



# COMP 273

## Virtual Memory & Cache Hierarchies

### Part 2

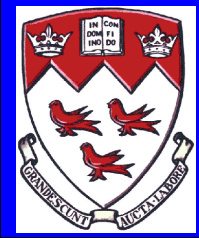
Prof. Joseph Vybihal



# Announcements

COMP 273

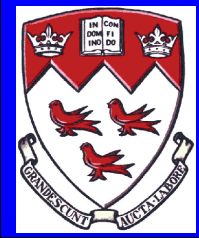
Introduction to Computer Systems

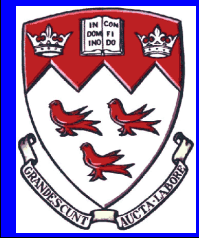




# Part 1

## Programming with Virtual Addresses





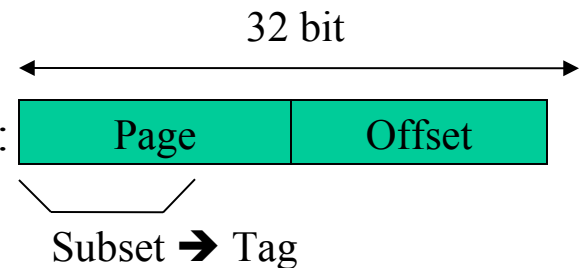
# Virtual Addressing

```
.text  
.globl Start
```

Compiled as  
offsets

```
Start: :  
:  
lw $1, Value  
:  
:  
  
.data  
Value: .asciiz "Value = "
```

Loaded into RAM as Virtual Addresses:





# Architecture

TLB CACHE

Valid	Dirty	TAG	Page No.
1	0	tag	A
0	1	tag2	B

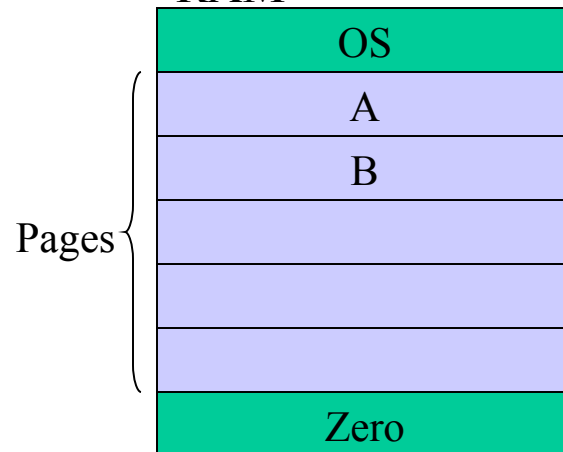
Cache

Valid	TAG	Data
1	tag	A

Page Table RAM

Valid	Address
1	A
1	B
0	C

RAM





# Computing Real Addresses

## Version One

Tag TLB	Cache Index
---------	-------------

TLB hit & in cache

## Version Two

Page <u>TLB</u> "A"	V.A. Offset
---------------------	----------------

TLB hit & not in cache

## Version Three

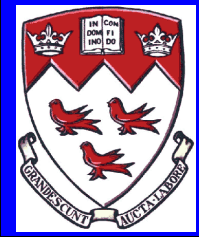
Page <u>Table</u> "A"	V.A. Offset
-----------------------	----------------

TLB miss & in RAM

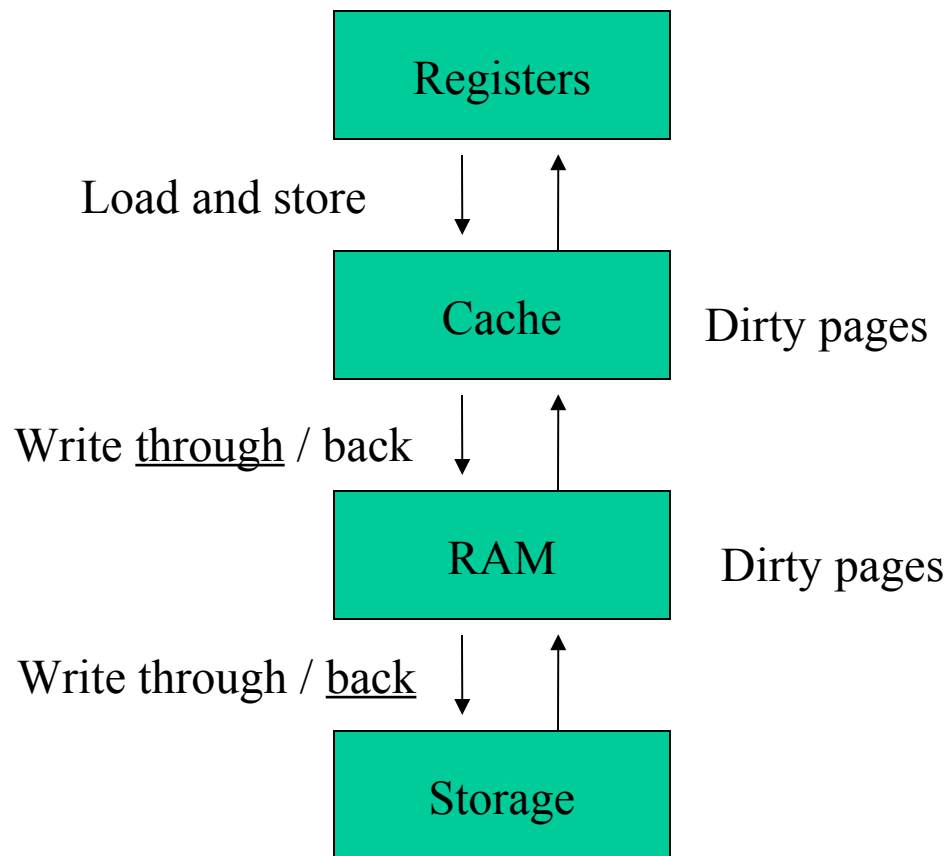


# OS Interaction

- CPU Controllable
  - RAM Page Table register present
  - TLB present
- OS Controllable
  - Page faults
  - RAM Page Table
  - Dirty bit management
    - Write back (copy page to storage)
    - Write through (access storage directly)



