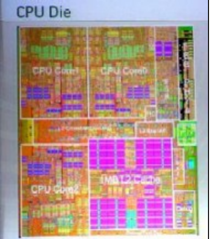
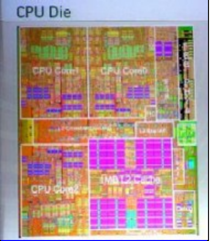


COMP 273
Introduction to Computer Systems

COMP 273

Cache Memory

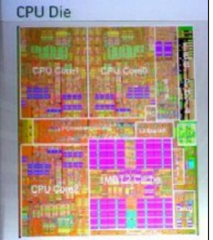
Prof. Joseph Vybihal



COMP 273
Introduction to Computer Systems

Announcements

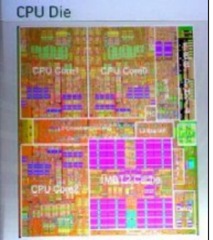
- Course evaluation



At Home

- Using the supplied code with the simulator, manipulate the cache.

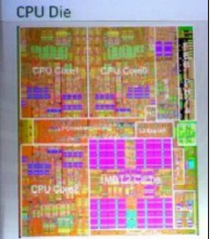
- Textbook:
  - See MIPS Run; By Sweetman; Morgan Kaufmann Publishers, ISBN 1-55860-410-3 Chapter 4



COMP 273
Introduction to Computer Systems

Part 1

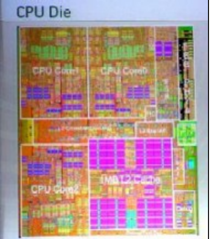
Cache Basics



Why use a cache?

- To speed up the operation of the computer

- Primary problem: it is smaller than the program, wouldn't you spend more time in the RAM?



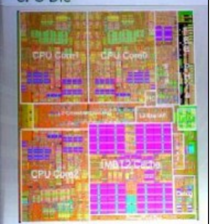
Principle of Locality

- Temporal Locality
  - Item will be referenced again soon
  - Example: Library and functions
- Spatial Locality
  - Adjacent items will probably be executed next
  - Example: Loops and functions

The Storage Conundrum

- We need fast and voluminous storage to run our applications adequately
- Two issues at hand:
  - Speed of access
  - Amount of storage
- Speed and storage physically work against each other, when compared to cost in dollars

