CS / EE 320
Computer Organization and Assembly Language

Lecture 25
Spring 2024

Shahid Masud

Topics: Cache/Memory Update Strategies, Cache Performance, Multi Level Caches

Topics

- Examples of calculating different parameters in Cache Design
- Cache Replacement: LRU, LFU, FIFO, Random schemes
- Cache Writes – Write Back and Write Through schemes
- Some Examples of Cache Performance
- Introduction to Multi-level Caches
- Examples of Cache Performance

Quiz 6 Next Monday

Dealing with Cache Miss and Cache Line Replacement

Cache Misses

- On cache hit, CPU proceeds normally
- On cache miss
  - Stall the CPU pipeline
  - Fetch block from next level of hierarchy
  - Instruction cache miss
    - Restart instruction fetch
  - Data cache miss
    - Complete data access

Sources of Misses

- Compulsory misses (aka cold start misses)
  - First access to a block
- Capacity misses
  - Due to finite cache size
  - A replaced block is later accessed again
- Conflict misses (aka collision misses)
  - In a non-fully associative cache
  - Due to competition for entries in a set
  - Would not occur in a fully associative cache of the same total size

Replacement Algorithms (1) Direct mapping

- No choice
- Each block only maps to one line
- Replace that line

Replacement Algorithms (2)
Associative & Set Associative

- Hardware implemented algorithm (speed)
- Least Recently used (LRU)
- e.g. in 2 way set associative
    - Which of the 2 block is lru?
- First in first out (FIFO)
    - replace block that has been in cache longest
- Least frequently used
    - replace block which has had fewest hits
- Random

Write Policy

- Must not overwrite a cache block unless main memory is up to date
- Multiple CPUs may have individual caches
- I/O may address main memory directly

Write-through scheme

- All writes go to main memory as well as cache
- Multiple CPUs can monitor main memory traffic to keep local (to CPU) cache up to date
- Lots of traffic
- Slows down writes

Write-Through

- On data-write hit, could just update the block in cache
    - But then cache and memory would be inconsistent
- Write through: also update memory
- But makes writes take longer
    - e.g., if base CPI = 1, 10% of instructions are stores, write to memory takes 100 cycles
        - Effective CPI = 1 + 0.1×100 = 11
- Solution: write buffer
    - Holds data waiting to be written to memory
    - CPU continues immediately
        - Only stalls on write if write buffer is already full

Write-back

- Updates initially made in cache only
- Update bit for cache slot is set when update occurs
- If block is to be replaced, write to main memory only if update bit is set
- Other caches get out of sync
- I/O must access main memory through cache
- N.B. 15% of memory references are writes

Write-Back

- Alternative: On data-write hit, just update the block in cache
  - Keep track of whether each block is dirty
- When a dirty block is replaced
  - Write it back to memory
  - Can use a write buffer to allow replacing block to be read first

Write Miss

- What should happen on a write miss?
- Alternatives for write-through
  - Allocate on miss: fetch the block
  - Write around: don’t fetch the block
    - Since programs often write a whole block before reading it (e.g., initialization)
- For write-back
  - Usually fetch the block

Block Placement

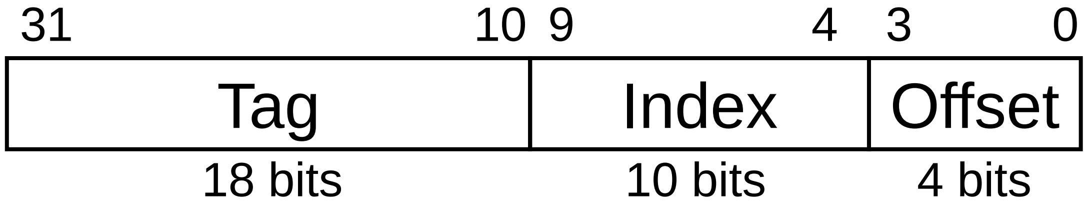
- Determined by associativity
  - Direct mapped (1-way associative)
    - One choice for placement
  - n-way set associative
    - n choices within a set
  - Fully associative
    - Any location
- Higher associativity reduces miss rate
  - Increases complexity, cost, and access time

Cache Design Trade-offs

Design change	Effect on miss rate	Negative performance effect
Increase cache size	Decrease capacity misses	May increase access time
Increase associativity	Decrease conflict misses	May increase access time
Increase block size	Decrease compulsory misses	Increases miss penalty. For very large block size, may increase miss rate due to pollution.

