Full Name:_____

Andrew ID (print clearly!):_____

740: Computer Architecture, Fall 2013

SOLUTIONS TO Midterm I

October 23, 2013

Instructions:

- Make sure that your exam has 15 pages and is not missing any sheets, then write your full name and Andrew login ID on the front.
- This exam is closed book. You may not use **any** electronic devices. You may use one **single-sided** page of notes that you bring to the exam.
- Write your answers in the box provided below the problem. If you make a mess, clearly indicate your final answer.
- Be concise. You will be penalized for excessive verbosity. Use no more than 15 words per answer, unless otherwise stated.
- The exam lasts 1 hour 20 minutes.
- The problems are of varying difficulty. The point value of each problem is indicated. Do not spend too much time on one question. Good luck!

Problem	Your Score	Possible Points
1		50
2		30
3		26
4		20
5		14
6 (BONUS)		(20)
Total		140 (+20)

Please read the following sentence carefully and sign in the provided box:

"I promise not to discuss this exam with other students until Friday, October 25."

Signature:

Problem 1: Potpourri (50 pts)

A) [8 pts] Amdahl's Law

What assumption is made by Amdahl's Law about the parallelizable fraction of a program?

That it is perfectly parallelizable.

What are the three major reasons why this assumption may not hold?

1) **Synchronization overhead**

2) **Load imbalance overhead**

3) **Resource sharing overhead**

B) [9 pts] Locking

Give three reasons why a lock may be required statically for program correctness but may not be needed dynamically?

1) **Threads may not update the shared data protected by the lock.**

2) **Threads may update disjoint parts of the shared data structure protected by the lock.**

3) **Threads may not contend for the lock.**

C) [10 pts] Memory Consistency

Consider the following statement:

"A sequentially consistent multiprocessor guarantees that different executions of the same multithreaded program produce the same architecturally-exposed ordering of memory operations."

1) Is this statement true or false?

CIRCLE ONE: **TRUE** **FALSE**

2) Explain your reasoning (less than 15 words).

Sequential consistency makes no guarantees across different executions. (It is about the ordering of operations within the *same* execution)

3) Why do we want the property described above; i.e., the property that "different executions of the same multithreaded program produce the same architecturally-exposed ordering of memory operations"?

Debugging ease.

D) [9 pts] SLE vs. TM

1) What is the major difference between speculative lock elision (SLE) and transactional memory (TM)?

TM requires the programmer to mark transactions. SLE preserves conventional lock based programming.

2) What benefit does TM provide that SLE does not?

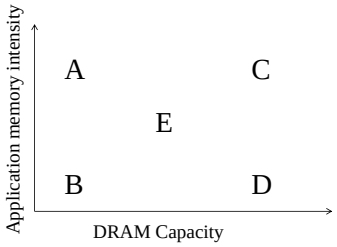
No need for reasoning about locks and getting them correct.

3) What benefit does SLE provide that TM does not?

No need to modify existing lock-based program.

E) [14 pts] A Refreshing Problem

Recall from lecture that RAIDR is a mechanism that uses profiled DRAM cell retention times to identify and skip unnecessary refreshes. Below is a plot with two independent variables—application memory intensity and DRAM capacity.



1) Identify the point (A, B, C, D, or E) where a mechanism like RAIDR would buy the most performance relative to a system without RAIDR.

CIRCLE ONE: **A** **B** **C** **D** **E**

CIRCLE ONE: **A** **B** **[C]** **D** **E**

Why (15 words or less)?

There is much DRAM to refresh and many memory accesses with which refreshes would contend.

2) Identify the point (A, B, C, D, or E) where the most energy is spent on refreshes (relative to total DRAM energy).

CIRCLE ONE: **A** **B** **C** **D** **E**

CIRCLE ONE: **A** **B** **C** **[D]** **E**

Why (15 words or less)?

There is much DRAM to refresh and few memory accesses, so refresh energy dominates.

