

ECE 3724/CS 3124 Test #2 – Spring 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: (70 pts)

- a. (5 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. (8 pts) Write a PIC18 assembly code fragment to implement the following:

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

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

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

d. (8 pts) Implement the *doadd* subroutine in PIC18 assembly language. Assume the parameters have been initialized by the calling function. Do NOT forget that this is a subroutine!!!!!!

```
// doadd function
doadd (unsigned int *ptrA, unsigned int *ptrB){
    *ptrA = *ptrA + *ptrB;
}
```

```
;parameter space for doadd subroutine
CBLOCK 0x020
ptrA:2, ptrB:2 , ; ptrA, ptrB contains pointers to
integers
ENDC
```

- e. (8 pts) Implement the following in PIC18 assembly, which is a call to the subroutine *doadd* of the previous problem. The assembly code should work regardless of where the parameter block for main is located. The '*&p*' and '*&q*' passes the addresses of variables *p* and *q* to the *doadd* subroutine (these are the **ptr_a*, **ptr_b* parameters).

```
main() {  
  int p,q;  
  //call function  
  doadd( &p, &q);  
}
```

```
;allocation for main  
CBLOCK 0x????  
p:2, q:2:  
ENDC  
;parameter space for doadd subroutine  
CBLOCK 0x020  
ptra:2, ptrb:2 , ; ptra, ptrb contains pointers to integers  
ENDC
```

- 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 || k) {  
  ..operation 1...  
  ..operation 2....  
}
```

g. (5 pts) Write a PIC18 assembly code fragment to implement the following:

```
signed int s, p, q;
```

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

Assume the following memory contents at the START of EACH of these code fragments for problems g to h.

```

CBLOCK 0x015A
s:1, p:1, q:1,      ; char s,p,q;
r:2,                ; unsigned int r;
t:4                 ; unsigned long t
ENDC
Assume the following initializations:
s = 0x39;
p = 0x5A;
q = 0xA5;
r = 0x3044; (this will be stored in little ENDIAN order!!)
t = 0xA5DC39FF; (this will be stored in little ENDIAN order!!)

```

W register = 0x02

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 0x15B = 0x??)

h. (5 pts)

```

lfsr      FSR1, s
movff    PLUSW1, p

```

FSR1 = _____
 Location _____ = _____

i. (5 pts)

```

movlw    low q
movwf    FSR1L
movlw    high q
movwf    FSR1H
movff    POSTDEC1, s

```

FSR1 = _____
 Location _____ = _____

j. (5 pts)

```

movff    r+1, r

```

Location _____ = _____

k. (5 pts)

```

lfsr      FSR1, t
movff    POSTINC1, s

```

FSR1 = _____
 Location _____ = _____

Part II: (30 pts) Answer 10 out of the next 12 questions. Cross out the 2 questions that you do not want graded. Each question is worth 3 pts.

1. What return address is pushed on the stack for the instruction CALL 0x0300 if the location of the call instruction is 0x0154?
2. The value 0xED is a two's complement, 8-bit number. What is the decimal value?
3. Give the value of -6 as a 16-bit two's complement number.
4. Give the V, N flag settings after the operation $0x80 + 0x7F$.
5. Give the V, N flag settings after the operation $0x7F + 0x10$.

6. In the code below, what is the value of *i* when the loop is exited? Give the value in HEX!!!

```
signed char i;

i = 0x01;
while (i > 0) {
    i = i << 1;
}
```

7. For the C code and CBLOCK show below, what is the value of *ptr* after the statement '*ptr++*'? Careful, *ptr* is pointer to type *int*.

```
int *ptr;
char a[4];
int b[4];

ptr = b;
ptr ++;
```

```
CBLOCK 0x200
ptr:2, a:4, b:8
ENDC
```

8. Write the CBLOCK that allocates space for the C variables below in a similar manner as done for problem 7.

```
long *ptr;
char a[4];
long b[4];

ptr = b;
ptr ++;
```

```
CBLOCK 0x200

ptr:____, a: _____, b: _____

ENDC
```

9. Write a simple PIC18 code fragment that will force return address stack underflow.

10. Give the machine code for the 'bov 0x208' instruction below given the locations shown:

location		
0x0200	bov	0x208
0x0202	???	
0x0204	???	
0x0206	???	
0x0208	incf	0x002,f

11. Write a PIC18 assembly code fragment to implement the following.

```
signed long k,j;
```

```
k = k & j;
```

12. When does a *call* instruction have to be used instead of an *rcall* instruction?