Microprocessor & Computer Architecture (μpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

Microprocessor & Computer Architecture (µpCA)

Unit 3: Performance Analysis

Dr. D. C. Kiran

Department of Computer Science and Engineering

Syllabus

~~**Unit 1: Basic Processor Architecture and Design**~~

~~**Unit 2: Pipelined Processor and Design**~~

Unit 3: Memory

- ~~Memory Hierarchy~~
- ~~Principles of Locality~~
- ~~Cache Design Principles~~

Mapping Functions

- ~~Direct Mapping~~
- ~~Full Associative Mapping~~
- ~~Set Associative Mapping~~
- ~~Cache Replacement Policy~~
- ~~Read & Write Policy~~
- **Performance Analysis**

Unit 4: Input/Output Device Design

Unit 5: Advanced Architecture

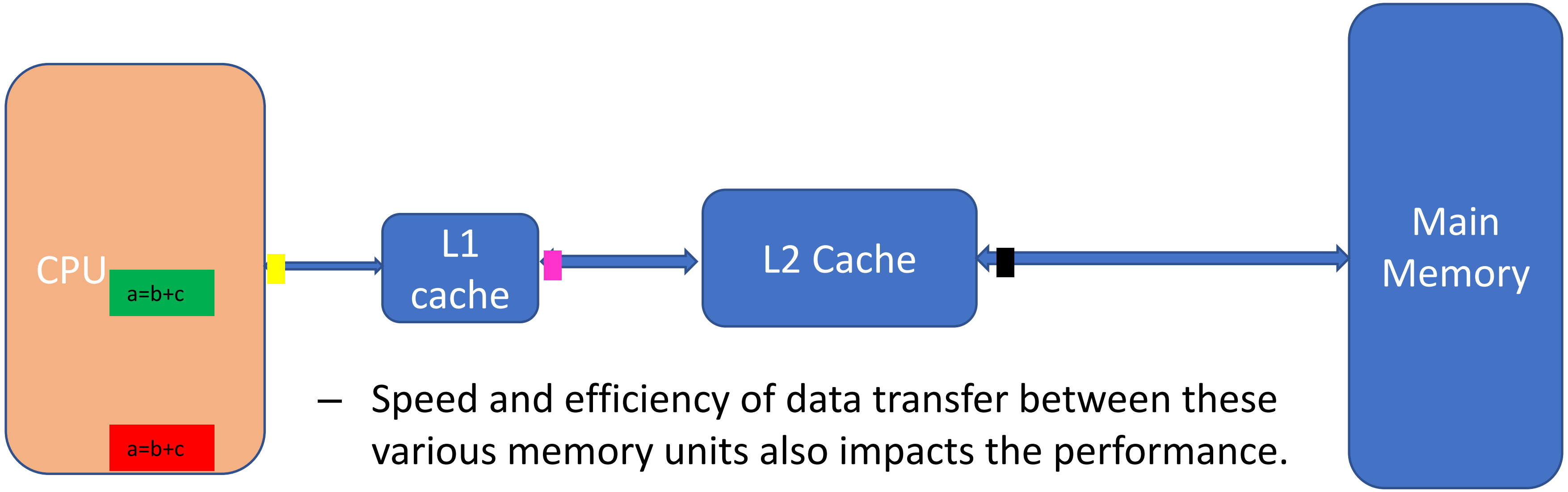
Performance Analysis Known Facts

- User expect best possible performance at lowest possible cost.

- Performance depend on:-
  - ✓ How fast the Data / Instructions can be provided for execution?
  - ✓ How fast the Instructions can be executed?

- Memory ***Hierarchy*** is to balance Size, Cost and Speed

- However, data need to be transferred between various unit in the ***Hierarchy***



  – Speed and efficiency of data transfer between these various memory units also impacts the performance.

