CMU 18-447 Introduction to Computer Architecture, Spring 2013

Midterm Exam 2

Date: Wed., 4/17

Instructor: Onur Mutlu
TAs: Justin Meza, Yoongu Kim, Jason Lin

Name:

Legibility & Name (5 Points):

Problem 1 (90 Points):

Problem 2 (35 Points):

Problem 3 (35 Points):

Problem 4 (50 Points):

Problem 5 (50 Points):

Problem 6 (55 Points):

Total (320 Points):

Instructions:

1. This is a closed book exam. You are allowed to have two letter-sized cheat sheets.
2. No electronic devices may be used.
3. This exam lasts 1 hour and 50 minutes.
4. Clearly indicate your final answer for each problem.
5. Please show your work when needed.
6. Please write your initials at the top of every page.
7. Please make sure that your answers to all questions (and all supporting work that is required) are contained in the space required.

Tips:

• **Be cognizant of time.** Do not spend too much time on one question.
• **Be concise.** You will be penalized for verbosity.
• **Show work when needed.** You will receive partial credit at the instructors’ discretion.
• **Write legibly.** Show your final answer.
1. Potpourri [90 points]

(a) **DRAM** [4 points]

What is the purpose of the precharge command in DRAM?

(b) **TLBs** [3 points]

What does a TLB cache?

(c) **SIMD Processing** [12 points]

Suppose we want to design a SIMD engine that can support a vector length of 16. We have two options: a traditional vector processor and a traditional array processor.

Which one is more costly in terms of chip area (circle one)?

- The traditional vector processor
- The traditional array processor
- Neither

Explain:

Assuming the latency of an addition operation is five cycles in both processors, how long will a `VADD` (vector add) instruction take in each of the processors (assume that the adder can be fully pipelined and is the same for both processors)?

For a vector length of 1:

- The traditional vector processor: ____________________________
- The traditional array processor: ____________________________

For a vector length of 4:

- The traditional vector processor: ____________________________
- The traditional array processor: ____________________________
For a vector length of 16:

The traditional vector processor: 

The traditional array processor: 

(d) **Page Replacement** [16 points]

Suppose we have a demand paging system with a physical memory of 4 GB and a virtual page size of 4 KB. An ambitious system software designer aims to implement true LRU as the page replacement policy. What is the minimum number of bits required to store the necessary information to enable this policy? Show your work.

Where should these bits be stored?

Suppose another engineer wants to use the Victim-NextVictim policy we discussed in class. What is the minimum number of bits required to store the necessary information to enable this policy? Show your work.

Which of the above policies performs better (circle one)?

- True LRU
- Victim-NextVictim
- It depends

Explain briefly your reasoning:
(e) **Cache Mystery** [15 points]

SlickMarketing, Inc. markets its newly-designed Super-Duper™ processor as having a 128 KB, 2-way cache with 5-cycle access latency. The cache block size is 64 bytes.

A programmer writes a program that repeatedly accesses (in a loop) only the following two unique cache blocks. The loop is executed for billions of iterations.

A, B

What do you expect the average memory access time to be?

In the steady state, the programmer measures the average memory access time to be 7.5 cycles.

What can you say about this processor? Please describe everything you can say, concretely, but be concise. Show your work.

(f) **Memory Channels** [8 points]

Assume that we have a byte-addressable processor with a 4 KB page size. The virtual address is 64 bits. The physical memory is 4 GB. The processor employs a 3-level cache hierarchy where cache block size is 256 bytes. The processor has 4 on-chip memory controllers, each controlling an independent memory channel, consisting of a 1 GB DIMM. Consecutive cache blocks in physical memory space are located in consecutive channels.

Say that the operating system wants to allocate the physical memory pages of an application to only two DIMMs on memory channels 0 and 1. Describe how the operating system can accomplish this. Be concise and concrete.
(g) **Bulk Synchronous Parallel Processing [20 points]**

Assume we have parallelized a program into eight threads. Each thread performs the *exact same computation* on a different part of a very large data set partitioned across the threads. The threads communicate at the end of the parallel portion by synchronizing at a barrier. After that, a single thread runs and aggregates the results generated by each parallel thread. Many parallel computations are structured in a similar way, a model called “Bulk Synchronous Parallel” processing.

