CSE 141: Introduction to Computer Architecture Spring 2024

Final Examination

Instructor: Jishen Zhao **Due on:** June 7, 2024 @ 11:59PM *(96 points)*

Name: Andrew Onozuka
PID: A16760043
Email: ronozuka@ucsd.edu

Q1	6	
Q2	15	
Q3	12	
Q4	18	
Q5	10	
Q6	20	
Q7	15	
Total	**96**	

"While taking this examination, I have not witnessed any wrongdoing, nor have I personally violated any conditions of this course's integrity policy."

If you can honestly attest to the statement above, **write "I excel with Integrity" below and sign.**

If you do not write and sign, the instructor will contact you by e-mail to request you clarify/explain.

I excel with Integrity

Signature: Ryo Andrew Onozuka

Instructions:

- This exam is open book and open notes. Show your work and insert your answer in the space(s) provided. Please provide details on how you reach a result unless directed by the question as not to.
- The exam totals 96 points. It counts for 24% of your course grade. Please submit answers to the following questions as a PDF via Gradescope by June 7, 2024 at 11:59 PM. Policy of late submissions is the same homework assignments. Handwritten or typed responses are accepted.

Reminder:

- As a reminder, we will drop the lowest homework grade if class evaluation response rate is above 85%. This class will adopt the new SET (Student Evaluation of Teaching). You should have received an email with instructions on filling out the evaluation.

Q1 (6 points): Short answers (no further explanation needed).

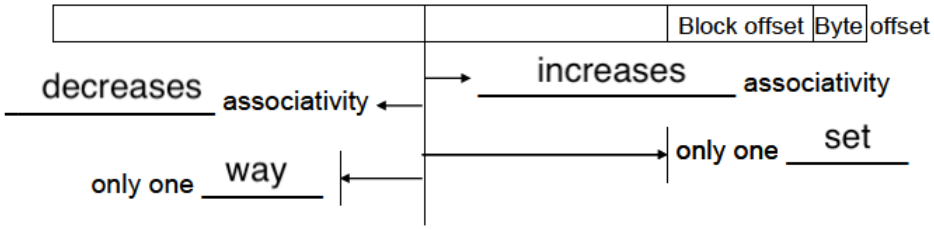
A. Allow cache and memory to be inconsistent, i.e., write the data **only** into the cache block. Is this write-back or write through?

write-back

B. Require cache and memory to be consistent, i.e., always write the data into both the cache block and the next level in the memory hierarchy. Is this write-back or write through?

write-through

C. Consider the diagram below that shows the dividing line between the bits used for tag compare and those used to select the cache set. Fill in the lines indicating whether the associativity **increases** or **decreases** and whether the resulting cache has only one **way** or only one **set**.



Q2 (15 points): Cache Performance

Assuming the base CPI_{base} (no stall) of a pipeline is 1. A program has 25% of load/store instructions. The processor only has one level of cache, i.e., an L1 instruction cache and an L1 data cache. Cache miss rate and penalty are as following:

- L1 instruction cache : $\%_{miss} = 2\%$, $t_{miss} = 30$ cycles
- L1 data cache: $\%_{miss} = 30\%$, $t_{miss} = 30$ cycles

What is the CPI?

$\text{CPI} = \text{CPI}_{base} + (\%_{accessed} \times \%_{miss} \times t_{miss})$
$\text{CPI} = 1 + (100\% \times 2\% \times 30 \text{ cycles/instruction}) + (25\% \times 30\% \times 30 \text{ cycles/instruction})$
$\text{CPI} = 1 + 0.6 + 2.25$

CPI = 3.85

Q3 (12 points): Cache tag overhead

Assume:

- A processor has a 64KB 4-way set associative cache
- The cache access uses physical addresses only
- A physical address is 48 bits long
- Each block holds 64 bytes of data
- Tag overhead includes the valid bit and tag bits

How much is the tag overhead in percent (no need to consider dirty bit)?