# Memory Hierarchy







# Part 2

## Calculating Cache Performance

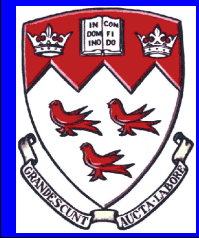
# Properties of Cache Performance

- CPU Time =  
$$(\text{Prog. execution cs} + \text{MemoryStall cs}) * \text{csTime}$$
- MemoryStall cs = ReadStall cs + WriteStall cs
- ReadStall cs =  
$$\text{ProgramReads} * \text{ReadMissRate} * \text{ReadMissPenalty}$$
- WriteStall cs =  
$$(\text{ProgramWrites} * \text{WriteMissRate} * \text{WriteMissPenalty}) + \text{WriteBufferStalls}$$



# Basic Cache Performance Calculations

- Assume:
  - Instruction cache miss rate is 2%
  - Data cache miss rate is 4%
  - Cycles per Instruction (CPI) = 2
  - Miss penalty of 40 cs
  - Loads and stores is 36% of instructions
- Speed improvement with a perfect cache with zero misses?
  - Instruction miss cycles =  $N * 2\% * 40 = 0.8N$
  - Data miss cycles =  $N * 36\% * 4\% * 40 = 0.58N$
  - Totals memory stalls =  $0.8N + 0.58N = 1.38N$
  - CPI & Stalls =  $2 + 1.38 = 3.38$
  - Ratio =  $(\text{CPU Stall} / \text{CPU perfect}) = 3.38 / 2 = 1.69$





# Faster Execution?

- Assume:
  - CPI increased from 2 to 1
- Improvement:
  - CPI with stalls =  $1 + 1.38 = 2.38$
  - Perfect cache =  $2.38 / 1 = 2.38$
- Implication
  - Time spent on memory stalls =
    - $1.38 / 3.38 = 41\%$ , to
    - $1.38 / 2.38 = 58\% !!$

Optimizing execution ALONE does not help.



# Increased Clock Rate

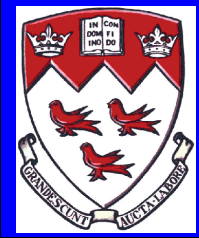
- Assume:
  - Same scenario as in previous examples
  - Clock rate doubled
- How much faster will the new machine be?
  - Miss penalty of 80 cs (given new speed)
  - Miss CPI =  $(2\% * 80) + (36\% * 4\% * 80) = 2.75$
  - New machine CPI =  $2 + 2.75 = 4.75$
  - Performance fast  
Performance slower
$$IC * CPI_{\text{slow}} * CS / IC * CPI * (CS / 2) =$$
$$3.38 / 4.75 * 0.5 = \underline{1.41} !!$$

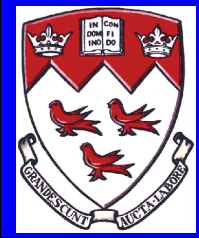
Not by a factor of 2.



# What to do?

- Did it help...
  - Perfect? 1.38
  - No optimization? 3.38
  - Make instruction exec faster? 2.38
  - Speed up clock? 1.41