You run this program on a multi-core processor with 8 cores, where the only shared resource between the cores is the memory controller and the DRAM system, and measure the following execution time, in seconds, for each thread. There is a single memory controller and it employs the FR-FCFS request scheduling policy.

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>7</td>
<td>10</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>

What could be three reasons T3 takes longer than others? The first one is given to you.

1. Interference in the memory controller that delays T3 more than others

2. 

3. 

Suppose you would like to tackle the first reason. How would you modify the memory controller to improve the program’s performance?

Now, say you have another program that works in the same way. You run this program on the same multi-core processor and measure the following execution time, in seconds, for each thread:

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

How would you change the scheduling policy in an attempt to improve the program’s performance? Explain concisely.
Why can your proposal improve performance? Be concise.

Can your proposal hurt performance (circle one)?

YES  NO

Why or why not?

(h) **Pointers [12 points]**

Assume we have a byte-addressable computer with a 64-bit virtual address space and a 4KB page size. Assume that the data is laid out such that it is aligned at the data type size boundaries.

The following shows the contents of the valid translations cached in the TLB for the running program:

<table>
<thead>
<tr>
<th>VPN</th>
<th>PFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0007577123451</td>
<td>0x00011</td>
</tr>
<tr>
<td>0x005abcd0123a1</td>
<td>0xC5789</td>
</tr>
</tbody>
</table>

The following shows the contents of a 64-byte L3 cache block starting at address 0xC5789100. The L3 is physically addressed.

0005abcd0123a100 005abcd01230aaa fffffffffffffffff 00000000C5789100
000000000011000 00000000C5789808 005abcd012000000 000757712340000d

(i) Identify and circle the values that are likely to be pointers.

(ii) Assume that a load instruction accesses this cache block. Describe a simple mechanism that detects in hardware which values in the cache block are pointers.
2. Cache Enigma [35 points]

A processor has a 4-way set-associative L1 cache that can house 4 blocks in total. The access latency to this cache is 1 cycle. The replacement policy is true LRU. The processor is known to not employ any prefetching mechanism.

The processor also has a 16-way set-associative L2 cache that can house 128 blocks in total. The access latency to this cache is 20 cycles.

A programmer writes a test program that in a loop repeatedly accesses only the following data cache blocks (assume billions of iterations are run):

A, B, C, D, E, F

where A, . . . , F are different cache block addresses.

In the steady state (i.e., after the loop has executed for a few iterations), the programmer finds out that the average memory access time is 1 cycle.

Then, the programmer writes another program that in a loop repeatedly accesses only the following data cache blocks:

A, B, C, D, E, F, G, H

In the steady state (i.e., after the loop has executed for a few iterations), the programmer finds out that the average memory access time is 20 cycles.

(a) What can you say about this processor? (I.e., what is going on?)

Please describe everything you can say, concretely, but be concise.

(b) Based on the above information, what do you expect the average memory access time of yet another program that in a loop repeatedly accesses only the following data cache blocks?

A, B, C, D, E

Explain:
(c) Again, based on the above information, what do you expect the average memory access time of yet another program that in a loop repeatedly accesses only the following data cache blocks?

A, B, C, D, E, F, G

Explain:

(d) Finally, again, based on the above information, what do you expect the average memory access time of yet another program that in a loop repeatedly accesses only the following data cache blocks?

A, B, C, D, E, F, G, H, I

Explain:
3. GPUs and SIMD [35 points]

We define the *SIMD utilization* of a program run on a GPU as the fraction of SIMD lanes that are kept busy with *active threads* during the run of a program.

The following code segment is run on a GPU. Each thread executes a **single iteration** of the shown loop. Assume that the data values of the arrays A, B, and C are already in vector registers so there are no loads and stores in this program. (Hint: Notice that there are 4 instructions in each thread.) A warp in the GPU consists of 64 threads, and there are 64 SIMD lanes in the GPU.

```c
for (i=0; i<1,024,768; i++) {
    if (A[i] > 0) {
        A[i] = A[i] * C[i];
        B[i] = A[i] + B[i];
        C[i] = B[i] + 1;
    }
}
```

(a) How many warps does it take to execute this program?

(b) When we measure the SIMD utilization for this program with one input set, we find that it is 67/256. What can you say about arrays A, B, and C? Be precise.

A:

B:

C:

(c) Is it possible for this program to yield a SIMD utilization of 100% (circle one)?

