

# CSCE 4610/5610: Computer Architecture

## Review

- Register Renaming
- Memory Hierarchy

- Direct Mapped Cache
- Set Associative Cache
- Fully Associative Cache

- Block (line) size – large or small
- Memory Access time
  - Miss rate
  - Miss penalty
- Bandwidth vs Latency

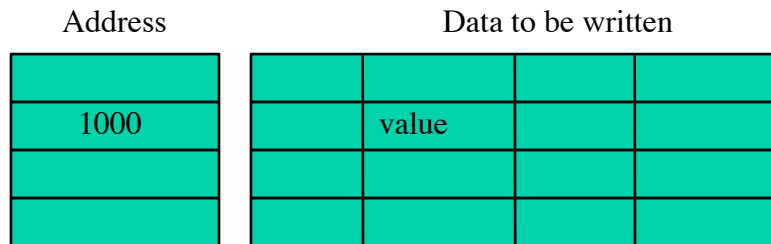
## Reading vs Writing

- Write through vs Write Back
- Cache allocate vs No allocate

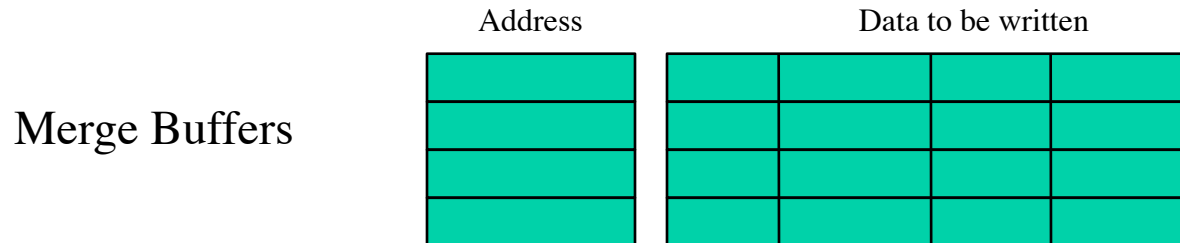
# CSCE 4610/5610: Computer Architecture

Write through using write buffers

Cache is updated immediately and the new value is placed in a buffer, awaiting write to Memory (or L2 cache)



Should we write 4 bytes at a time if the next write is not to the same cache block?



# CSCE 4610/5610: Computer Architecture

Cache performance evaluation:

Cache miss rate

Miss Penalty

Memory Access time = (time on hit \* hit rate) + (miss penalty \* miss rate)

Miss penalty includes the time to transfer the entire data from memory

Also includes the time to translate address in main memory

Main memory uses physical addresses

Caches may use either physical or virtual addresses

May need more than one memory cycle to fetch a cache line

For example Cache line may be 32Bytes but the memory bandwidth is only 16 bytes

So larger cache lines may lead to longer penalty

Any reason for larger cache lines?

# CSCE 4610/5610: Computer Architecture

Latency vs Bandwidth?

Bandwidth -- how many bytes can we transfer per cycle or second

Latency -- how long before even the first bit (or byte) is available

Miss Penalty also depends on the write policy

Write Back: If the line to written back is dirty, we need to write the dirty line back to main memory -- two memory accesses on dirty

Write-through: read and write miss penalties are equal assuming write buffers are adequate

For example if on average 25% cache lines are dirty on every miss you have 1.25 memory accesses if we use write-back

# CSCE 4610/5610: Computer Architecture

Miss rate depends on the cache size, line size and set associativity

Split cache vs Unified Cache

See the example

36% are load/store instructions  
miss penalty is 100 cycles

Should we use unified 32KB cache or separate  
16KByte data and 16Kbyte instruction cache?

Unified: Miss rate = 43.3 misses per 1000 instructions  
Each instruction has 1.36 memory accesses  
Miss rate =  $[43.3/1000]/1.36 = 0.0318$  or 3.18%

# CSCE 4610/5610: Computer Architecture

## Split cache vs Unified Cache

Instruction cache: 3.82 misses per 1000 instructions

$$\text{Miss rate} = 3.82/1000 = 0.00382$$

Data cache: 40.9 misses per 1000 instructions

$$\text{Miss rate} = [40.9/1000]/0.36 = 0.114$$

How do we combine these miss rates into a single number?

