Computer Architecture Lab 4

Name: BRYSON KATUU

Reg No: SCT212-0205/2021

**Problem**

Assume we have a computer where the CPI is 1.0 when all memory accesses (including data and instruction accesses) hit in the cache. The cache is a unified (data + instruction) cache of size 256 KB, 4-way set associative, with a block size of 64 bytes. The data accesses (loads and stores) constitute 50% of the instructions. The unified cache has a miss penalty of 25 clock cycles and a miss rate of 2%. Assume 32-bit instruction and data addresses.

a. What is the tag size for the cache?

b. How much faster would the computer be if all memory accesses were cache hits?

**SOLUTION.**

The number of bits needed for the block offset is calculated as **log (64) = 6 bits**.

The total number of sets in the cache is **256 KB ÷ (64 bytes × 4 ways) = 1024 sets**, or **1K**.

So, the number of index bits is **log (1024) = 10 bits**.

This means the number of tag bits is **32 - (10 + 6) = 16 bits**, assuming 32-bit addresses.

Now for performance:

* The overall **CPI (Cycles Per Instruction)** is the sum of the base execution CPI and the extra stall cycles due to cache misses:

**CPI = CPI\_execution + StallCycles\_per\_Instruction**

* If the cache always hits, then **CPI = 1**.
* But when there’s a miss rate, we calculate the stalls like this:

**StallCycles per Instruction = (average memory accesses per instruction) × miss rate × miss penalty**

On average, each instruction has:

* + 1 access for the instruction itself
  + 0.5 access for data (assuming 50% of instructions are loads/stores)

So: **1 + 0.5 = 1.5 accesses per instruction**

Then: **StallCycles = 1.5 × 0.02 × 25 = 0.75**

Therefore: **CPI = 1 + 0.75 = 1.75**

**Problem**

You purchased an Acme computer with the following features:

• 95% of all memory accesses are found in the cache.

• Each cache block is two words, and the whole block is read on any miss.

• The processor sends references to its cache at the rate of 109 words per second.

• 25% of those references are writes.

• Assume that the memory system can support 109 words per second, reads or writes.

• The bus reads or writes a single word at a time (the memory system cannot read or

write two words at once).

• Assume at any one time, 30% of the blocks in the cache have been modified.

• The cache uses write allocate on a write miss.

You are considering adding a peripheral to the system, and you want to know how much

of the memory system bandwidth is already used. Calculate the percentage of memory system bandwidth used on the average in the two cases below. Be sure to state your assumptions.

a. The cache is write through.

b. The cache is write back.

**SOLUTION.**

The **cache miss rate** is **5%** (or 0.05).

Each **cache block** holds **2 words**, which equals **8 bytes**.

The processor sends **10⁹ memory operations per second**.

**25%** of these memory operations are **writes**, so there are **0.25 × 10⁹ writes/sec**.

The **bus** is only capable of transferring **one word at a time** to or from memory.

**30%** of the cache blocks are modified and would need to be written back when evicted (for write-back caches).

The cache follows a **write-allocate** policy, meaning that on a write miss, the block is loaded into the cache before the write.

**Breakdown of Access Fractions:**

* **Read hits** = 75% (reads) × 95% (hit rate) = **0.7125**
* **Read misses** = 75% × 5% = **0.0375**
* **Write hits** = 25% × 95% = **0.2375**
* **Write misses** = 25% × 5% = **0.0125**

**a. Write-Through Cache Behaviour:**

* **Read hit**: No access to main memory.
* **Read miss**: Memory sends **2 words** (entire block) to cache.
* **Write hit**: One word is written **directly to memory**.
* **Write miss**: Memory first sends **2 words** to cache, then cache writes **1 word** to memory.

Average words transferred = 0.7125 ∗ 0 + 0.0375 ∗ 2 + 0.2375 ∗ 1 + 0.0125 ∗ 3 = 0.35

Average bandwidth used = 0.35 ∗ 109

* **Bandwidth used** = 0.35 × 10⁹ words/sec
* So, the **fraction of memory bandwidth used** = 0.35 × 10⁹ / 10⁹ = 0.35 (or **35%**)

**b. Write-Back Cache Behaviour:**

* **Read hit** and **write hit**: No memory access.
* **Read miss**:
  + If the block being replaced is **modified**, write **2 words** back to memory, then load **2 words** from memory.
  + If the block is **clean**, just load **2 words** from memory.
* **Write miss**: Same as read miss, since it's write-allocated.

Average words transferred = 0.7125 ∗ 0 + 0.0375 ∗ (0.7 ∗ 2 + 0.3 ∗ 4) + 0.2375 ∗ 0 + 0.0125 ∗ (0.7 ∗ 2 + 0.3 ∗ 4) = 0.13 Average bandwidth used = 0.13 ∗ 109

* **Bandwidth used** = 0.13 × 10⁹ words/sec
* So, the **fraction of memory bandwidth used** =  
  0.13 × 10⁹ / 10⁹ = 0.13 (or **13%**)

Comparison: From the calculations above, we can see that the **write-through cache consumes more than twice** the memory bandwidth compared to the **write-back cache**. This makes write-back more efficient in terms of bandwidth usage.