

Circle one: JONES section / REESE section

ECE 3724 Test #2 – Fall 2006 – 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. *For any signed right shifts, assume that the sign bit must be preserved by the assembly code that you write.*

Part I: (82 points)

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

```
unsigned long i, k;      // this is a LONG, be careful  
  
k = k - i;
```

```
movf i,w  
subwf k,f  
movf i+1,w  
subwfb k+1,f  
movf i+2,w  
subwfb k+2,f  
movf i+3,w  
subwfb k+3,f
```

b. (7 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 int j,k;  
  
while (j == 0 || k != 0) {  
    ...operation 1...  
    ...operation 2...  
};
```

```
op_top      movf j,w  
            iorf j+1,f  
            bz    loop_body  
            movf k,w  
            iorf k+1,w  
            bz    loop_end  
  
loop_body  
            ...operation 1..  
            ...operation 2..  
            bra loop_top  
  
loop_end
```

- 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;

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

```
loop_top    ..operation 1
            ..operation 2
            movf i,w
            subwf k,w      ;k-i
            bov   V_1
            bnn  loop_top  ;true V=0,N=0
            bra  loop_exit
V_1
loop_exit   bn   loop_top  ;true V=1,N=1
            ....other instr...
```

- d. (8 points) Implement the *strswap()* function given below. Assume FSR0 already contains the pointer value for *char *Va* on function entry, and FSR1 contains the pointer value for *char *Vb* but that the pointer value for *char *Vc* is passed in the CBLOCK. In the subroutine, use FSR2 to implement the pointer operations for *char *Vc*.

```
void vecAdd(unsigned char* Va,
            unsigned char* Vb, unsigned char* Vc, unsigned char length)
{
    while (length)
    {
        *Vc = *Va + *Vb;
        Va++;
        Vb++;
        Vc++;
        length--;
    }
}
```

```
; Parameter block for the vecAdd function
CBLOCK 0x040
    length, Vc:2;   Space for parameters
ENDC
```

```
vecAdd      movff Vc, FSR2L
            movff Vc+1, FSR2H

loop_top    movf length,w
            bz   exit
            . movf INDF0,w      ;;could have used POSTINC0 here for more efficiency
            addwf INDF1,w      ;;could have used POSTINC1 here for more efficiency
            movwf INDF2       ;;could have used POSTINC2 here for more efficiency
            movf POSTINC0,w   ;;not needed if POSTINC0 used previously
            movf POSTINC1,w   ;;not needed if POSTINC1 used previously
            movf POSTINC2,w   ;;not needed if POSTINC2 used previously
            decf length, f
            bra  loop_top

exit        return
```

- e. (8 points) Implement the main() code below in PIC assembly. Pass the value for “char *Va” directly in FSR0, for “char *Vb” directly in FSR1. Pass the value for “char *Vc” and “char length” using the CBLOCK space for “vecAdd”.

```

void vecAdd(unsigned char* Va,
            unsigned char* Vb, unsigned char* Vc, unsigned char length)
{
    // some code
}

char vec1[]={120,3,10,23, 24};
char vec2[]={4,89, 12,39,210};
char vec3[5];

main()
{

    vecAdd(&vec1[0], &vec2[0], &vec3[0], 5);

}

```

```

Parameter block for the vecAdd function
CBLOCK 0x040
    length, Vc:2; Space for parameters
ENDC

CBLOCK 0x000 ;space for main
vec1:5, vec2:5, vec3:5
ENDC

```

```

    lfsr FSR0,vec1
    lfsr FSR1,vec2
    movlw low vec3
    movwf Vc
    movlw high vec3
    movwf Vc+1
    movlw 5
    movwf length
    call vecAdd

```

- f. (7 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 p, k;

if (p != k)
{
    ...operation 1...
    ...operation 2...
}

```

```

    movf p,w
    subwf k,w
    bnz if_body
    movf p+1,w
    subwf k+1,w
    bz if_end
if_body
    ...operation 1...
    ...operation 2....
if_end

```

g. (10 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). WARNING: This code is written in a ‘strange’ way to determine if you really understand what the call/return statements do. Fill in the blanks until you either run out of spaces or execute the instruction at location “Start3”.

```

Start1:    call subA1
Start2:    nop
Start3:    nop

subA1:    movlw 2      ;w = 2
subA2:    call subB1
subA3:    return
subA4:    nop

subB1:    decf WREG, w;   ;w=w-1
subB2:    bnz  subA2
subB3:    return
subB4:    nop

```

| Label | Instruction |
|------------|-------------|
| 1: Start1 | call subA1 |
| 2: subA1 | movlw 2 |
| 3: subA2 | call subB1 |
| 4: subB1 | decf wreg,w |
| 5: subB2 | bnz subA2 |
| 6: subA2 | call subB1 |
| 7: subB1 | decf wreg,w |
| 8: subB2 | bnz subA2 |
| 9: subB3 | return |
| 10: subA3 | return |
| 11: subA3 | return |
| 12: Start2 | nop |

