150
movff PLUSW1, 0x0153
```

FSR1 = _____

Location _____ = _____

j. (4 points)

```
lfsr FSR1, 0x0152
movff 0x0154, PREINC1
```

FSR1 = _____

Location _____ = _____

k. (4 points)

```
lfsr FSR1, 0x0151
movff POSTINC1, 0x0154
```

FSR1 = _____

Location _____ = _____

l. (4 points) (somewhat of a trick question; be aware that FSR0L occupies location 0xFE9)

```
lfsr FSR1, 0x0153
movff INDF1, FSR0L
```

FSR1 = _____

Location _____ = _____

- f. For the comparison $P \geq Q$, the required subtraction operation is $P - Q$. Give the N,V flag settings for the TRUE case and give two different numerical values (IN HEX!!) for P, Q that show why both flag cases are needed.

- g. A 16-bit comparison test ($>$, \geq) requires only a single 16-bit subtraction and a C flag check. However, the equality test ($p == q$) CANNOT be written as:

```
movf      q, w
subwf     p, w
movf      q+1, w
subwfb    p+1, w      ;;sub with borrow
bz        p_equal_q
```

where the code at `p_equal_q` is executed if p is equal q. Give numerical values for p, q that proves that this code does not work and explain why.

- h. If the PIC's data memory were expanded to 2^{16} locations, how would the size of the FSR1 register change? Would it require a FSR1U (upper byte), in addition to the pre-existing FSR1H (high byte) and FSR1L (low byte)? Why or why not?