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. In fill-in questions, the number of blanks shown has no relation to the number of characters in the answer. Note that earlier questions tend to be easier.

1. Fill in the blanks with the names of programs one can run on the PCs in CSIF:

- (a) (5) A program you can use to see "how many people you are taking turns with" right now on a given machine is ____________________.
- (b) (5) A program whose output you can watch as another program's memory usage changes over time is ____________________.
- (c) (5) If program X called execve() to start program Y, and Y did the same thing to start program Z, you would see this relationship between X, Y and Z if you were to run the program ________________________.

2. Remember, the MIPS and Intel chips are completely different; a machine-program which runs on one has no hope of running on the other. The two chips have different register sets, different instructions (operations, op codes), etc. However, most chip types are at least vaguely similar — they have registers, they access memory, they do arithmetic, they do jumps, etc. For each of the two MIPS instructions below, give an Intel instruction which is similar.

- (a) (5) lw \$3, 200(\$4)
- (b) (5) addu \$3, \$3,12

3. (15) Number the bits of a word from bit 31 (most significant bit) to bit 0 (least significant bit). Write AT&T Intel assembly code (just a few lines of code, not a full program or subroutine) which will put 1 or 0 in EBX, depending on whether bit 5 is a 1 or 0. Write your code carefully on scratch paper first, and then copy it clearly to this exam sheet.

4. (15) Consider the little C program in Sec. 5.5.3 of the OS unit handed out in class. Suppose that in addition to the array q, some other **int** arrays had also been declared. For instance (this is just an example!),

```
int x[500],q[200],y[1520];
```

No other changes are made to the program.

We are interested in i_{seg}, the value of i when the seg fault occurs. Find the largest possible value that i_{seg} might have. Do your work on scratch paper first, then copy it clearly to this exam sheet.

5. Suppose on a PC running Linux I/O device number 6 causes an interrupt, at a time when the contents of various registers are: c(EAX) = 0x1216, c(EBX) = 0xfffffff0, c(ESP) = 0x6448, c(IDT) = 0x100, c(CR3) = 0x8888. (You may not use all of these.) Assume there is no paging.

- (a) (15) Give the specific numerical values, if any, which we know for sure will go in the MAR during the processing of the interrupt.

- (b) (15) Give the specific numerical values, if any, which we know for sure will go into the MDR during that processing.

6. (15) Write assembly code which might appear in the portion of Linux that handles page faults. Specifically, write code which will update the entry in the page table for the page which caused the fault, i.e. the page brought in from disk. Assume the format in Sec. 5.5.1 of the revised OS handout distributed in class, and assume that previous code has already set two variables in the data segment, virtpagenum and physpagenum, for the page in question. Assume that paging is turned off during this period of time. Do not refer to "PTR"; use the specific Intel register which plays this role. Write your code carefully on scratch paper first, and then copy it clearly to this exam sheet.

Solutions:

1.a. w

1.b. top

1.c. pstree

2.a. For example:

```
movl 200(%eax), %ebx
```

2.b. For example:

```
addl $12, %ebx
```

3. For example:

```
andl $0x20, %eax
shrl $5, %eax
movl %eax, %ebx
```

4. Since the page size is 4096, each page can contain 1024 **int**s. The largest value of i_{seg} will occur if just one element of q is in the page (at the beginning of the page), with the rest of the page not being part of q. This will occur if q[199] is at the beginning of the page, followed by 1023 words which aren't in q but which won't produce access violations, since they are in the same page. The seg fault will then occur the very first time we try to access the next page, i.e. 1024 words after q[199]. Thus the largest possible value of i_{seg} is 199+1024 = 1223.

5. When the interrupt occurs, the CPU will first save the three register values on the stack, so we know for sure that the MAR will contain 0x6444, 0x6440 and 0x643c. Then the CPU will assert IACK, and the I/O device will send its number, in this case 6, along the data bus – so we know for sure that the MDR will have 6 in it. Then the CPU will check the information at c(IDT)+8*6 = 0x130, so that value will go in the MAR.

6.

```
# the page table entry we want is at c(CR3)+4*virtpagenum
```

```
movl %cr3, %eax
movl virtpagenum, %ebx
shrl $2, %ebx
addl %ebx, %eax
# now EBX points to that entry

# erase the physical page number in the entry, but make sure to preserve
# the other information
andl $0x0000fff, (%eax)

# now put in the new physical page number into that entry, again being
# careful to preserve the other information
movl physpagenum, %ebx
shll $12, %ebx
orl %ebx, (%eax)

# finally, change Bit 11 to 1, indicating the page is now resident
orl $2048, (%eax)
```