

ECE 3724/CS 3124 Test #2 – Summer 2005- Reese

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 pts)

- a. (6 pts) Write a PIC18 assembly code fragment to implement the following.

```
signed int i, k;
```

```
i = k << 1;
```

- b. (8 pts) 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. (6 pts) Write a PIC18 assembly code fragment to implement the following:

```
signed char j, k;
```

```
while ( k >= j) {  
    operation 1...  
    operation 2...  
}
```

```
loop_top:  
    movf    ____,w  
    b____  ____,w  
    b____  L1  
    b____  loop_body ;if true, loop body  
    bra    loop_exit ;exit  
L1  
    b____  loop_exit ;if false, exit  
loop_body:  
    ...code for operation 1...  
    ...code for operation 2....  
loop_exit  
    ....rest of code....
```

- d. (8 pts) Implement the *doshift* subroutine in PIC18 assembly language. Assume the value of *ptr* has been passed in the FSR0 register by the calling subroutine. Do not forget that this is a subroutine!!!!!!

```
// shift function  
doshift (unsigned int *ptr){  
  
    *ptr = (*ptr) >> 1;  
  
}
```

- e. (9 pts) Implement the *main()* code below in PIC assembly. You MUST pass the parameters for the *a_sub()* function using the CBLOCK locations for the function *a_sub()*. You CANNOT just use FSR0 for passing the *ptr* value to *a_sub()*.

```
a_sub (char c, long *ptr){
// some code ..... //

}
```

```
CBLOCK 0x040 // parm. block for a_sub
c: 1, ptr: 2
ENDC
```

```
main() {
char p;
long k;
// some code that initializes
// p, k ....., don't worry about this
//now, call a_sub() function
```

```
CBLOCK 0x060 // parm. block for main
// define p, k space here, fill in blanks
p: ____, k: _____

ENDC
```

```
a_sub( p, &k);
}
```

- f. (8 pts) 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 == 0) && (k != 0) ) {
    ..operation 1...
    ..operation 2....
}
```

g. (6 pts) Write a PIC18 assembly code fragment to implement the following:

```
long p, q;
```

```
p = p - q;
```

h. (15 pts)

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

```
CBLOCK 0x0150
r:2, t:4, s:1, ptra:2, ptrb:2
ENDC
C code:
unsigned int r;
signed long t;          // this is SIGNED!!!!!!
signed char s;         // this is SIGNED!!!!!!
signed char *ptrs;
unsigned int *ptrb;
r = 256;                // specified in decimal!!   (3 pts)
t = -2;                 // specified in decimal!!   ( 3 pts)
s = -49;                // specified in decimal!!!! (3 pts)
ptrs = &s;              (3 pts)
ptrb = &r;              (3 pts)
ptrb++;                (3 pts)
```

Location	Contents (MUST BE GIVEN IN HEX!!!!)
0x0150	_____
0x0151	_____
0x0152	_____
0x0153	_____
0x0154	_____
0x0155	_____
0x0156	_____
0x0157	_____
0x0158	_____
0x0159	_____
0x015A	_____

i. (16 pts)

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!!!

Memory:

0x0100 0x45
0x0101 0xFF
0x0102 0xBA
0x0103 0x3C
0x0104 0x64

W register = 0x03

j. (4 pts)

```
lfsr      FSR1, 0x0101
movff    PREINC1, 0x0100
```

FSR1 = _____
Location _____ = _____

k. (4 pts)

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

FSR1 = _____
Location _____ = _____

l. (4 pts)

```
lfsr      FSR1, 0x0103
movff    POSTDEC1, 0x100
```

FSR1 = _____
Location _____ = _____

m. (4 pts) (careful on this one!!!!)

```
lfsr      FSR1, 0x0103
movff    FSR1H, 0x100
```

FSR1 = _____
Location _____ = _____

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

1. Fill in memory location below, and either a CALL or RCALL instruction (use mnemonic, not machine code) such that a value of 0x0104 is pushed as the return address on the stack

Mem location

instruction

2. Write an 8-bit addition such that afterwards, both the V and the N flags are set.

+

afterwards, V=1,N=1

3. Given an N-bit number, what number range can I represent using 2's complement encoding?

4. In the code below, what is the value of i when the loop is exited? Give the value in either hex or decimal.

```
signed char i;

i = 0x80;
while (i <= -32) {
    i = i >> 1;
}
```

5. Give the machine code for the 'bnn 0x200' instruction below given the locations shown:

location	
0x0200	decf 0x02,f
0x0202	???
0x0204	???
0x0206	???
0x0208	bnn 0x200

6. On the PIC18, can I nest subroutine calls as deep as I want? (i.e. subroutine A calls subroutine B calls Subroutine C calls Subroutine Detc). If NO, then why? Be detailed.