Cache Control

- Example cache characteristics
  - Direct-mapped, write-back, write allocate
  - Block size: 4 words (16 bytes)
  - Cache size: 16 KB (1024 blocks)
  - 32-bit byte addresses
  - Valid bit and dirty bit per block
  - Blocking cache
    - CPU waits until access is complete

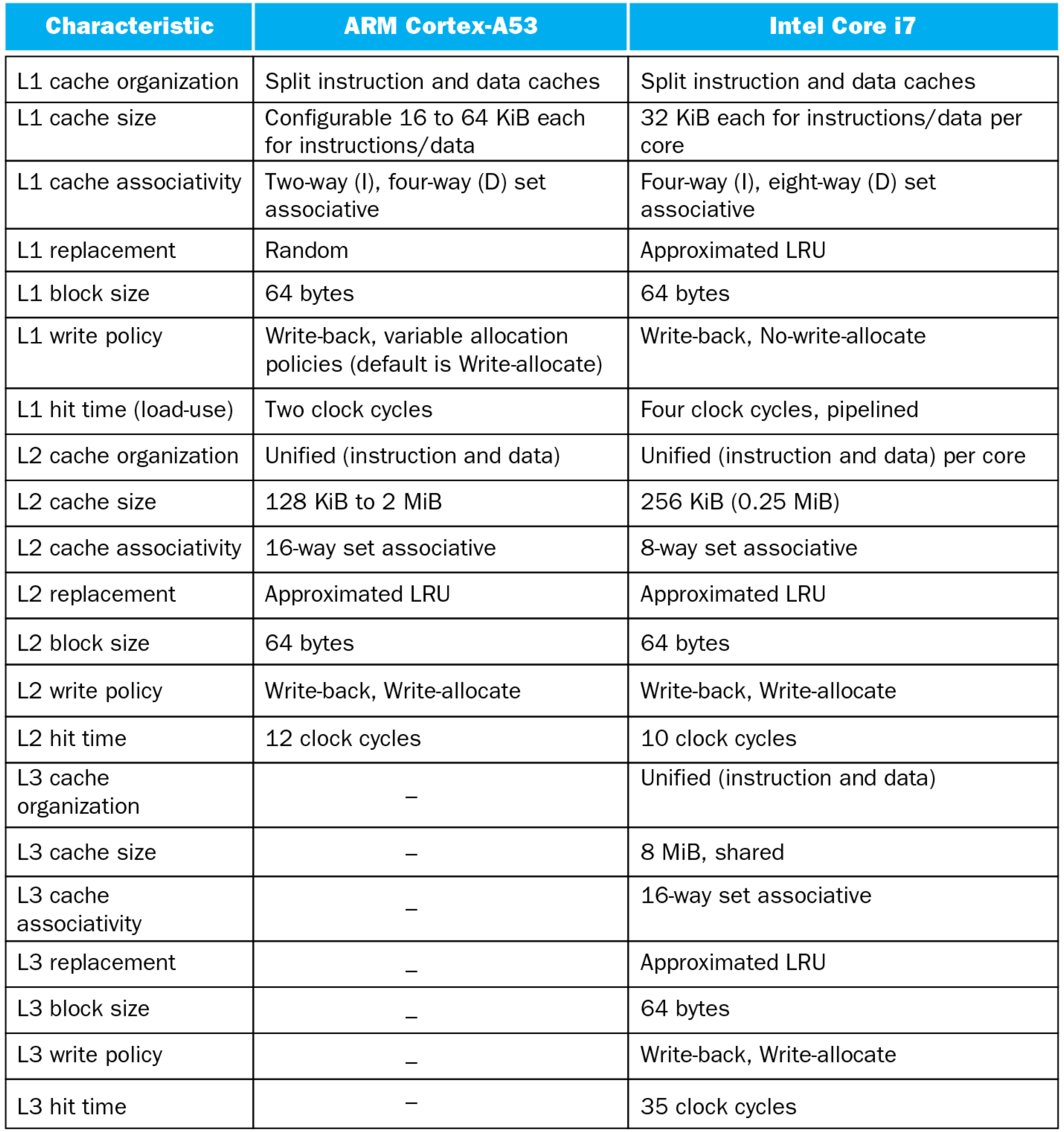


31 10 9 4 3 0

Tag | Index | Offset

18 bits | 10 bits | 4 bits

Multilevel On-Chip Caches



Characteristic	ARM Cortex-A53	Intel Core i7
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	Configurable 16 to 64 KiB each for instructions/data	32 KiB each for instructions/data per core
L1 cache associativity	Two-way (I), four-way (D) set associative	Four-way (I), eight-way (D) set associative
L1 replacement	Random	Approximated LRU
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, variable allocation policies (default is Write-allocate)	Write-back, No-write-allocate
L1 hit time (load-use)	Two clock cycles	Four clock cycles, pipelined
L2 cache organization	Unified (instruction and data)	Unified (instruction and data) per core
L2 cache size	128 KiB to 2 MiB	256 KiB (0.25 MiB)
L2 cache associativity	16-way set associative	8-way set associative