CPU Die

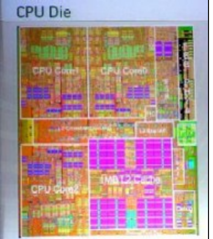


COMP 273
Introduction to Computer Systems

Storage Comparison

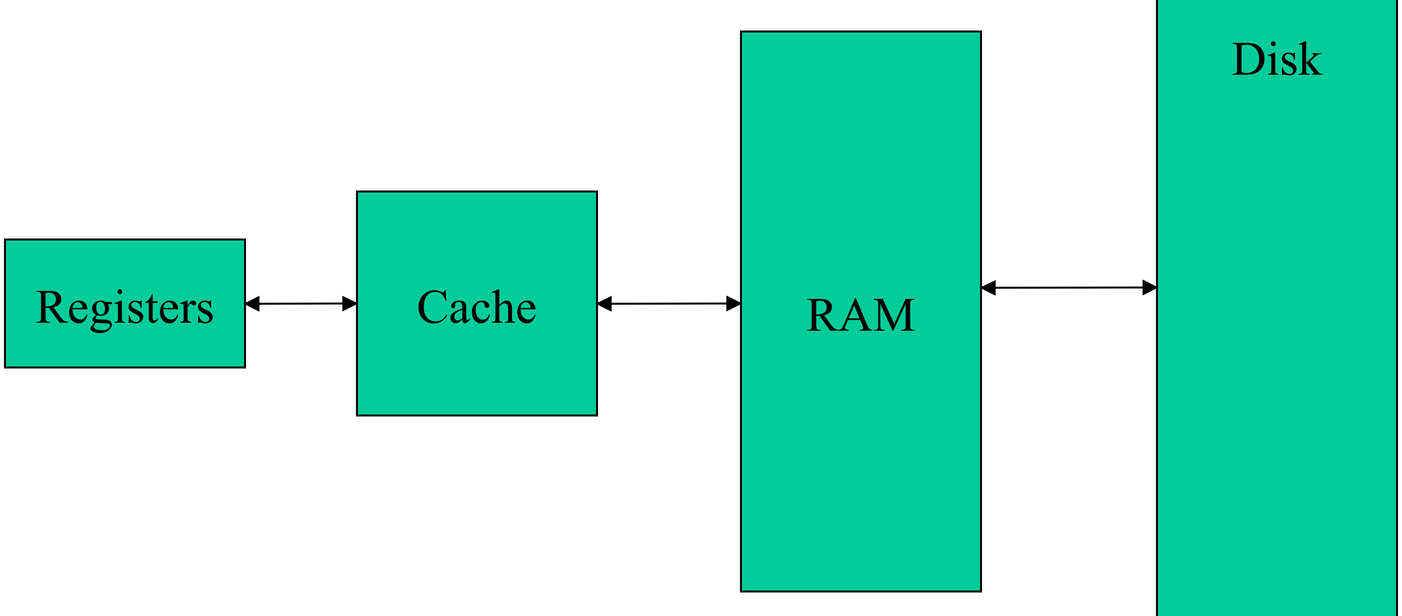
Storage	Technology	Speed	Cost
CPU Registers	Flip-flops	1 – 5 ns	\$250 - \$300
Cache	SRAM – transistors + power	5 – 25 ns	\$100 - \$250
RAM	DRAM – capacitors + refresh	30 – 120 ns	\$5 - \$10
Disk	Magnetic charge – mechanical	10 Million ns – 20 million ns	\$0.1 - \$0.2

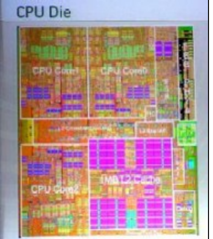
~ Per 100 Meg



Memory Hierarchy

- To manage efficiently, memory is organized into layers.
- Where each layer of memory is optimally geared to execute a program



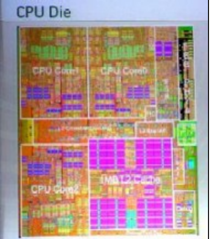


COMP 273
Introduction to Computer Systems

Cache Levels

TABLE 4.1 Cache evolution in MIPS CPUs

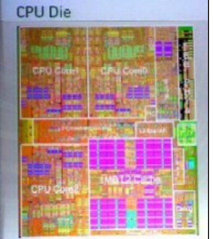
CPU (MHz)	*Primary*				*Secondary*			*Tertiary*		
	Size I-cache	*Size D-cache*	*direct/ n-way*	*on-chip?*	*Size*	*direct/ n-way*	*on-chip?*	*Size*	*direct/ n-way*	*on-chip?*
R3000-33	32K	32K	Direct	Off						