$\#\text{ of Sets} = \frac{\text{Cache Size}}{\text{Block Size} \times \text{Associativity}}$
$\#\text{ of Sets} = \frac{64\text{ KB}}{64\text{ bytes} \times 4\text{ way}}$
$\#\text{ of Sets} = \frac{64\text{ bytes} \times 1024}{64\text{ bytes} \times 4\text{ way}}$
$\#\text{ of Sets} = 256$
$\text{Index Bits} = log_2(256) = 8$
$\text{Block Offset Bits} = log_2(64) = 6$
Tag Bits = Address Bits - Index Bits - Block Offset Bits
Tag Bits = 48 - 8 - 6 = 34 bits
Total Tag Overhead/Block = Tag Bits + Valid Bit = 34 + 1 = 35
$\text{Total Tag Overhead}_{bits} = \#\text{ of Sets} \times \text{Associativity} \times \text{Total Tag Overhead/Block}$
$\text{Total Tag Overhead}_{bits} = 256 \times 4 \times 35$
$\text{Total Tag Overhead}_{bits} = 35840\text{ bits}$
$\text{Total Tag Overhead}_{bytes} = \frac{35840\text{ bits}}{8}$
$\text{Total Tag Overhead}_{bytes} = 4480\text{ bytes}$
$\text{Tag Overhead}_{percent} = \frac{\text{Total Tag Overhead}_{bytes}}{\text{Cache Size}} \times 100$
$\text{Tag Overhead}_{percent} = \frac{4480\text{ bytes}}{64\text{ KB}} \times 100$
$\text{Tag Overhead}_{percent} = \frac{4480\text{ bytes}}{65536\text{ bytes}} \times 100$

$$\boxed{\textbf{Tag Overhead}_{percent} = \textbf{6.8359375\%}}$$

Q4 (18 points): Multicore: Cache Coherence

Assuming write-back caches, private L1 data caches (L1Ds) (no L2), shared memory, and a MESI coherence policy, indicate in the cache tables what would be the values (V) of the variables X and Y and their coherency (C) state (either M, E, S or I) in the L1D's for a dual core processor with the following sequence of reads and writes. As shown in the table, the coherency (C) state in Step 0 (the initial coherency state) of both variables in both caches is I (invalid). If a value doesn't change, you may leave that entry blank. You may use either the state transition diagram or the table given below to reason about cache coherence.

Step 0: Initially, X =3, Y = 5 in the shared memory, processor caches are empty

Step 1: Core 1 reads X (from the shared memory)

Step 2: Core 2 reads X

Step 3: Core 1 writes X = 2

Step 4: Core 1 writes Y = 6

Step 5: Core 2 reads Y

Please fill in the following cache and shared memory tables (no further explanation for your answers is required).

Core 1's L1D Cache (private)												
	Step 0		Step 1		Step 2		Step 3		Step 4		Step 5	
	V	C	V	C	V	C	V	C	V	C	V	C
X	–	I	3	E	3	S	2	M	2	M	2	M
Y	–	I	–	I	–	I	–	I	6	M	6	S

Core 2's L1D Cache (private)												
	Step 0		Step 1		Step 2		Step 3		Step 4		Step 5	
	V	C	V	C	V	C	V	C	V	C	V	C
X	–	I	–	I	3	S	3	I	3	I	3	I
Y	–	I	–	I	–	I	–	I	–	I	6	S

Shared memory						
	Step 0	Step 1	Step 2	Step 3	Step 4	Step 5
	V	V	V	V	V	V
X	3	3	3	3	3	3
Y	5	5	5	5	5	6

Q5 (10 points): Memory consistency

Two threads (A and B) are concurrently running on a dual-core processor that implements a sequentially consistent memory model. Assume that the value at address (R10) is initialized to 0. The instruction `st immediate, (R10)` writes an immediate number into the memory address stored in R10.

