CMPE 110 Homework #3

John Allard
February 23nd, 2015
jhallard@ucsc.edu

1. **Question #1 - Basic Cache**

Consider a 512-KByte cache with 32 word cache lines.This cache uses write-back scheme, and the address is 32 bits wide. (The three tables for parts A, B, and C have been condensed into a single table found below part C).

Note - Since it is not stated, I am assuming that the lower 2 bits are hard-coded as `00`, which means all accesses are word-aligned. I am also including both the word offset and byte offset in my calculation of cacheline offset, as we are supposed to do as stated in question 436 on Piazza.

(a) **Question 1.A Direct-Mapped, Cache fields** Assume the cache is direct mapped. Fill in the table below to specify the size of each address field.

Explanation - For a direct-mapped cache, we need to have a 5-bit word offset because each cache-line is 32 words wide, we then need the 2 hard-coded bits to account for byte offset within each word, for a total of 7 bits for cacheline offset. Since our cache is 52-KByte large, and each line is 32 words, we have 4,096 cache lines. The amount of bits needed to address this amount of cache-lines is 12 ($2^{12} = 4,096$). This leaves 13 bits for the tag.

(b) **Question 1.B Fully Associative Cache Fields**
Assume the cache is fully associative, fill in the table below to specify the size of each address field.

Explanation - For a fully-associative cache, we still need 7 bits for the offset, which determines which of the 32 words in a cache-line our data is stored at. Since fully-associative caches do not use an index, this leaves 25 bits left for the tag.

(c) **Question 1.C 8-Way Set-Associative, Cache Fields**
Assume the cache is 8-way associative, fill in the table below to specify the size of each address field.

Explanation - Once again, we need 7 bits for the offset because each cache-line is 32-words wide. With 512Kbyte of memory and an 8-way set-associative cache, we will need 11 bits to state which set we are looking for ($2^9 = 4,096/8$). Finally we have 16 bits left to form the tag.

Table 1: Q 1A, 1B, and 1C Calculations

Field	Size (1A)	Size (1B)	Size (1C)
Cache line offset	7	7	7
Cache line index	12	0	9
Tag	13	25	16

(d) **Question 1.D Direct Mapped Cache Transactions**

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 (aka cold). 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]

Table 2: Q1D Calculations

Address	Request	Cacheline Ind	Valid	Modified	Tag	Data	Caused Replace	Write-back?
0x128	read	2	1	0	0	M[0x100]	0	0
0xF40	write	30	1	1	0	D[0xF00]	0	0
0xC00024	read	0	1	0	24	M[0xC00000]	0	0
0x014	write	0	1	1	0	D[0x000]	1	0
0x100F44	read	30	1	1	2	D[0x100F00]	1	1

Explanation -

(e) **Question 1.E 8-Way Set Associative, Cache Transactions**
Assume the cache is 8-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].

Table 3: Q1D Calculations

Address	Request	Cacheline Ind	Valid	Modified	Tag	Data	Caused Replace	Write-back?
0x128	read	2	0	0	0	M[0x100]	0	0
0xF40	write	30	1	1	0	D[0xF00]	0	0
0xC00024	read	0	1	0	96	M[0xC00000]	0	0
0x014	write	0	1	1	0	D[0x000]	0	0
0x100F44	read	30	1	0	16	D[0x100F00]	0	0

Explanation -

(f) **Question 1.F Overhead**
What is the overhead and actual size of the direct-mapped cache? What is the overhead and actual size of the 8-way set-associative cache? Does the structure change the overhead in terms of number of memory bits?

