**Computer Architecture Lab 4**

**Name: Eddy Kimani**

**Reg No.: SCT212-0596/2021**

**E1: CPI**

1. What is the tag size of the cache?

* Cache type: Unified, 4-way set associative
* Cache size: 256 KB = 2¹⁸ bytes
* Block size: 64 bytes = 2⁶ bytes
* Miss rate: 2% (for all accesses)
* Miss penalty: 25 cycles
* CPI (ideal, all hits): 1.0
* 50% of instructions are data accesses (loads/stores)
* Address size: 32 bits

**Step 1: Block Offset**

Block size = 64 bytes = 2^6

Block offset = 6 bits

**Step 2: Number of sets**

Total cache size = 256 KB = 2^18 bytes

Block size = 2^6 bytes

Cache has 4-way associativity, thus, number of sets = (2^18 / 2^6) / 4 = 2^12 sets

**Step 3: Tag Bits**

Total address size = 32 bits

Tag = 32 – (Set index + Block offset) = 32 – (12 + 6)

= 14 bits

1. How much faster would the computer be if all memory access were cache hits

Let’s assume:

* For every instruction, there’s 1 instruction fetch
* 50% of instructions also have 1 data access

Therefore,

Average memory access per instruction = 1 fetch + 0.5 data = 1.5 memory accesses

**CPI with Misses**

Miss rate = 2% = 0.02

Miss penalty = 25 cycles

Extra CPI due to misses = 1.5 x 0.02 x 25 = 0.75

CPI = 1.0 + 0.75 = 1.75

**Speedup**

Speedup = Performance with hits / Performance with misses

Speedup = 1.75 / 1.0 = 1.75

The computer would be 1.75 times faster if all memory accesses were cache hits

**E2: Memory system bandwidth percentage**

* Memory reference rate = 10⁹ words/sec
* Cache hit rate = 95%, miss rate = 5%
* Write ratio = 25% of memory references
* Read ratio = 75%
* Cache block size = 2 words
* Bus supports 1 word per operation
* Memory system bandwidth = 10⁹ words/sec (read or write)
* Cache write policy: first write-allocate
* At any time, 30% of blocks are dirty (for write-back)
* On a miss, the entire block (2 words) is fetched from memory

Let’s assume

Memory access per second = 10⁹ memory access/sec

Writes/sec = 0.25 x 10⁹ = 2.5 x 108

Reads/sec = 0.75 x 10⁹ = 7.5 x 108

1. **Cache is write-through**

**Read Hits**

95% of 7.5 x 108 = 7.125 x 108  (no memory traffic)

**Read Misses**

5% of 7.5 x 108 = 3.75 x 107

Each read miss loads 2 words (1 block) so, memory traffic = 3.75×10⁷ × 2 = 7.5×10⁷ words/sec

**Write Hits**

95% of 2.5 x 108 = 2.375 x 108

In write-through, every write goes to memory so, memory traffic = 2.375x108 words/sec

**Write Misses**

5% of 2.5×10⁸ = 1.25×10⁷

In Write-allocate, read full block (2 words), then write 1 word

Total traffic = 2 (read) + 1 (write) = 3 words per write miss

Traffic = 1.25×10⁷ × 3 = 3.75×10⁷ words/sec

**Total memory traffic**

7.5 x 107 (read miss) + 2.375 x 108 (write hits) + 3.75 x 107 (write misses) = 3.5 x 108 words/sec

**Bandwidth usage**

(3.5 x 108) / 109 = 35%

1. Write-back

**Read Hits**

95% of 7.5 x 108 = 7.125 x 108  (no memory traffic)

**Read Misses**

5% of 7.5 x 108 = 3.75 x 107

Each read miss loads 2 words (1 block) so, memory traffic = 3.75×10⁷ × 2 = 7.5×10⁷ words/sec

**Write Hits**

Write-back, write goes to cache so no memory traffic

**Write Misses**

1.25×10⁷ write misses/sec

Write-allocate → must read 2 words

Afterward, write is to cache, not memory

Traffic = 1.25×10⁷ × 2 = 2.5×10⁷ words/sec

**Write-backs**

Dirty blocks evicted on replacement

From 5×10⁷ misses/sec, assume 30% are dirty

Write-back cost = 2 words per dirty block

Write-back traffic = 0.3 × 5×10⁷ × 2 = 3×10⁷ words/sec

**Total memory traffic**

7.5 x 107 (read miss) + 2.5 x 107 (write miss) + 3 x 107 (write-back) = 1.3 x 108 words/sec

**Bandwidth usage**

(1.3 x 108 ) / 109 = 13 %

**E3: Write-through and write-back cache**

CPU Performance Equation

CPU Time = Instruction Count (IC) x CPI x Clock Time

CPI = CPI execution + Stall Cycles per Instruction

**Given**,

Instruction cache miss penalty = 50 cycles

Data cache read hit = 1 cycle

Data cache write hit = 2 cycles

Data cache miss penalty = 50 cycles (write-through), 50 or 100 cycles (write-back, 50% dirty blocks)

**Miss rates**

Instruction cache (MRI) = 0.5%

Data cache (MRD) = 1%

**Instruction mix**

Loads = 26%

Stores = 9%

Others = 65%

**Base execution CPI with no stalls**

CPI = (0.26 x 1) + (0.09 x 2) + (0.65 x 1) = 1.09

**Stall cycles per instruction**

**Write-through cache**

Stall = (0.005 x 50) + 0.01 x [(0.26 x 50) + (0.09 x 50)] = 0.25 + 0.175 = 0.425

CPI = 1.09 + 0.425 = 1.515

**Write-back cache**

Data misses may cost 50 cycles (clean) or 100 cycles (dirty)

Average penalty = 150 / 2 = 75

Stall = (0.005 x 50) + 0.01 x [(0.26 x 75) + (0.09 x 75)] = 0.25 + 0.2625 = 0.5125

CPI = 1.09 + 0.5125 = 1.6025

(1.6025 – 1.515) / 1.515 0.0578 6%

Write-back cache system is approximately 6% slower that write-through