Problem 2: Cache (30 pts)

Assume we have a dual-core processor with a 2MB shared set-associative last-level cache that has 64B cachelines and 4096 sets. The cache uses LRU cache replacement policy.

A) [2 pts] What is the associativity of the last-level cache (how many ways are there in a set)? Show your calculation.

8

Assume we have an OS which does not support virtual memory and all cores write directly to physical memory. Sleep-deprived Hongyi wrote a simple program called USELESS and he wants to run 2 USELESS processes simultaneously on the processor.

```
function USELESS(int processID)
{
  ADDRESS base = 1048576 * processID; // This is a physical address

  malloc(base, sizeof(BYTE) * 2097152); // Alloc some physical space

  int i = 0;

  while (1) {

    *(base + (i % 6) * 262144)++; // access physical memory directly (no virtual memory)

    i++;
  }
}
```

Hint—Power of 2 table:

2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}
2048	4096	8192	16384	32768	65536	131702	262144	524288	1048576	2097152

B) [4 pts] What is the steady-state last-level cache miss rate of a USELESS process when it is running alone on the system? Show your work.

0%

Continued . . .

C) [18 pts] In order to maximize the system performance running two USELESS processes, Hongyi considers three configurations:

- Static cache partitioning: Two processes execute concurrently on separate cores sharing the last-level cache and the cache is statically partitioned. Each process is assigned an equal number of ways in each set.
- Utility-based cache partitioning: Two processes execute concurrently on separate cores sharing the last-level cache and the cache is partitioned using the Utility Based Cache Partitioning mechanism that was discussed in class and that was also part of your required readings.
- Thread multiplexing: Both processes are active in the system but only one process executes at a given time. Every 1000 cache accesses, the other process is switched in. Assume that the context switch does not incur any overhead or extra cache accesses to switch in the other thread.

Calculate the steady-state cache miss rates of both processes for each configuration, and show your work:

1) Static cache partitioning:

100%
100%

2) Utility-based cache partitioning:

100%
0%

3) Thread multiplexing:

0.6%
0.6%

D) [6 pts] Hongyi also proposes a fourth configuration with virtual memory support:

- Page coloring: Two processes execute concurrently on separate cores sharing the last-level cache and the cache is partitioned using the Page Coloring mechanism that was discussed in class.

Calculate the steady-state cache miss rate of both processes for this new configuration, and show your work:

0%
0%

Problem 3: Research is Fun! (26 pts)

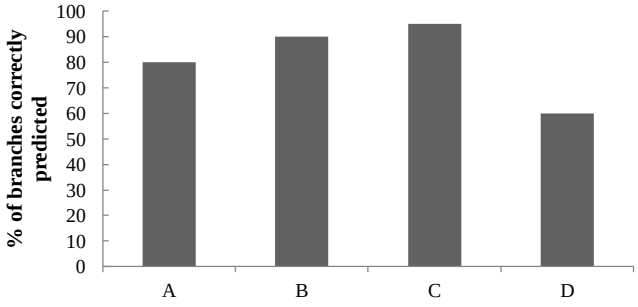
A researcher is studying compression in on-chip caches. You may assume it's very similar to Base-Delta-Immediate compression studied in this course, with an implementation that allows one base per cache line. She's considering four workloads: A, B, C, and D. She builds a baseline System X, without compression, and compares this against System Y, which is employing on-chip compression in the last-level cache (the modifications required to support compression, such as doubling the number of tags, are the only differences between System X and System Y).

- Figure 0 shows the branch prediction accuracy in System X of the four workloads.
- Figure 1 shows the accuracy of System X's stream prefetcher. As a reminder, a stream prefetcher identifies that the processor is continuously accessing consecutive cache lines (i.e., streaming) and prefetches future cache lines into the cache before the processor requests them.
- Figure 2 shows the misses per thousand instructions (MPKI) in the last-level cache (LLC) of System X.
- Figure 3 shows the effective cache capacity in the compressed LLC in System Y.
- Figure 4 shows the instructions per cycle (IPC) of System Y normalized to the IPC of System X.
- Figure 5 shows the normalized LLC MPKI of System Y normalized to the LLC MPKI of System X.