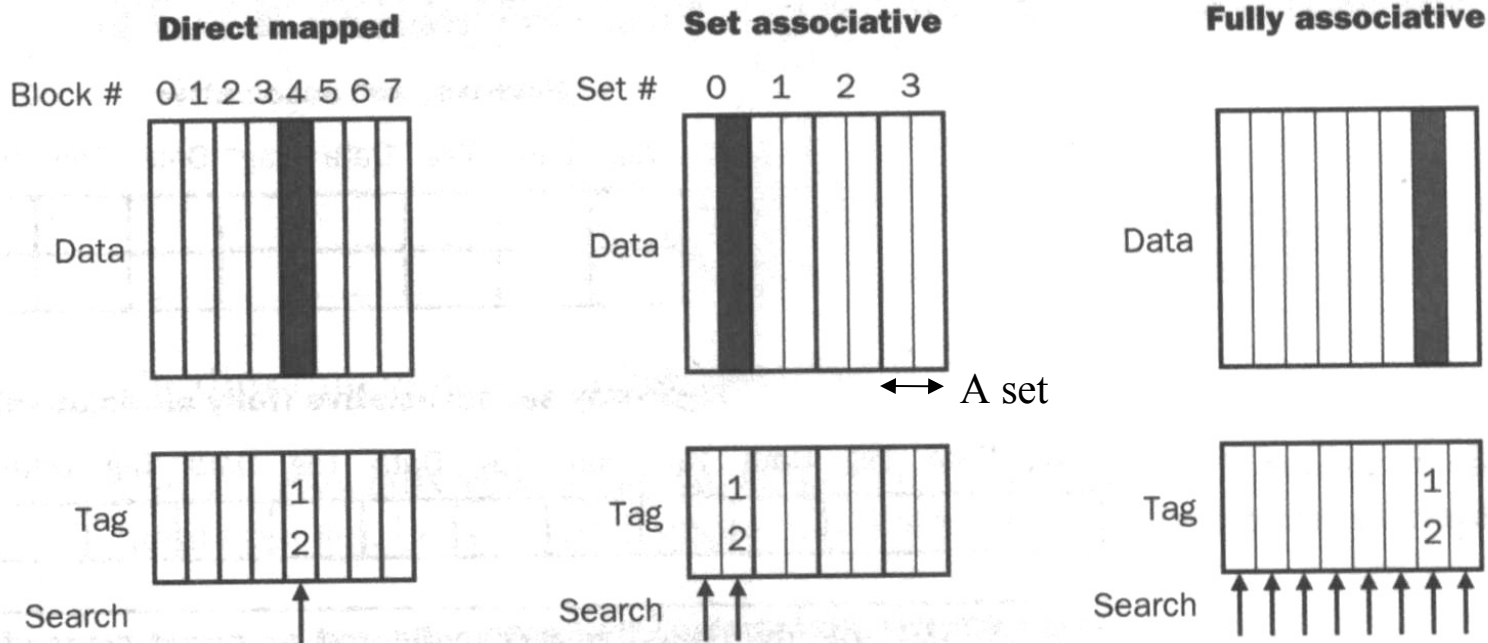
# Part 3

## Reducing Cache Misses



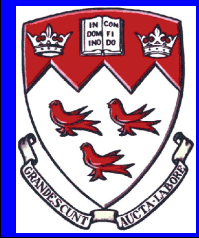
# Associativity

Assume address 12



Direct: Block No. MODULO Number of cache blocks

Assoc: Block NO. MODULO Number of sets in cache







# Associative Caches

## One-way set associative (direct mapped)

Block Tag Data

0		
1		
2		
3		
4		
5		
6		
7		

## Two-way set associative

Set Tag Data Tag Data

0				
1				
2				
3				

## Four-way set associative

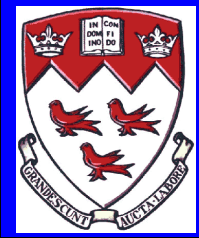
Set Tag Data Tag Data Tag Data Tag Data

0									
1									

## Eight-way set associative (fully associative)

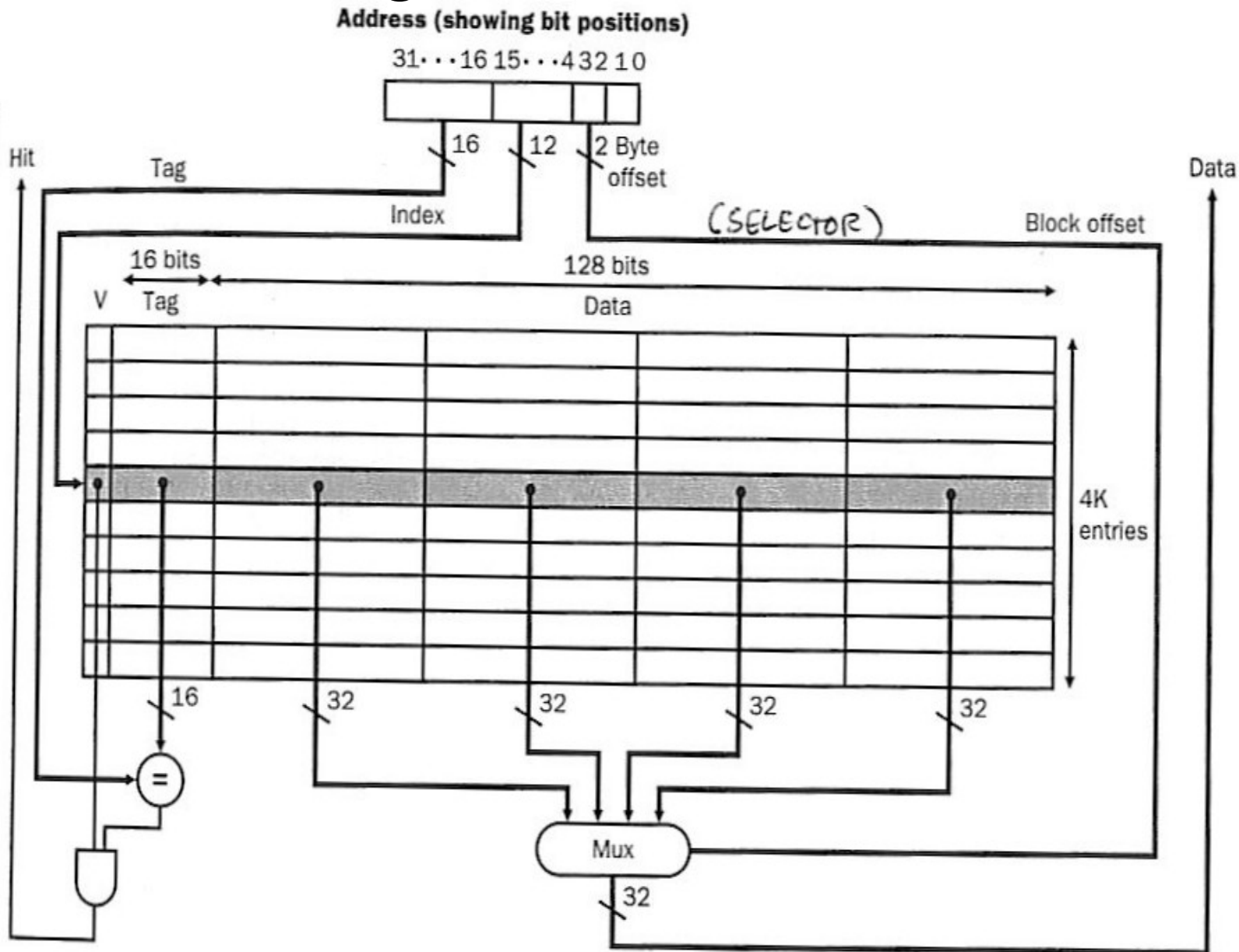
Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--



