

# CSCI 210: Computer Architecture

## Lecture 34: Caches III

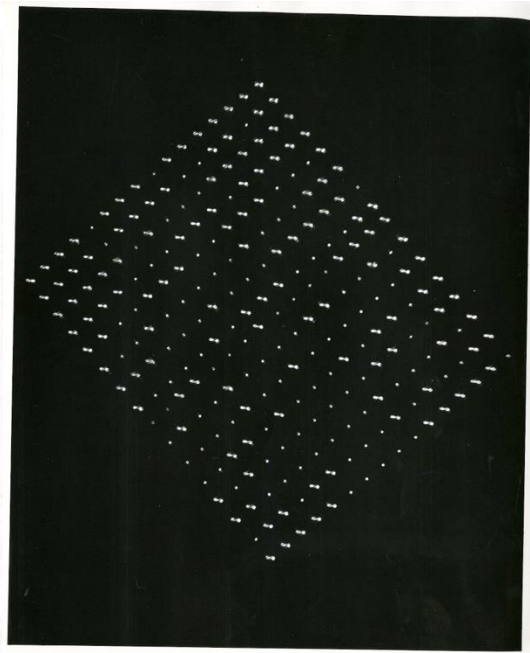
Stephen Checkoway

Slides from Cynthia Taylor

# CS History: The Williams Tube



ArnoldReinhold,  
CC BY-SA 3.0



National Institute of  
Standards and  
Technology, Public  
domain, via Wikimedia  
Commons

- First random-access storage device
- Developed in 1946
- Displays a grid of dots over a cathode ray tube (using an electron beam to strike phosphor)
- Each dot represents a bit
- Each dot creates a small static electricity charge
- Charge at each location is read by a metal sheet in front of the display
- Needs to be periodically refreshed as charge fades over time

# Three types of cache misses

- Compulsory (or cold-start) misses
  - first access to the data.
- Capacity misses
  - we missed only because the cache isn't big enough.
- Conflict misses
  - we missed because the data maps to the same index as other data that forced it out of the cache.

block address of misses

4  
8  
12  
4  
8  
20  
4  
8  
20  
24  
12  
8  
4

tag	data

DM cache

# Cache miss example (from StackOverflow)

32 kB direct-mapped cache

1. You repeatedly iterate over a 128 kB array
  - All misses but the first access to each block are capacity misses because the array does not fit in cache; the first are compulsory misses
2. You iterate over two 8 kB arrays that map to the same cache indices
  - These are conflict misses because if you changed the locations of the arrays to be consecutive, then both would fit in the cache

# Cache Miss Type

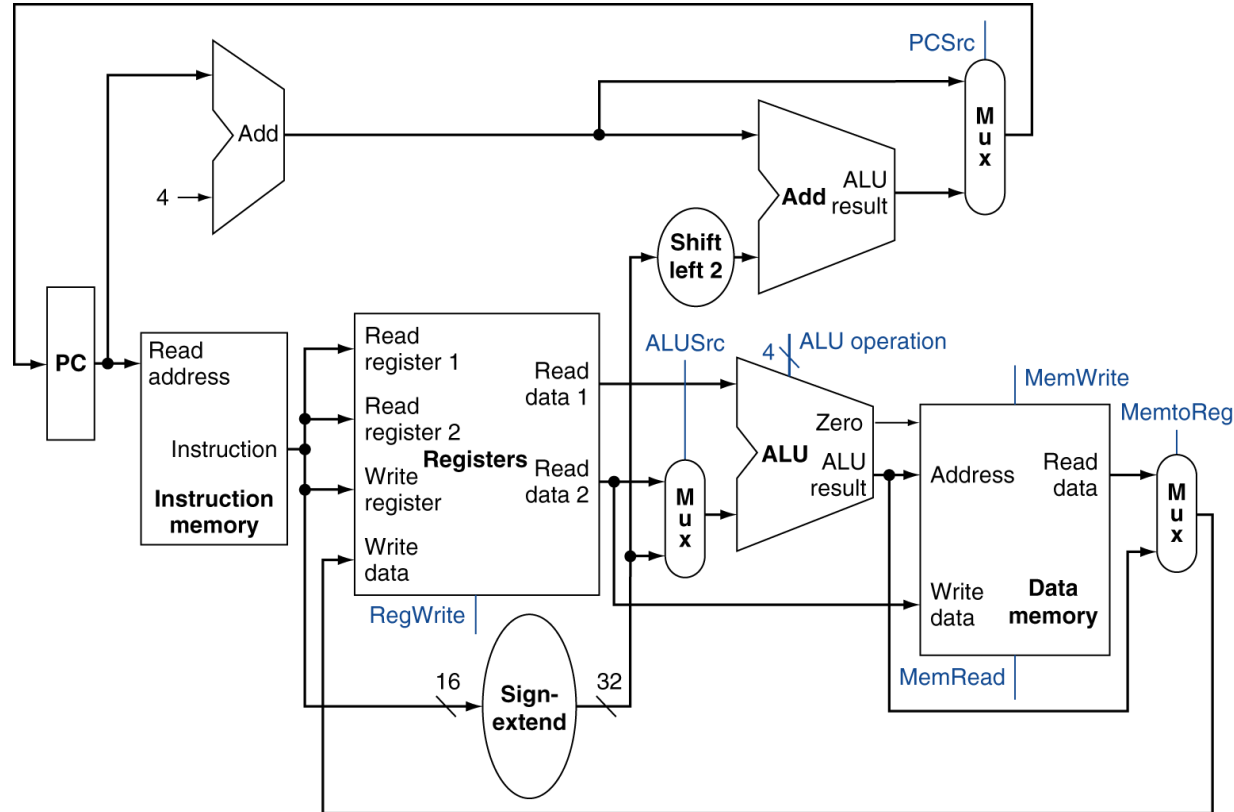
Suppose you experience a cache miss on a block (let's call it block A). You have accessed block A in the past. There have been precisely 1027 different blocks accessed between your last access to block A and your current miss. Your block size is 32-bytes and you have a 64 kB cache (recall a kB = 1024 bytes). What kind of miss was this?

