CSCE 692 HW Set 2 - Exercises

(Chapter 2 and Appendix B)

(100 Points)

Due: NLT 1100 Thursday, 29 January 2019

**Exercises :**

**B.1 (a, b, c, d)** [5/5/5/7]

**B.2 (a, b)** [7/7]

**B.5 (a, b, c, d)** [12/12/12/8]

**2.17 (a, b, c)** [5/5/10]

**Instructions:**

* Write your name at the top and a page number on each page
* Please show your work (in some cases, no work = no credit)
* Clearly indicate your answers
* Explain assumptions and provide references (sources) for any additional information you had to research in order to complete any problem.

**Notes:**

1. **Answer all sub questions.**
2. **B.2, To simplify grading, adapt the general format of Fig B.30.**
3. **B.5, Problem Modification: Write buffer eliminates stalls for 97% of writes.**
4. **CACTI,** 
   1. **Use CACTI data provided below**
   2. **Let assumed CPU cycle time be random cycle time (ps) for each case.**

CACTI inputs/outputs

Access Time (ps)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Cache Size | | |  |  |  |
| Associativity | 16KB | 32 KB | 64 KB | 128 KB | 1 MB | 2 MB | 4 MB | 8 MB |
| 1 | 301.4 | 387.6 | 477.7 | 731.6 | 1705 | 2014 | 2501 | 3010 |
| 2 | 322.2 | 406.5 | 523.6 | 771.8 | 1736 | 2046 | 2626 | 3046 |
| 4 | 344.2 | 429.1 | 520.6 | 778.3 | 1737 | 2044 | 2625 | 3069 |
| 8 | 561.6 | 574.2 | 663.0 | 917.6 | 1749 | 2053 | 2546 | 3056 |

Cycle Time (ps)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Cache Size | | |  |  |  |
| Associativity | 16KB | 32 KB | 64 KB | 128 KB | 1 MB | 2 MB | 4 MB | 8 MB |
| 1 | 183.9 | 183.9 | 238.9 | 183.9 | 127.2 | 127.2 | 127.2 | 128.2 |
| 2 | 181.3 | 183.9 | 236.2 | 183.9 | 127.2 | 127.2 | 121.4 | 127.2 |
| 4 | 240.8 | 181.3 | 236.2 | 181.3 | 127.2 | 127.2 | 121.4 | 124.7 |
| 8 | 149.1 | 155.2 | 181.3 | 155.2 | 124.7 | 124.6 | 124.7 | 124.7 |

B.1 [5/5/5/7] <B.1> You are trying to appreciate how important the principle of locality is in justifying the use of a cache memory, so you experiment with a computer having an L1 data cache and a main memory (you exclusively focus on data accesses). The latencies (in CPU cycles) of the different kinds of accesses are as follows: cache hit, 1 cycle; cache miss, 110 cycles; main memory access with cache disabled, 105 cycles.

1. [5] <B.1> When you run a program with an overall miss rate of 3%, what will the average memory access time (in CPU cycles) be?

.97\*1 + .03\*110 = 4.27 cycles

1. [5] <B.1> Next, you run a program specifically designed to produce completely random data addresses with no locality. Toward that end, you use an array of size 1 GB (all of which fits in the main memory). Accesses to random elements of this array are continuously made (using a uniform random number generator to generate the elements indices). If your data cache size is 64 KB, what will the average memory access time be?

Hit Rate = 64 KB / 1 GB = 64 KB / (1 \* 1024^2) KB = 0.000061

Miss Rate = (1 – Hit Rate) = (1 – 0.000061) = 0.999939

AMAT = 1 \* Hit Rate + 110 \* Miss Rate = 1 \* 0.000061 + 110 \* 0.999939

AMAT = 109.9934 Cycles

1. [5] <B.1> If you compare the result obtained in part (b) with the main memory access time when the cache is disabled, what can you conclude about the role of locality in justifying the use of cache memory?

Because the AMAT with a cache is 109 cycles, while the AMAT with the cache disabled is 105 cycles, locality would not justify the use of a cache because it would increase the average access time. There would be no reason to deal with the increased overhead to lose efficiency.

1. [7] <B.1> You observed that a cache hit produces a gain of 104 cycles (1 cycle vs. 105), but it produces a loss of 5 cycles in the case of a miss (110 cycles vs. 105). In the general case, we can express these two quantities as G (gain) and L (loss). Using these two quantities (G and L), create a general formula to identify the highest miss rate after which the cache use would be disadvantageous.