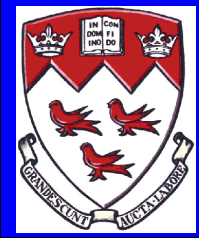


# Addressing in Direct Blocked Cache



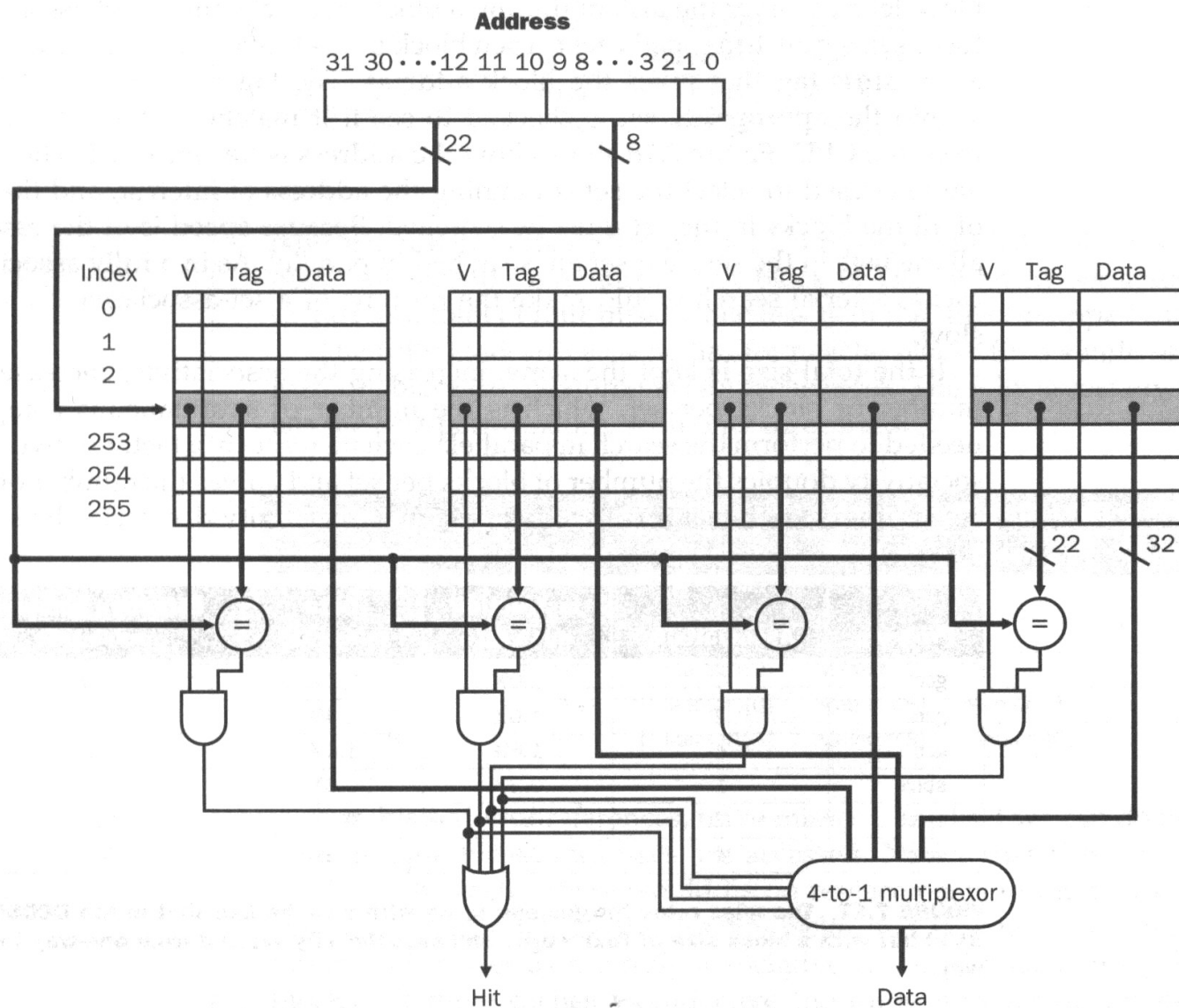
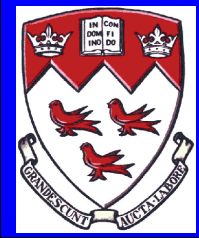
COMP 273

