Name: ____________________________

Directions: Work only on this sheet (on both sides, if needed); do not turn in any supplementary sheets of paper. There is actually plenty of room for your answers, as long as you organize yourself BEFORE starting writing. In order to get full credit, SHOW YOUR WORK.

1. (25) Suppose c(ESP) = 5000, c(5000) = 240 and c(240) = 8800. Consider the execution (excluding the instruction fetch) of

```
addl $8, (%esp)
```

(a) What values will go onto the address bus during the execution of this instruction?

(a) What values will go onto the data bus during the execution of this instruction?

2. (25) Fill in the blanks: The IDT register is written to by the __________ at __________ time, and is read by the hardware at the times that __________ occur.

3. (25) Suppose **x()** calls **y()**, which in turn calls **z()**, with **x()** and **y()** being written in C, and with **z()** in assembly language. Suppose the declaration of **x()** starts with

```
int x(int a, int b, int c)
```

Give **at most four lines** of assembly code, to be placed at the very beginning of **z()**, which will copy the third parameter of **x()** (formally named **c** above, but not referrable to as such from within **z()**, for reasons of generality), to the EDX register. You may use any registers as "scratch" space except ESP and EBP.

4. (25) The code below is made up, but could be part of Linux's context switch code, preceding the real code shown on p.10 of our PLN unit on OS. It would choose the process to be given the next turn, and would point EBX and ECX to the proper values. Assume that: each process' TSS is 812 bytes long; the current state of the process, 'R' or 'S' for Run or Sleep, is stored 240 bytes from the beginning of the TSS; the process number is stored in the word 88 bytes from the beginning of the TSS; the variable **currturn** in the **.data** segment gives the address of the TSS of the process whose turn it is currently (at the time **u**'s turn ends on p.9, **currturn** points to **u**'s TSS, and the code below will ultimately set **currturn** to **v**'s TSS). The TSSs are assumed to form an array, i.e. they are stored contiguously.

```
   # saving of register values on stack,
   # code not shown
   movl __________, %ecx
   movl %ecx, %ebx
top:
   # check if at end of array;
   # if so, go to array start (code not shown)
   addl __________, %ebx
   cmpb 'R', __________(%ebx)
   jz got_one
   jmp top
got_one:
   movl __________, __________
   # restoring of register values from stack, not shown
```

Solutions:

1.a 5000, 5000

1.b 240, 248

2. OS, boot, interrupts

3. Note that EBP still has the value placed into it by **y()**.

```
movl (%ebp), %ebx
movl 8(%ebx), %ebx
```

Due to an error in the PLN, the value 16 was accepted in place of 8.

4.

```
   # saving of register values on stack,
   # code not shown
   movl currturn, %ecx
   movl %ecx, %ebx
top:
   # check if at end of array;
   # if so, go to array start (code not shown)
   addl $812, %ebx
   cmpb 'R', 240(%ebx)
   jz got_one
   jmp top
got_one:
   movl %ebx, currturn
   # restoring of register values from stack, not shown
```