# COSE222: Computer Architecture
# Assignment #4

## Due: December 15, 2020 (Tuesday) 11:59pm on Blackboard

## Solutions (Total score: 121)

Please answer for the questions. Write your student ID and name on the top of the document. Submit your homework with "**PDF**" format only. (You can easily generate the pdf files from Microsoft Word or HWP. You can also handwrite your answers to scan the handwritten documents with "**PDF**" format. You may use the document capture applications such as "Office Lens" for scanning your documents with your smartphones.)

The answer rules:
(1) You can write answers in both Korean and English.
(2) Please make your final answer numbers have two decimal places.
(3) Performance of A is improved by *NN* % compared to performance B if PerfA / PerfB = 1.*NN*.

1. The following code is written in C, where elements within the same **row** are stored contiguously. Assume each element of arrays is a 64-bit integer. The data type of all variables is a 64-bit integer also. Cache blocks are allocated on write miss. (*Hint: in this code, there are 5 variables: a, b, i, j, and sum*)

```
int sum;
for (i = 0; i < 1024; i++)
    sum = 0;
    for (j = 0; j < 1024; j++)
        sum += a[i][j];
    b[i] = sum;
```

(a) Which variable references exhibit temporal locality? [2]

i, j, sum

(b) Which variable references exhibit spatial locality? [2]

a, b

(c) Let's focus on the data transfers for arrays a and b. How many ld and sd instructions are issued while executing this code? [4]

Array a: number of ld instructions = 1024 x 1024 = $2^{20}$ (or 1048576), no sd instructions

Array b: no ld instructions, number of sd instructions = 1024

(d) Let's focus on the cache miss rates for arrays a and b. Let us assume the cache block size is 32 bytes and elements of arrays a and b are aligned within a cache block. Calculate the compulsory (cold) miss rate for arrays a and b respectively. (Hint: assume that the cache size is infinite). [8]

A single cache block includes 4 elements of an array. Thus the cache access events will be miss/hit/hit/hit for one block.

Cold miss rate of A: 25 %

Cold miss rate of B: 25%

(e) In this problem let's focus on the miss rates of arrays a and b. Assume the cache block size is 32 bytes and the cache has 128 blocks. The cache is a fully associative cache. Calculate the capacity miss rates for arrays a and b. [8]

For array a, the miss rate is 25% and this result is equal to the cold miss. Thus, capacity miss is 0%.

For array b, the miss rate is 100% since the allocated blocks for array b will be evicted before it is re-referenced. Thus, capacity miss is 75%. (25% comes from cold miss)

Capacity miss rate of A: 0%

Capacity miss rate of B: 75%

2. Below is a list of 64-bit memory address references, given as **word addresses**. (1 word = 64-bits)

`0xFD, 0xBA, 0x2C, 0xB5, 0x0E, 0xBE, 0x58, 0xBF, 0x02, 0x2B, 0xB4, 0x03`

(a) Let us assume a direct-mapped cache has 16 blocks and a single block includes 2 word. What is the size of this cache? [2]

16 x 16 bytes = 256 bytes

(b) For each of these references, identify the binary word address, the tag, and the index given a direct-mapped cache with 16 two-word blocks. Also list whether each reference is a hit or miss, assuming the cache is initially empty. [6]

| Word address | Binary address | Tag | Index | Hit/Miss |
|---|---|---|---|---|
| 0xFD | 111 1110 1 | 7 | E | M |
| 0xBA | 101 1101 0 | 5 | D | M |
| 0x2C | 001 0110 0 | 1 | 6 | M |
| 0xB5 | 101 1010 1 | 5 | A | M |
| 0x0E | 000 0111 0 | 0 | 7 | M |
| 0xBE | 101 1111 0 | 5 | F | M |
| 0x58 | 010 1100 0 | 2 | C | M |
| 0xBF | 101 1111 1 | 5 | F | H |
| 0x02 | 000 0001 0 | 0 | 1 | M |
| 0x2B | 001 0101 1 | 1 | 5 | M |
| 0xB4 | 101 1010 0 | 5 | A | H |
| 0x03 | 000 0001 1 | 0 | 1 | H |

(c) Calculate the miss rate (in percentage) of the above cache. [2]

Miss rate = 9 / 12 = 0.75, 75%

(d) Assume that the hit time of the above cache is one cycle and it takes 8 cycles to read **one** word from the memory (i.e. this value decides the miss penalty of the cache). Calculate the average memory access time (AMAT) of the cache for the given reference stream. [2]

Miss penalty for this cache is 16 since one cache block includes 2 words.
AMAT = hit time + miss rate x miss penalty = 1 + 0.75x16 = 13