Introduction to Computer Systems





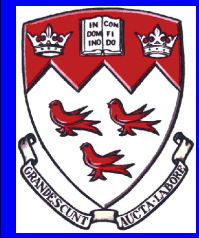
# Addressing in Associative Cache

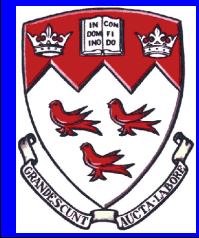




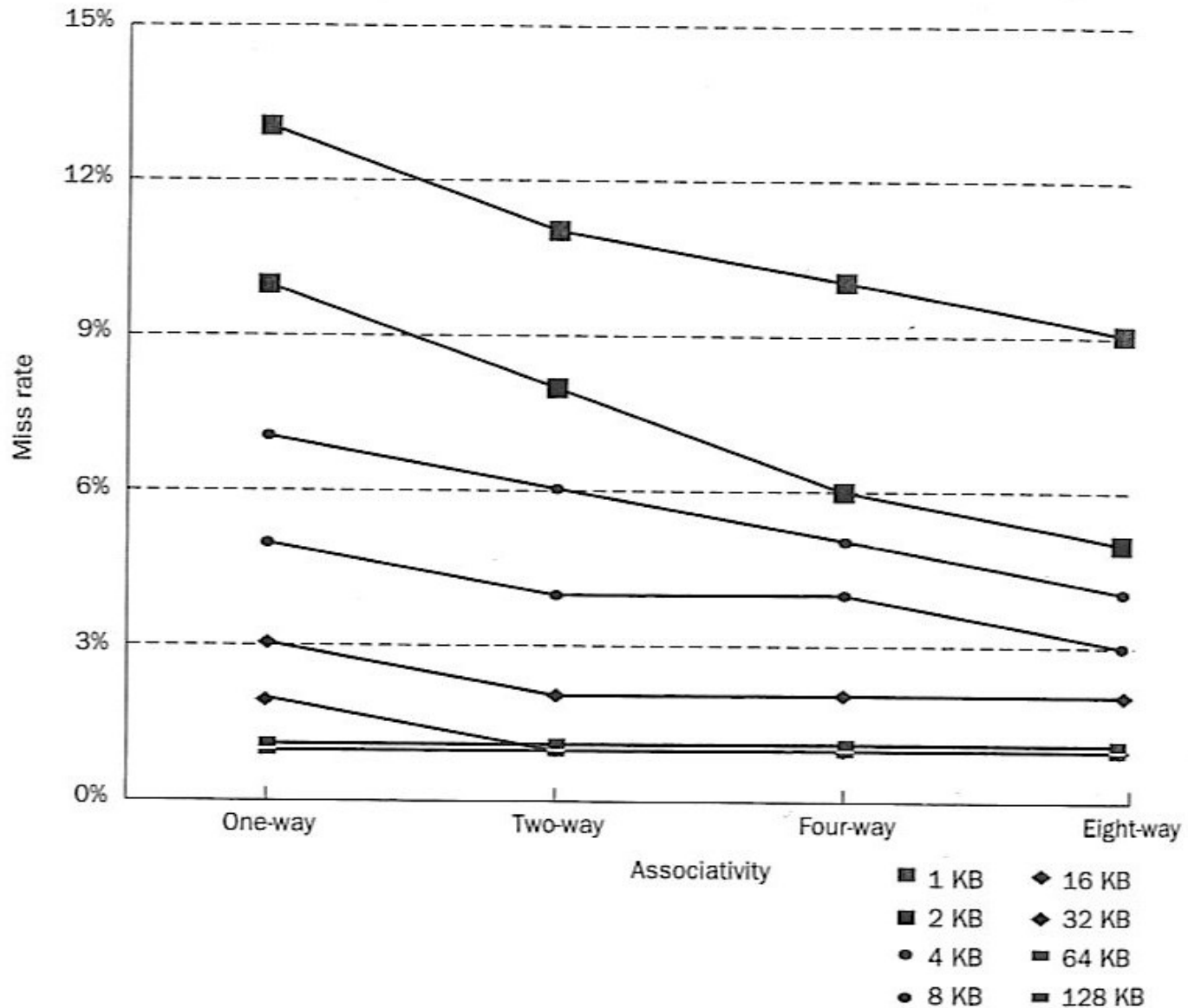
# Why would this help?

- Time Locality
  - Code tends to be iterative
- Space Locality
  - Code tends to be sequential
- Question:
  - Where does it break down?
    - GOTO statements
    - Function calls
    - Global variables
    - Page sizes