L2 replacement	Approximated LRU	Approximated LRU
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L2 hit time	12 clock cycles	10 clock cycles
L3 cache organization	–	Unified (instruction and data)
L3 cache size	–	8 MiB, shared
L3 cache associativity	–	16-way set associative
L3 replacement	–	Approximated LRU
L3 block size	–	64 bytes
L3 write policy	–	Write-back, Write-allocate
L3 hit time	–	35 clock cycles

Concluding Remarks

- Fast memories are small, large memories are slow
  - We really want fast, large memories
  - Caching gives this illusion
- Principle of locality
  - Programs use a small part of their memory space frequently
- Memory hierarchy
  - L1 cache ↔ L2 cache ↔ … ↔ DRAM memory ↔ disk
- Memory system design is critical for multiprocessors

- **Multi Level Caches**
- **Performance of Cache and Memory System**
- **Examples of Cache Performance**

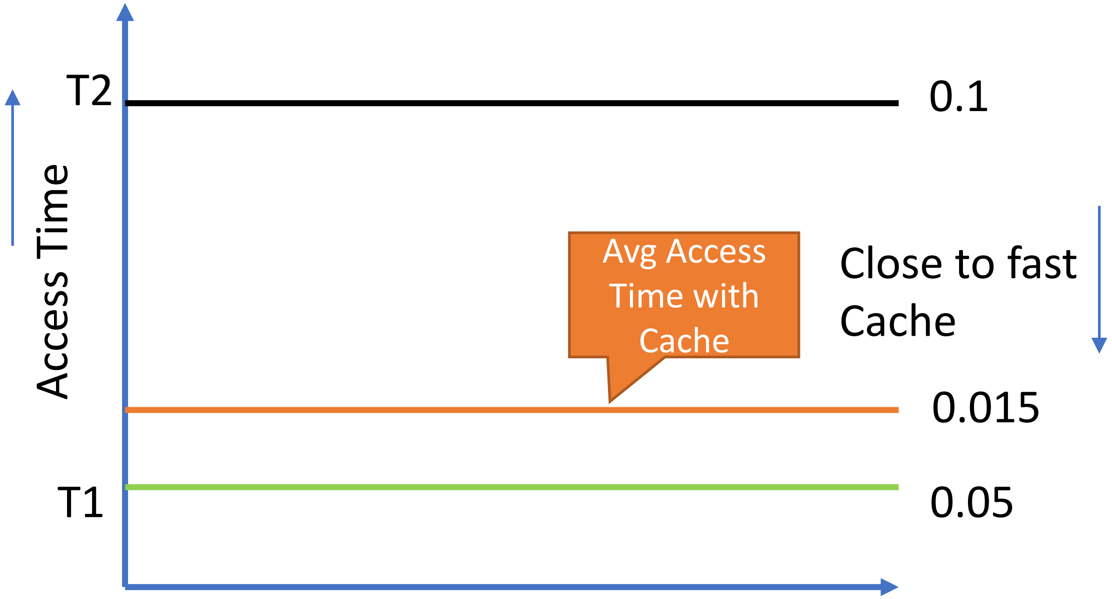
Example 1 of Cache Performance

Question

Given that:
T1 is access time from Cache = 0.01 µsec
T2 is access time from Main Memory = 0.1 µsec
95% accesses are from Cache
5% accesses are from Main Memory
Find the Average Time seen by CPU to Access one word?