YES  NO
If YES, what should be true about arrays A, B, C for the SIMD utilization to be 100%? Be precise.

A: 

B: 

C: 

If NO, explain why not.

(d) Is it possible for this program to yield a SIMD utilization of 25% (circle one)?

YES  NO

If YES, what should be true about arrays A, B, and C for the SIMD utilization to be 25%? Be precise.

A: 

B: 

C: 

If NO, explain why not.
4. Memory Scheduling [50 points]

Row-Buffer Conflicts. The following timing diagram shows the operation of a single DRAM channel and a single DRAM bank for two back-to-back reads that conflict in the row-buffer. Immediately after the bank has been busy for 10ns with a READ, data starts to be transferred over the data bus for 5ns.

(a) Given a long sequence of back-to-back reads that always conflict in the row-buffer, what is the data throughput of the main memory system? Please state your answer in gigabytes/second.

(b) To increase the data throughput, the main memory designer is considering adding more DRAM banks to the single DRAM channel. Given a long sequence of back-to-back reads to all banks that always conflict in the row-buffers, what is the minimum number of banks that is required to achieve the maximum data throughput of the main memory system?
**Row-Buffer Hits.** The following timing diagram shows the operation of the single DRAM channel and the single DRAM bank for four back-to-back reads that hit in the row-buffer. It is important to note that row-buffer hits to the same DRAM bank are pipelined: while each `READ` keeps the DRAM bank busy for 10ns, up to at most half of this latency (5ns) can be overlapped with another read that hits in the row-buffer. (Note that this is different from Lab 6 where we unrealistically assumed that row-buffer hits are non-pipelined.)

![Timing Diagram](image)

(c) Given a long sequence of back-to-back reads that always hits in the row-buffer, what is the data throughput of the main memory system? Please state your answer in gigabytes/second.

(d) When the maximum data throughput is achieved for a main memory system that has a single DRAM channel and a single DRAM bank, what is the bottleneck that prevents the data throughput from becoming even larger? **Circle** all that apply.

<table>
<thead>
<tr>
<th>BANK</th>
<th>COMMAND BUS</th>
<th>ADDRESS BUS</th>
<th>DATA BUS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Memory Scheduling Policies.** The diagram below shows the memory controller’s request queue at time 0. The shaded rectangles are read requests generated by thread T0, whereas the unshaded rectangles are read requests generated by thread T1. Within each rectangle, there is a pair of numbers that denotes the request’s (BankAddress, RowAddress). Assume that the memory system has a **single** DRAM channel and **four** DRAM banks. Further assume the following.

- All the row-buffers are **closed** at time 0.
- Both threads start to stall at time 0 because of memory.
- A thread continues to stall until it receives the data for all of its requests.
- Neither thread generates more requests.

<table>
<thead>
<tr>
<th>(0,0)</th>
<th>(3,0)</th>
<th>(2,4)</th>
<th>(1,9)</th>
<th>(0,7)</th>
<th>(0,0)</th>
<th>(0,0)</th>
<th>(0,0)</th>
</tr>
</thead>
</table>

Youngest Oldest
(e) For the FCFS scheduling policy, calculate the memory stall time of $T_0$ and $T_1$.

$T_0$:

$T_1$:

(f) For the FR-FCFS scheduling policy, calculate the memory stall time of $T_0$ and $T_1$.

$T_0$:

$T_1$:
(g) For the PAR-BS scheduling policy, calculate the memory stall time of $T_0$ and $T_1$. Assume that all eight requests are included in the same batch.
5. Code Optimizations [50 points]

Assume an in-order, non-pipelined machine. The ISA for this machine consists of only the following five instructions that never generate exceptions. In this syntax, $x$ is a register, whereas $y$ and $z$ are either registers or constants. Note that this machine is able to execute branches and jumps instantaneously.

