CMPE 110 Computer Architecture
Fall 2016, Homework #3

Computer Engineering 110

UC Santa Cruz

November 16, 2016

Name: **Seongwoo Choi**
Email: scho29@ucsc.edu

Submission Guidelines:

- **This homework is due on Friday 11/16/2016 by 11:59pm.**
- **The homework must be submitted only on ecommons. Late submission are accepted on eCommons till 11/16/2016, 11:59pm with penalty. Do not email your HW to the instructor or TAs.**
  - **Please write your name and your UCSC email address**
- **The homework should be "readable" without too much effort**
  - **The homework must be typed and submitted as a single file in PDF or .doc format**
  - **Please name your homework file cmpe110-hw3-yourcruzid.pdf**
- **Provide details on how to reach a solution. An answer without explanation gets no credit. Clearly state all assumptions.**

- **Points: 64 = 30 + 15 + 19**

Question	Part A	Part B	Part C	Part D	Part E	Part F	Total
1							
2				-	-	-	
3					-	-	
4					-	-	
Total							

Seongwoo Choi
Homework 3
(Student ID: 1368039)
(scho29)

Question 1. Cache Mapping and Access (24 points)

Consider a 512-KByte cache with 64-word cachelines (a cacheline is also known as a cache block, each word is 4-Bytes). This cache uses write-back scheme, and the address is 32 bits wide.

Question 1.A Direct-Mapped, Cache Fields (4 points)

Assume the cache is direct-mapped. Fill in the table below to specify the size of each address field.
Field Size (bits)

Field	Size (bits)
Cacheline Offset	8
Cacheline Index	11
Tag	13

Offset = $\log_2(64 \times 4) = 8$ bits. Index = $[\log_2 (512\ KB/\ 64*4)] = 11$ bits. Tag = 32 - 8 - 11 = 13 bits.

Question 1.B Fully-Associative, Cache Fields (4 points)

Assume the cache is fully-associative. Fill in the table below to specify the size of each address field.

Field	Size (bits)
Cacheline Offset	8
Cacheline Index	0
Tag	24

Offset = $\log_2(64 \times 4) = 8$ bits. Same as in part A. Fully-associative caches do not use the index field. Tag = 32 - 8 = 24 bits.

Question 1.C 16-Way Set-Associative, Cache Fields (4 point)

Assume the cache is 16-way set-associative. Fill in the table below to specify the size of each address field.

Field	Size (bits)
Cacheline Offset	8
Cacheline Index	7
Tag	17

Offset = [log2 (64 - 4)] = 8 bits. Same as in part A. Index = [log2 (512 KB/16x64x4)] = 7 bits.

Tag = 32 - 8 - 7 = 17 bits.

Question 1.D Direct-Mapped, Cache Transactions (4 points)

Assume the cache is direct-mapped. Fill in the table on the next page to identify the content of the cache after each of the following memory accesses. Assume the cache is initially empty (also called a cold cache"). Specify if an entry causes another line to be replaced from the cache, and if an entry has to write its data back to memory. For the data column, specify the data in the block by referring to its address like M[address]. Write accesses will modify the data, so let's indicate the data after a write access with D[address].

Address	Request Type	Cache line Index	Hit or Miss?	Modified	Tag	Data	Caused Replace?	Write-back to Memory?
0x128	read	0x1	Miss	0	0x0	M[0x100]	no	no
0xF40	write	0xf	Miss	1	0x0	D[0xF00]	no	no
0xA00051	read	0x0	Miss	0	0x14	M[0xA00000]	no	no
0x093	write	0x0	Miss	1	0x0	D[0x000]	yes	no
0x4000B44	read	0xb	Miss	0	0x80	M[0x4000B00]	no	no

Question 1.E 16-Way Set-Associative, Cache Transactions (4 points)

Assume the cache is 16-way set-associative. Fill the table below to identify the content of the cache after each of the following memory accesses. Assume the cache is empty in the beginning (also known as cold cache). Specify if an entry causes another line to be replaced from the cache, and if an entry has to write its data back to memory. For the data column, specify the data in the block by referring to its address like M[address]. Write accesses will modify the data, so lets indicate the data after a write access with D[address].