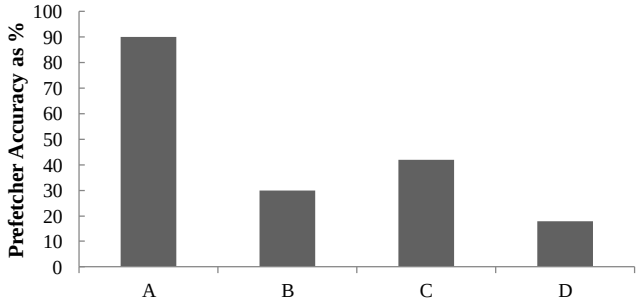
Answer the following questions, providing the most likely explanation for each considering the information provided in the figures.

Continued . . .

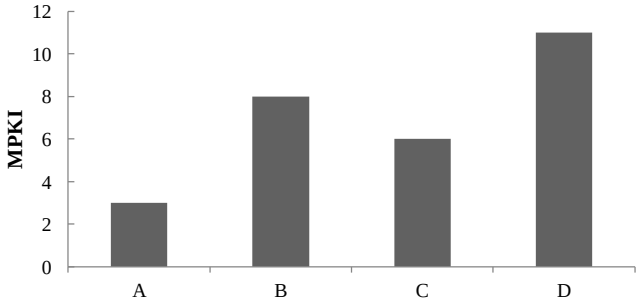
0. Branch prediction accuracy in System X



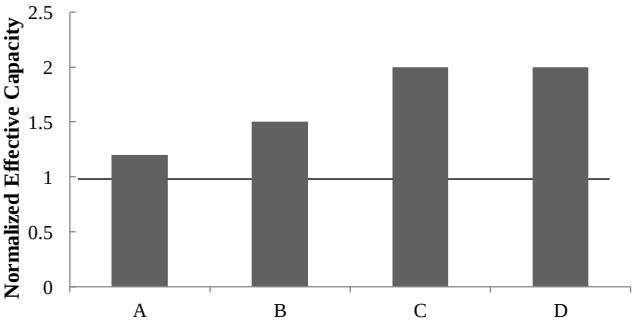
1. Prefetcher accuracy in System X



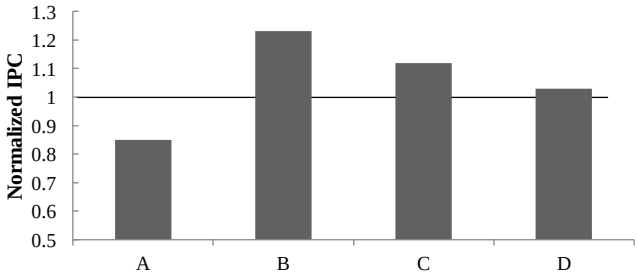
2. LLC MPKI in System X



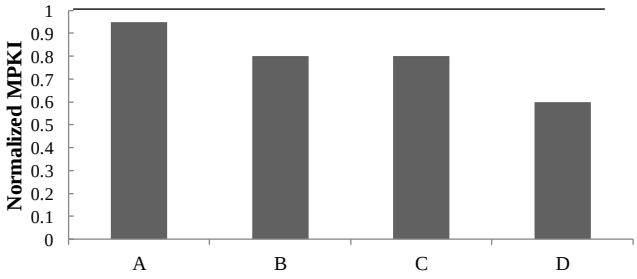
3. Effective LLC cache capacity in System Y



4. Performance (IPC) of System Y normalized to System X IPC



5. LLC MPKI in System Y normalized to LLC MPKI in System X



A) [4 pts] Why might the normalized IPC of workload A be less than 1.0? (15 words or less)

Decompression latency

B) [4 pts] Why might the normalized IPC of workload B be greater than the normalized IPC of workload C? (15 words or less)

B is more sensitive to cache size

C) [4 pts] Why might the normalized IPC of workload C be greater than the normalized IPC of workload D? (15 words or less)

