CS61C Spring 2025 — Caches — Discussion 6

1 Memory Quantities

1.1 Convert the following quantities into the number of bytes each term represents (you may leave your answers in terms of powers of 2).

a) 1 KiB

Answer: 2^{10} Bytes

We can use the 61C reference card (or the precheck worksheet) for the values of the SI prefixes. A Kibi- is 2^{10}, so 1 KiB = 2^{10} bytes.

b) 32 MiB

Answer: 2^{25} Bytes

We know that one MiB = 2^{20} Bytes, so we have $32 \times 2^{20} = 2^5 \times 2^{20} = 2^{25}$ bytes.

c) 16 Gib

Answer: 2^{31} Bytes

16 Gib = $2^4 \times 2^{30} = 2^{34}$. Notice that we have 16 Gib which is 16 Gibi*bits* – one byte is $8 = 2^3$ bits, so 2^{34} / $2^3 = 2^{31}$ Bytes

d) 20 KiB

Answer: 5×2^{12} Bytes

We can factor 20 into 4×5 for a solution in terms of powers of 2. 20 KiB = $(4 \times 5) \times 2^{10} = (2^2 \times 5) \times 2^{10} = 5 \times 2^{12}$ Bytes.

1.2 Rewrite the following quantities using IEC Prefixes.

a) 2048 B

Answer: 2 KiB

$2048 = 2^{11} = 2^{10+1}$

$= 2^1 \times 2^{10} = 2$ KiB.

b) 2^{38} B

Answer: 256 GiB

2^{38} can be rewritten as $2^{30+8} = 2^8 \times 2^{30} = 256 \times 2^{30} = 256$ GiB.

2 Cache T/I/O

2.1 Assume we have a 32-bit system with a direct-mapped, byte-addressed cache with capacity 32B and line size of 8B. Of the 32 bits in each address, which bits do we use to find the tag, index, and offset of the cache?

Tag: 27 bits → address[31:5]
Index: 2 bits → address[4:3]
Offset: 3 bits → address[2:0]
We can start by finding which bits correspond to the offset bits. The number of offset bits is just dependent on the line size, so since our lines are size 8B, we need $\log_2(8) = 3$ bits to differentiate the 8 bytes in the line, so we have 3 offset bits. In this case, the offset is the 3 least significant bits. Denoting the most significant bit (MSB, on the left) as it 31 and the least significant bit (LSB, on the right) as bit 0, our offset bits are bits 0, 1, and 2.

We can determine the number of index bits we need from the number of sets our cache has. Since our cache is direct-mapped, the number of sets is the same as the number of lines, so we just need to figure out how many lines our cache has. We see that num lines = cache size/line size, so our cache has 32/8 = 4 lines. We need $\log_2(4) = 2$ bits to differentiate the 4 lines, so we have 2 index bits.

From our T:I:O breakdown, we can see that the offset bits are the least significant bits and the next set of least significant bits is the index bits. We calculated that there were 3 offset bits, so our index bits will start at bit 3 (remember the least significant bit is bit 0!). Since we have 2 index bits, this means that we can find the index bits at bits 3 and 4.

From our T:I:O breakdown, we can see that the tag bits are the most significant bits. Our tag is the remainder most-significant bits, so we can find our tag bits at bits 5-31.

2.2 Assume that we have the same cache scheme as the previous part (direct-mapped byte-addressed cache with capacity 32B and line size of 8B). Do the following:

- Decode the tag, index, and offset bits for each address
- Classify each of the following accesses as a cache hit (H) or a cache miss (M).
  - For any misses, list out what type of miss it is (Compulsory (C), Noncompulsory (NC)).

Tip: Use the space below to draw out your cache!