# Performance

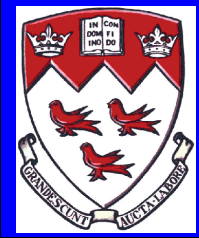




# Part 4

## Some Real Hardware

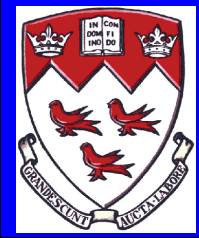
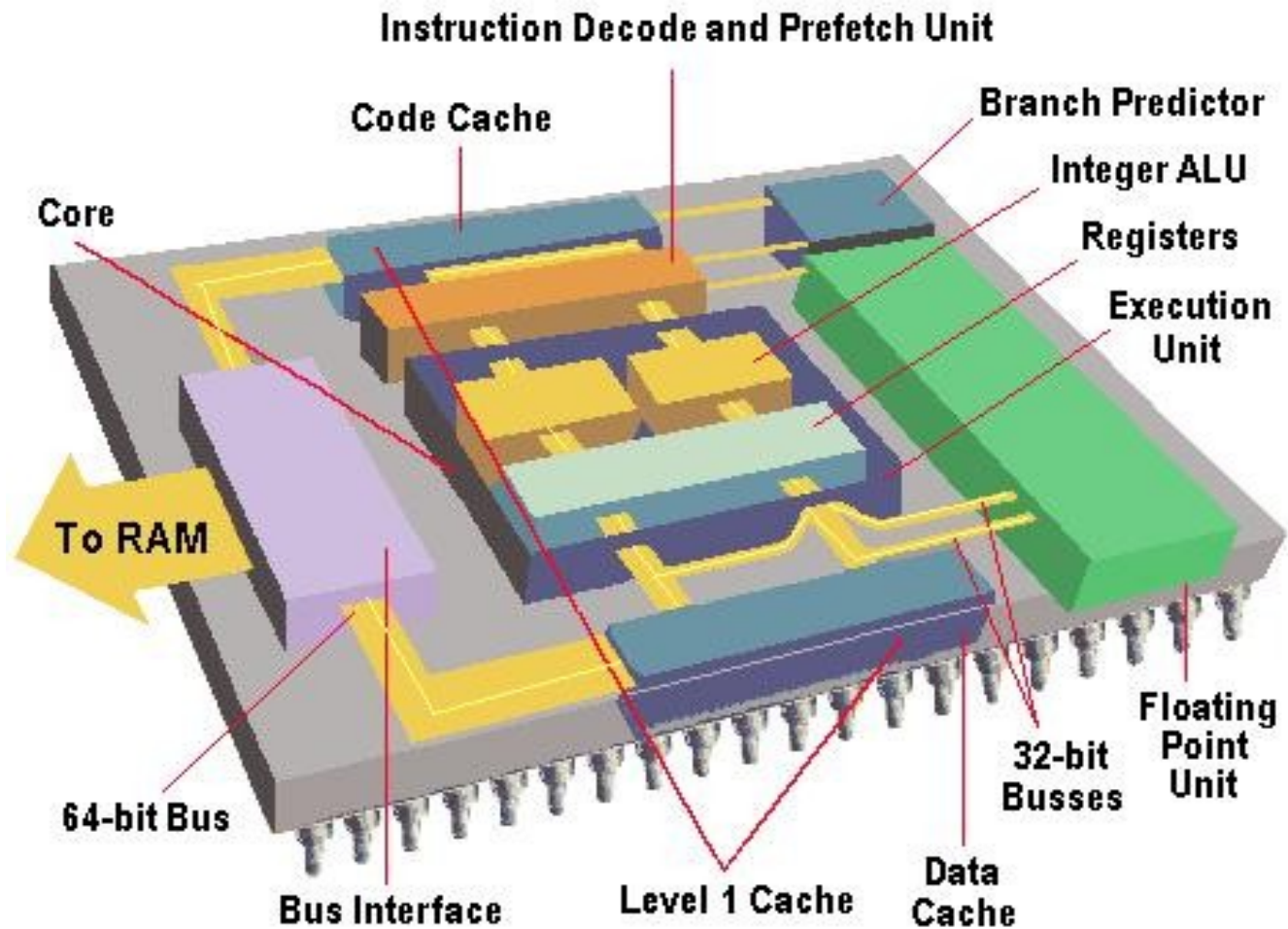
Old Statistics





