Computer Architecture Lab 5

Name: BRYSON KATUU

Reg No: SCT212-0205/2021

**Problem**:

Consider a computer system with a first-level data cache with the following characteristics: size: 16KBytes; associativity: direct-mapped; line size: 64Bytes; addressing: physical. The system has a separate instruction cache, and you can ignore instruction misses in this

problem. This system is used to run the following code:

for (i=0; i<4096; i++)

X[i] = X[i] \* Y[i] + C

Assume that both X and Y have 4096 elements, each consisting of 4 bytes (single precision floating point). These arrays are allocated consecutively in physical memory. The assembly code generated by a naive compiler is the following:

loop: lw f2, 0(r1) # load X[i]

lw f4, 0(r2) # load Y[i]

multd f2, f2, f4 # perform the multiplication

add f2, f2, f0 # add C (in f0)

sw 0(r1), f2 # store the new value of X[i]

addi r1, r1, 4 # update address of X

addi r2, r2, 4 # update address of Y

addi r3, r3, 1 # increment loop counter

bne r3, 4096, loop # branch back if not done

a. How many data cache misses will this code generate? Breakdown your answer into the

three types of misses. What is the data cache miss rate?

b. Provide a software solution that significantly reduces the number of data cache misses.

How many data cache misses will your code generate? Breakdown the cache misses into the

three types of misses. What is the data cache miss rate?

c. Provide a hardware solution that significantly reduces the number of data cache misses.

You are free to alter the cache organization and/or the processor. How many data cache

misses will your code generate? Breakdown the cache misses into the three types of misses.

What is the data cache miss rate?

**SOLUTION.**

1. **Original Case (Baseline Behaviour):**

Types of Cache Misses:

* Cold Misses:  
  Since data is loaded into the cache for the first time every 16 iterations, both X and Y cause:  
  2 × (4096 ÷ 16) = 512 cold misses
* Conflict Misses:  
  Each store to X replaces a Y element, and every one of the remaining 15 Y elements per block also gets evicted by stores to X, leading to:  
  4096 + (15/16) × 4096 = 7936 conflict misses
* Total Misses:  
  512 + 7936 = 8448 misses
* Miss Rate:  
  There are 3 × 4096 = 12288 total memory accesses (loads from X and Y, and store to X)  
  So:  
  Miss rate = 8448 / 12288 = 0.6875 (68.75%)

1. **Software Optimization Approaches**

**Option 1: Interleave X and Y in Memory**

* Store elements of X and Y alternately in memory to share cache lines.
* **Cold Misses**: Every 8 iterations leads to a miss on X:  
  4096 ÷ 8 = 512
* **Conflict Misses**: 0 (conflicts avoided due to interleaving)
* **Total Misses**: 512
* **Miss Rate**: 512 / 12288 = 0.0417 (4.17%)

**Option 2: Separate Arrays in Memory by a Non-16KB Offset**

* Keep X and Y far enough apart in memory to prevent cache interference.
* **Cold Misses**: 2 × (4096 ÷ 16) = 512
* **Conflict Misses**: 0
* **Total Misses**: 512
* **Miss Rate**: 0.0417 (4.17%)

1. **Hardware Optimization Approaches**

**Option 1: Double the Cache Size to 32KB**

* Increases capacity to avoid evictions.
* **Cold Misses**: 2 × (4096 ÷ 16) = 512
* **Conflict Misses**: 0
* **Total Misses**: 512
* **Miss Rate**: 0.0417 (4.17%)

**Option 2: Use Set-Associative Cache**

* Allows multiple blocks to share a set, reducing conflict.
* **Cold Misses**: 512
* **Conflict Misses**: 0
* **Miss Rate**: 0.0417 (4.17%)

**Option 3: Increase Block Size by a Factor of z**

* Larger blocks can reduce cold misses but may cause more conflicts:
  + **Cold Misses**: 4096 ÷ (16 × z)
  + **Conflict Misses**: 2 × 4096 = 8192  
    (each load from Y collides with X, and each store to X evicts Y)

**Option 4: Add a Next-Line Prefetcher**

* Automatically fetches the next cache line to reduce misses:
  + **Cold Misses**: 2 × (4096 ÷ 32) = 256
  + **Conflict Misses**:  
    4096 + (31/32) × 4096 = 8064
  + **Total Misses**: 256 + 8064 = 8320
  + **Miss Rate**: 8320 / 12288 ≈ 0.667 (66.7%)

**Option 5: Use a Victim Cache**

* Small fully associative cache to catch evicted blocks:
  + **Cold Misses**: 4096 ÷ 16 = 256  
    (X misses every 16th time; the rest hit in the victim cache)
  + **Conflict Misses**: 4096  
    (loads from Y still clash with X)
  + **Total Misses**: 256 + 4096 = 4352

**Miss Rate**: 4352 / 12288 ≈ 0.3542 (35.42%)