Address	**T/I/O**	**Hit / Miss (C / NC)**
`0x00000004`	Tag 0, Index 0, Offset 4	Miss (Compulsory)
`0x00000005`	Tag 0, Index 0, Offset 5	Hit
`0x00000068`	Tag 3, Index 1, Offset 0	Miss (Compulsory)
`0x000000C8`	Tag 6, Index 1, Offset 0	Miss (Compulsory)
`0x00000068`	Tag 3, Index 1, Offset 0	Miss (Noncompulsory)
`0x000000DD`	Tag 6, Index 3, Offset 5	Miss (Compulsory)
`0x00000045`	Tag 2, Index 0, Offset 5	Miss (Compulsory)
`0x000000CF`	Tag 6, Index 1, Offset 7	Miss (Noncompulsory)
`0x000000F3`	Tag 7, Index 2, Offset 3	Miss (Compulsory)

3 Cache Associativity

3.1 Here's some practice involving a **2-way set associative** cache. This time we have an 8-bit address space, 8 B lines, and a cache size of 32 B.

- Decode the tag, index, and offset bits for each address
- Classify each of the following accesses as a cache hit (H) or a cache miss (M).
  - For any misses, list out what type of miss it is (Compulsory (C), Noncompulsory (NC)).

Assume that we have an LRU replacement policy (in general, this is not always the case).

Address	T/I/O	Hit / Miss (C / NC)
`0b0000 0100`	Tag 0000, Index 0, Offset 100	Miss (Compulsory)
`0b0000 0101`	Tag 0000, Index 0, Offset 101	Hit
`0b0110 1000`	Tag 0110, Index 1, Offset 000	Miss (Compulsory)
`0b1100 1000`	Tag 1100, Index 1, Offset 000	Miss (Compulsory)
`0b0110 1000`	Tag 0110, Index 1, Offset 000	Hit
`0b1101 1101`	Tag 1101, Index 1, Offset 101	Miss (Compulsory)
`0b0100 0101`	Tag 0100, Index 0, Offset 101	Miss (Compulsory)
`0b0000 0100`	Tag 0000, Index 0, Offset 100	Hit
`0b0011 0000`	Tag 0011, Index 0, Offset 000	Miss (Compulsory)
`0b1100 1011`	Tag 1100, Index 1, Offset 011	Miss (Noncompulsory)
`0b0100 0010`	Tag 0100, Index 0, Offset 010	Miss (Noncompulsory)

Since our cache is 2-way set associative, there are 2 lines in a set. Given the cache size and the line size, we have 32 / 8 = 4 lines. Thus, there are 4 / 2 = 2 sets in our cache. We need $\log_2(2) = 1$ bit to differentiate the 2 sets, so we have 1 index bit. Our line size of 8 B means we have $\log_2(8) = 3$ offset bits, and that the rest of our bits are our tag bits. Therefore, our TIO breakdown means bits 0, 1, and 2 are our offset bits, the only index bit is bit 3, and bits 4-7 being the tag bits.

3.2 What is the hit rate of our above accesses?

$\frac{3 \text{ hits}}{11 \text{ accesses}} \approx 27.3\%$ hit rate

4 Code Analysis

Given the follow chunk of code, analyze the hit rate given that we have a byte-addressed computer with a total memory of **1 MiB**. It also features a **16 KiB** Direct-Mapped cache with **1 KiB** lines. Assume that your cache begins cold.

```
#define NUM_INTS 8192    // 2^13
int A[NUM_INTS];         // A lives at 0x10000
int i, total = 0;
for (i = 0; i < NUM_INTS; i += 128) {
    A[i] = i;            // Line 1
}
for (i = 0; i < NUM_INTS; i += 128) {
    total += A[i];       // Line 2
}
```

4.1 How many bits make up a memory address on this computer?

We take $\log_2(1 \quad \text{MiB}) = \log_2(2^{20}) = 20$.

4.2 What is the T/I/O breakdown?

Offset = $\log_2(1 \quad \text{KiB}) = \log_2(2^{10}) = 10$
Index = $\log_2(\frac{16 \text{ KiB}}{1 \text{ KiB}}) = \log_2(16) = 4$
Tag = $20 - 4 - 10 = 6$

4.3 Calculate the cache hit rate for the line marked Line 1:

The integer accesses are $4 * 128 = 512$ bytes apart, which means there are 2 accesses per line. The first accesses in each line is a compulsory cache miss, but the second is a hit because `A[i]` and `A[i+128]` are in the same cache line. Thus, we end up with a hit rate of **50%**.