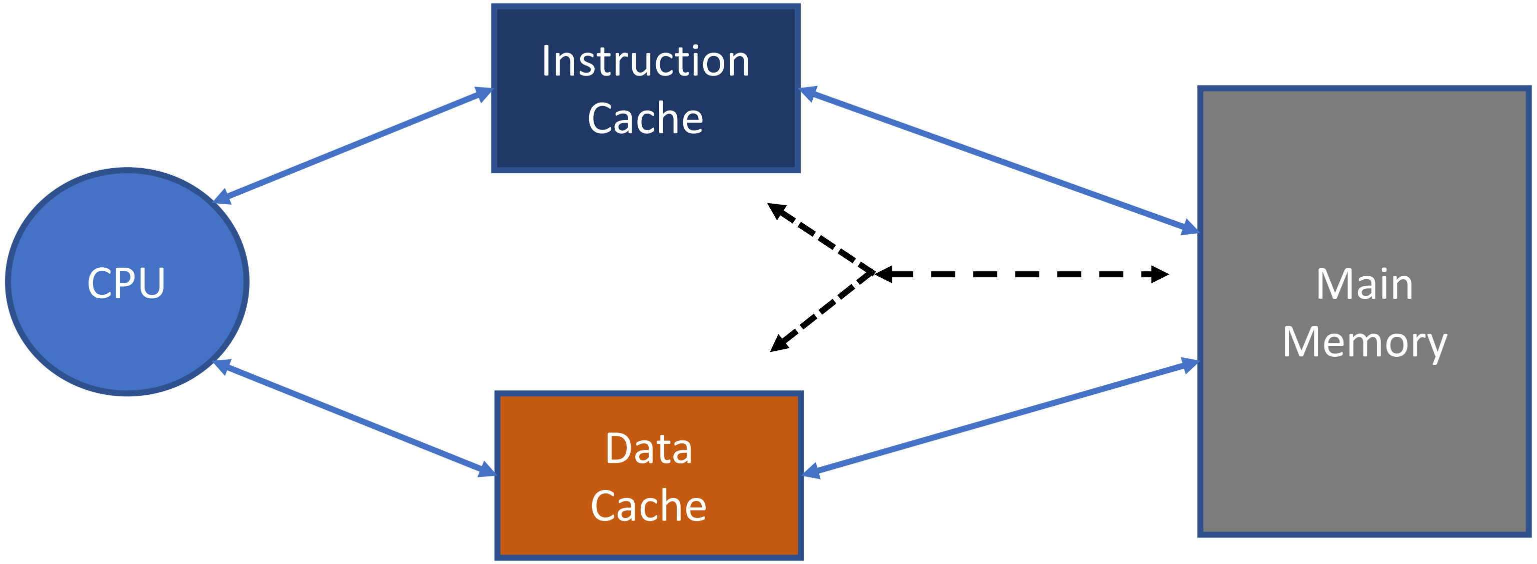
Solution

Average time to access one word:
Tavg= (0.95 x 0.01) + (0.5 x 0.1)
Tavg= 0.095 + 0.0055
Tavg= 0.015 µsec



Split Caches

Separate Cache for Instructions and Data to take advantage of pipelining
In case the program has too many localized accesses to data only or instructions only,
Then there may be no performance improvement