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.

```
signed int a[2];
signed int *ptrb;
signed long b;
signed char c;
signed long *ptrb;
```

```
CBLOCK 0x020
  a:4, ptrb:2,b:4,c:1,ptrb:2
ENDC
```

```
a[0] = 5;
a[1] = -6;
ptrb = &a[1];
*ptrb = *ptrb + 1;
ptrb--;
ptrb = &b;
b = a[1] << 1;
c = *ptrb; //type conv. in C causes ptrb to be treated as char * here
ptrb++;
```

Location Contents (**MUST** be given in hex)

| | | |
|--------|-------------|--|
| 0x0020 | <u>0x05</u> | ; a[0]=5, LSB a[0] = 0x05 |
| 0x0021 | <u>0x00</u> | ; MSB a[0] = 0x00 |
| 0x0022 | <u>0xFB</u> | ; a[1] = -5 (-6 +1); LSB a[1] = -5 = 0xFB |
| 0x0023 | <u>0xFF</u> | ; MSB a[1] = 0xFF |
| 0x0024 | <u>0x20</u> | ;ptrb final = 0x0020; LSB 0x20 |
| 0x0025 | <u>0x00</u> | ; MSB = 0x00 |
| 0x0026 | <u>0xF6</u> | ; b = -10= 0xFFFFFFFF6; LSB b = 0xF6 |
| 0x0027 | <u>0xFF</u> | ; b 2nd byte = 0xFF |
| 0x0028 | <u>0xFF</u> | ; b 3rd byte = 0xFF |
| 0x0029 | <u>0xFF</u> | ; b 4th byte = 0xFF |
| 0x002A | <u>0x05</u> | ;c = 5 = 0x05 |
| 0x002B | <u>0x2A</u> | ; ptrb final = 0x0026 +4 = 0x002A; LSB = 0x2A |
| 0x002C | <u>0x00</u> | ; ptrb MSB = 0x00 |

- i. (12 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.

W register = 0x01

Memory:

| | |
|--------|------|
| 0x02F0 | 0x03 |
| 0x02F1 | 0x01 |
| 0x02F2 | 0xF0 |
| 0x02F3 | 0x02 |
| 0x02F4 | 0xAC |

- i. (4 points)

```
movff 0x2F2, FSR1L
movff 0x2F3, FSR1H
movff 0x2F4, POSTINC1
```

FSR1 = 0x02F1

Location 0x02F0 = 0xAC

- j. (4 points)

```
lfsr FSR1, 0x02F2
movff POSTDEC1, 0x02F0
```

FSR1 = 0x2F1

Location 0x2F0 = 0xF0

- k. (4 points)

```
lfsr FSR1, 0x02F3
movff PREINC1, 0x2F1
```

FSR1 = 0x2F4

Location 0x2F1 = 0xAC

Part II: (16 points) Answer 4 of the next 6 questions. Cross out the 2 questions you do not want graded. Each question is worth 4 points.

- a. Write PIC18 assembly code for the C statement below assuming that FSR0 already contains the address of "ptr".

```
int *ptr;  
ptr++;
```

```
movf POSTINC0,w  
movf POSTINC0,w    ;;increment twice since point to int
```

- b. Write PIC18 assembly code for the C statement below assuming that FSR0 already contains the address of "ptr".

```
int *ptr;  
*ptr = *ptr+1;
```

```
movlw 0  
incf POSTINC0,f    ;increment LSByte  
addwfc POSTINC0,f  ;increment MSByte
```

- c. (1) Write an addition of two 2's complement 8-bit numbers that will produce the following flag conditions: $V = 1, N = 0, C = 1, Z = 0$. (2) What range of numbers can be represented in 6 bits using 2's complement encoding? Give the value range, do not give an equation.

(1) $0x80 + 0xFF = 0x7F$ $V=1, N=0, C=1, Z = 0$

(2) in 6 bits, from -32 to +31

- d. Give the machine code for the 'bnz' instruction in the following code fragment:

```
        bnz there
        call subA
there:   incf j,f
```

```
branch_target = PC+2+2*N
N = (branch_target - (PC+2))/2
  = ( PC+6 - PC -2)/2 = 4/2 = 2

offset is 2.  Encoding for 'bnz' with an offset of '2' is
0xE102
```

- e. Write assembly code for the following:

```
signed int k;

k = k >> 1;
```

```
bcf     STATUS,C
btfsc   k+1,7
bsf     STATUS,C
rrcf    k+1, f
rrcf    k, f
```

- f. Write assembly code for the following:

```
signed int k;

k = k << 1;
```

```
bcf     STATUS,C
rlcf    k,f
rlcf    k+1,f
```