MACQUARIE UNIVERSITY

Division of Information and Communication Sciences

END-OF-YEAR EXAMINATIONS 2005

Unit:	COMP226 - Computer Architecture
Date:	Thursday, 24 November, 2005, 9:20 am
Time Allowed:	Three hours + ten minutes reading time
Number of Questions:	Seven (7)
Total Marks:	70
Instructions:	Answer ALL questions.
	The examination is in two sections. Section A is on SPARC assembler, Digital Logic, and Pipelining. Section B is on Memory Management, I/O and Systems Programming.
	Answer Sections A and B in separate books. Indicate the section clearly on the outside of each book. Ensure that your name and student number are clearly marked on each answer book.
	If you wish to spread your time evenly, you should spend about 25 minutes on each question.
Materials permitted:	Calculators are not permitted.
	Dictionaries are not permitted.

SECTION A. SPARC Assembler, Pipelining and Digital Logic

QUESTION 1. (10 marks)

For each of the following, work out your solution first on scrap paper and then copy it neatly into your book. Comment your solutions clearly.

Write a fragment of SPARC assembler which will swap two 32-bit values stored in memory at locations labelled `first` and `second`.

Now write another fragment that will swap two 8-bit values stored in memory at locations `third` and `fourth`.

Lastly, write a third fragment which will swap an 8-bit and a 32-bit value. You may assume that the 32-bit value is an integer between 0 and 127. The 8-bit value is a signed integer. The 8-bit value will be originally stored at absolute location number 8, and the 32-bit value at absolute location number 32.

Be sure that the numerical values are preserved (so that for example the 8-bit number has the same value whether it is read as a byte from location 8 before your program executes, or as a word at location 32 after your program executes). Write a brief paragraph explaining how your code preserves the numerical values.

QUESTION 2. (10 marks)

Assemble the following fragment of SPARC assembly code into SPARC machine code.

```
        .text
        cmp     %r3,0
        be      fin
        nop
fori:   add     %r1,%r2,%r1
        subcc   %r3,1,%r3
        bne     fori
        nop
fin:    ta    0
```

Show your working for each line of code, and then present your final answer as eight lines of 32 bit words written in hexadecimal.

The SPARC reference card is included at the back of this examination paper.

QUESTION 3. (10 marks)

Consider the following code fragment executing on a machine with the usual five stage pipeline. Suppose that `%r2` initially contains 1, `%r3` initially contains 6, and `%r5` initially contains 6, and that the word with value 1 is stored at location 0.

```
        ld [%r0], %r3           ! line 1
  t1:   add %r3, %r2, %r3       ! line 2
        subcc %r3, %r5, %r0     ! line 3
        blt t1                  ! line 4
        inc %r2                 ! line 5
```

Assume that the branch takes four cycles to complete and has just one branch delay slot. Report what happens if the code is run on
(a) (4 marks) a machine with no forwarding and no hardware interlocks

(b) (3 marks) a machine with no hardware interlocks, but with forwarding

(c) (3 marks) a machine with both hardware interlocks and forwarding

For each one you should report the final values of the three relevant registers and the number of times that the loop code is executed. In each case explain your answer. You will most likely find it useful to draw up a table with the instructions down the left hand side and the progress of time (in pipeline stages) across the top. You may if you wish refer to instructions by the line numbers provided.

QUESTION 4. (10 marks)

(a) (5 marks)
Use the Hardware Description Language presented in lectures to describe what happens on each of the five stages of the standard pipeline when the instruction

```
add %r1,16,%r1
```

is executed.

(b) (5 marks)
Draw a diagram which shows how to construct an `AND` gate using three transistors. Next draw diagrams which show how to construct a half-adder and a 3-input 8-output decoder. You may use `AND`, `OR` and `NOT` gates.

SECTION B. Memory Management, I/O and Systems Programming

(Use a separate book)

QUESTION 5. (10 marks)

Consider a 2-way set associative physical cache with 64 byte line size. Suppose that the cache can hold 1MB of data. Suppose that a machine uses this cache and has 32 bit addresses used to access a 4GB memory space.

(a) (2 marks) Compute the size of the offset, tag and index fields

(b) (4 marks) How many bits in total will the cache need to store? Explain your working (and you may leave the final answer as an arithmetic expression if you wish since you don't have a calculator).

(c) (4 marks) What is the effective access time for a program which accesses in turn memory locations 40 to 65 and then loops 10 times accessing in turn memory locations 60 to 70 each time through the loop if the hit time is 10ns and the miss time (main memory access time) is 1000ns. Assume that the cache is initially empty and (as always!) show your working. Again you may leave your final answer as an arithmetic expression if you wish.

QUESTION 6. (10 marks)

Suppose that a hard disk drive has 2000 sectors per track and that each sector holds 4KB. Assume that the disk has 6 surfaces and 2000 tracks on each surface. What is the approximate capacity of the disk?

If the disk rotates at 6000 rpm, what is the average rotational delay? Express your answer in microseconds. If a cache hit takes 10ns, how many times slower than a cache hit is a disk access even assuming that the heads are already in the correct position?

If the heads move at 100 tracks per ms, what is the approximate worst case access time (assuming that the disk is already spinning and ignoring the time it takes for the heads to get up to full speed).

QUESTION 7. (10 marks)

Write a C program which accepts an arbitrary number of filenames on the command line. Your code should add a newline character at the end of each of the files specified on the command line but otherwise leave the files unchanged. Make sure that your program includes code to deal appropriately with missing files or misspelt filenames.

Work out your solution first on scrap paper and then copy it neatly into your book. Comment your solution clearly.

.

.