Microprocessor & Computer Architecture (μpCA)

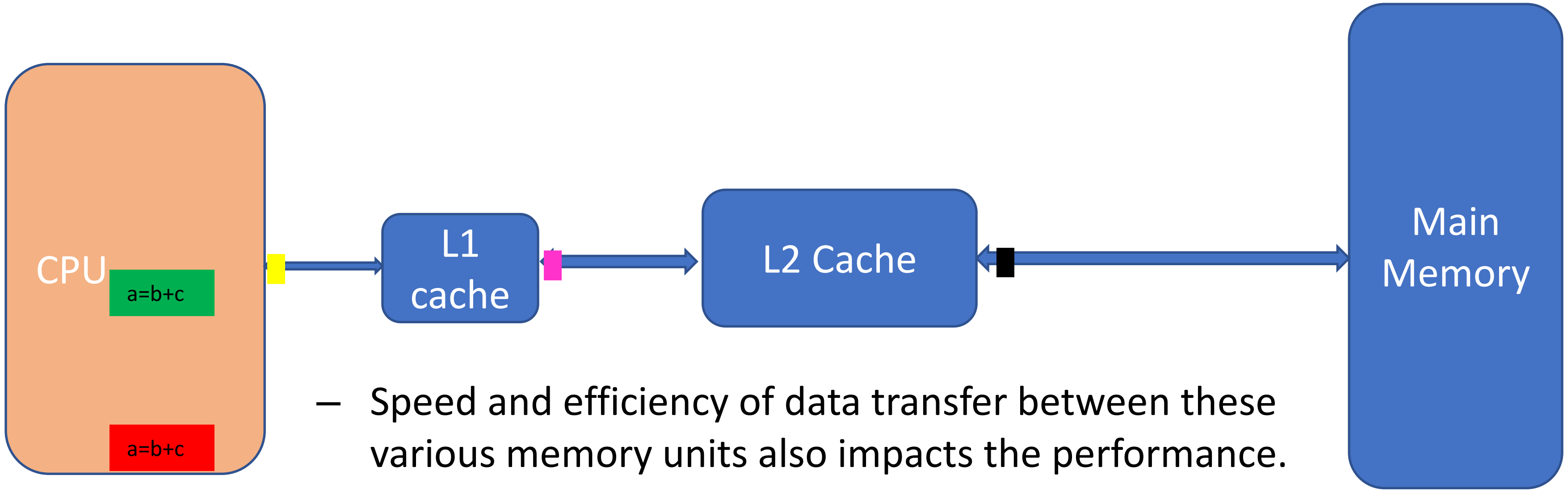
Performance Analysis

To examine the performance of a memory system, the important factors are:

— How long does it take to send data from the cache to the CPU? (Hit Time) [Indexing, Tag Comparison and Transfer)

— How long does it take to copy data from memory into the cache? (Miss Penalty) [# Additional Cycles required to fetch data from next level/s of memory when Miss]

— How often do we have to access main memory? (Miss Rate)

