CMPE 110 Computer Architecture
Fall 2016, Homework #3

Computer Engineering
UC Santa Cruz

November 2, 2016

Question 1. Cache Mapping and Access (16 points)

SOLUTION:

Question 1.A Direct-Mapped, Cache Fields (2 points)

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

Offset = $\lceil \log_2(64 \times 4) \rceil = 8$ bits.
Index = $\lceil \log_2 \frac{512KB}{64 \times 4} \rceil = 11$ bits.
Tag = 32 - 8 - 11 = 13 bits.

Question 1.B Fully-Associative, Cache Fields (2 points)

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

Offset = $\lceil \log_2(64 \times 4) \rceil$ = 8 bits.
Fully-associative caches do not use the Index field.
Tag = 32 - 8 = 24 bits.

Question 1.C 16-Way Set-Associative, Cache Fields (2 point)

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

Offset = $\lceil \log_2(64 \times 4) \rceil$ = 8 bits.
Index = $\lceil \log_2 \frac{512KB}{16\times 64\times 4} \rceil$ = 7 bits.
Tag = 32 - 8 - 7 = 17 bits.

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

Address	Request Type	Cachline 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)

Address	Request Type	Cachline 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 (2 points)

Directed-mapped:

Cacheline size: no. of words × size of word × 8 bits/byte = 2048 bits.
Overhead: (Tag + Valid + Modified) × no. of cache lines = 13 + 1 + 1 = 15 bits per cacheline × 2048 cache lines.
Actual Size = (no. of cachelines × size of cacheline) + (no. of cachelines × overhead fields size) = (2048 × 2048) + (2048 × 15).
Final Answer in terms of bits: 4194304 + 30720 = 4225024 bits.
Final Answer in terms of Bytes: 512 KB + 3.75 KB = 515.75 KB.

16-way Set-Associative:

Cacheline size: no. of words × size of word × 8 bits/byte = 2048 bits.
Overhead: (Tag + Valid + Modified + Lowest LRU bits for 16-way replacement policy) = 13 + 1 + 1 + 4 = 19 bits per cacheline × 2048 cache lines.
Actual Size = (no. of cachelines × size of cacheline) + (no. of cachelines × other fields size) = (2048 × 2048) + (2048 × 19).
Final Answer in terms of bits: 4194304 + 38912 = 4233216 bits.
Final Answer in terms of Bytes: 512 KB + 4.75 KB = 516.75 KB.

Yes, the structure of the cache changes the overhead because the size of the tags are different and set-associative cache needs state elements to keep track of replacement policy, whereas direct-mapped does not.

Question 2. Average Memory Access Time (12 points)

SOLUTION:

Question 2.A Ideal System (4 points)

In an idea system with no cache misses, systems behave according to base CPI.
(Execution Time)$_A$ = IC $\times$ CPI $\times$ clock cycle time = IC$_A$ $\times$ 1 $\times$ 500 ps = (1.25 $\times$ IC$_B$) $\times$ 1 $\times$ 500 ps.
(Execution Time)$_B$ = IC $\times$ CPI $\times$ clock cycle time = IC$_B$ $\times$ 2 $\times$ 400 ps.
Speedup = (Execution Time)$_B$ / (Execution Time)$_A$.
Speedup = $\frac{2\times400}{1.25\times1\times500}$ = 1.28.
Computer A is faster by 28%.

Question 2.B No L2 Cache (4 points)

Since Base CPI of Computer B is 2 cycles/instructino, L1 data cache hit time being 2 cycles does not stall the pipeline.
CPI = Base CPI + stall$_{IL1}$+ stall$_{DL1}$ = Base CPI + (miss rate$_{IL1}$ $\times$ miss penalty$_{IL1}$) + (30% $\times$ (miss rate$_{DL1}$ $\times$ miss penalty$_{DL1}$).
The miss penalty for IL1 and IL2 is to access main memory.
CPI$_A$ = 1 + 0.02 $\times$ 250 + 0.3 $\times$ 0.08 $\times$ 250 = 12 cycles/instruction.
CPI$_B$ = 2 + 0.02 $\times$ 300 + 0.3 $\times$ 0.04 $\times$ 300 = 11.6 cycles/instruction.
(Execution Time)$_B$ = IC $\times$ CPI $\times$ clock cycle time = IC$_B$ $\times$ 11.6 $\times$ 400 ps.
(Execution Time)$_A$ = IC $\times$ CPI $\times$ clock cycle time = IC$_A$ $\times$ 12 $\times$ 500 ps = (1.25 $\times$ IC$_B$) $\times$ 12 $\times$ 500 ps.
Speedup = (Execution Time)$_A$ / (Execution Time)$_B$.
Speedup = $\frac{1.25\times12\times500}{11.6\times400}$ = 1.6164.
Computer B is faster by 61.64%.

