School of Computer Science
Georgia Institute of Technology

CS4290/CS6290, Fall 2011
Hyesoon Kim, Instructor
Nagesh B Lakshminarayana, TA

November 10, 2011

Name :______________________________

T-square Account Name:______________________________

Problem 1 (15 points):________

Problem 2 (10 points):________

Problem 3 (10 points):________

Problem 4 (5 points):________

Problem 5 (10 points):________

Problem 6 (Extra 3 points):________

Total (50 points):____________

Note: Please make sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Note: Please make sure your GTID is recorded on each sheet of the exam.

GTID:______________________________

Problem 1 (15 points):

Part a. (5 points) Explain three metrics for measuring the effectiveness of prefetching mechanisms.

Part b. (5 points) Discuss whether write-allocate or no-write-allocate policy changes whether a processor can retire store instructions before it gets data from the main memory.

Part c. (5 points) DRAM has a destructive read characteristic. What does the DRAM controller do because of the destructive read characteristic?

GTID:______________________________

Problem 2 (10 points):

Part a. (5 points) Explain what false sharing is. Provide an example code that generates false sharing.

Part b. (5 points) These days some processors have limited power budget, so they cannot turn on all the cores with the highest frequency. Let's say a machine has 4 cores. When only 2 cores are active, the maximum frequency of the core is 2GHz. What will be the highest frequency when we use all 4 cores. To make the problem simple, we assume that power is always equally distributed among the cores.

GTID:______________________________

Problem 3 (10 points):

The following table shows the result profiling of a program.

FuncA	65%
FuncB	25%
FuncC	10%

Note that FuncA, FuncB, FuncC must be executed in sequence. Assume that if a parallelized section is executed on an n-core machine, the parallelized section is executed n times faster than the sequential version of the code. All the machines have IPC 1.

Part a. (3 points) After parallelizing only FuncA, we run the program on a 1GHz 128-core machine. What is the maximum speedup over 1GHz 1-core machine?

Part b. (7 points) Assume that only FuncA and FuncB are parallelized. Among the three machines in the below table, which one do you choose to run the program to get the best performance? For a heterogeneous machine, one function can run on only CPUs or GPUs. You must show the work.

Machine A	1GHZ-1 CPU and 500 MHz 512-core GPU
Machine B	2GHz 32-core CPU
Machine C	4GHz 1-core CPU and 1GHz 16-core GPU

GTID:______________________________

Problem 4 (5 points):

A computer has a 16KB write-back, write-allocate cache and 1MB physical memory. Each cache block is 64B, the cache is 4-way set associative and uses the true LRU replacement policy. Assume a 24-bit virtual address space and byte-addressable memory. The page size is 16KB. The cache uses a virtual tag and physical index. How big (in bits) is the tag store? The cache uses MSI protocol and assume that we need 2 bits to indicate cache coherence states.

GTID:________________________________

Problem 5 (10 points):

Assume that there are two processors and each processor has a cache which is described in the previous question. The processors use the MSI coherence protocol. The coherence stats are denoted M, S, and I for Modified, Shared, and Invalid.
A sequence of one or more CPU operations is specified as P#: op, address, ← value. P# designates the CPU (e.g., P0), op is the CPU operation (LDB: load ← byte, STB: store byte). address denotes the memory address, and value indicates the new word to be assigned on a write operation.
Each action occurs after the previous action is finished. (1) Show only the cache blocks that change, for example P0.B0: (I, 1, 0x01) indicates that CPU P0's block B0 (0 is the index number) has the final state of I, tag of 1, and data 0x1. Assume that the memory is initialized by all 0 at the beginning of the program. (2) Show the final state of the caches after all the following operations. (coherence state, cache tags, data, and index). To simplify the problem you do not need to draw the entire cache. You just need to fill out the relevant entries.

Processor	LDB/STB	addr	coherence
P0	LDB	0x88001773	
P0	STB	0x88001788 ← 0x4	
P1	STB	0x88001774 ← 0x5	
P1	LDB	0x88001772	
P1	STB	0x88001788 ← 0x6	
P0	LDB	0x88001773	
P1	LDB	0x88001770	

Processor 1's cache

Block ID	Coherence state	Tag	Data	Index

Processor 2's cache

Block ID	Coherence stage	Tag	Data	Index

GTID:________________________________

Problem 6 (Extra 3 points):

How do you insert prefetch requests for the following code? If you build a hardware prefetcher, what kind of prefetcher will you build?

```
// main function
.......
for (i=0; i < m; i++) {

    int val;

    val= test(val, i);

}

.....


int test(int val, int n)

{

 for (j = n; j < k; j++) {

  Node *node = tree[j];
  int cost1 = node->head->cost;
  int cost2 = node->tail->cost;

  return MAX3(cost1, cost2, val);

 }

}
```