Where are we?

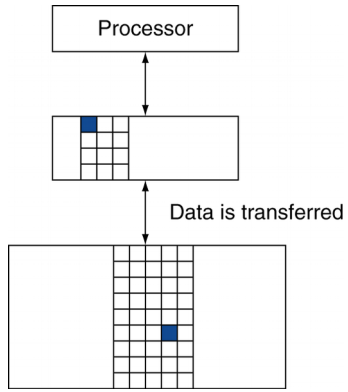
Memory technologies

IDEAL

Fast as SRAM

Large as magnetic disc

Memory Hierarchy



1. Where can a block be placed? **(Q1)**
2. How is a block found? **(Q2)**
3. What block is replaced on a miss? **(Q3)**
4. How are writes handled? **(Q4)**

Direct mapped cache memory

Computer Memory

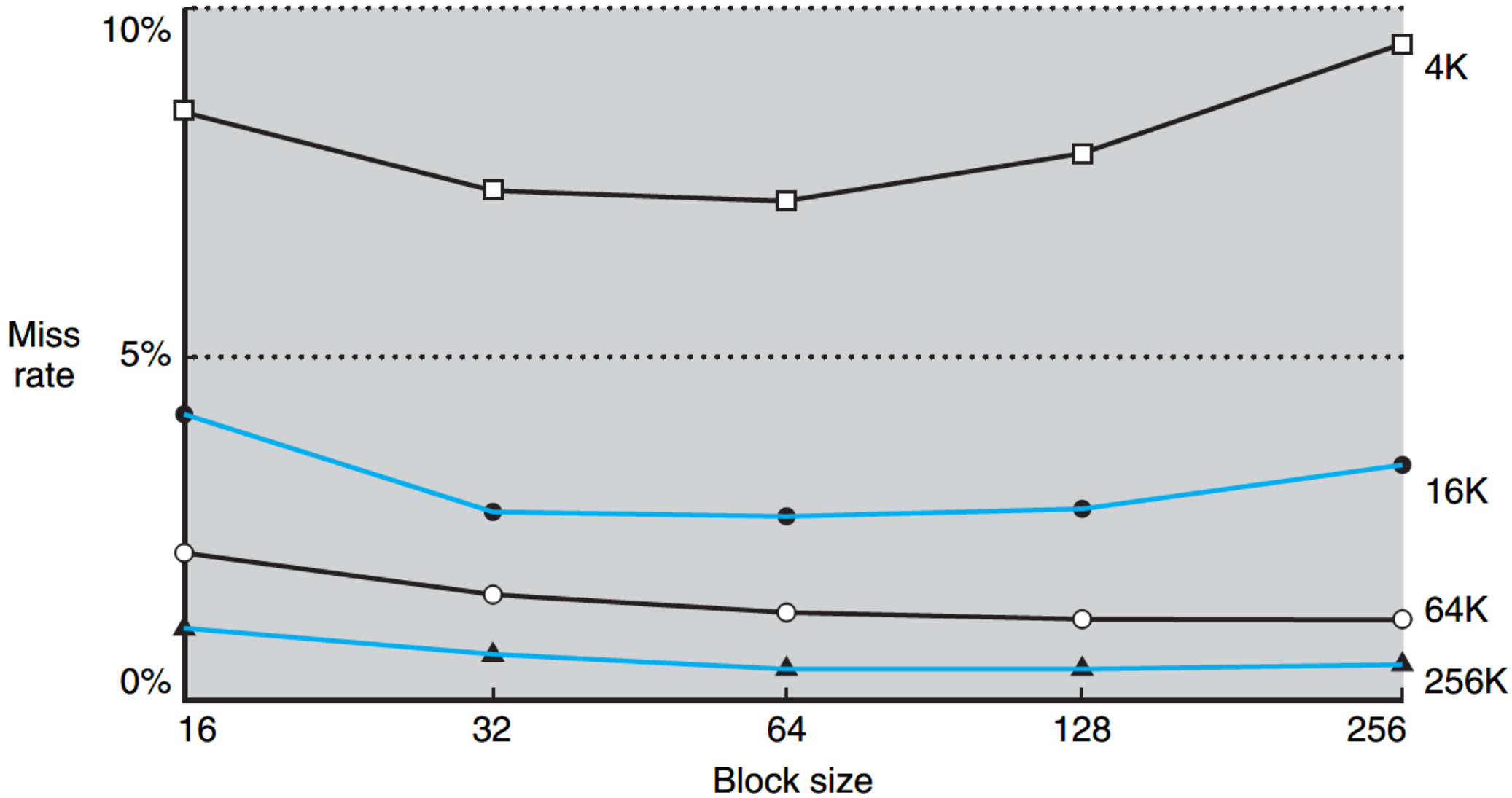
Dr. Ahmed Zahran

WGB 182

a.zahran@cs.ucc.ie

Large Block Size: good or bad?