R3052-33	8K	2K	Direct	On						
R4000-100	8K	8K	Direct	On	1M	Direct	Off			
R4600-100	16K	16K	Two-way	On						
R10000-250	32K	32K	Two-way	On	4M	Two-way	Off			
R5000-200	32K	32K	Two-way	On	1M	Direct	Off			
RM7000-xxx	16K	16K	Four-way	On	256K	Four-way	On	8M	Direct	Off



How to optimally use cache?

Problems …

- Issues:
  - What to load?
    - Can we make an educated guess?
  - How to address?
    - Should we match with RAM's address?
  - <u>Hit to Miss ratio</u> (cache miss rate)
    - Hit implies we find the instruction in the cache
    - Miss implies we need to go to the RAM
  - Cache miss/refill penalty



Operations in a Miss

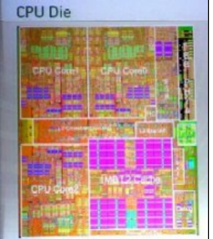
1. Access RAM (PC – 4, or jump address)
2. Wait for RAM to complete read
3. Put the data into cache & update table
4. Restart the instruction (from cache)

Miss penalty = Cycles to upload data to cache

Cost of miss = Miss frequency * Miss penalty

Program speed = n + m*penalty, where n & m are no. of instructions

(compare with no cache?)