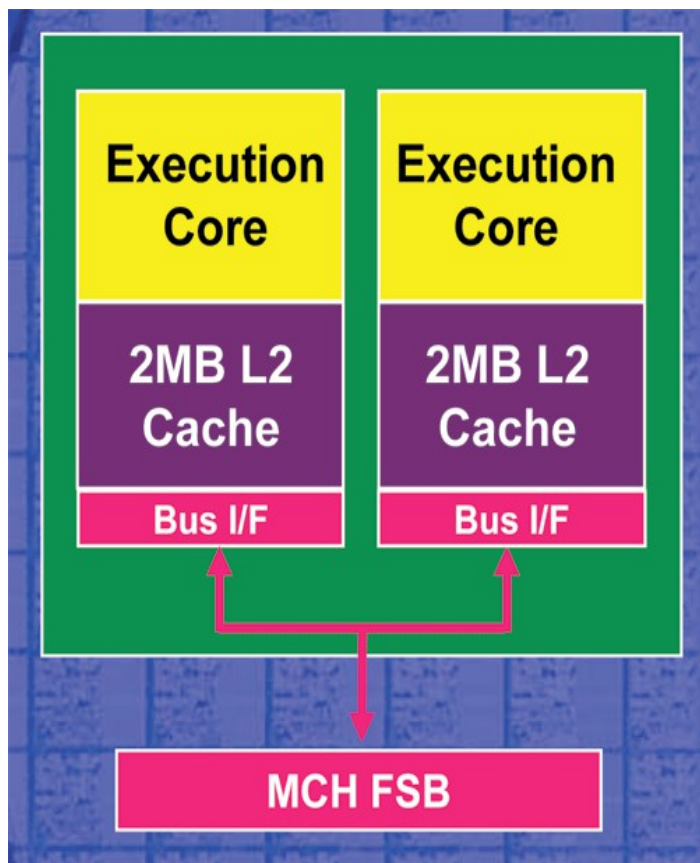


# Standard CPU

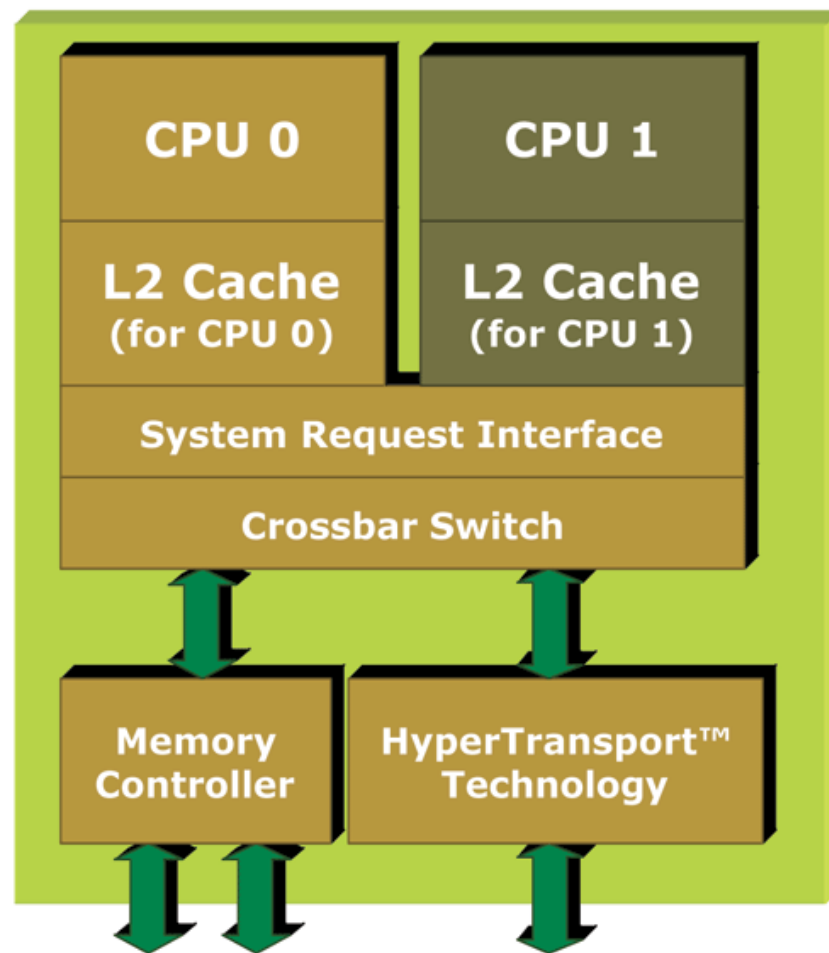




# Dual Core



**Intel's Pentium D dual core architecture**



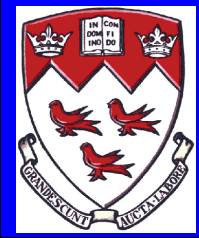
**AMD Athlon™ 64 X2  
Dual-Core Processor Design**





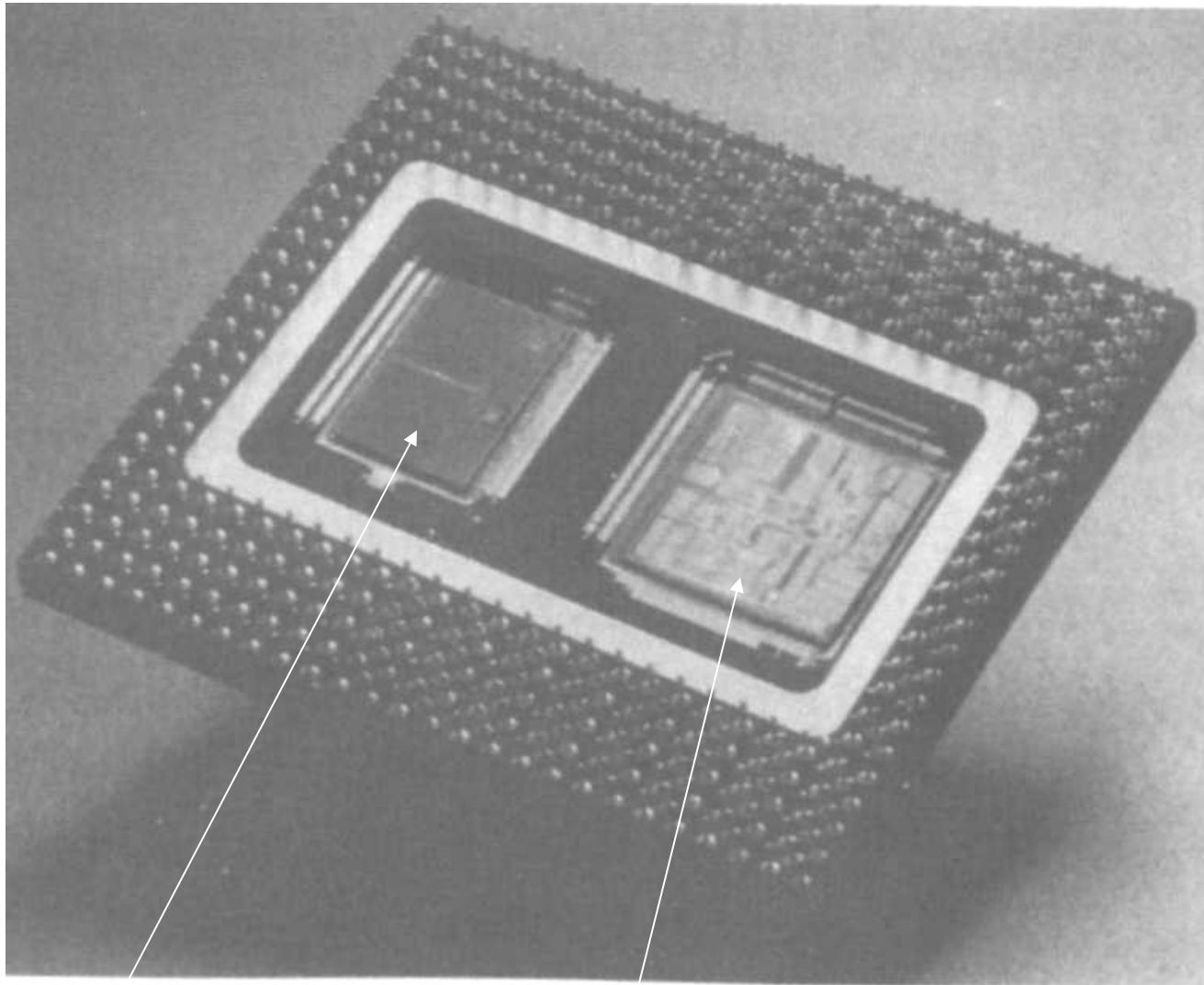
Characteristic	Intel Pentium Pro	PowerPC 604
Virtual address	32 bits	52 bits
Physical address	32 bits	32 bits
Page size	4 KB, 4 MB	4 KB, selectable, and 256 MB
TLB organization	A TLB for instructions and a TLB for data Both four-way set associative Pseudo-LRU replacement Instruction TLB: 32 entries Data TLB: 64 entries TLB misses handled in hardware	A TLB for instructions and a TLB for data Both two-way set associative LRU replacement Instruction TLB: 128 entries Data TLB: 128 entries TLB misses handled in hardware

