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. (5) A keyboard, for example, has an "address" (more likely more than one), but we don't use the term *address* in that context on Intel-based machines. Instead, we use the term _____.

2. (10) It was speculated in class that the JVM is big-endian, judging from material in our PLN on JVM. Cite the specific sentence in that PLN that implicitly tells us that the JVM is big-endian.

3. (5) Suppose you run **as -a** on an assembly language program on a UNIX/MIPS machine. In the listing for the **.text** segment, in most cases the memory offsets of consecutive instructions will differ by _____ (fill in a number).

4. It is possible that an instruction might trigger more than one page fault. For each of the Intel instructions below, state the maximum number of page faults that instruction could cause, including Steps A, B and C (but don't break your number down according to step). Assume no cache and no instruction prefetching.

(a) (5) **movl %eax, (%ebx)**

(b) (5) **addl %eax, (%ebx)**

(c) (5) **iret**

5. (10) Suppose the maintainers of the Linux OS kernel were to change the source code for the **struct** for a TSS to look like this:

```
struct tss {
   int turns;
   ...
   int esp;
   ...
};
```

Here the entry **turns** records how many turns (i.e. quanta or timeslices) this process has completed so far. Give a *single* line of assembly language, to be inserted somewhere in the code on p.9 of our unit on OS, which would do something with that entry.

6. Look at the code on p.21 of our unit on OS. Suppose that just prior to the fetch of the INT instruction, we have the following register values: c(PC) = 0x2088, c(ESP) = 0x602c, c(IDT)+8*0x80 = 0x112288, c[c(IDT)+8*0x80] = 0x484746. The machine code for an INT instruction has the format 11001101IMM1. Show the contents of the following just after Step C is done:

(a) (10) PC

(b) (10) ESP

(c) (10) top of the stack

7. The following questions are on our unit on the JVM:

(a) (5) Suppose (in this part only) that the call to **Min()** in **main()** on p.5 had been

```
X = Min(X,Y);
```

State the line number(s)/offsets on pp.5-6 for the instructions which would change.

(b) (10) Here is part of the **od -t x1** listing of the file **Minimum.class**

```
...
240 0a 00 00 00 54 00 02 00 03 00 00 00 0e 1a 1b a2
260 00 08 1a 3d a7 00 05 1b 3d 1c ac 00 00 00 02 00
300 0b 00 00 00 0e 00 03 00 00 00 11 00 0a 00 12 00
320 0c 00 13 00 0c 00 00 00 20 00 03 00 00 00 0e 00
340 19 00 14 00 00 00 00 00 0e 00 1a 00 14 00 01 00
360 07 00 07 00 1b 00 14 00 02 00 01 00 1c 00 00 00
400 02 00 1d
```

Give the op codes, in hex, for **goto** and **ireturn**.

(c) (10) Suppose in some program there is an **if_cmpge** instruction with offset 0x20000. Then the branch target can be anywhere from _____ to _____, inclusive (fill in two hex addresses).

Solutions:

1. port

2. p.7: "Thus the entire instruction should be 0xa20008..."

3. 4

4.a. 2 (instruction fetch, operand)

4.b. 2 (instruction fetch, operand; note that though operand is accessed twice, it could only produce one page fault)

4.c. 3 (instruction fetch, operands; note that though there are 3 operands pushed onto the stack, only 2 page faults could occur from them)

5.

```
incl 0(%ebx)
```

6. 0x484746, 0x6020, 0x208a

7.a. 19

7.b. 0xa7, 0xac

7.c. The lower bound is 0x00020000 + 0xffff80000 = 0x00018000 (see the material on **sign extension** in our unit on machine language). The upper bound is 0x00020000 +0x00007fff = 0x00027fff.