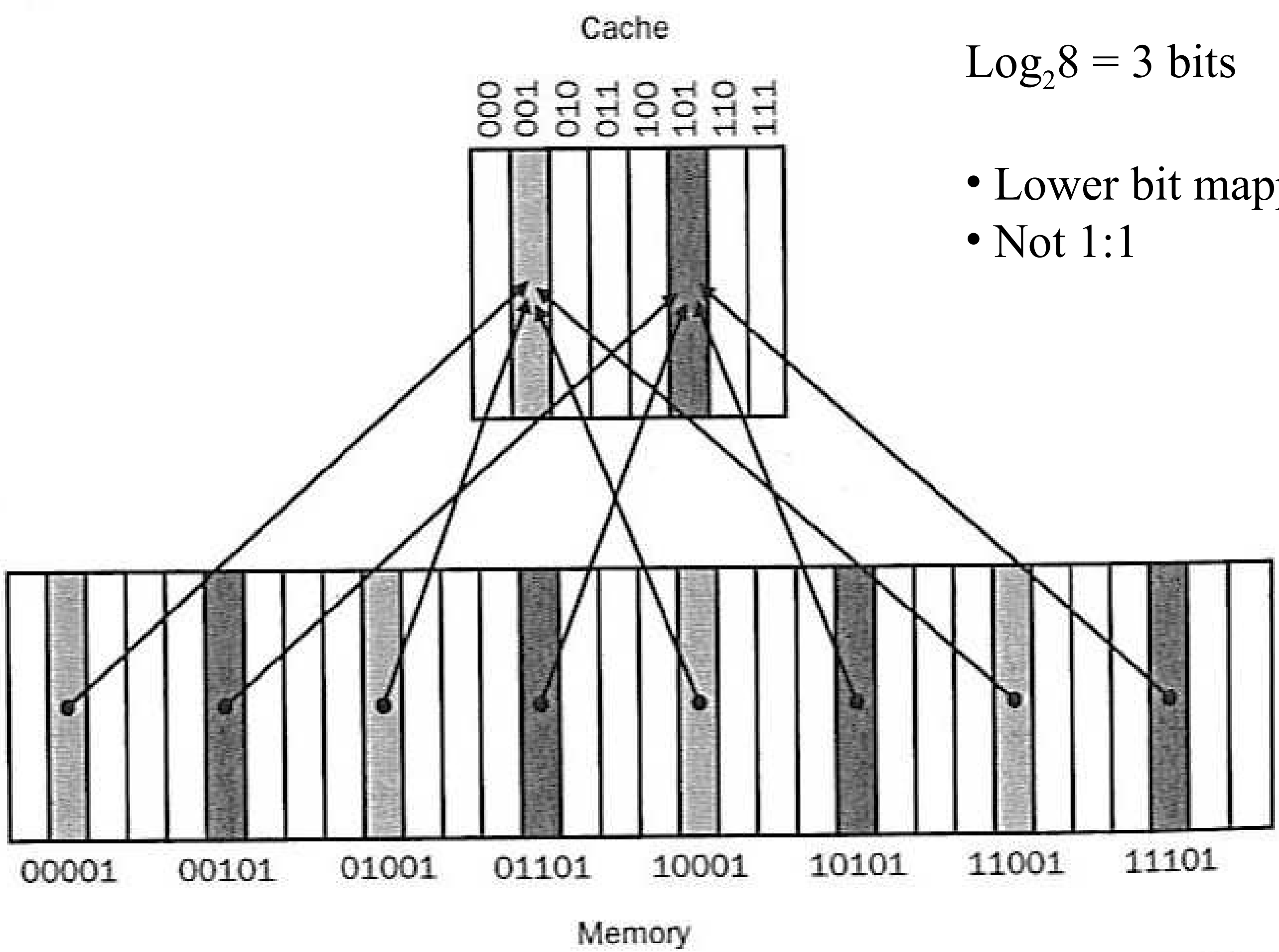


Part 2

Cache Mechanics

Addressing

Block address MODULO Number of cache blocks in the cache



$Log_2 8 = 3$ bits

- Lower bit mapping
- Not 1:1

Basic Structure & Operation

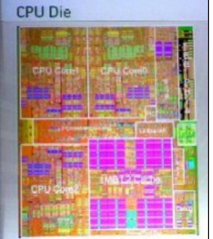
E.g.

Decimal address of reference	Binary address of reference	Hit or miss in cache	Assigned cache block (where found or placed)
22	10110_{two}	miss (7.6b)	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	miss (7.6c)	$(11010_{two} \bmod 8) = 010_{two}$
22	10110_{two}	hit	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	hit	$(11010_{two} \bmod 8) = 010_{two}$
16	10000_{two}	miss (7.6d)	$(10000_{two} \bmod 8) = 000_{two}$
3	00011_{two}	miss (7.6e)	$(00011_{two} \bmod 8) = 011_{two}$