C is more sensitive to cache size

Assume just for the next two subquestions that most of Workload C's data is of type Flow and most of Workload D's data is of type Node:

```
struct Flow { // defined in Workload C
  long flow_time;
  Pipe * inlet;
  Pipe * outlet;
  char identifier;
  float flow_rate;
}

struct Node { // defined in Workload D
  Node * right_sibling;
  Ancestor * parent;
  Descendent * child;
  Node * left_sibling;
  Node * metadata;
}
```

D) [4 pts] Just from looking at the above code, which workload's data is likely more compressible?

CIRCLE ONE: **C** **D**

Why? (15 words or less)

Pointers often have low dynamic range, so D has more compressible data.

E) [5 pts] For a new workload E, is it possible that the LLC MPKI in System Y is greater than the LLC MPKI in System X?

CIRCLE ONE: **YES** **NO**

Why or why not? (15 words or less)

Yes, depends on replacement policy.

Continued . . .

F) [5 pts] The results in the figures were determined through simulation. To increase simulation speed, the researcher was using a fixed 300 cycle latency for all memory requests. Now, she decides to model the DRAM system accurately. Across all workloads, the average memory access latency with this new model is 300 cycles. Which workload's performance in System X do you expect to change the most, compared to the old simulation results? State your assumptions for partial credit (15 words or less).

CIRCLE ONE: **A** **B** **C** **D**

A, high streaming prefetcher accuracy may indiciate good row buffer locality (average latency will be less than 300 cycles)

Problem 4: Cache coherence: MESI (20 pts)

Assume you are designing a MESI snoopy bus cache coherence protocol for write-back private caches in a multi-core processor. Each private cache is connected to its own processor core as well as a common bus that is shared among other private caches.

There are 4 input commands a private cache may get for a cacheline. Assume that bus commands and core commands will not arrive at the same time:

```
CoreRd:  Read request from the cache's core
CoreWr:  Write request from the cache's core
BusGet:  Read request from the bus
BusGetI: Read and Invalidate request from the bus (invalidates shared data in
the cache that receives the request)
```

There are 4 actions a cache can take while transferring to another state:

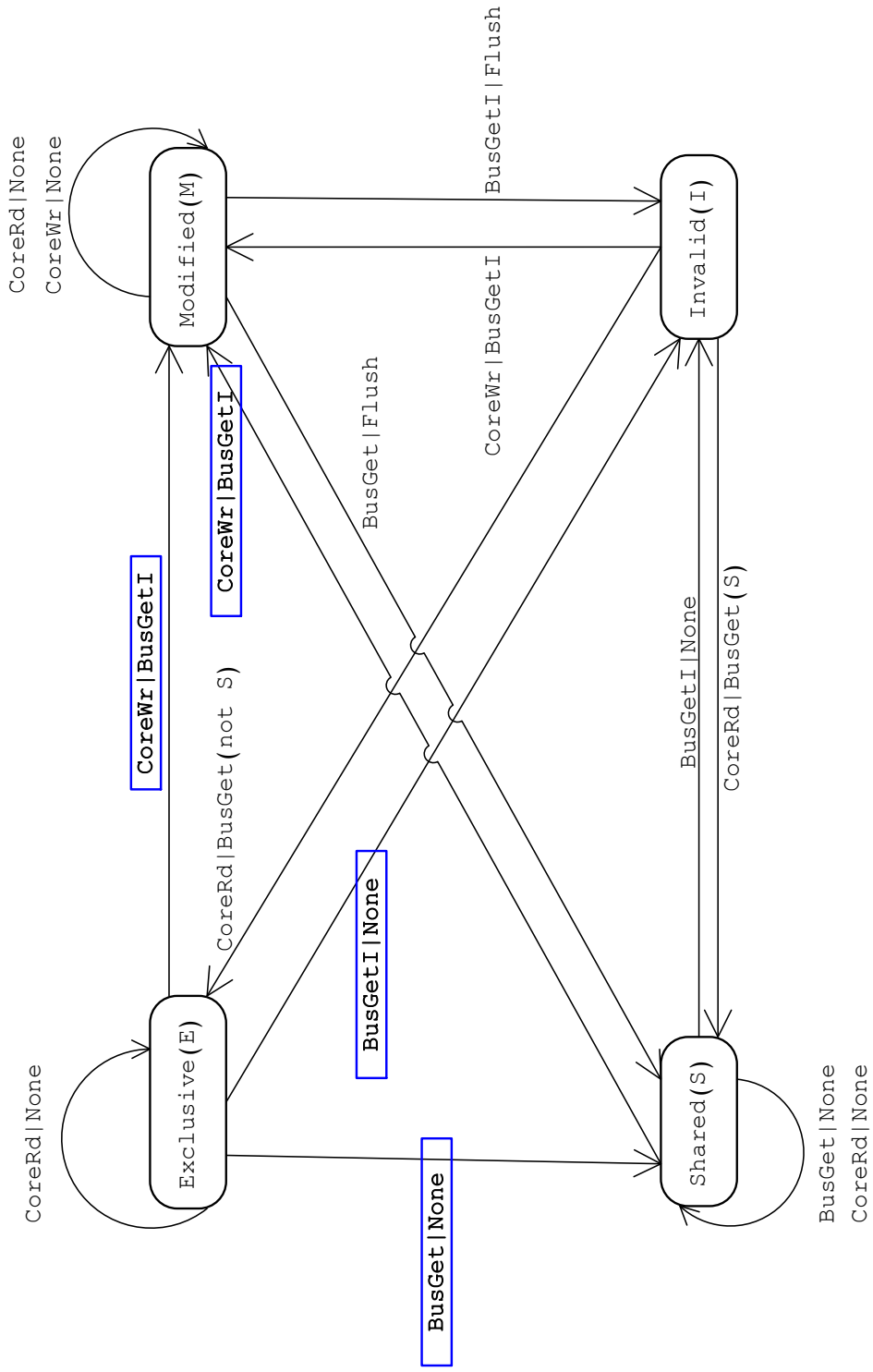
```
Flush:  Write back the cacheline to lower-level cache
BusGet:  Issue a BusGet request to the bus
BusGetI: Issue a BusGetI request to the bus
None:    Take no action
```

There is also an "is_shared (S)" signal on the bus. S is asserted upon a BusGet request when at least one of the private caches shares a copy of the data (BusGet (S)). Otherwise S is deasserted (BusGet (not S)).

Assume upon a BusGet or BusGetI request, the inclusive lower-level cache will eventually supply the data and there are no private cache to private cache transfers.

On the next page, Hongyi drew a MESI state diagram. There are 4 mistakes in his diagram. **Please show the mistakes and correct them.** You may want to practice on the scratch paper first before finalizing your answer. If you made a mess, clearly write down the mistakes and the changes below.

Continued . . .


Modified(M)
Invalid(I)
Exclusive(E)
Shared(S)
CoreRd|None
CoreWr|None
BusGetI|Flush
CoreWr|BusGetI
BusGet|Flush
CoreWr|BusGetI
CoreWr|BusGetI
CoreRd|BusGet(not S)
BusGetI|None
CoreRd|BusGet(S)
BusGetI|None
BusGet|None
CoreRd|None
BusGet|None
CoreRd|None

Problem 5: Heterogeneous Multicore (14 pts)

Suppose you have three different core designs which you can use to build a heterogeneous multicore system.

- An OoO core (OoO-Fixed) that can execute instructions out-of-order.
- An in-order core (IO-Fixed) that can execute instructions in order. The area of IO-Fixed is 1/4th the size of OoO-Fixed.
- Morphy: a hybrid core which can dynamically switch between two modes of execution: out-of-order with a single thread (OoO-Morphy) and in-order with 4 threads (IO-Morphy).