- Larger blocks should reduce miss rate
  - Due to spatial locality
- But in a fixed-sized cache
  - Larger blocks ⇒ fewer of them
    - More competition ⇒ increased miss rate
- Larger miss penalty
  - Can override benefit of reduced miss rate
  - ***Early restart*** and critical-word-first can help to reduce the penalty


10%
5%
0%
Miss rate
4K
16K
64K
256K
16
32
64
128
256
Block size

What happens on cache misses?

1. Stall the CPU pipeline
   - Send original PC value to memory i.e., current PC -4
2. Fetch block from lower level of hierarchy
   - Instruct memory to perform a read and wait for memory to complete access
   - Cache update
     - Put data from memory into data portion of entry
     - Update the tag and Turn the valid bit on
3. Resume execution
   1. ***Instruction cache miss:*** restart instruction fetch
   2. ***Data cache miss:*** complete data access [read or write]

Cache Write

write hit

(Block in cache)

- ***Write-through*** technique: Update both cache and next lower level memory hierarchy

- ***Write-back*** technique: Update slower memory level when the entire block is replaced

When to update the main memory?
Consistent vs. speed

write miss

(Block is not in cache)

Write-allocate: load the block in the cache and update in the cache

No write allocate: update the written address using write through without loading the block

Write-through technique

Consistent

slow

- Example: if base CPI = 1, 10% of instructions are stores, write to memory takes 100 cycles
  - Effective CPI = 1 + 0.1×100 = 11
- Speeding write-through using a ***write buffer***
  - A buffer holds data waiting to be written to memory
  - CPU continues immediately
    - Only stalls on write if write buffer is already full

Write-Back

- **Fast cache writes** but **a higher miss penalty**
  - Write every block back to main memory
  - Doubles a miss penalty, why?
- Reducing miss penalty
  - A ***dirty bit*** is set whenever a cache block is updated
  - When an updated (dirty) block is replaced
    - Write it back to memory
    - Can use a write buffer to allow replacing block to be read first
- A good option with frequent block writes and/or slower memory

Measuring Cache Performance

Measuring Cache Performance

CPU time = (# CPU execution cycles + CPU stall cycles) * cycle period

$$\text{Memory-stall clock cycles} = (\text{Read-stall cycles} + \text{Write-stall cycles})$$

$$\text{Read-stall cycles} = \frac{\text{Reads}}{\text{Program}} \times \text{Read miss rate} \times \text{Read miss penalty}$$

$$\text{Write-stall cycles} = \left(\frac{\text{Writes}}{\text{Program}} \times \text{Write miss rate} \times \text{Write miss penalty} \right) + \text{Write buffer stalls}$$

Can be ignored with sufficiently deep buffer and/or fast writing procedures

Measuring Cache Performance

$$\text{Memory-stall clock cycles} = \frac{\text{Memory accesses}}{\text{Program}} \times \text{Miss rate} \times \text{Miss penalty}$$

- Example:
  - Instruction-cache miss rate = 2%
  - Data-cache miss rate = 4%
  - Miss penalty = 100 cycles
  - Base CPI (ideal cache) = 2
  - Load & stores are 36% of instructions
- Stall cycles
  - 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

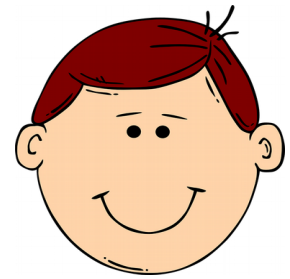
What if the CPU CPI is 1 (faster processor)?

Average 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

Good memory performance

1. *A small hit time*
2. *A low miss ratio*
3. *A small miss penalty*



Performance Concerns

- 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 behaviour when evaluating system performance

Reading

- Sections 5.3 and 5.4