16	10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$
18	10010_{two}	miss (7.6f)	$(10010_{two} \bmod 8) = 010_{two}$

MAP IN USE unique DATA

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory(10110_{two})
111	N		

Cache Memory



Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

a. The initial state of the cache after power-on

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory(10110_{two})
111	N		

b. After handling a miss of address (10110_{two})

Index	V	Tag	Data
000	N		
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

c. After handling a miss of address (11010_{two})

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

d. After handling a miss of address (10000_{two})

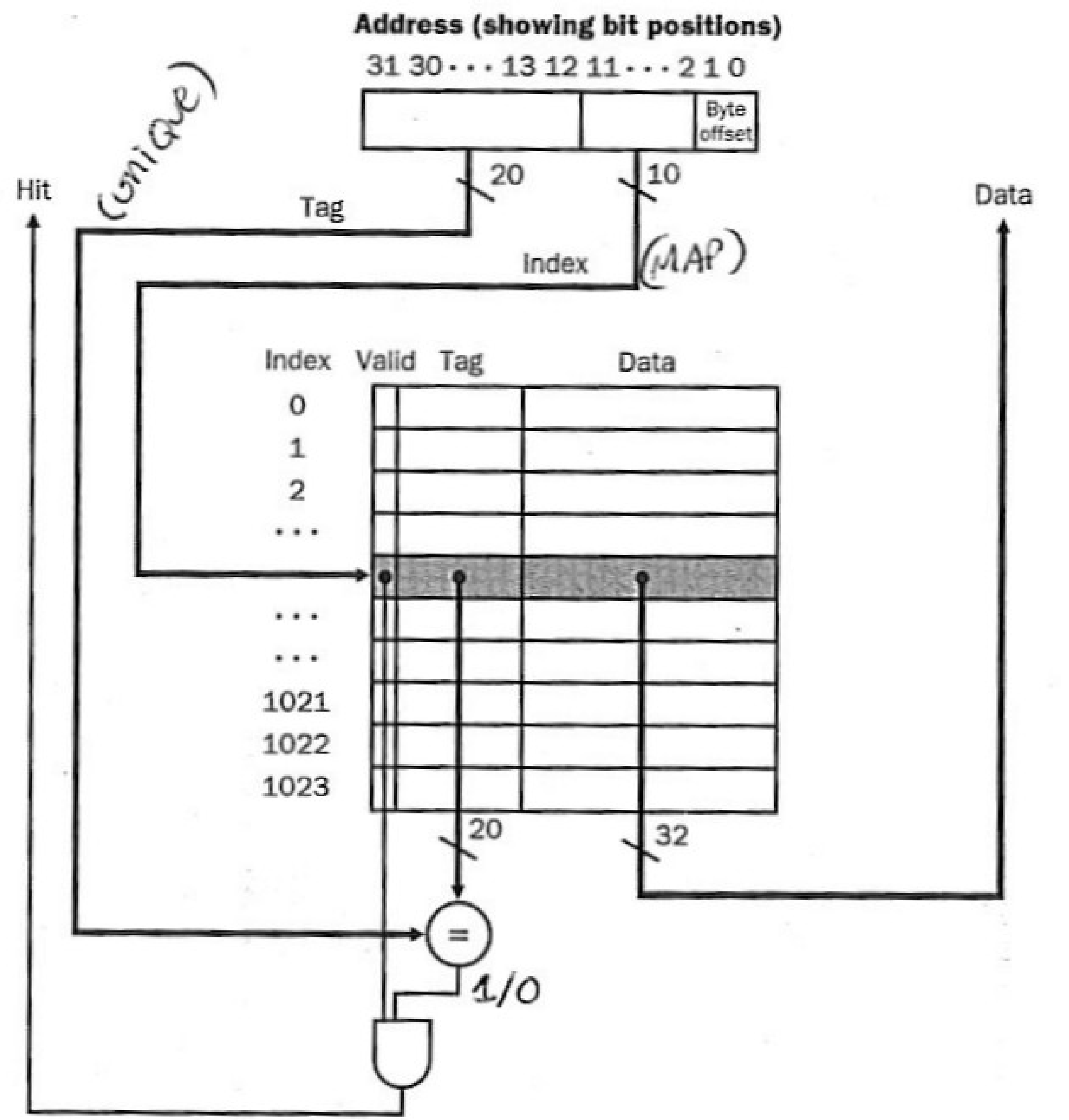
Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