(e) Assume that the direct-mapped cache has the same size of the data memory, but the size of a single cache block is increased to **4 words**. For each of the above references, identify the binary word address, the tag, and the index of the cache. Also list if each reference is a hit or a miss, assuming the cache is initially empty. [6]

| Word address | Binary address | Tag | Index | Hit/Miss |
|---|---|---|---|---|
| 0xFD | 111 111 01 | 7 | 7 | M |
| 0xBA | 101 110 10 | 5 | 6 | M |
| 0x2C | 001 011 00 | 1 | 3 | M |
| 0xB5 | 101 101 01 | 5 | 5 | M |
| 0x0E | 000 011 10 | 0 | 3 | M |
| 0xBE | 101 111 10 | 5 | 7 | M |
| 0x58 | 010 110 00 | 2 | 6 | M |
| 0xBF | 101 111 11 | 5 | 7 | H |
| 0x02 | 000 000 10 | 0 | 0 | M |
| 0x2B | 001 010 11 | 1 | 2 | M |
| 0xB4 | 101 101 00 | 5 | 5 | H |
| 0x03 | 000 000 11 | 0 | 0 | H |

(f) Calculate the miss rate (in percentage) of the above cache. [2]

Miss rate = 9 / 12 = 0.75, 75%

(g) Assume that the hit time of the above cache is one cycle and it takes 8 cycles to read **one** word from the memory (this configuration is the same to the above problem d). Calculate the average memory access time (AMAT) of the cache for the given reference stream. [2]

Miss penalty for this cache is 32 since one cache block includes 4 words.
AMAT = hit time + miss rate x miss penalty = 1 + 0.75x32 = 23

3. For a direct-mapped cache design with a 64-bit address, the following bits of the address are used to access the cache. (1 word = 64-bits)

| Tag | Index | Offset |
|---|---|---|
| 63-10 | 9-5 | 4-0 |

(a) What is the cache block size (in words)? [4]

One word size is 8 bytes, so a single word occupies 3 bits of the address field. So, the remaining 2 bits in the offset field are assigned to block offset. Thus one block includes 4 ($2^2$) words.

(b) How may blocks does the cache have? [4]

Index field size is 5 bits. Thus, there are 32 ($2^5$) blocks in the cache.

(c) What is the ratio between total bits required for such a cache implementation over the data storage bits? Let us assume each cache block includes 1-bit "valid" field. [8]

Each cache block has 32 bytes (256 bits) of data. One cache block includes 1-bit of valid bit and 54-bit of tag field, so total 55 bits. Thus, the ratio of total bits to storage bits is (256 + 55) / 256 = 1.21

Beginning from power on, the following **byte-addressed** cache references are recorded.

```
0x000, 0x004, 0x010, 0x084, 0x0E8, 0x0A0, 0x400, 0x01E, 0x08C, 0xC1C, 0x0B4,
0x884
```

(d) For each reference, complete the following table. "Replace" represents which bytes replaced if any. [6]

| Address | Tag | Index | Offset | Hit/Miss | Replace |
|---|---|---|---|---|---|
| 0x000 | 0x0 | 0x00 | 0x00 | M | |
| 0x004 | 0x0 | 0x00 | 0x04 | H | |
| 0x010 | 0x0 | 0x00 | 0x10 | H | |
| 0x084 | 0x0 | 0x04 | 0x04 | M | |
| 0x0E8 | 0x0 | 0x07 | 0x08 | M | |
| 0x0A0 | 0x0 | 0x05 | 0x00 | M | |
| 0x400 | 0x1 | 0x00 | 0x00 | M | 0x000~0x01F |
| 0x01E | 0x0 | 0x00 | 0x1E | M | 0x400~0x41F |
| 0x08C | 0x0 | 0x04 | 0x0C | H | |
| 0xC1C | 0x3 | 0x00 | 0x1C | M | 0x000~0x01F |
| 0x0B4 | 0x0 | 0x05 | 0x14 | H | |
| 0x884 | 0x2 | 0x04 | 0x04 | M | 0x080~0x09F |

(e) What is the hit ratio? [2]

4/12 = 0.33, 33%

(f) List the final state of the cache, with each valid entry represented as a record of <index, tag, data>. For example, [5]

```
    <0, 3, Mem[0xC00]-Mem[0xC1F]>
```

```
<0, 3, Mem[0xC00]~Mem[0xC1F]>
<4, 2, Mem[0x880]~Mem[0x89F]>
<5, 0, Mem[0x0A0]~Mem[0x0BF]>
<7, 0, Mem[0x0E0]~Mem[0x0FF]>
```

4. Cache access time is usually proportional to the capacity of cache. Assume that main memory accesses take 50 ns and that 36% of all instructions access data memory. The following table shows data for L1 cache attached to each of two processors, P1 and P2.

| Processor | L1 size | L1 miss rate | L1 hit time |
|---|---|---|---|
| P1 | 2 KB | 8% | 0.5 ns |
| P2 | 4 KB | 4% | 0.8 ns |

