**TANIA MAINA.**

**SCT212-0179/2021.**

**BCT2408: COMPUTER ARCHTECTURE.**

E1

a)

Given:

* Cache size = 256 KB = 2²⁸ bytes
* Cache associativity = 4-way set associative
* Block size = 64 bytes = 2⁶ bytes
* Instruction/Data address = 32 bits
* Miss penalty = 25 cycles
* Miss rate = 2%
* Base CPI (if all hits) = 1.0
* 50% of instructions access memory (i.e., data loads/stores)

To determine the number of blocks:

Total blocks = Cache size / Block size  
 = 256 KB / 64 B  
 = (2²⁸) / (2⁶)  
 = 2²² blocks

To determine the number of sets

In a 4-way set associative cache:  
 Number of sets = Total blocks / Associativity  
 = 2²² / 4 = 2²⁰ sets

Given a 32-bit address, we divide it as:

Offset bits = log₂(Block size) = log₂(64) = 6 bits

Index bits = log₂(Number of sets) = log₂(2²⁰) = 20 bits

Tag bits = Remaining bits = 32 - 6 - 20 = 6 bits

*Tag size is 6 bits*

b)

Compare actual CPI (with misses) vs ideal CPI (all hits = 1.0) and then compute the speedup.

To understand the cost of a miss:

Miss rate = 2% = 0.02  
 Miss penalty = 25 cycles  
 Memory access frequency = 50% = 0.5 of all instructions

Only memory accesses can have cache misses.

So, average miss cycles per instruction =  
 = Miss rate × Miss penalty × Memory access frequency  
 = 0.02 × 25 × 0.5 = 0.25 cycles per instruction

So, actual CPI = base CPI + miss penalty  
 = 1.0 + 0.25 = 1.25

**To compute speedup:**

Speedup = CPI (no misses) / CPI (with misses)  
 = 1.0 / 1.25 = 0.8

This means the actual system is 0.8× the speed of the ideal.  
 To get how much faster the system would be if all accesses were hits:

That’s the reciprocal: = 1.25 / 1.0 = 1.25× faster

The computer would be 1.25× faster if all memory accesses were cache hits.

E2

Given:

* Cache hit rate = 95%
* Miss rate = 5%
* Cache block = 2 words
* Access rate = 10⁹ words/sec
* Write frequency = 25%
* Memory bandwidth = 10⁹ words/sec
* Bus is 1 word wide
* 30% of blocks in cache are dirty (used in write-back only)
* Write allocate on write miss

**CASE (a): Write-Through Cache**

**Characteristics of Write-Through:**

* All writes go to both cache and memory.
* On a read miss, the entire block (2 words) is read from memory.
* On a write miss, the block is first read (2 words), then write-through occurs (1 word write).

**To analyze the memory traffic from each type of operation:**

**Reads (75% of accesses)**

95% are hits → no memory traffic

5% are misses → need to read 2 words from memory (the block)  
 → Miss rate = 0.05 × 75% = 3.75% of total accesses  
 → Memory traffic = 0.0375 × 10⁹ accesses/sec × 2 words = 75 million words/sec

**Writes (25% of accesses)**

Write-through: all writes go to memory.

Write hits (95%) → 0.25 × 0.95 × 10⁹ = 237.5 million writes/sec

Write misses (5%) →

Write allocate → read 2 words (entire block)

Then write the word to memory

Therefore:

Write misses = 0.25 × 0.05 × 10⁹ = 12.5 million accesses/sec

Memory traffic from write misses = 12.5M reads (2 words each = 25M) + 12.5M writes  
 → 25M + 12.5M = 37.5 million words/sec

Total write traffic = 237.5M (from write hits) + 37.5M = 275 million words/sec

**Total memory traffic:**

Read traffic: 75 million

Write traffic: 275 million

Total = 350 million words/sec

**Percentage of bandwidth used:**

350 million / 1000 million=0.35=35%

35% of memory bandwidth is used with a write-through cache

**CASE (b): Write-Back Cache**

**Characteristics of Write-Back:**

Writes are only sent to memory when a dirty block is evicted

Write hits: no memory traffic

Write misses: load block (2 words), write to cache

Evictions cause dirty blocks to be written back (2 words per dirty block)

We’ll estimate memory traffic per second based on misses and dirty block write-backs.