e. After handling a miss of address (00011_{two})

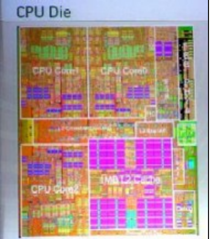
Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	10_{two}	Memory (10010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

f. After handling a miss of address (10010_{two})

Basic Access Architecture

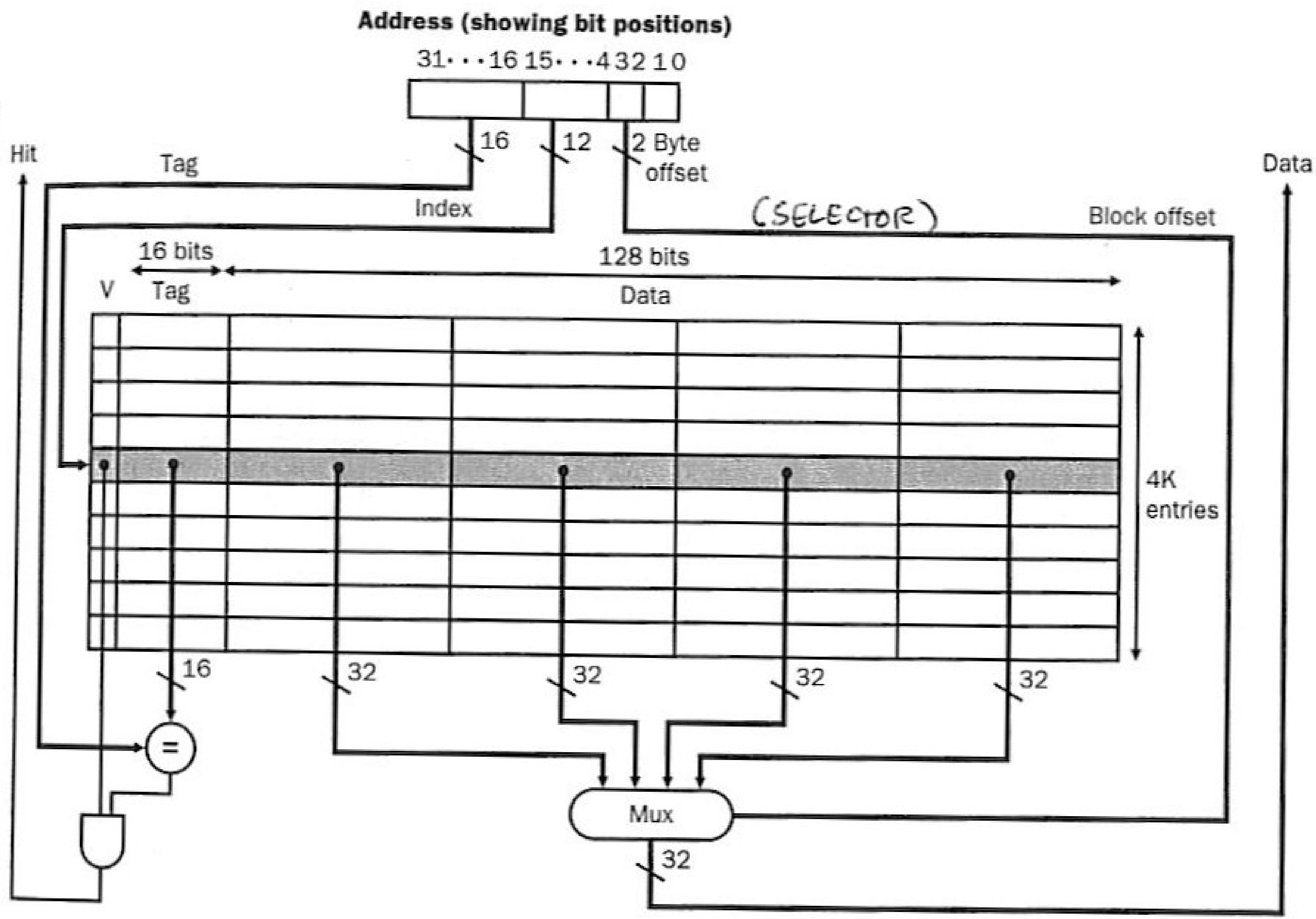


COMP 273
Introduction to Computer Systems



Locality Access

(Read/Miss = 4 word read, a block read)