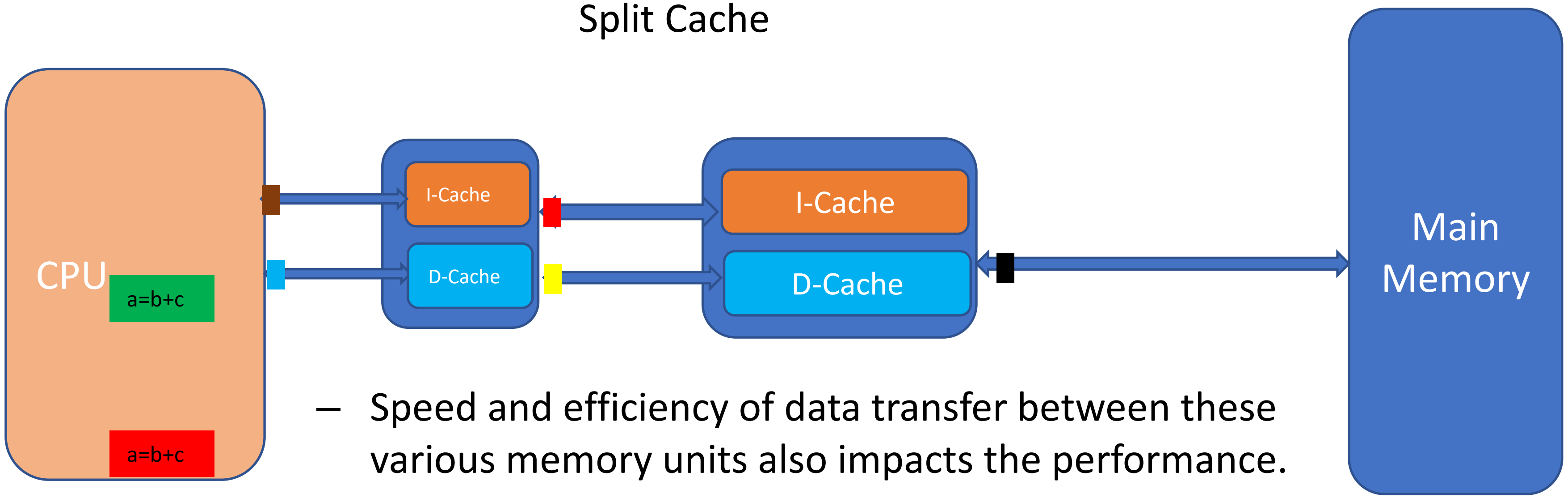


– Speed and efficiency of data transfer between these various memory units also impacts the performance.

Microprocessor & Computer Architecture (μpCA)

Performance Analysis With I-Cache & D-Cache

Split Cache



- Speed and efficiency of data transfer between these various memory units also impacts the performance.

Memory Access include ***Instruction Fetch*** and ***Data Access*** for Load and Store.

Miss Rate of Instruction Access and Data Access need not same.

Miss Penalty of Instruction Access and Data Access need not be same.

Miss Penalty

Three steps are taken when a cache needs to load data from the main memory.

1: Sending Address to RAM
2: Accessing data from RAM
3: Receiving data from RAM

Example: If the buses from the CPU to the cache and from the cache to RAM are all one word wide. Memory accesses take 15 cycles, If the cache has one-word blocks, How much time required for filling a block from RAM **(*i.e.*, the miss penalty)?**

1. It takes 1 cycle to send an address to the RAM.
2. If there is a 15-cycle latency for each RAM access.
3. It takes 1 cycle to return data from the RAM.

Miss Penalty is: 1 + 15 + 1 = 17 clock cycles

Miss Penalty

Example2: If the buses from the CPU to the cache and from the cache to RAM are all one word wide. Memory accesses take 15 cycles, If the cache has **four-word blocks**, How much time required for filling a block from RAM **(*i.e.*, the miss penalty)?**

Solution 1: If the cache has four-word blocks, then loading a single block would need four individual main memory accesses, and a miss penalty of 68 cycles!

4 x (1 + 15 + 1) = 68 clock cycles

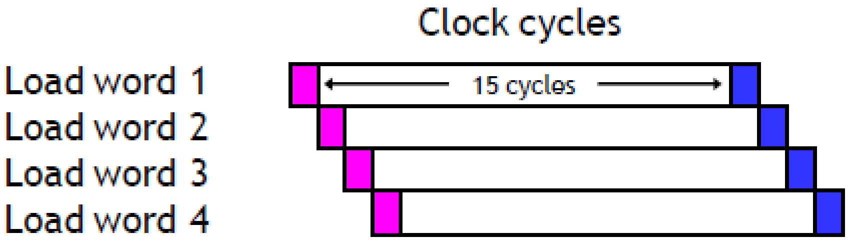
Solution 2: Widen the Bus to read multiple word in single shot.

1 + 15 + 1 = 17 cycles

But it will be Costly!☹

Miss Penalty

Solution 3: Create individual bank for each word and Overlap the Latencies.



