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.

```assembly
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.

```assembly
int i, k;
if (i != k) {
    ..operation 1...
    ..operation 2....
}
```
c. (8 pts) Write a PIC18 assembly code fragment to implement the following:

```assembly
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!!!!!!

```assembly
; parameter space for doadd subroutine
CBLOCK 0x020
ptra:2, ptrb:2 ; ptra, ptrb contains pointers to integers
ENDC
```

```c
// doadd function
doadd (unsigned int *ptra, unsigned int *ptrb){
    *ptra = *ptra + *ptrb;
}
```
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 `*ptra` *ptrb` parameters).

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

---

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.

```assembly
int i, k;

if (i || k) {
    ..operation 1...
    ..operation 2....
}
```
g. (5 pts) Write a PIC18 assembly code fragment to implement the following:

```assembly
    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!!)
- \( t = 0x5A5C99FF; \) (this will be stored in little ENDIAN order!!)

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!!

```c
signed char i;

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

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

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

ptr = b;
ptr ++;
```

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

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

ptr = b;
ptr ++;
```

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:

<table>
<thead>
<tr>
<th>location</th>
<th>instruction</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0200</td>
<td>bov</td>
<td>0x208</td>
</tr>
<tr>
<td>0x0202</td>
<td>??</td>
<td></td>
</tr>
<tr>
<td>0x0204</td>
<td>??</td>
<td></td>
</tr>
<tr>
<td>0x0206</td>
<td>??</td>
<td></td>
</tr>
<tr>
<td>0x0208</td>
<td>incf</td>
<td>0x002,f</td>
</tr>
</tbody>
</table>

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

```assembly
signed long k, j;
k = k & j;
```

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