Address	Request Type	Cacheline Index	Hit or Miss?	Modified	Tag	Data	Caused Replace?	Write-back to Memory?
0x128	read	0x1	Miss	0	0x0	M[0x100]	no	no
0xF40	write	0xf	Miss	1	0x0	D[0xF00]	no	no
0xA00051	read	0x0	Miss	0	0x140	M[0xA00000]	no	no
0x093	write	0x0	Miss	1	0x0	D[0x000]	no	no
0x4000B44	read	0xb	Miss	0	0x800	M[0x4000B00]	no	no

Question 1.F Overhead (4 points)

What is the overhead and actual size of the direct-mapped cache? What is the overhead and actual size of the 16-way set-associative cache? Does the structure change the overhead in terms of number of memory bits?

Direct-mapped:
Cache line size: no. of words x size of word x 8 bits/byte = 2048 bits

Overhead: (Tag + Valid + Modified) x no. of cache lines = 13 + 1 + 1 = 15 bits per cache line 2048 cache lines.

Actual Size = (no. of cache lines x size of cache line) + (no. of cache lines x overhead field size) = (2048 x 2048) + (2048 x 15)

Final Answer in terms of bits: 4194304 + 30720 = 4225024 bits

16-way Set-Associative:
Cache line size: no. of words x size of word x 8 bits/byte = 2048 bits

Overhead: (Tag + Valid + Modified + Lowest LRU bits for 16-way replacement policy) = 13 + 1 + 1 + 4 = 19 bits per cache line 2048 cache lines.

Actual Size = (no. of cache lines x size of cache line) + (no. of cache lines x other fields size) = (2048 x 2048) + (2048 x 19)

Final Answer in terms of bits: 4194304 + 38912 = 4233216 bits

The answer is yes because in this case the structure of the cache changes the overhead. It is changing because the size of the tags are different and set-associative cache needs state elements to keep track of replacement policy, whereas in direct-mapped, it does not need to be.

Question 2. Average Memory Access Time (12 points)

Considered two pipelined machines A and B, described in the table below.

Property	Computer A	Computer B
ISA	MIPS	x86
Clock Frequency	2 GHz	2.5 GHz
Base CPI	1 cycle/instruction	2 cycles/instruction
L1 Instruction Cache Hit Time	1 cycle	1 cycle
L1 Instruction Cache Miss Rate	2%	2%
L1 Data Cache Hit Time	1 cycle	2 cycles
L1 Data Cache Miss Rate	8%	4%
L2 Hit Time	15 cycles	12 cycles
L2 Global Miss Rate	3%	4%
Main Memory Access Time	250 cycles	300 cycles

Seongwoo Choi
Homework 3
(Student ID: 1368039)
(scho29)

Question 2.A Ideal System (4 points) (Visited Lawrence Chong's MSI session)
Assume a perfect memory system (100% of memory accesses hit in the L1 caches) and perfect branch prediction. Assume that MIPS programs execute 1.25x as many instructions as x86 programs. Which machine has the faster execution time and what is the speedup?

In this case, we need to assume in an idea system that there is no cache misses, and systems behave according to base CPI.

(Execution Time)B = IC × CPI × clock cycle time = ICB × 2 × 400 ps

(Execution Time)A = IC × CPI × clock cycle time = ICA × 1 × 500 ps = (1.25 × ICB) × 1 × 500 ps

Speedup = (Execution Time)B / (Execution Time)A

Speedup = $\frac{2\times400}{1.25\times1\times500}$ = 1.6

Computer A is faster by 60%

Question 2.B No L2 Cache (4 points)
Now assume each machine has only L1 caches (split iL1 cache and dL1 cache), no L2 cache, perfect branch prediction, and that 30% of the instructions are loads/stores. Which machine has the faster execution time and what is the speedup?