Cache is disadvantageous when the losses outweigh the gains, and that occurs for any miss rate over the miss rate at which the gains equal the losses

0 = (1 – Miss Rate) \* G – Miss Rate \* L

**General Equation:** Miss Rate \* L = (1 – Miss Rate) \* G

**Miss rate with G = 104, L = 5:**Miss Rate \* 5 = (1 – Miss Rate) \* 104 🡪 Miss Rate = 104/109 = 0.9541

For the access times in this problem, a cache is disadvantageous if the miss rate > 0.9541.

B.2 [7/7] For the purpose of this exercise, we assume that we have 512-byte cache with 64-byte blocks. We will also assume that the main memory is 2 KB large. We can regard the memory as an array of 64-byte blocks: M0, M1, ..., M31. Figure B.30 sketches the memory blocks that can reside in different cache blocks if the cache was direct-mapped.

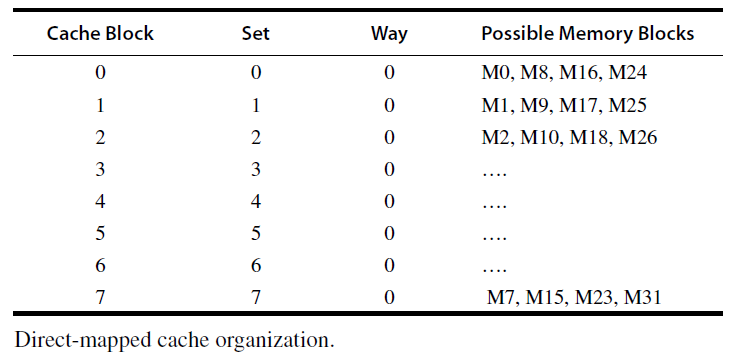


Figure B.30 (from 6th Edition). Memory blocks distributed to direct-mapped cache

1. [7] Show the contents of the table if cache is organized as a fully associative cache.

|  |  |  |  |
| --- | --- | --- | --- |
| **Cache Block** | **Set** | **Way** | **Possible Memory Blocks** |
| 0 | 0 | 0 | M0, M1, M2, … , M31 |
| 1 | 0 | 1 | … |
| 2 | 0 | 2 | … |
| 3 | 0 | 3 | … |
| 4 | 0 | 4 | … |
| 5 | 0 | 5 | … |
| 6 | 0 | 6 | … |
| 7 | 0 | 7 | … |

Fully associative – any memory block can go in any cache block

1. [7] Repeat part (a) with the cache organized as a four-way set associative cache.

|  |  |  |  |
| --- | --- | --- | --- |
| **Cache Block** | **Set** | **Way** | **Possible Memory Blocks** |
| 0 | 0 | 0 | M0, M2, M4, … , M30 |
| 1 | 0 | 1 | … |
| 2 | 0 | 2 | … |
| 3 | 0 | 3 | … |
| 4 | 1 | 0 | M1, M3, M5, … , M31 |
| 5 | 1 | 1 | … |
| 6 | 1 | 2 | … |
| 7 | 1 | 3 | … |

Four-way – 8 cache blocks 🡪 2 sets with 4 blocks/set

B.5 [12/12/12/8] You are building a system around a processor with in-order execution that runs at 1.1 GHz and has a CPI of 1.35 excluding memory accesses. The only instructions that read or write data from memory are loads (20% of all instructions) and stores (10% of all instructions). The memory system for this computer is composed of a split L1 cache that imposes no penalty on hits. Both the I-cache and D-cache are direct mapped and hold 32 KB each. The I-cache has a 2% miss rate and 32-byte blocks, and the D-cache is write-through (no-write allocate) with a 5% read miss rate and 16-byte blocks. There is a write buffer on the D-cache that eliminates stalls for 97% of all writes. The 512 KB write-back (write allocate), unified L2 cache has 64-byte blocks and an access time of 15 ns. It is connected to the L1 cache by a 128-bit data bus that runs at 266 MHz and can transfer one 128-bit word per bus cycle. Of all memory references sent to the L2 cache in this system, 80% are satisfied without going to main memory. Also, 50% of all blocks replaced are dirty. The 128-bit-wide main memory has an access latency of 60 ns, after which any number of bus words may be transferred at the rate of one per cycle on the 128-bit-wide 133 MHz main memory bus.