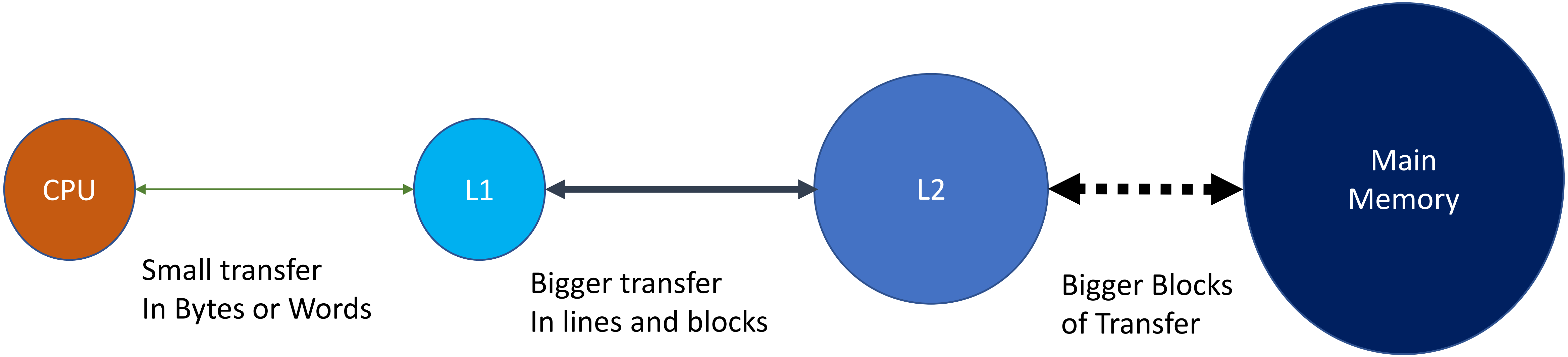


Multilevel Caches

- Primary cache attached to CPU
  - Small, but fast
- Level-2 cache services misses from primary cache
  - Larger, slower, but still faster than main memory
- Main memory services L-2 cache misses
- Some high-end systems include L-3 cache

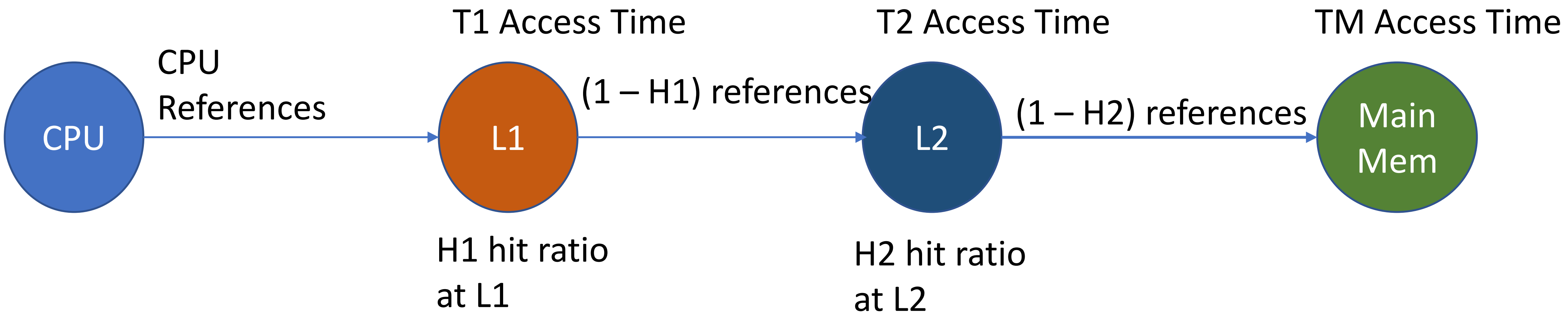
Multi Level Caches

Main purpose is to reduce Miss Penalty



CPU
L1
L2
Main Memory
Small transfer In Bytes or Words
Bigger transfer In lines and blocks
Bigger Blocks of Transfer

Multi Level Caches - Performance



Average Access Time in Multilevel Cache:

Tavg = T1 + (1 – H1) x T2 + (1 – H1) x (1 – H2) x TM

(1 – H1) = **miss rate at L1** = MRL1

Tavg = T1 + MRL1 x T2 + MRL1 x MRL2 x TM

MRL1 x MRL2 = Global Miss Rate at L2

Example 1, find Tavg in 2 level cache

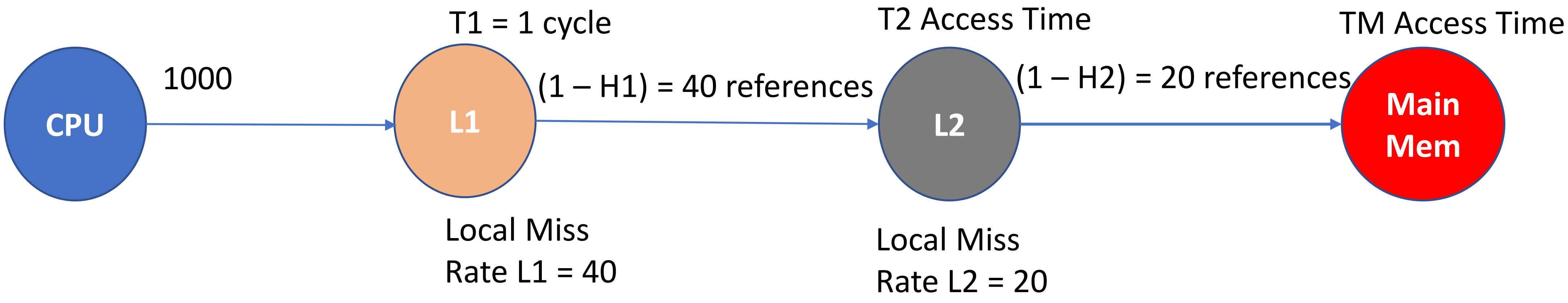
- CPU generates 100 memory references. 80 are hit in L1, 20 are miss in L1. 16 are hit in L2 and 4 are miss in L2. Access time of L1 is 1 cycle. Access time of L2 is 10 cycles. Access time of Main Mem is 100 cycle. Find average memory access time Tavg?

- Tavg = T1 + MRL1 x T2 + MRL1 x MRL2 x TM
- =1 + (20/100) x 10 + [(20/100)x(4/20)] x 100
- =1 + 2 + 4 = 6 cycles Avg Access time

Note:
MRL1 = Miss Rate at Cache L1

Example 2, find Tavg in 2 level cache

- CPU generates 1000 memory references. Number of Miss in L1 is 40. Number of miss in L2 is 20. Access time of L1 is 1 cycle. Access time of L2 is 10 cycle. Find Tavg when Miss Penalty of L2 is 100 cycle.



Tavg = T1 + MRL1 x T2 + MRL1 x MRL2 x TM
=1 + (40/1000) x 10 + [(40/1000) x (20/40)] x 100
=1 + 0.4 + 2 = 3.4 cycles Avg Access time

Measuring Cache Performance

- Components of CPU time
  - Program execution cycles
    - Includes cache hit time
  - Memory stall cycles
    - Mainly from cache misses
- With simplifying assumptions:

$$\text{Memory stall cycles}$$

$$= \frac{\text{Memory accesses}}{\text{Program}} \times \text{Miss rate} \times \text{Miss penalty}$$

$$= \frac{\text{Instructio ns}}{\text{Program}} \times \frac{\text{Misses}}{\text{Instructio n}} \times \text{Miss penalty}$$

Cache Performance Example

- 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
  - Ideal CPU is 5.44/2 =2.72 times faster

Average Memory Access Time

- Hit time is also important for performance
- Average memory access time (AMAT)
  - AMAT = Hit time + Miss rate × Miss penalty
- Example
  - CPU with 1ns clock, hit time = 1 cycle, miss penalty = 20 cycles, I-cache miss rate = 5%
  - AMAT = 1 + 0.05 × 20 = 2ns
    - 2 cycles per instruction

Performance Summary

- When CPU performance increased
  - Miss penalty becomes more significant
- Decreasing base CPI
  - Greater proportion of time spent on memory stalls
- Increasing clock rate
  - Memory stalls account for more CPU cycles
- Can’t neglect cache behavior when evaluating system performance

Multilevel Cache CPI Example

- Given
  - CPU base CPI = 1, clock rate = 4GHz
  - Miss rate/instruction = 2%
  - Main memory access time = 100ns
- With just primary cache
  - Miss penalty = 100ns/0.25ns = 400 cycles
  - Effective CPI = 1 + 0.02 × 400 = 9

Readings

- Chap 5 of P&H Textbook