Homework 2

Shenyao Jin

shenyaojin@mines.edu

February 14, 2025

1 Problem 1

Question

Compute cache performance is influenced by several parameters. For each of the following parameters, describe the issues that arise if their value is either too small or too large:

1. Cache size
2. Line size
3. Associativity

Answer

1. **Cache size:**
   - **Too small:** A small cache leads to a high miss rate since frequently accessed data may not fit, increasing the number of memory accesses and slowing performance.
   - **Too large:** While a larger cache reduces misses, it also increases access latency and power consumption. Moreover, excessive size may lead to inefficiencies due to increased tag overhead.
2. **Line size:**
   - **Too small:** Small cache lines may result in excessive tag overhead and inefficient spatial locality utilization, increasing miss rates for workloads with strong locality.
   - **Too large:** Large cache lines improve spatial locality but may increase conflict misses due to fewer available cache blocks. Additionally, they may cause unnecessary data fetching (cache pollution).
3. **Associativity:**
   - **Too low:** Low associativity increases conflict misses, as multiple addresses may map to the same cache set, leading to frequent evictions.
   - **Too high:** Higher associativity reduces conflict misses but increases lookup time and power consumption due to complex replacement policies.

2 Problem 2

Question 1

What is the L1 D-cache miss rate for the `matrix_add` function? How many misses are due to compulsory misses? How many are conflict misses?

Answer

To analyze the L1 D-cache miss rate, we consider the following details:

- The cache is **64KB**, **2-way set associative**, with **64-byte blocks**.
- The arrays `a`, `b`, and `c` are each **128 × 128** integers, where each integer is **4 bytes**.

- The memory addresses of `a`, `b`, and `c` are `0x10000`, `0x20000`, and `0x30000`, respectively.

Step 1: Calculate Data Footprint and Block Usage

- Each row of the array has 128 integers $\times 4$ bytes $= 512$ bytes.
- Since each cache block is 64 bytes, each row occupies $512/64 = 8$ cache blocks.
- The total data footprint of one array is:

$$128 \times 512 = 65536 \text{ bytes} = 64KB$$

- Since we have three arrays, the total data footprint is:

$$3 \times 64KB = 192KB$$

- The L1 cache can only hold 64KB, so it cannot fit all three arrays at once.

Step 2: Analyze Cache Misses

- The cache is **2-way set associative**, meaning that each memory address maps to a specific set with two possible slots.
- The three arrays `a`, `b`, and `c` are spaced apart in memory, but they map to the same cache sets due to their addresses (`0x10000`, `0x20000`, `0x30000`).
- Since each set can only hold two blocks at a time, when accessing `c[i][j]`, it often replaces `a[i][j]` or `b[i][j]`, leading to **conflict misses**.

Step 3: Breakdown of Misses

- **Compulsory Misses:** Each element is accessed for the first time, leading to an initial miss. Since each matrix has $128 \times 128 = 16,384$ elements and each cache block holds $64/4 = 16$ elements, the total compulsory misses across all three matrices is:

$$\frac{128 \times 128}{16} \times 3 = 3072$$

- **Conflict Misses:** Due to limited associativity, repeated accesses evict previous data. Given the mapping pattern, nearly every access results in a miss, leading to approximately:

$$16,384 \text{ additional misses}$$

Final Miss Rate:

$$\frac{(3072 + 16,384)}{3 \times 16,384} \approx 39.1\%$$

Thus, the L1 D-cache miss rate is approximately **39.1%**, with **3072 compulsory misses** and **16,384 conflict misses**.

Question 2

If the L1 hit time is 1 cycle and the L1 miss penalty is 20 cycles, what is the average memory access time?

Answer

The average memory access time (AMAT) is calculated using the formula:

$$AMAT = \text{Hit time} + (\text{Miss rate} \times \text{Miss penalty})$$

From Question 1, we determined that:

1. **L1 hit time** = 1 cycle
2. **L1 miss rate** $39.1\% = 0.391$
3. **L1 miss penalty** = 20 cycles

Substituting these values:

$$AMAT = 1 + (0.391 \times 20)$$

$$AMAT = 1 + 7.82 = 8.82 \text{ cycles}$$

Thus, the average memory access time (AMAT) is **8.82 cycles**.

3 Problem 3

Question 3

You are given a cache that has 16 byte blocks, 512 rows, and is 2-way set associative. Integers are 4 bytes. Give the C code for a loop that has a 100% miss rate in this cache but whose hit rate rises to almost 100% if you double the size of the cache.

Answer

```
#define ARRAY_SIZE 8192  // Larger than cache
int arr[ARRAY_SIZE];

for (int i = 0; i < ARRAY_SIZE; i += 1024) {
    arr[i] += 1;
}
```