1 + 15 + (4 x 1) = 20 cycles

- The magenta cycles represent sending an address to a memory bank.
- Each memory bank has a 15-cycle latency, and
- It takes another cycle (shown in blue) to return data from the memory.

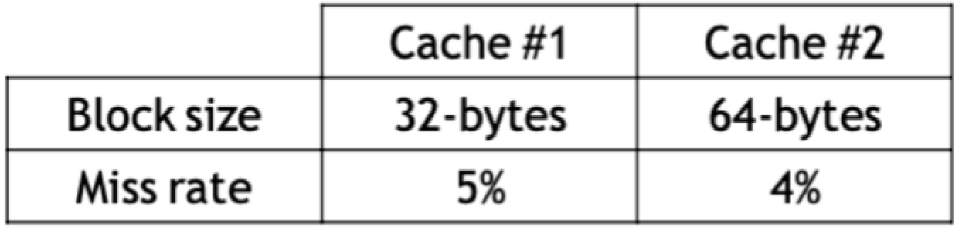
This is the same basic idea as pipelining!
—As soon as we request data from one memory bank, we can go ahead and request data from another bank as well.
—Each individual load takes 17 clock cycles, but four overlapped loads require just 20 cycles.

Average Memory Access Time (AMAT)

AMAT = Hit Time + (Miss Rate x Miss Penalty)

- This is just averaging the amount of time for cache hits and the amount of time for cache misses.
- Lower AMAT is better
- Reduce Miss Penalty or Miss Rate

CACHE PERFORMANCE – EXAMPLE 1



	Cache #1	Cache #2
Block size	32-bytes	64-bytes
Miss rate	5%	4%

Which cache configuration would be better?

Assume both caches have single cycle hit times.

Memory accesses take 15 cycles, and the memory bus is 8-bytes wide

CACHE PERFORMANCE – EXAMPLE 1

Cache #1 : 32-byte memory access takes 20 cycles:
1 (send address) + 15 (memory access) + 4 (four 8-byte transfers)

Miss Penalty=1+15+32B/8B=20cycles

- AMAT=1+0.05x20=2

Cache #2 :64-byte memory access takes 24 cycles:
1 (send address) + 15 (memory access) + 8 (four 8-byte transfers)

Miss Penalty = 1 + 15 + 64B/8B = 24 cycles

- AMAT = 1 + 0.04 x 24 = 1.9

Execution Time

We Know that

CPU Execution Time = IC x Overall CPI x Cycle time

Overall CPI = (Base CPI + CPU Stalls)

How does cache hits and misses affect system performance?

Overall CPI = (Base CPI + CPU Stalls + Memory Stalls)

Memory Stall Cycles = Memory Accesses x Miss Rate x Miss Penalty

Note: Memory Access= Instruction Fetch + Data Referred during Load or Store

Cache Performance Example 2

Assume that 33% of the instructions in a program are data accesses. The cache hit ratio is 97% and the hit time is one cycle, but the miss penalty is 20 cycles.

Solution

Memory Stall Cycles = Memory Accesses x Miss Rate x Miss Penalty

= 0.33 x 0.03 x 20 cycles
= 0.2 cycles

CPI = (Base CPI + CPU Stalls + Memory Stalls)

CPI = [1 + 0 + 0.2] = 1.2

CPU Execution Time = IC x 1.2 x cycle Time

This code is 1.2 times slower than a program with a “perfect” CPI of 1!

Memory systems are a Bottleneck

What if we could *double* the CPU performance so the CPI becomes 0.5, but memory performance remained the same?

CPI= 0.5+0.2= 0.7
CPU time = IC x (0.7) x Cycle time

- The overall CPU time improves by just 1.2/0.7 = 1.7 times ☹

CACHE PERFORMANCE – EXAMPLE 3

Assume we have a computer where the cycles per instruction (CPI) is 1.0 when all memory accesses hit in the cache. The cache is a split cache (D-Cache + I-Cache). The only data accesses are loads and stores, and these total 50% of the instructions. If the miss penalty is 25 clock cycles and the miss rate is 2%, how much faster would the computer be if all instructions were cache hits?

