Name:

Directions: Work on this sheet (both sides, if needed) only; **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, **WRITE LEGIBLY** (∞ points off for illegible handwriting!), and **SHOW YOUR WORK**. Make sure you work on the easier problems first (which will usually be the earlier problems, and within multi-part problems, the earlier parts).

1. Consider a virtual-memory machine which has 16-bit addresses, thus a capacity of 2^{16} bytes of physical memory, and page size 2048 ($= 2^{11}$).

(a) (10 pts) How many lines will the page table have?

(b) (10 pts) Suppose a certain game program consists of 8192 ($= 2^{13}$) bytes of code and 3000 bytes of data. Uncle Bill is playing the game on the machine. Then Aunt Hillary starts to play too. Assuming that the operating system tries to conserve memory usage, how many lines will Hillary's page table have in common with Bill's?

2. (15 pts) Look at Fig.7.2.3, p.504 of Patterson & Hennessy. Suppose this simple machine has 8-bit addresses and 8-bit words. Find the total number of flip-flops needed to implement the 2-way set-associative configuration. Assume a write-back policy, and remember that each tag will also include a Valid bit. Give your answer in unsimplified form—for example, $2 + 3 \times 7$ instead of 23—in order to show your work.

3. (15 pts) On p.300 of the PC book, write a C-language version (i.e. no assembly language) of PUTCURSOR. You still will call the BIOS.

4. (15 pts) In the game program, pp.281-287 of the PC book, write NEWDIR as a C function, instead of as an assembly-language macro. (_INIT, KISR, etc. will remain as assembly language). Show not only the C function, but also <u>all</u> changes (especially deletions) to the assembly-language code.

5. Consider a CPU with the following structure. Instructions are 16 bits wide, labeled Bits 15-0. There are various instruction formats, determined by Bits 15-14; a 00 in that field means that the given instruction will access the system bus. In such cases, Bits 13-12 form the op code, with the following codings: 00 for an IOL instruction, 01 for IOS, 10 for ML, and 11 for MS. The 'L' and 'S' in these four instruction mnemonics mean "load" (read from the chip) and "store" (write to the chip), respectively. There are two system bus lines, $IO/\bar{M}$ and $L/\bar{S}$, related to these four instructions. For instance if $IO/\bar{M}$ has a 0 on it and $L/\bar{S}$ has a 1, then we will read from a memory chip. Bits 11-0 of the instruction contain the I/O or memory address. The system bus has address lines A11-A0 and data lines D7-D0. In our system we will have two I/O chips, each containing a single port, to be at addresses 0x000 and 0x001, respectively. We will also have two 2Kx8 memory chips, and will use low-order interleaving, i.e. one chip will contain the even-numbered addresses and the other chip the odd-numbered ones. All four chips have the following pins: CS, D7-D0, $R/\bar{W}$. The I/O chips have no address pins, while the memory chips have address pins AM10-AM0.

(a) (10 pts) Show the connections of the four chips to the bus, including any "glue" (i.e. miscellaneous gates) needed.

(b) (10 pts) Show the connections <u>within</u> the CPU between the instruction register IR (which consists of Bits 15-0) and the system bus, for the case of the 00 instruction format.

6. (15 pts) Look at Prog. 8.4 in the PC book. Suppose the operating system's data segment has the following labels: FIRSTPR, the offset of the first line in the process table; CURRPR, the offset of the page table line belonging to the current process; and LASTPR, the offset of the last line of the process table. Each line of the process table is 100D bytes long, and the register-save area within each line begins at byte 12D of that line. Show the assembly code corresponding to the comment, "look at the process table...," and the code, later in this same interrupt routine, needed to update CURRPR.