CPI = Base CPI +$stall_{IL1}$+$stall_{DL1}$ = ($missrate_{IL1}$ x $miss\ penalty_{IL1}$)+(30%($missrate_{DL1}$ x $misspenalty_{DL1}$)

The miss penalty for IL1 and IL2 is to access main memory.

CPI_A = (1 + 0.02 x 250 + 0.3 x 0.08 x 250) x 1.25= 12 cycles/instruction

CPI_B = (2 + 0.02 x 300 + 0.3 x 0.04 x 300) = 11.6 cycles/instruction

(Execution Time)B = IC × CPI × clock cycle time = ICB × 12.5 × 400 ps

(Execution Time)A = IC × CPI × clock cycle time = ICA × 12 × 500 ps = (1.25 × ICB) × 12 × 500 ps

Speedup = $(Execution\ Time)_A$ / $(Execution\ Time)_B$

Speedup = $\frac{1.25\times12\times500}{11.6\times400}$ = 1.61637931

Computer B is faster by 61.64%

Question 2.C Full System (4 points) (Visited Lawrence Chong's MSI session for help)
Now assume each machine has both L1 caches and a unied L2 cache, and that 30% of the instructions are loads/stores. Which machine has the faster execution time and what is the speedup?

CPI = Base CPI + $miss\ rate_{L1}$ x ($hit\ time_{L2}$ + $miss\ rate_{L2local}$ + $miss\ penalty_{L2}$)

Miss $rate_{L1}$ should include both IL1 and DL1 accesses.

CPIA = 1 + 0.02 x (15 + 0.03 x 250) + 0.3 x 0.08 x (15 + 0.03 x 250) = 1.99

CPIB = 2 + 0.02 x (12 + 0.04 x 300) + 0.3 x 0.04 x (12 + 0.04 x 300) = 2.768

(Execution Time)B = IC x CPI x clock cycle time = IC_B x 2.768 x 400 ps

(Execution Time)A = IC x CPI x clock cycle time = IC_A x 1.99 x 500 ps = (1.25 x IC_B) x 1.99 x 500 ps

Speedup = (Execution Time)A / (Execution Time)B

Speedup = $\frac{1.25 \times 1.99 \times 500}{2.768 \times 400}$

Computer B is faster by 12.33%

Question 3. Cache and AMAT (12 points) (deleted)

A processor has two levels of caches:

A 2-way set-associative L1 cache that can house 4 blocks in total. The access latency

to this cache is 1 cycle.

A 16-way set-associative L2 cache that can house 128 blocks in total. The access latency

to this cache is 20 cycles.

Both L1 and L2 caches employ LRU replacement policy. The processor does not employ any prefetching mechanism.

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

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 large number of iterations), the programmer observes that the average memory access time (AMAT) is 1 cycle.

Then, the programmer writes another program that repeatedly accesses only the following

data cache blocks in a loop:

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

In the steady state, the programmer observes that the AMAT is 20 cycles.

Question 3.A (3 points)

Why do the two programs have different AMAT? Please explain.

In this problem, we can say that the L1 cache can hold only 4 blocks. Therefore, a victim cache with an access latency of 1 cycle is present. This victim cache is accessed in parallel with the access to L1 cache. Cache

blocks evicted from the L1 cache are inserted into the victim cache. Based on the assumption of the access latencies from above, we can conclude that the victim cache can hold either 2 or 3 cache blocks.

Question 3.B (3 points)

Based on the above information, what do you expect the average memory access time of yet another program that repeatedly accesses only the following data cache blocks in a loop?
A, B, C, D, E Please explain.

The answer is 1 cycle. This is because that the 5 cache blocks from above can all fit in the L1 cache and victim cache, and therefore all cache block accesses are hits in the steady state.