(a) Assuming that the L1 hit time determines the cycle time for P1 and P2, what are their respective clock rates? [2]

P1: 1/0.5ns = 2.00 GHz. P2: 1/0.8ns = 1.25 GHz

(b) What is the Average Memory Access Time (AMAT) for P1 and P2 (in cycles)? [4]

P1: miss penalty is 50 ns / 0.5 ns = 100, so AMAT = 1 + (100 x 0.08) = 9

P2: miss penalty is 50 ns / 0.8 ns = 62.5, so AMAT = 1 + (63 x 0.04) = 3.52

(c) Assuming a base CPI is 1.0 without any memory stalls, what is the total CPI for P1 and P2? Which processor is faster? When we say a "base CPI of 1.0", we mean instructions complete in one cycle, unless either the instruction access or the data access causes a cache miss. [4]

P1: 1 + 0.08x100 + 0.36x0.08x100 = 11.88, CPI x clock period = 11.88 / 2G = 5.94x10$^{-9}$

P2; 1 + 0.04x63 + 0.36x0.04x63 = 4.43, CPI x clock period = 4.43 / 1.25G = 3.54x10$^{-9}$

Hence, P2 is faster

For the next problems, we will consider the addition of an L2 cache to P1 (to presumably make up for its limited L1 cache capacity). Use the L1 cache capacities and hit times from the previous table when solving these problems. The L2 miss rate indicated is its local miss rate, namely the L2 miss counts divided by the total L2 access counts.

| L2 size | L2 miss rate | L2 hit time |
|---|---|---|
| 1 MB | 80% | 5.0 ns |

(d) What is the AMAT for P1 with the addition of an L2 caches? Is the AMAT better or worse with the L2 cache? [4]

L2 hit time is 5.0 ns, which is 10 cycles. AMAT = 1 + 0.08x10 + 0.08x0.8x100 = 8.2. Better

(e) Assuming a base CPI of 1.0 without any memory stalls, what is the total CPI for P1 with the addition of an L2 cache? [4]

CPI = 1 + (AMAT – 1) + 0.36x(AMAT – 1) = 10.79

5. Consider the following program and cache behaviors.

| Data reads per 1000 instructions | Data writes per 1000 instructions | Instruction cache miss rate | Data cache miss rate | Block size (bytes) |
|---|---|---|---|---|
| 250 | 100 | 0.3% | 2.0% | 64 |

(a) Suppose a CPU with a write-through, write-allocate cache achieves a CPI of 2. Assume that the size of a single write request generated by one data write is 8 bytes (64-bit) for write-through policy. What are the read and write bandwidths (measured by bytes per cycle) between RAM and the cache? (Assume each miss generates a request for one block.) [8]

As CPI is 2, on average 0.5 instructions access the instruction cache per cycle. Thus the read bandwidth by instruction cache misses: 0.5 x 0.003 x 64 = 0.096 bytes/cycle

The read bandwidth by data reads: 0.5 x 0.25 x 0.02 x 64 = 0.16 bytes/cycle

The read bandwidth by data writes (note it is write-allocate cache): 0.5 x 0.1 x 0.02 x 64 = 0.064 bytes/cycle. Therefore the total read bandwidth = 0.096 + 0.16 + 0.064 = 0.32 bytes/cycle

The data writes also generate write bandwidth by write-through policy. Thus the write bandwith by data writes: 0.5 x 0.1 x 8 = 0.4 bytes. This is the same to the total write bandwidth.

(b) For a write-back, write-allocate cache, assuming 30% of replaced data cache blocks are dirty. Note that a whole evicted block is written to memory with the write-back cache if the evicted block is dirty. What are the read and write bandwidths needed for a CPI of 2? [8]

The read bandwidth is the same to the above question. (0.32 bytes/cycle)

With a write-back cache write requests are generated to memory only when a cache miss occurs and a replaced cache block is dirty. Note that the cache block replacement is caused by both read and write requests. Thus the write bandwidth: 0.5 x (0.25 + 0.1) x 0.02 x 0.3 x 64 = 0.0672 bytes/cycle

6. Let us assume that the size of virtual address is 48-bits and the size of physical memory is 4 GB. Word size is 64-bits and page size is 4 KB. All addresses are byte-addressed.

(a) What is the maximum size of the virtual memory supported by this system? [2]

256 TB

(b) What is the size of physical address? [2]

32-bits

(c) Let us assume the TLB has 512 entries and TLB is two-way set associative. Which virtual address bits are used to index the TLB? Which virtual address bits are used as tag of the table? [8]

Virtual address bits [47:12] are used as a virtual page number. As TLB has 512 entries with two-way set associativity, TLB has 256 sets (= 8-bits).

Index field: V19 ~V12

Tags: V47 ~ V20