The implementations of out-of-order execution in OoO-Morphy and OoO-Fixed are the same, except OoO-Morphy requires the ability to switch between out-of-order and in-order modes. Likewise, the implementations of in-order execution in IO-Morphy and IO-Fixed are the same except for the ability to switch modes.

Answer the following:

A) [7 pts] The *peak single-threaded performance* of OoO-Morphy mode **could be** less than the *peak single-thread performance* of OoO-Fixed.

CIRCLE ONE: **TRUE** **FALSE**

Why? Explain your reasoning.

True, the OoO-Morphy may have a slower peak frequency due to extra logic for morphing.

B) [7 pts] Imagine a heterogeneous multicore system *Fixed* with 12 IO-Fixed cores and 1 OoO-Fixed core. Imagine a homogeneous multicore system *Morphy* with four Morphy cores.

Suppose we want to accelerate critical sections on both systems using an OoO core.

Could the same critical section that is accelerated run faster on System *Fixed* than on System *Morphy*? Why? Explain.

CIRCLE ONE: **YES** **NO**

True, from above, if OoO-Fixed has a higher peak frequency than OoO-Morphy.

Could the same critical section that is accelerated run faster on System *Morphy* than on System *Fixed*? Why? Explain.

CIRCLE ONE: **YES** **NO**

True, due to preservation of cache contents when IO-Morphy changes to OoO-Morphy.

Problem 6: Remember This (BONUS) (20 pts)

A researcher has developed a new type of nonvolatile memory, BossMem. He is considering BossMem as a replacement for DRAM. BossMem is 10x faster (all memory timings are 10x faster) than DRAM, but since BossMem is so fast, it has to frequently power-off to cool down. Overheating is *only a function of time*, not a function of activity—an idle stick of BossMem has to power-off just as frequently as an active stick. When powered-off, BossMem retains its data, but can't service requests. Both DRAM and BossMem are banked and otherwise architecturally similar. To the researcher's dismay, he finds that a system with 1GB of DRAM performs considerably better than the same system with 1GB of BossMem.

A) [4 pts] What can the researcher change or improve in the core (he can't change BossMem or anything beyond the memory controller) that will make his BossMem perform more favorably compared to DRAM, realizing that he will have to be fair and evaluate DRAM with his enhanced core as well? (15 words or less)

Prefetcher degree or other speculation techniques so that misses can be serviced before memory powered off

B) [4 pts] A colleague proposes he build a hybrid memory system, with both DRAM and BossMem. He decides to place data that exhibits low row buffer locality in DRAM and data that exhibits high row buffer locality in BossMem. Assume 50% of requests are row buffer hits. Is this a good or bad idea?

CIRCLE ONE: **GOOD** **BAD**

Show your work.

No, it may be better idea to place data with high row buffer locality in DRAM and low row buffer locality data in BossMem since row buffer misses are less costly

C) [4 pts] Now a colleague suggests trying to improve the last-level cache replacement policy in the system with the hybrid memory system. Like before, he wants to improve the performance of this system relative to one that uses just DRAM and he will have to be fair in his evaluation. Can he design a cache replacement policy that makes the hybrid memory system look more favorable?

CIRCLE ONE: **YES** **NO**

In 15 words or less, justify NO or describe a cache replacement policy that would improve the performance of the hybrid memory system more than it would DRAM.

Yes, this is possible. Cost-based replacement where cost to replace is dependent on data allocation between DRAM and BossMem

Continued . . .

D) [4 pts] In class we talked about another nonvolatile memory technology, phase-change memory (PCM). Which technology, PCM, BossMem, or DRAM requires the greatest attention to security?

CIRCLE ONE: **PCM** **BossMEM** **DRAM**

What is the vulnerability (less than 10 words)?

PCM is nonvolatile and has potential endurance attacks.

E) [4 pts] Which is likely of least concern to a security researcher?

CIRCLE ONE: **PCM** **BossMEM** **DRAM**

Why (less than 10 words)?

DRAM is likely least vulnerable, as BossMem also has nonvolatility concerns.