SOLUTION:

First compute the performance for the computer that always hits:

CPU execution time = IC × (CPI + Memory stall cycles) × Clock cycle

= IC × (CPI + 0) × Clock cycle

= IC × 1.0 × Clock cycle

Now for the computer with the real cache, first we compute memory stall cycles:

CPU Execution Time = IC x (Base CPI + CPU Stalls + Memory Stalls) x Cycle time

Cache Performance – Example 3

CPU Execution Time = IC x (Base CPI + CPU Stalls + Memory Stalls) x Cycle time

CPU Execution Time = IC x (1 + 0 + Memory Stalls) x Cycle time

Memory Stall Cycles = Memory Accesses x Miss Rate x Miss Penalty
= (1+0.5) x .02 x 25
= 0.75

// Where memory access (Instruction Fetch + Data Access)

CPU Execution Time = IC x (1 + 0 + 0.75) x Cycle time
= IC x (1.75) x Cycle time

The performance ratio is the inverse of the execution times:

$$Speedup = \frac{\text{CPU Execution Time } \textit{with real cache}}{\text{CPU Execution Time } \textit{with all hits}} = \frac{1.75}{1} = 1.75$$

Hence, The computer with no cache misses is 1.75 times faster.

Cache Performance Example 4

Given

- I-cache miss rate = 2%
- D-cache miss rate = 4%
- Miss penalty = 100 cycles
- Base CPI (ideal cache) = 2
- Load & stores are 36% of instructions

Miss cycles per instruction

- I-cache: 0.02 × 100 = 2
- D-cache: 0.36 × 0.04 × 100 = 1.44

Actual CPI = 2 + 2 + 1.44 = 5.44

Speedup = IC X 5.44 X Clock Cycle/ IC X 2 X Clock Cycle

Conclusion: Ideal CPU is 5.44/2 =2.72 times faster

CACHE PERFORMANCE – EXAMPLE 5

Consider a pipelined processor that has an average CPI of 1.8 without accounting for memory stalls. I-Cache has a hit rate of 95% and the D-Cache has a hit rate of 98%. Assume that memory reference instructions account for 30% of all the instructions executed. Out of these 80% are loads and 20% are stores. On average, the read-miss penalty is 20 cycles and the write-miss penalty is 5 cycles. Compute the effective CPI of the processor accounting for the memory stalls.

Note:
Avg CPI = 1.8;
I-cache miss ratio = (1-0.95) = **0.05**;
D-cache miss ratio = (1-0.98) = 0.02; 30% mem references (80% loads & 20% stores)
Read miss penalty = 20 cycles; Write miss penalty = 5 cycles;

CACHE PERFORMANCE – EXAMPLE 4

Cost of instruction misses = cache miss rate * read miss penalty
= 1***0.05*** 20
= 1 cycle per instruction

- Cost of data read misses = fraction of memory reference instructions in program *
fraction of memory reference instructions that are **loads** *
D-cache miss rate * **Read miss penalty**
= 0.3 * 0.8 * 0.02 * 20
= 0.096 cycles per instruction
- Cost of data write misses = fraction of memory reference instructions in the program *
fraction of memory reference instructions that are **stores** *
D-cache miss rate * **Write miss penalty**
= 0.3 * 0.2 * 0.02 * 5
= 0.006 cycles per instruction

Effective CPI = Avg CPI + Effect of I-Cache on CPI + Effect of D-Cache on CPI
= 1.8 + 1 + 0.096 + 0.006
= **2.902**

Next Session

Cache Optimizations

THANK YOU

Dr. D. C. Kiran

Department of Computer Science and Engineering

dckiran@pes.edu

9829935135