Characteristic	Intel Pentium Pro	PowerPC 604
Cache organization	Split instruction and data caches	Split instruction and data caches
Cache size	8 KB each for instructions/data	16 KB each for instructions/data
Cache associativity	Four-way set associative	Four-way set associative
Replacement	Approximated LRU replacement	LRU replacement
Block size	32 bytes	32 bytes
Write policy	Write-back	Write-back or write-through





# INTEL CPU CHIP



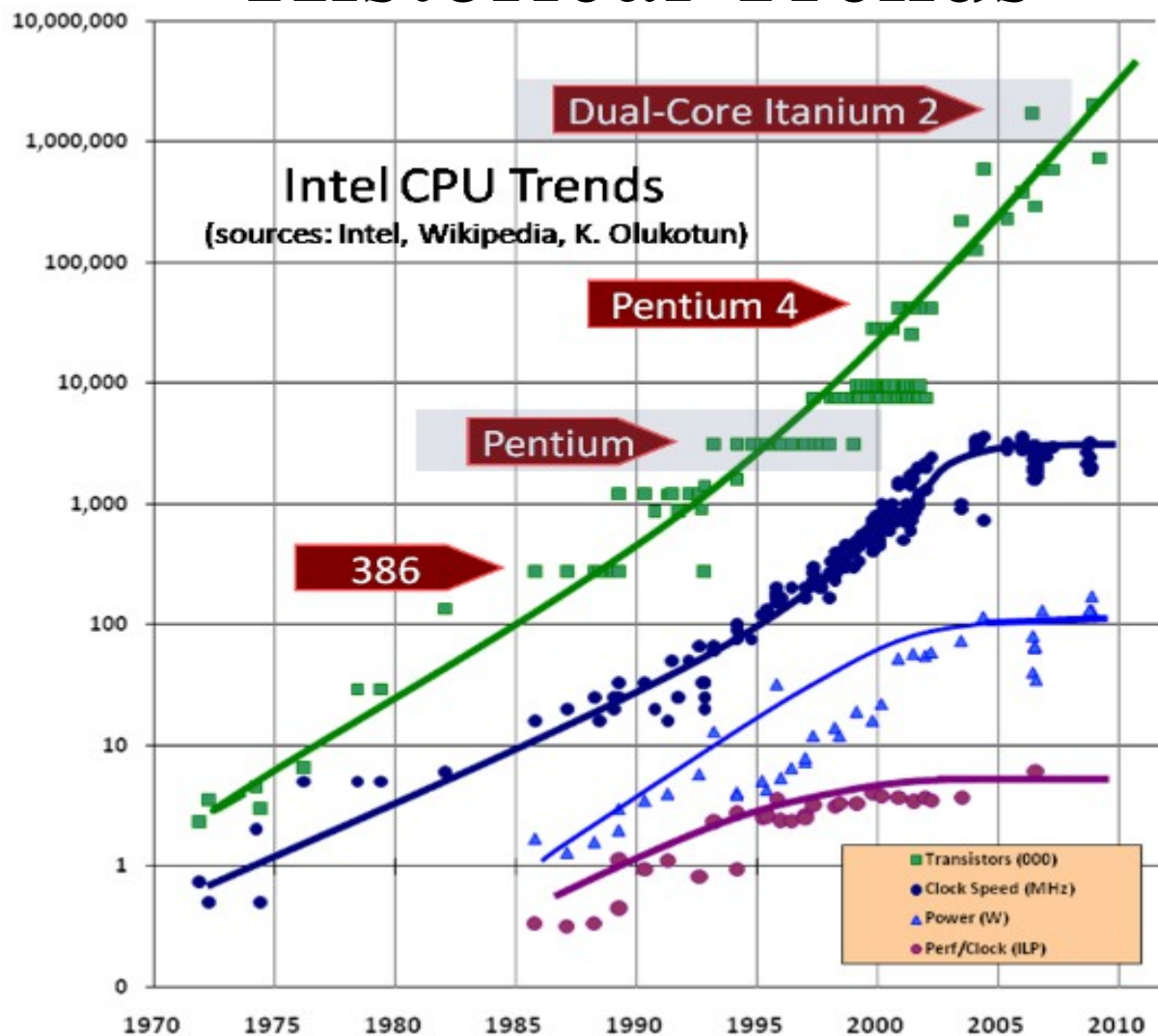
On-board Cache

The CPU



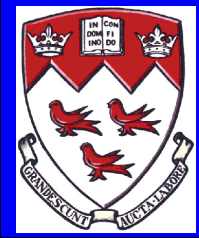


# Historical Trends



Moor's first law: CPU power doubles every 2 years

Moors 2<sup>nd</sup> law: CPU manufacturing cost doubles every 3 years



# Trends

## Evolution of Computer Power/Cost

### Brain Power Equivalent per \$1000 of Computer

**MIPS per \$1000** (1998 Dollars)