Thread A (core 1)			
1:	st	0x1,	(R10)
2:	ld	R1,	(R10)
3:	st	0x2,	(R10)
4:	ld	R2,	(R10)

Thread B (core 2)			
1:	st	0x3,	(R10)
2:	ld	R3,	(R10)
3:	st	0x4,	(R10)
4:	ld	R4,	(R10)

After both threads have finished executing, you find that (R1, R2, R3, R4) = (1, 2, 3, 4). How many different instruction orderings of the two threads produce this result (please show all possible orderings)?

6 different instruction orderings

[1] $(A_1, A_2), (A_3, A_4), (B_1, B_2), (B_3, B_4)$

[2] $(A_1, A_2), (B_1, B_2), (A_3, A_4), (B_3, B_4)$

[3] $(A_1, A_2), (B_1, B_2), (B_3, B_4), (A_3, A_4)$

[4] $(B_1, B_2), (A_1, A_2), (A_3, A_4), (B_3, B_4)$

[5] $(B_1, B_2), (A_1, A_2), (B_3, B_4), (A_3, A_4)$

[6] $(B_1, B_2), (B_3, B_4), (A_1, A_2), (A_3, A_4)$

Q6 (20 points): Main Memory

Given a system with 2 memory channels and 4 DRAM DIMMs (2 DIMMs per channel), each DIMM has:

- 1 rank per DIMM
- 8 chips per rank
- 8 bits per column
- 8 banks per chip
- 32,768 rows per bank
- 2,048 columns per bank

A) What is the total amount (bytes) of physical memory in the system?

Memory per chip = Banks per chip × Rows per bank × Columns per bank × Bits per column
Memory per chip = 8 × 32768 × 2048 × 8 = 4 Gbits
Memory per DIMM = Memory per chip × Ranks per DIMM × Chips per rank
Memory per DIMM = 4 Gbits × 1 × 8 = 32 Gbits
Total physical memory = Memory per DIMM × DIMMs
Total physical memory = 32 Gbits × 4 = 128 Gbits = 16 GBytes

16 GBytes

B) What is the minimum number of physical address bits needed to address this much memory?

Minimum number of physical address bits needed = log_2(16 GB)
Minimum number of physical address bits needed = $log_2(16 \times 1024 \times 1024 \times 1024)$

Minimum number of physical address bits needed = **34 bits**

C) With the number of physical address bits obtained in B), also assume

- The physical address space has 1M (i.e., 1048576) pages (physical frames)
- Virtual addresses have 64 bits

What is the maximum number of pages in the virtual address space?

Page size = $\frac{\text{Total memory}}{\text{Number of pages}} = \frac{17{,}179{,}869{,}184 \text{ bytes}}{1{,}048{,}576}$ = 16,384 bytes

Number of pages in virtual address space = $\frac{2^{64}}{\text{Page size}} = \frac{2^{64}}{16{,}384} = \frac{2^{64}}{2^{14}} = 2^{50}$

The maximum number of pages in the virtual address space is $\mathbf{2^{50}}$ **pages**.

Q7 (15 points): GPUs

A) Which ones of the following are the basic ideas of GPU processor architecture design? (Multiple selections, no further explanation needed.)

- ✓ Interleave processing of many fragments on a single core to avoid stalls caused by high latency operations
- × Make common case fast
- ✓ Remove components that help a single instruction stream fast
- ✓ Amortize cost/complexity of managing an instruction across many ALUs

A,C,D

B) Assume a hypothetical GPU with the following characteristics:

- Clock rate 1 GHz
- 32 simultaneous instruction streams
- Contains 8 SIMD cores, each containing 32 single-precision floating-point units, each instruction performs one single-precision floating-point operation (either a multiply or an add)
- 1024 concurrent (but interleaved) instruction streams

What is the peak single-precision floating-point throughput for this GPU in GFLOP/sec, assuming that all memory latencies can be hidden?

1 GHz $\times$ 8 $\times$ 32 = **256 GFLOPS/sec**