We need on average 1.36 memory access per instruction

$$1/1.36 = 0.74 \text{ to access an instruction}$$

$$0.36/1.36 = 0.26 \text{ to access data}$$

$$\text{Total miss rate} = 0.00382*0.74 + 0.114*0.26 = 0.0324 \text{ or } 3.24\%$$

Unified Cache is slightly better (3.18% vs 3.24%)

# CSCE 4610/5610: Computer Architecture

But let us consider actual CPI related performance

Access time:  $(\text{hit\_time} * \text{hit\_rate}) + (\text{mss\_penalty} * \text{miss\_rate})$

Note how the textbook computes effective memory access time

$$= \text{Hit\_time} + (\text{miss\_rate}) * (\text{miss\_penalty})$$

Instead of

$$= (\text{Hit\_time}) * (1 - \text{Miss\_rate}) + (\text{Miss\_rate}) * (\text{Miss\_Penalty})$$

Split cache = 4.24

Unified -- need to include an extra cycle for load/store since we have a structural hazard when using a single cache

$$= 4.44$$

So, miss rate alone is not a good indicator of cache performance

# CSCE 4610/5610: Computer Architecture

Let us look at another example of evaluating the impact of cache.

We are comparing direct mapped and 2-way set associate cache  
2-way as lower miss rate (1% compared to 1.4% for direct mapped)  
But slower clock (1.25 compared to 1).

Other parameters given. The CPI = 2, 1ns per cycle  
Miss penalty = 75ns  
1.5 memory accesses per instruction (unified cache?)

$$\text{CPU}_{1\text{-way}} = \text{IC} * (2 * 1\text{ns} + (1.5 * 0.014 * 75\text{ns})) = 3.58 * \text{IC ns}$$

$$\text{CPU}_{2\text{-way}} = \text{IC} [2 * 1.25\text{ns} + (1.5 * 0.01 * 75\text{ns})] = 3.63 * \text{IC ns}$$

Note again how we are calculating effective memory access time

# CSCE 4610/5610: Computer Architecture

Let us look at another example from page 295

2-way vs 4-way

Miss rate for 2-way is 0.049 and for 4-way it is 0.044

4-way is 1.1 times slower (in terms of clock) as compared to 2-way and miss penalty is 10 cycles

$$\begin{aligned}\text{Access time for 2-way} &= \text{Hit-time} + \text{Miss-rate} * \text{Miss penalty} \\ &= 1 + 0.049 * 10 = 1.49\end{aligned}$$

$$\text{Access time for 4-way} = 1.1 + 0.044 * 10 = 1.54$$

Note how the book computes. It is stated that the miss penalty is 10 cycles based on the faster clock, because the access on a miss remains constant in terms of actual time to get the data. If slower clock, fewer cycles are needed

For 4-way, miss penalty will be 9 cycles

$$\text{Access time for 4-way} = 1.1 + 0.044 * 9 = 1.5$$

# CSCE 4610/5610: Computer Architecture

Note in general we should reduce miss rate and miss penalty

CPI in presence cache misses:

Instruction cache misses

Data cache misses for load and store

If you have a unified cache, we need to add an extra cycle for Load/store due to structural hazard

That is in addition to stalls on data cache misses

Types of Cache misses

Cold start or Compulsory

Conflict

Capacity

Conflicts may be reduced with  
higher associativity

Particularly if we want multithreading  
or out of order execution.

Page 290.

# CSCE 4610/5610: Computer Architecture

In general we need to look at techniques to reduce miss rates and reduce miss penalty

For example, we can have multiple level cache to reduce effective penalty  
Or have a very large level 1 cache to reduce miss rates.

**For project: You can experiment with cache simulators  
use the simulators with SimpleScalar  
or CACTI**

**Run large benchmarks with different cache organizations**

11 techniques to improve cache performance in Chapter 5



# CSCE 4610/5610: Computer Architecture

Let us briefly talk about localities

- Instructions have good spatial locality

  - Instructions of a loop body exhibit temporal locality

- Arrays have good spatial localities

What about Scalar data items

  - say loop indexes, constants etc?

Consider data structures and linked lists or even objects in OO

  - Do they exhibit either spatial or temporal localities?

New ideas to aid memory performance in such situations

  - Data relocation (memory forwarding)

  - Data pre-fetching (jump pointers)

  - Data flattening