Question 2.C Full System (4 points)

CPI = Base CPI + miss rate$_{L1}$ $\times$ (hit time$_{L2}$ + miss rate$_{L2}$ $\times$ miss penalty$_{L2}$).
miss rate$_{L1}$ should include both IL1 and DL1 accesses.
miss rate$_{L1}$ = miss rate$_{IL1}$ + 30% $\times$ miss rate$_{DL1}$.
CPI$_A$ = 1 + (0.02 + 0.3 $\times$ 0.08) $\times$ (15 + 0.03 $\times$ 250) = 1.99.
CPI$_B$ = 2 + (0.02 + 0.3 $\times$ 0.04) $\times$ (12 + 0.04 $\times$ 300) = 2.768
(Execution Time)$_B$ = IC $\times$ CPI $\times$ clock cycle time = IC$_B$ $\times$ 2.768 $\times$ 400 ps.
(Execution Time)$_A$ = IC $\times$ CPI $\times$ clock cycle time = IC$_A$ $\times$ 1.99 $\times$ 500 ps = (1.25 $\times$ IC$_B$) $\times$ 1.99 $\times$ 500 ps.
Speedup = (Execution Time)$_A$ / (Execution Time)$_B$.
Speedup = $\frac{1.25\times1.99\times500}{2.768\times400}$ = 1.1233.
Computer B is faster by 12.33%.

Question 3. Cache and AMAT (12 points)

SOLUTION:

Question 3.A (4 points)

The L1 cache can hold at most 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 access latency from the first program, the victim cache can hold at least 2 cache blocks. The second program should access at least one index in the cache and hence two cache blocks. In the worst case, all 8 data cache blocks content for the position in the same two cache blocks whereas the other two are left unused. Based on the access latency from the second program, the victim cache can hold at most 5 cache blocks.

Question 3.B (4 points)

1 cycle. The 5 cache blocks from above can all fit in the L1 cache and victim cache, and thus all cache block accesses are hits (in the steady state).

Question 3.C (4 points)

Either 1, 20 cycles or in between. Since LRU replacement policy is performed, during the loop execution, one data cache block either always stays in the cache without being replaced, or is replaced by others in every loop. It is impossible that one data cache block is swapped out in only part of the loops. There are four situations as follows.

- If every data cache block is swapped out in all loops and there are enough cache blocks in the victim cache, the access time is 1 cycle.
- If every data cache block is swapped out in all loops and there are no enough cache blocks in the victim cache, the access time is 20 cycles.
- If only a fraction of data cache blocks are swapped out in all loops and there are enough cache blocks in the victim cache, the access time is 1 cycle.
- if only a fraction of data cache blocks are swapped out in all loops and there are no enough cache blocks in the victim cache, the access time falls between 1 cycle and 20 cycles.

Question 3.D (4 points)

20 cycles. We can reach to this result by contradiction. We first assume the average memory access time is less than 20 cycles for the program in Question 3.D. Since the data cache blocks used in this program is a superset of $\{A, B, C, D, E, F, G, H\}$, the average memory access time should be at least 20 cycles, which contradicts the assumption made above. Therefore, the average memory access time is equal or more than 20 cycles. Since L2 cache is 16-way, L2 conflict and capacity cache misses will never happen in this program, and therefore the average memory access time is never beyond 20 cycles. As a result, we get the result of 20 cycles.

Question 4. Caching Basics (12 points)

SOLUTION:

Cache block size - 32 bytes
For sequence 1, 4 out of the 8 accesses (specifically those to addresses 2, 4, 8 and 16) can hit in the cache, as the hit ratio is 0.5. With any other cache block size but 32 bytes, the hit ratio is either smaller or larger than 0.5. Therefore, the cache block size is 32 bytes.

Associativity - 4
For sequence 2, blocks 0, 1024, 3072 and 4096 are the only ones that are reused and could potentially result in cache hits when they are accessed the second time. Three of these four blocks should hit in the cache when accessed for the second time to give a hit rate of 0.33 (3/9).

Given that the block size is 8 and for either cache size (256B or 512B), all of these blocks map to set 0. Hence, 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, the hit rate for this sequence is 3/9. Therefore, an associativity of 4 is the only one that could potentially give a hit rate of 0.33 (3/9).

Total cache size - 2048 B

For sequence 3, a total cache size of 1024 B will give a hit rate of 0.33 with a 4-way associative cache and 32 byte blocks regardless of the replacement policy, which is smaller than 4/9. Therefore, the total cache size is 2048 bytes.

Replacement policy - LRU

For the aforementioned cache parameters, all cache lines in sequence 4 map to set 0. If a FIFO replacement policy were used, the hit ratio would be 3/8, whereas if an LRU replacement policy were used, the hit ratio would be 1/4. Therefore, the replacement policy is LRU.