Name ______________________

Computer Architecture and Implementation
EE 7700-4
Final Examination

7 December 1998, 15:00-17:00 CST

Problem 1	________	(20 pts)
Problem 2	________	(20 pts)
Problem 3	________	(20 pts)
Problem 4	________	(40 pts)
Exam Total	________	(100 pts)

Alias ______________________

Problem 1: The program below runs on a processor using PAp branch prediction and a JRS confidence estimator. The PAp branch predictor uses four branch outcomes (per branch history register) and the tables are large enough so there is no interference between branch instructions. The branch predictor uses two-bit saturating counters. The mispredict distance counter table has one entry per instruction (also no interference). The high confidence threshold is five. All table entries are zero when the code below starts executing.

```
/*             0  1  2  3  4  5  6  7   */
int *bt[] = { 0, 1, 1, 0, 1, 1, 0, 0 }; /* 8-element array. */

for(i=0; i<100; i++) for(j=0; j<8; j++) if( bt[j] ) x[i]++;
```

(*a*) Find the prediction accuracy of the branch in the `if` statement over the entire execution of the code (`i=0` through `99`). (7 pts)

(*b*) What is the PVP and sensitivity of the JRS confidence estimator on the branch in the `if` statement from `i=50` through `i=99`? (7 pts)

(*c*) Is the program above a good demonstration of the JRS confidence estimator? If it's not describe how the program could be modified to better demonstrate the confidence estimator. (6 pts)

Problem 2: The DLX assembler program below examines elements of an array. The program is to be run on a simultaneous multithreading processor based on a 4-way superscalar processor similar to the one described in Tulleson 96. The program runs as four threads, the value of `r1` when the code below starts is different in each thread.

The instruction fetch unit can fetch four consecutive and aligned instructions per cycle. Round-robin thread selection is used.

There are an unlimited supply of functional units. The results from ALU operations and loads that hit the cache are available in next cycle.

The memory system can support up to two outstanding cache misses. Cache miss delay is 50 cycles (regardless of address).

There are a total of 40 reservation stations. Reservation stations are not dedicated to a particular functional unit so an instruction can move from any reservation stations to any functional unit.

Branch prediction is perfect.

```
 ! r1: Base address.     (Each thread has own value.)
 ! r2: Stride, in bytes. (Same in all threads.)
 ! r3: Last value in array.  (Same in all threads.)
LOOP:
 lw  r7, 0(r1)        ! Load value.
 add r1, r1, r2
 beqz r7, SKIP        ! Goto SKIP if r7 zero
 addi r4, r4, #1
 add r5, r5, r7
SKIP:
 seq r6, r7, r3       ! Look for terminating value.
 beqz r6, LOOP        ! Branch taken if value not found.
```

The questions below ask for scenarios in which certain things become execution bottlenecks or are operating well. The answers might specify the state of the cache (which lines are present), the values in registers, values loaded from memory, etc. as long as they are not specified in the comments and consistent with the code.

(*a*) Under what conditions would instruction fetch bandwidth limit performance? What is that worst-case fetch bandwidth? Under what conditions would instruction fetch bandwidth be at a maximum? What is the maximum fetch bandwidth? (8 pts) (*b*) Under what conditions would the round-robin thread selection policy limit performance? Which thread selection policy would provide better performance? (6 pts)

(*c*) Consider the conditions under which the round-robin selection limits performance as asked for above. Would the performance limit be as severe if the memory system could simultaneously service more than two cache misses? Explain. (6 pts)

Problem 3: In a *forwarding* directory coherence protocol a miss by one processor to a block in an exclusive state in another processor is serviced by having a message sent directly from the cache holding the block to the one needing it. (Other messages are sent; since they are part of the solution they can't be described here.) Add forwarding to the directory cache coherence protocol presented in the text. The new protocol should do the following:

- Forward for both read and write misses to exclusive blocks. On a read miss the exclusive block should be changed to shared; of course for a write miss the exclusive block should be changed to invalid.
- It's possible that a block to be forwarded is evicted just before a forwarding message arrives. The protocol should work correctly in this case.
- Be sure that a second read miss (at some other processor) after a block enters the exclusive state is handled correctly.
- Be sure that it is not possible to have two caches simultaneously holding exclusive copies of the same block for an unlimited amount of time. (*Hint: This might occur in an incomplete design if there are write misses at two different processors at the same time.*)

The solution should show new states and other information needed at the caches and memories, new protocol messages, and new state transitions. State transitions should indicate how the directory is changed.

Provide time diagrams showing states and messages sent for each of the situations indicated in the bulleted items above. Show the messages and states, but do not show times (*e.g.*, 12 cycles). (20 pts)

Problem 4: Answer each question below.

(*a*) Each line in the diagrams below shows, in program order, reads and writes issued by a processor. The letter in parenthesis indicates the address, the value written is shown with an arrow. Position indicates program order on one processor but not necessarily time or order between processors. Before the code is run address a holds 1, address b holds 2, address c holds 3, and address e holds 4. After this initialization the memory at these addresses is only changed by the writes shown below. (8 pts)

Indicate values returned by reads which could occur on a sequentially consistent memory system. Write the values next to the reads.

Proc. 1:	W(a)$\leftarrow$ 11	W(b)$\leftarrow$ 12	R(c)	W(e)$\leftarrow$ 14	
Proc. 2:	R(b)	R(a)	R(e)	W(c)$\leftarrow$ 13	R(b)

Indicate values returned by reads which could occur on a processor consistent (total store order) memory system but not a sequentially consistent memory system.

Proc. 1:	W(a)$\leftarrow$ 11	W(b)$\leftarrow$ 12	R(c)	W(e)$\leftarrow$ 14	
Proc. 2:	R(b)	R(a)	R(e)	W(c)$\leftarrow$ 13	R(b)

Indicate values returned by reads which could occur on a partial store order memory system but not a processor consistent system.

Proc. 1:	W(a)$\leftarrow$ 11	W(b)$\leftarrow$ 12	R(c)	W(e)$\leftarrow$ 14	
Proc. 2:	R(b)	R(a)	R(e)	W(c)$\leftarrow$ 13	R(b)

(*b*) Consider the following modification to the directory cache coherence protocol presented in the text. On a write to a block in the shared state, rather than invalidating other shared copies, the protocol will send the new value to each cache holding a shared copy. Is the memory system under this protocol coherent? If not, provide an example. (8 pts)

(*c*) A lock is implemented on a multiple-processor Tera MTA by a synchronizing store (using full/empty bits) and on a multiprocessor using store-conditional and load linked instructions:

```
TEST:
 ll   r2, 0(r1)
 bnez r2, TEST     ! If locked, check again.
 addi r2, r0, #1   ! It's not locked! Prepare a "locked" value.
 sc   r2, 0(r1)    ! Try to write locked value.
 beqz r2, TEST     ! If store failed, try again.
```

Suppose when the lock is held multiple processors simultaneously attempt to obtain the lock (and so must wait). Compare the amount of memory traffic generated on the two machines over time.

After a long interval, the processor holding the lock releases it. Describe what happens, including the amount of memory traffic generated, on the two machines. (8 pts) (*d*) Describe possible execution speed advantages or disadvantages of having smaller traces and larger traces in a trace processor, assuming hardware cost is kept roughly constant (same number of PEs, functional units, etc.). (8 pts)

(*e*) Both a finite context method data predictor and a stride data predictor could predict the value of a loop index. Give two advantages of a stride data predictor for predicting a loop index. Be specific. (8 pts)