*A useful tool for solving this type of problem is to extract all of the available information from the problem description. It is possible that not all of the information will be necessary to solve the problem, but having it in summary form makes it easier to think about*

Processor – 1.1 GHz

CPI = 1.35 🡪 excluding memory accesses

Loads – 20%, Stores – 10%

**L1 Cache:**

Cache Capacity – 32 KB

I-Cache – 2% miss rate, 32 Byte blocks (1024 blocks)

D-Cache – write through, 5% read miss rate, 16 Byte blocks (2048 blocks)

D-Cache – write buffer 🡪 0 penalty for 97% of writes

**Connection**: 128-bit bus @ 266 MHz, one 128-bit word/bus cycle

**L2 Cache:**

512 KB unified L2 cache: 64 byte blocks 🡪 8192 blocks, access time = 15 ns

80% satisfied at L2; 50% replaced are dirty

**Main Memory:**

128-bit bus; latency = 60 ns

One word/cycle on 128-bit-wide 133 MHz bus

**Assumptions:**

* Write buffer eliminates stalls only **to** the D-Cache, not from the D-Cache to main memory
* Instruction accesses have a 50% chance of needing write-back to main memory
* No write back is needed on data writes because data is being written already.

1. [12] What is the average memory access time for instruction accesses?

Time for hit at L1 = 0

Miss Rate at L1 = 0.02

L2 Access = 15 ns

Cycle time L2 🡪 L1 bus:

L2 Transfer Time:

L2 Miss Rate = 0.20

L3 Access time = 60 ns

Cycle time on L2 🡪 L3 bus:

L3 Transfer Time:

L3 Write Back – 50% of blocks are dirty

1. [12] What is the average memory access time for data reads?

L1: 95% hit rate, penalty = 0

L2: 15 ns access

L3: 60 ns access

Write back penalty on 50% of blocks in main memory::

1. [12] What is the average memory access time for data writes?

L2 access = 15 ns

L2 Transfer Time = 3.7594 ns

L3 access = 60 ns

L3 Transfer Time = 30.075 ns

L3 Write Back = 0.50\* 90 ns = 45 ns

1. [8] What is the overall CPI, including memory accesses?

CPI = 1.35 excluding mem access

Overall CPI:

2.17 [5/5/10] <2.2> The following questions investigate the impact of small and simple caches using CACTI and assume a 32 nm (0.032 μm) technology.

1. [5] <2.2> Compare the access times of 64 KB caches with 64 byte blocks and a single bank. What are the relative access times of two-way and four- way set associative caches in comparison to a direct mapped organization?

Direct mapped – 477.7 ps, Two-Way – 523.6 ps, Four-way – 520.6 ps.

The direct mapped cache performed better than all other cache organizations. The two-way access time was 45.9 ps longer than direct mapped, while the 4-way was 42.9 ps longer.

1. [5] <2.2> Compare the access times of four-way set associative caches with 64-byte blocks and a single bank. What are the relative access times of 32 KB and 64 KB caches in comparison to a 16 KB cache?

The access time of the 16 KB cache was 344.2 ps. The 32 KB cache had an access time of 429.1 ps, while the 64 KB cache was 520.6 ps. Using the 16 KB cache as the baseline produces relative access times as follows:

Thus, in this case the smallest cache had the smallest access time.

1. [10] <2.2> For a 64 KB cache, find the cache associativity between 1 and 8 with the lowest average memory access time given that misses per instruction for a certain workload suite is 0.00664 for direct mapped, 0.00366 for two- way set associative, 0.000987 for four-way set associative, and 0.000266 for eight-way set associative cache. Overall, there are 0.3 data references per instruction. Assume cache misses take 10 ns in all models. To calculate the hit time in cycles, assume for each associativity, the CPU will be built to have a cycle time equal to the random cycle time found using CACTI, which corresponds to the maximum frequency that cache configuration can operate without any bubbles in the pipeline.

Note, I will show the calculation for direct mapped below, and the rest will be in the table:

Table 1: AMATs for different associativities of 64KB caches

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Associativity: | Cycle Time (ps) | Misses/Inst. | Miss Rate | AMAT (ps) |
| 1 | 238.9 | 0.006640 | 0.022133 | 454.942 |
| 2 | 236.2 | 0.003660 | 0.012200 | 355.318 |
| 4 | 236.2 | 0.000987 | 0.003290 | 268.323 |
| 8 | 181.3 | 0.000266 | 0.000887 | 190.009 |

The cache associativity with the lowest AMAT is a 8-way associative. For this example, the higher the associativity, the lower the AMAT. This is most likely due to the significantly lower miss rate for the higher associativities.