<table>
<thead>
<tr>
<th>Instruction Name</th>
<th>Syntax</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move</td>
<td>$x = y$</td>
<td>1</td>
</tr>
<tr>
<td>Addition</td>
<td>$x = y + z$</td>
<td>2</td>
</tr>
<tr>
<td>Multiplication</td>
<td>$x = y \cdot z$</td>
<td>5</td>
</tr>
<tr>
<td>Branch-If-Greater-Than</td>
<td>if ($y &gt; z$)</td>
<td>0</td>
</tr>
<tr>
<td>Jump</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Consider the following assembly program that is written in the machine’s ISA. For any initial values of registers $a$ and $b$, the purpose of the program is to compute the final value and store it in register $f$. After the program terminates, it is considered to have executed correctly if and only if the values stored in registers $a$, $b$, and $f$ are correct.
(a) How many cycles does the machine take to execute the assembly program, when \( a \) and \( b \) are initialized to 0 and 1, respectively?
You decide to make simple optimizations to the program to reduce its execution time. The optimizations involve only removing, modifying, and/or moving some of the already existing instructions. However, there are two restrictions: you may not move instructions out of the loop and you may not add completely new instructions.

(b) Show the optimized assembly program.

(c) How many cycles does the machine take to execute the optimized assembly program, when \( a \) and \( b \) are initialized to 0 and 1?
After learning about superblocks, you decide to optimize the program even further to reduce its execution time. In order to form the superblock(s), assume that you run the program once beforehand when $a$ and $b$ are initialized to 0 and 1. During this profile run, if a branch is biased in either direction by more than 60\%, it is included in the superblock(s). However, there are two restrictions: you may not move instructions out of the loop and you may not unroll the loop.

(d) Show the superblock-optimized assembly program. **Circle** the superblock(s).
(e) How many cycles does the machine take to execute the superblock-optimized assembly program, when $a$ and $b$ are initialized to 0 and 1?

(f) If you had used traces to optimize the program instead of superblocks, would the execution time increase, decrease, or stay the same compared to (e)? Choose one and explain briefly why.

(g) If you had used hyperblocks to optimize the program instead of superblocks, would the execution time increase, decrease, or stay the same compared to (e)? Choose one and explain briefly why.
6. Running Ahead [55 points]

Consider the following program, running on an in-order processor with no pipelining:

```
LD  R1  ←  (R3)  // Load A
ADD R2  ←  R4, R6
LD  R9  ←  (R5)  // Load B
ADD R4  ←  R7, R8
LD  R11 ←  (R16) // Load C
ADD R7  ←  R8, R10
LD  R12 ←  (R11) // Load D
ADD R6  ←  R8, R15
```

Assume that all registers are initialized and available prior to the beginning of the shown code. Each load takes 1 cycle to execute, and each add takes 2 cycles to execute. Loads A through D are all cache misses. In addition to the 1 cycle taken to execute each load, these cache misses take 6, 9, 12, and 3 cycles, respectively for Loads A through D, to complete. For now, assume that no penalty is incurred when entering or exiting runahead mode.

*Note: Please show all your work for partial credit.*

(a) For how many cycles does this program run without runahead execution?
(b) For how many cycles does this program run with runahead execution?

(c) How many additional instructions are executed in runahead execution mode?
(d) Next, assume that exiting runahead execution mode incurs a penalty of 3 cycles. In this case, for how many cycles does this program run with runahead execution?

(e) At least how many cycles should the runahead exit penalty be, such that enabling runahead execution decreases performance? Please show your work.
Which load instructions cause runahead periods? Circle all that did:

<table>
<thead>
<tr>
<th>Load A</th>
<th>Load B</th>
<th>Load C</th>
<th>Load D</th>
</tr>
</thead>
</table>

For each load that caused a runahead period, tell us if the period generated a prefetch request. If it did not, explain why it did not and what type of a period it caused.

<table>
<thead>
<tr>
<th>Load A:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load B:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load C:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load D:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

(g) For each load that caused a runahead period that did not result in a prefetch, explain how you would best mitigate the inefficiency of runahead execution.

<table>
<thead>
<tr>
<th>Load A:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load B:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load C:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load D:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
(h) If all useless runahead periods were eliminated, how many additional instructions would be executed in runahead mode?

How does this number compare with your answer from part (c)?
(i) Assume still that the runahead exit penalty is 3 cycles, as in part (d). If all useless runahead execution periods were eliminated (i.e., runahead execution is made efficient), for how many cycles does the program run with runahead execution?

How does this number compare with your answer from part (d)?

(j) At least how many cycles should the runahead exit penalty be such that enabling efficient runahead execution decreases performance? Please show your work.
Stratchpad
Initials:

Stratchpad
Stratchpad