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**.

ASSUME THROUGHOUT THAT WE ARE DEALING WITH ORDINARY PCS RUNNING LINUX. By the way, instructions like `pushl w`, where `w` is a label in the .data section, are OK.

1. Consider the assembly language presentation of the compiled code for **NumNode.java** on p.17 of our PLN on the JVM.

(a) (5) Give the JVM machine code in hex for the **if_cmpge** instruction which starts at offset 21 of **Insert()**.

(b) (5) Give the line number in the source code on p.16 whose compiled code includes this same **if_cmpge** instruction.

(c) (10) Suppose we are currently just about to execute the **putfield** instruction in offset 6 of **NumNode()**. Draw the contents—in terms of variables in the source code on p.16—of the operand stack at that point (with the stack growing downward in your picture).

2. (10) Say an interrupt occurs during the execution of some instruction. During Step D of that instruction, how many bytes are read from memory? How many bytes are written?

3. For each of the following instructions, state how many bytes of memory are read, and how many are written. Do not count instruction fetches.

(a) (10) **addl $8, (%eax)**

(b) (10) **inb $50, %al**

4. (5) Consider the threads example from discussion section. Suppose we execute the program as

```
a.out /dev/pts/3 /dev/pts/2 /dev/pts/4 /dev/pts/6
```

and run **ps** (with whatever options are needed for threads). How many threads will show up?

5. (15) Write OS code that launches a new user process (as opposed to giving an already-existing process a turn). Assume that the OS has stored the addresses of the processes' stack and first-executed instruction at the OS' variables **stackstart** and **progstart**, respecitively, and that the initial value of EFLAGS is in **initeflags**. Assume also that the values in the registers EAX, EBX and so on will be initialized by the process itself. You may have at most six instructions, hopefully fewer.

6. (10) Consider our very first assembly language example program, in Section 4 of our PLN on Linux assembly language. This code calculated the sum of a four-element array. I've revised the code, so that it accomplishes this same goal but in a different way. Here is the core of the output from **as -a**:

```
                    _start:
0000 B8040000               movl $4, %eax
     00

0005 BB000000               movl $0, %ebx
     00
000a 031D0000       top:    addl x, %ebx
     0000
0010 80050C00               _______, top+_____
     000004
0017 48                     decl %eax
0018 75F0                   jnz top
001a 891D1000       done:   movl %ebx, sum
```

Fill in the blanks.

7. Consider a page fault ISR on an Intel machine, written in assembly language.

(a) (10) At what address will the ISR start? Use the c() notation in your answer, e.g. c(EBX)+8888*c(2000).

(b) (10) Suppose for some reason we want the ISR to check whether the instruction that triggered the interrupt was a JNZ. We'll put 1 in EAX if that is the case, and put 0 there otherwise. Fill in the blank:

```
    movl $0, %eax
    ------------------------------
    jnz q
    movl $1, %eax
 q:
```

Solutions:

1.a 0xa2000f

1.b 25

1.c

```
this
V
```

2. 12 bytes written, none read

3.a 4 bytes read, 4 bytes written

3.b no bytes read or written

4. 5 including **main()**

5.

```
movl stackstart, %esp
pushl initeflags
pushl $0
pushl progstart
iret
```

6. The program literally modifies itself.

```
addb $4, top+2
```

7.a

```
c(c(IDT) + 8 * 0xe)

    cmpb (%cr2), $0x75
```