COMP 273
Introduction to Computer Systems

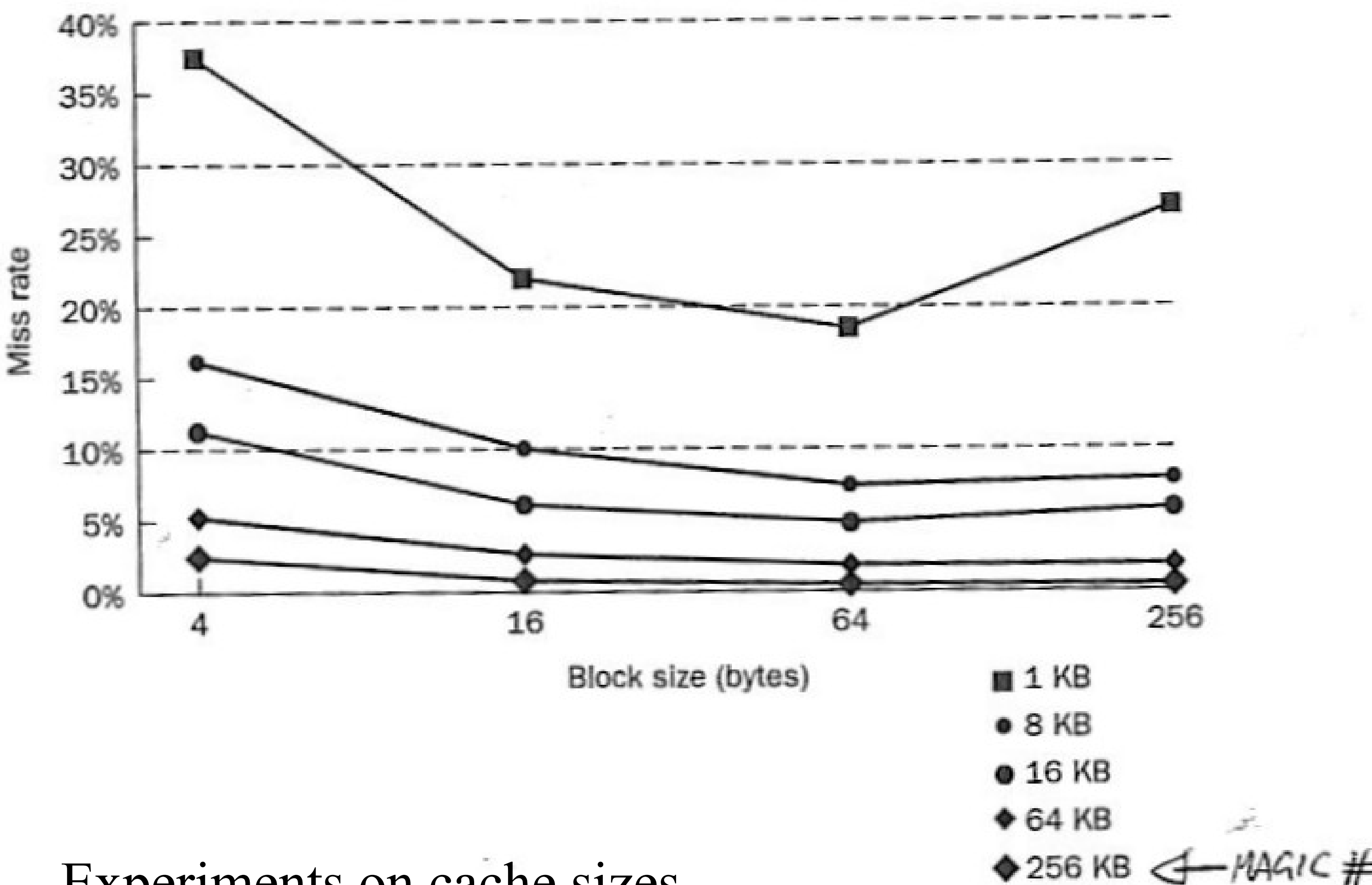
Performance

Program	Block size in words	Instruction miss rate	Data miss rate	Effective combined miss rate
gcc	1	6.1%	2.1%	5.4%
	4	2.0%	1.7%	1.9%
spice	1	1.2%	1.3%	1.2%
	4	0.3%	0.6%	0.4%

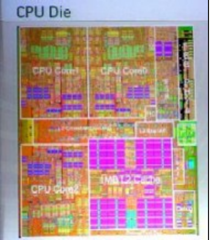
Just temporal

Spatial locality advantage

COMP 273
Introduction to Computer Systems



Experiments on cache sizes



COMP 273
Introduction to Computer Systems

Part 3

Technology to support caches

Bus Designs

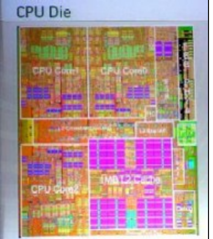
- One-Word-Wide memory organization
  - RAM → Bus → Cache → CPU
- Wide memory organization
  - RAM → multi-word bus → Cache → Multiplexor → CPU
- Interleaved memory organization
  - RAM banks → Bus → Cache → CPU

Assume:

- 1 cs to send an address to RAM
- 15 cs for each DRAM access find
- 1 cs to send a word of data

What improvement for 4 word lw?