Answer :

2. **Question #2 - Impact of Cache Access Time**
In this problem, we will be comparing various microarchitectures for a simple data cache. For all parts, the memory requests use a 32-bit address, although you should assume that all addresses are word aligned. Since words are four bytes, this means the bottom two bits of the address will always be zero. All caches contain exactly eight cache lines, and each cache line contains four words (i.e., each cache line is 16 bytes long). Thus the total cache capacity is $8 \times 16B = 128B$

This problem will require you to identify the critical path of a specific microarchitecture. The table below lists simplified delay equations for the cache hardware components. These delay equations are parameterized by the size of each component. Delay is measured in normalized gate delays, where 1τ is the delay of a single inverter driving four identical inverters. To simplify things, assume that the delay of a component is always the same regardless of the order in which different inputs arrive at a component. More specifically, the delay of a write access is the same regardless of whether the address arrives before the write data or vice versa. Also assume that we are using combinational memories (i.e., the address is set and the data is returned on the same cycle).

(a) **Question 2.A - Sequential Tag Check then Memory Access**

The diagram on the previous page illustrates two cache microarchitectures that serialize the tag check before data access. This means that for both reads and writes, the cache completely finishes the tag check and accesses the data memory only on a cache hit. figure (a) is for a directed-mapped cache, while figure (b) is for a two-way set-associative cache. We now want to determine the critical path and cycle time in units of τ for each cache microarchitecture. As an example, the table below shows the critical path and cycle time for the directed-mapped cache from (a). Note that because we are serializing the tag check before data access and because the delay equations are the same for both read and write accesses, the critical path is the same regardless of whether we are doing a read or a write. This is a simplification, but it will do for the purposes of this part. Note that the tag is 25 bits, but each row of the tag memory requires 26 bits since it must also include a valid bit. Create a table similar to the one in the example which identifies the critical path and cycle time in units of τ for the two-way set-associative cache in (b). Compare the cycle times of the two cache microarchitectures. What is the primary reason one microarchitecture is slower than the other microarchitecture?

(b) **Question 2.B - Parallel Read Hit Path**

The diagram on the next page illustrates two cache microarchitectures with parallel read hit paths and pipelined write hit paths. This means that for a single read request, the tag check is done in parallel with the data memory read access, while for for a single write request the tag check is done in stage M0 and the data memory write access is done in stage M1. Figure (a) is for a directed-mapped cache, while figure (b) is for a two-way set-associative cache. For this part we will focus just on the parallel read hit path for both the direct-mapped and set-associative caches. Create two tables similar to the one from the previous part which identifies the critical path and cycle time in units of for just the parallel read hit paths. Note that since the tag check and the data memory read access are done in parallel, you will need to examine both of these paths to determine which one is in fact the critical path. Compare the cycle times of the two cache microarchitectures. What is the primary reason one microarchitecture is slower than the other microarchitecture?

(c) **Question 2.C - Pipelined Write Hit Path**
For this part we will focus just on the pipelined write hit path for both the direct-mapped and set-associative caches shown on the next page. Create two tables similar to the one in the previous parts which identifies the critical path and cycle time in units of for just the pipelined write hit path. Note that since the tag check and the data memory write access happen in two different stages, you will need to examine both of these paths to determine which one is in fact the critical path. Compare the cycle times of the two cache microarchitectures. What is the primary reason one microarchitecture is slower than the other microarchitecture?

3. **Question #3 : Cache Hierarchies**
Assume a processor uses a dedicated L1 cache for instructions (IL1) and a dedicated L1 cache for data (DL1). The processor also uses a shared L2 cache that serves as an intermediate level between each of the L1 caches and memory.

The processor has a cycle time of 1 ns (i.e., operates at 1 GHz frequency). A hit to either IL1 or DL1 takes 1 cycle (time to look up the cache and return the data to processor). IL1 has 8% miss rate. DL1 has 15% miss rate. Assume load/store instructions comprise 25% of the total instructions. Hit access to L2 from either of the IL1 or DL1 caches takes 6 cycles (time to lookup the cache and return the data to either of higher level caches). L2 has total miss rate of 30%. From L2, it takes 50 cycles to access memory

(a) **Question 3.A AMAT**
What is the average memory access time?

Answer :

(b) **Question 3.B AMAT**
What is the average memory access time?

Answer :

4. **Question #4 :**

   (a) nothing

5. **Question #5 : Multipliers**

   (a) nothing

6. **Question #6 : Hazards**

   (a) nothing