Selection	Cache Miss
A	Compulsory
B	Capacity
C	Conflict
D	Both Capacity and Conflict
E	None of the above

Questions on associativity, replacement?

# **CACHE PERFORMANCE**

# I-cache vs D-cache



- Separate caches for instruction memory and data memory
- I-cache: instruction cache
- D-cache: data cache



# Measuring Cache Performance

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

Memory stall cycles

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

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Misses}}{\text{Instruction}} \times \text{Miss penalty}$$

# Miss Cycles Per Instruction

## 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

	I-cache	D-cache
A	$.02 * 100$	$.04 * 100$
B	.02	.04
C	$.02 * .36 * 100$	$.04 * .36 * 100$
D	$.02 * 100$	$.04 * .36 * 100$

# 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:  $0.02 \times 100 = 2$
  - D-cache:  $0.36 \times 0.04 \times 100 = 1.44$
- Actual CPI =  $2 + 2 + 1.44 = 5.44$

# Average Access Time

- Hit time is also important for performance
- Average memory access time (AMAT)
  - $AMAT = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$
- Example
  - hit time = 1 cycle, miss penalty = 20 cycles, l-cache miss rate = 5%
  - AMAT =

# Cache Speed Factors

- Memory lookup time
- Hit rate
- Size
- Frequency of collisions

# How Much Associativity

- Increased associativity decreases miss rate
  - But with diminishing returns
- Simulation of a system with 64 kB D-cache, 64-byte blocks  
Miss rate:
  - 1-way: 10.3%
  - 2-way: 8.6%
  - 4-way: 8.3%
  - 8-way: 8.1%

```
for(int i = 0; i<10,000,000;i++)  
    sum+=A[i];
```

Assume each element of A is 4 bytes and sum is kept in a register. Assume a direct-mapped 32 kB cache with 32 byte blocks. Which changes would help the hit rate of the above code?

<b>Selection</b>	<b>Change</b>
A	Increase to 2-way set associativity
B	Increase block size to 64 bytes
C	Increase cache size to 64 kB
D	A and C combined
E	A, B, and C combined

# Performance Summary

- When CPU performance increases
  - 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



**MAKING CACHES FASTER**

# Multilevel Caches

- Primary (or level-1) cache attached to CPU
  - Small, but fast
- Level-2 cache services misses from primary cache
  - Larger, slower, but still faster than main memory
- L-3 cache usually services multiple CPUs
- L-3 misses go to main memory

# Multilevel Cache performance

- For primary (L-1) cache:
  - Access time in cycles, typically 1
  - Miss rate (fraction of L-1 cache accesses which miss)
  - On a miss, the next level of the cache hierarchy is consulted
- For L-n cache for  $n > 1$ :
  - Access time in cycles
  - Miss rate (fraction of L-n cache accesses which miss)
  - On a miss, the next level of the cache hierarchy is consulted
- Memory
  - Access time in cycles

# Cache Example: L-1 only

- Given
  - CPU base CPI = 1
  - L-1 access time = 1 cycle
  - Miss rate = 10%
  - Main memory access time = 400 cycles
- With just a primary (L-1) cache
  - Effective CPI =  $1 + 0.10 * 400 = 40$

# Cache example: L-1 and L-2

- L-1:
  - Access time = 1 cycle (so included in the base CPI)
  - Miss rate = 10%
- L-2
  - Access time = 20 cycles
  - Miss rate = 4%
- Memory access time of 400 cycles
- $\text{CPI} = 1 + 0.10 * (20 + 0.04 * 400) = 4.6$   
[Compare to a CPI of 40 for L-1 only]

# Cache Example: L-1, L-2, L-3

- L-1: access time = 1 cycle; miss rate = 10%
- L-2: access time = 20 cycles; miss rate = 4%
- L-3: access time = 50 cycles; miss rate = 1%
- Memory access time = 400 cycles

With your group, work out what the CPI is assuming a base CPI of 1.

# Multilevel Cache Considerations

- Primary cache
  - Focus on minimal hit time
- L-2 cache
  - Focus on low miss rate to avoid main memory access
  - Hit time has less overall impact
- Results
  - L-1 cache usually smaller than a single cache
  - L-1 less associative than L-2

# Interactions with Advanced CPUs

- Out-of-order CPUs can execute instructions during cache miss
  - Pending store stays in load/store unit
  - Dependent instructions wait in reservation stations
    - Independent instructions continue



# Prefetching

- Hardware Prefetching
  - suppose you are accessing a single field in each object in an array of large objects
  - hardware determines the “stride” and starts grabbing values early
- Software Prefetching
  - Compiler adds extra instructions to load data before it is needed

Which data structure will have better memory access times assuming you have a prefetcher?

A. ArrayList

B. Linked List

C. There will not be any difference

# Writing Cache-Aware Code

- Focus on your working set
- If your “working set” fits in L1 it will be vastly better than a “working set” that fits only on disk.
- If you have a large data set – do processing on it in chunks.
- Think about regularity in data structures (can a prefetcher guess where you are going – or are you pointer chasing)

# Reading

- Next lecture: More Caches!