Question 3.C

Again, based on the above information, what do you expect the average memory access time of yet another program that repeatedly accesses only the following data cache blocks in a loop? A, B, C, D, E, F, G Please explain.

The answer for this problem is 'either 1 or 20 cycles'. If the victim cache holds 2 blocks, the 7 cache blocks from above cannot all fit in the two caches, resulting in all accesses being misses. If the victim cache holds 3 blocks, the 7 cache blocks from above can all fit in the two caches, resulting in all accesses being hits.

Question 3.D (3 points)

Finally, based on the above information, what do you expect the average memory access time of yet another program that repeatedly accesses only the following data cache blocks in a loop? A, B, C, D, E, F, G, H, I. Please explain.

The answer is 20 cycles because we know that it is impossible for all 9 cache blocks to fit in the two caches. Therefore, all accesses are misses in the steady state.

Question 4. Caching Basics (16 points)

Below, we have given you four different sequences of addresses generated by a program running on a processor with a data cache. Cache hit ratio for each sequence is also shown below.

Sequence No.	Address Sequence	Hit Ratio
1	0, 2, 4, 8, 16, 32, 64, 128	0.50
2	0, 1024, 2048, 3072, 4096, 3072, 2048, 1024, 0	0.33
3	0, 256, 512, 1024, 2048, 1024, 512, 256, 0	4/9
4	0, 512, 2048, 0, 1536, 0, 1024, 512	0.25

Seongwoo Choi
Homework 3
(Student ID: 1368039)
(scho29)

Assumptions: all memory accesses are one byte accesses. All addresses are byte addresses. Assuming that the cache is initially empty at the beginning of each sequence, nd out the following parameters of the processor data cache:

1) Associativity (1, 2 or 4 ways) (4 points)

2) Block size (1, 2, 4, 8, 16, or 32 bytes) (4 points)

3) Total cache size (1024 B or 2048 B) (4 points)

4) Replacement policy (LRU or FIFO) (4 points)

Please justify (explain) your answers.

Associativity - 4

In this case, there are blocks in 0, 512, 1024 and 1536. We know that these are the only ones that are reused in this cycle. Also, they could potentially result in cache hits by the time when they are accessed the second time. Among these blocks, three of these four blocks should hit in the cache by the time they are accessed for the second time to give a hit rate of 0.33. In that moment, if we assume that the block size is 8 and for either cache size (in this case either 1024B or 2048B), then we can say that all of these blocks map will set to 0. Furthermore, an associativity of 1 or 2 would cause at most one or two of these four blocks to be present in the cache when they are accessed for the second time, resulting in a maximum possible hit rate of less than 3/9. However, because we know that the hit rate for this sequence is 3/9 we can conclude that an associativity of 4 is the only one that could potentially give a hit rate of 0.33.

Cache block size - 32 bytes

In this second sequence of the problem, only 2 out of the 6 accesses (specifically those to addresses 2 and 4) can hit in the cache, as the hit ratio is 0.50. With any other cache block size but 32 bytes, we can say that the hit ratio is either smaller or larger than 0.50. Therefore, we can conclude that the cache block size is 32 bytes.

0	2	4	8	16	32	64	128	50%
m	h	h	h	h	m	m	m	

Total cache size - 2048 B

In the third sequence, we can conclude that 2048 B amount of a total cache or size of that total cache will result in a hit rate of 4/9 with a 4-way associative cache and 8 byte blocks regardless of the replacement policy. Therefore, we can say that the total cache size is 2048 bytes.

0	256	512	1024	2048	1024	512	256	0
0	1	0	0	0	1	1	1	0

Replacement policy - LRU

For this specific aforementioned cache parameters, we can say that all chae lines in sequence 4 map to set 0. In a hypothetical situation when a FIFO replacement policy were used, then the hit ratio would be 3/8 compared to an LRU replacement policy were used then the hit ratio would be ¼. Therefore, the replacement policy should be LRU.

0	512	2048	0	1536	0	1024	512	25%
m	m	m	h	m	h	m	m	