CS61c, Fall 1998 Midterm #2 Professor Fateman

Problem #1 (7 points)

Answer the following questions using the diagram of a full adder module:

```
                     A   B   CarryIn
                     |   |       |
                     V   V       V
               |------------------------|
CarryOut <<< |                            |
               |------------------------|
                              |
                              V
                             Sum
```

a. Using no more than 10 of these adders, and no more than 4 other logic symbols, draw a circuit that takes an 8 bit unsigned number X and returns the eight lower order bits of X multiplied by 5. Ignore overflow. For example, 5*0x04 is 0x14 (20 decimal), and 5*0x80 is 0x80.

b. By making the inputs and outputs specifically on your diagram, show how your circuit can multiply 40 (decimal) by 5 to get 200 (decimal).

Problem #2 (2 points)

CDR-coding can be explained as a way of saving memory.
a. How does it save memory?
b. Now that memory is so cheap, can you come up with a good reason to still use CDR coding?

Problem #3 (10 points)

a. Given the address 0xA2EC4A8F, compute the block offset, set number and tag for each of the following caches:

Size(bytes)	Width (bytes)	Policy	Offset	Set	Tag in HEXADECIMAL
4096	4	Direct-Mapped			
1024	16	4-way set associative			
2048	32	Fully Associative			

Problem #4 (7 points)

	Stride	4	8	16	32	64	128	256	512
Size									
32		205	212	206	-	-	-	-	-
64		221	203	213	202	-	-	-	-
128		214	243	232	212	204	-	-	-
256		227	235	241	233	216	231	-	-

512		598	523	543	1311	1285	1289	321	-
1024		543	521	514	1294	1314	1306	1432	345
2048		552	517	532	1398	1302	1403	1532	1356
4096		527	513	567	1302	1298	1198	1231	1323

Stride and Size are in bytes, times nanoseconds. For the following questions, circle the numbers (or rows/columns) you used to answer each question and label them a, b, c.

(a) What is the cache size in bytes?
(b) What is the block size in bytes?
(c) What is the allocation policy? (Direct-map, full associative or N-way associative)
(d) In the output we were able to give timing data that was down in the 200 nanosecond range. How were we able to do so, considering that no user-accesible clock on the computer we were testing is nearly so precise?

Problem #5 (10 points)

Answer each of these questions by filling in the ______ with one of these:
increase, decrease, or not change

Assume the cache size stays the same.

a. Increasing the size of the block size in a direct-mapped cache would _______ the number of bits for byte offset.
b. Increasing associativity from 2 to 4 way in a cache would ______ the circuit complexity.
c. Increasing the size of the block in a direct-mapped cache would _______ the number of bits for tag.
d. Increasing the size of the block in a direct mapped cache would ______ the cost of a cache miss.
e. Replacing a write-back strategy with a write-through strategy would ___ the cost for repeatedly updating a memory location.
f. Writing code in-line instead of calling procedures would ____ spatial locality.
g. Compulsory cache misses would ____ after the operating system switches to another process.
h. Capacity cache misses would ___ after the operating system switches to another process.
i. Changing from direct-mapped to 2-way set associative ______ collisions.
j. Rearranging code so that an "inner loop" has fewer instructions would ____ temporal locality.

Problem #6 (5 points)

a. The instruction LA $8, M assembles into two instructions, LUI and ORI. Assuming that M has address 0x00554444, write out a plausible sequence of instructions to accomplish the load address.
b. Now assume that the all the static data in this program must be relocated, inlcuding the location for M, and that the relocation amount is 0x01004000. What instructions will accomplish this same LA after relocation?
c. Relocation information must be made available for instructions int the .text segment, but not for branch instructions? Why?
d. Values computed by the assembler in the .data segment may also need relocation. Why?

Problem #7 (8points)

Asynchronous output requires two functions, F1 to put data into a buffer or queue, and F2 to send the data to an output device.
a. Who calls F1 and when?
b. Who calls F2 and when?
c. Sometimes calling F2 does nothing. Why might that happen?

d. MAL converts a "call" from a user program to print characters into a system call, which among other things changes from user to kernel mode. Give one reason for this design feature. (Hint: most systems are assumed to be capable of running multiple processes).

Problem #8 (5 points)

The proper declaration for printf in C is

```
int sprintf(char *outbuf, char *fmt, ...)
```

Where the declatrration ... means that the number and types of all but the first two arguments can vary.
a. In your implementation of sprintf, where exactly did you obtain the first argument?
b. Where exactly did you obtain the second through last arguments?
c. If the number of extra arguments exceeds the number of format directives, what happens?
d. If the number of format directivies exceed the number of extra arguments, what happens?
e. Can the `sprintf` program check to see that they are the same? (Why or why not?)

Problem #9 (5 points)

a. A typical bus transaction consists of twp parts. What are they?
b. Underline which type of bus usually has to be kept to a rather short length:
asynchronous otr synchronous (clocked).
c. What is the role of the cause register?
d. From your reading or lab, give an estimate (be sure to give UNITS: your values can be very rough) for the HP-UX hard disk latency _________ and bandwith _____________ .