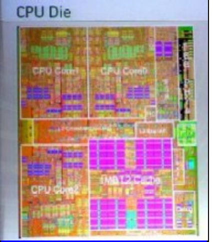
(one instruction but 4 word load)



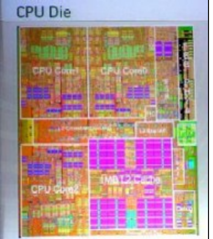
COMP 273
Introduction to Computer Systems

Part 4

Performance Calculations



Amdahl’s Law



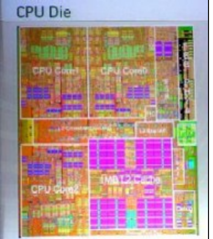
Amdahl's Law

- The speedup in performance is proportional to the new component and the actual fraction of work it carries out.

```
s = 1 / [ (1 - f) + f/k ]
```

Where:

- S is the speedup
- F is the fraction of work performed by the component
- K is the advertised speedup of the new component

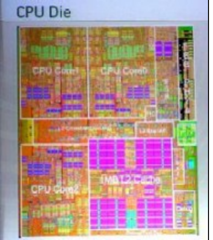


Example

- Assume your daytime processes spend 70% of their time running in the CPU and 30% waiting for service from a disk.
- You find a computer that functions 50% faster and costs $ 10,000.
- You find a new disk drive for $7,000 with a speed increase of 2.5
- What should you do?

Processor: f = .7, k=1.5
S = 1 / [(1 – 0.7) + 0.7/1.5] = 1.3 → 30% for $10,000 = 10000 / 30 = $333 per

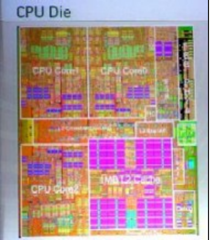
Disk: f = .3, k = 2.5
S = 1 / [(1 – 0.3) + 0.3/2.5] = 1.22 → 22% for $7,000 = 7000 / 22 = 318 per



Transfer Rate Calculations

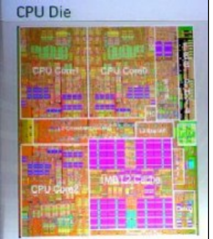
Polling Overhead

- Assume polling overhead takes 400cs on a CPU that runs at 500Mhz
- How much CPU time is used:
  - Poll a mouse 30 times per second
  - Floppy disk data transfer rate at 16-bits per tick and needs to move data at a rate of 50KB/sec.
  - Hard disk transfers at 4-word (32 bits per) chunks at 4MBytes/sec.



Answer

- Mouse
  - Polling = 30 * 400 = 12,000cs
  - Processor = 12000 / 500,000,000cs = 0.002%
- Floppy Disk
  - Polling = 50 / 2 = 25K times
    - 25K * 400 = 10,000,000cs
  - Processor = 10 / 500 = 2%
- Hard Disk
  - Polling = 4MB / 16Bytes = 250K times
    - 250K * 400 = 100,000,000
  - Processor = 100 / 500 = 20%



COMP 273
Introduction to Computer Systems

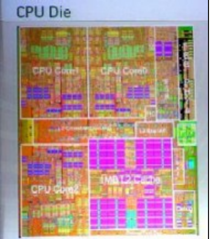
Interrupt Driven

- Assume 500MHz computer 4MB/s HDD 4-words
- Assume 500cs to handle interrupt
- Hard disk used 5% of the time
- Processor usage?

Data transfer = 4MB / 16B = 250K times
(interrupt at each of these moments, like in polling)
Polling = 250K * 400 = 100,000,000cs
Interrupt = 250K * 500 = 125,000,000cs !!

Polling / Processor = 100 / 500 = 20% (always)
Interrupt / Processor = 125 / 500 = 25% (at transfer)
Interrupt used only 5% of the time = 25% * 5% = 1.25% (actual)

Interrupt advantage over polling occurs during the time when no data transfer occurs (absence of overhead).



COMP 273
Introduction to Computer Systems

DMA

- Assume 500MHz, 4MB/sec DMA 1-byte
- DMA overhead (initialize) 1000cs
- Interrupt overhead (process at end) 500cs
- DMA transfer rate of 8 KB per cycle
- Processor usage?

DMA transfer = 8KB / 4MB/sec = 0.002 seconds (w/o CPU)

100% of the time = (1000 + 500) = 1500 cs

Processor = 1500 / 500000000 = 0.000003 => 0.0003%

Only for overhead … no transfer work