**To analyze the memory traffic from each type of operation:**

1. Read accesses (75%)

Miss rate = 5% of 75% = 3.75%

Read 2 words on each miss → 0.0375 × 10⁹ × 2 = 75 million words/sec

2. Write accesses (25%)

Write hits → stay in cache → 0 memory traffic

Write misses (5% of 25% = 1.25%) → write-allocate:

Load 2-word block from memory → 0.0125 × 10⁹ × 2 = 25 million words/sec

No write to memory immediately

→ Write traffic from write misses = 25 million words/sec

3. Dirty block evictions

5% of all memory accesses are cache misses.  
 That’s 0.05 × 10⁹ = 50 million misses/sec.

Assuming 30% of the replaced blocks are dirty, each dirty eviction writes 2 words.

→ 0.30 × 50M = 15 million evictions/sec  
 → Each writes 2 words → 30 million words/sec

**Total memory traffic:**

Read misses: 75 million

Write misses: 25 million

Dirty evictions: 30 million

Total = 130 million words/sec

Percentage of bandwidth used:

130 million / 1000 million=0.13=13%

3% of memory bandwidth is used with a write-back cache. Conclusively Write-back caches save significant memory bandwidth by avoiding unnecessary writes and buffering modified data until replacement.

E3.

Given:

Parameter, Value

Cache read hit time, 1 clock cycle

Write time (hit, both types), 2 clock cycles

Cache miss penalty (read miss), 50 clock cycles

Block write to memory, 50 clock cycles

Instruction cache miss rate, 0.5% = 0.005

Data cache miss rate, 1% = 0.01

Load instructions, 26% of all instructions

Store instructions, 9% of all instructions

Dirty blocks (write-back only), 50%

Write buffer never stalls CPU (write-through),

Definitions and Assumptions

We’re calculating effective average CPI overhead due to memory stalls.

Let’s break it into three parts:

* Instruction cache stalls
* Data read stalls (loads)
* Data write stalls (stores)

We'll compute this for both:

(A) Write-through cache

(B) Write-back cache

**(A) WRITE-THROUGH CACHE**

*1. Instruction Cache Misses*

Miss rate = 0.005

Penalty = 50 cycles

Contribution to CPI:

0.005×50=0.25 cycles/instruction

*2. Data Reads (Loads)*

Load freq = 26%

Miss rate = 0.01

Penalty = 50 cycles

0.26×0.01×50=0.13 cycles/instruction

*3. Data Writes (Stores)*

Store freq = 9%

All writes go to memory on write-through

CPU doesn't stall due to write buffer → no penalty

BUT write misses still incur a penalty

Miss rate = 0.01  
 On a write miss: read block (50 cycles), then write word (ignored due to buffer assumption)

→ Stall only due to read on write miss

0.09×0.01×50=0.045 cycles/instruction

*Total CPI Penalty for Write-Through:*

0.25+0.13+0.045=0.425 cycles/instruction

️ **(B) WRITE-BACK CACHE**

Key difference: only write misses or dirty block evictions cause memory access.

Also, CPU must wait for a write-back (no buffer assumption given here).

*1. Instruction cache misses*

Same as write-through:

0.005×50=0.25 cycles/instruction

*2. Data reads (Loads)*

Same load freq and miss rate.

0.26×0.01×50=0.13 cycles/instruction

*3. Data writes (Stores)*

Store freq = 9%

Write hits = no stall

Write misses = read block from memory (50 cycles)

1% of 9% = 0.0009 instructions cause write miss

→ Misses only result in read from memory on write-allocate

0.09×0.01×50=0.045 cycles/instruction

*4. Eviction of dirty blocks*

Every miss (load or store) may evict a block. If it's dirty, write-back occurs.

We assume 50% of evicted blocks are dirty.

Let’s estimate total cache misses per instruction:

Instruction miss rate = 0.005

Data read misses = 0.26 × 0.01 = 0.0026

Data write misses = 0.09 × 0.01 = 0.0009  
 → Total misses per instruction = 0.005 + 0.0026 + 0.0009 = 0.0085

*If 50% of evictions are dirty:*

0.0085×0.5×50=0.2125 cycles/instruction

*Total CPI Penalty for Write-Back:*

0.25+0.13+0.045+0.2125=0.6375 cycles/instruction

Conclusively;

Cache Type, Memory CPI Penalty

Write-through, 0.425

Write-back, 0.6375

The write-through cache performs better in this case because the write buffer prevents CPU stalls on writes, and dirty block write-backs in the write-back cache add significant latency.