4.4 Calculate the cache hit rate for the line marked Line 2:

The size of A is $8192 * 4 = 2^{15}$ bytes. This is exactly twice the size of our cache. At the end of Line 1, we have the second half of A inside our cache, but Line 2 starts with the first half of A. Thus, we cannot reuse any of the cache data brought in from Line 1 and must start from the beginning. Thus our hit rate is the same as Line 1 since we access memory in the same exact way as Line 1. We don't have to consider cache hits for total, as the compiler will most likely store it in a register. Thus, we end up with a hit rate of **50%**.

5 Review: Cache Performance

Recall that AMAT stands for Average Memory Access Time. The main formula for it is:

$$\text{AMAT} = \text{Hit Time} + \text{Miss Rate} * \text{Miss Penalty}$$

5.1 Suppose your system takes 100ns to access main memory. We decide to add a cache with a measured hit time of 25ns and miss rate of 25%. What is the average memory access time of the system?

Answer: 50ns

We are looking for a solution to the AMAT equation. The hit time for the new L1$ is 25ns. The miss rate is 25% and the miss penalty will be the 100ns required to access main memory in the case of a cache miss. Thus, our solution is $\text{AMAT} = 25\text{ns} + 0.25 * 100\text{ns} = 50\text{ns}$. By adding a cache, we have effectively halved the time spent waiting for memory accesses.

$\boxed{5.2}$ In a new 2-level cache system, after 100 total accesses to the cache system, we find that the L2$ (L2 Cache) ended up missing 20 times. What is the global miss rate of L2$?

Answer: 20%

$$\text{Global Miss Rate} = \frac{\text{Local Missed Accesses}}{\text{Total System Accesses}} = \frac{20}{100} = 20\%$$

$\boxed{5.3}$ Given the system from the previous subpart (100 total accesses, 20 L2$ misses), if L1$ had a local miss rate of 50%, what is the local miss rate of L2$?

Answer: 40%

$$\text{Local Miss Rate} = \frac{\text{Local Missed Accesses}}{\text{Local Cache Accesses}} = \frac{20}{50\% * 100} = \frac{20}{50} = 40\%$$

We know that L2$ is accessed when L1$ misses, so if L1$ misses 50% of the time, that means we access L2$ 50 times, of which we ended up having 20 misses in L2$.

For the following subparts, suppose we have a new system that consists of:

1. An L1$ that has a hit time of 2 cycles and a local miss rate of 20%
2. An L2$ that has a hit time of 15 cycles and has a global miss rate of 5%
3. Main memory where accesses take 100 cycles

$\boxed{5.4}$ What is the local miss rate of L2$?

Answer: 25%

The number of accesses to the L2$ is the number of misses in L1$, so we divide the global miss rate of L2$ with the miss rate of L1$.

$$\text{L2\$ Local Miss Rate} = \frac{\text{Misses in L2\$}}{\text{Accesses in L2}} = \frac{\text{Misses in L2\$}}{\text{Total Accesses}} / \frac{\text{Misses in L1\$}}{\text{Total Accesses}} = \frac{\text{Global Miss Rate}}{\text{L1\$ Miss Rate}} = \frac{5\%}{20\%} = 0.25 = 25\%$$

$\boxed{5.5}$ What is the AMAT of the system?

Answer: $\text{AMAT} = 2 + 20\% \times (15 + 25\% \times 100) = 10$ cycles

The miss penalty of the L1$ is the "local" AMAT of the L2$.

$\boxed{5.6}$ Suppose we want to reduce the AMAT of the system to 8 cycles or lower by adding in a L3$. If the L3$ has a local miss rate of 30%, what is the largest hit time that L3$ can have?

Answer: 30 cycles

Let $H =$ hit time of the cache. Extending the AMAT equation so that the Miss Penalty of the L2$ is the "local" AMAT of the L3$, we can write:

$$\text{AMAT} = 2 + 20\% * (15 + 25\% * (H + 30\% * 100)) \leq 8$$

Solving for H, we find that $H \leq 30$. So, the largest hit time is 30 cycles.