EECS 322 Computer Architecture

Superpipline and the Cache

Instructor: Francis G. Wolff

wolff@eecs.cwru.edu

Case Western Reserve University

This presentation uses powerpoint animation: please viewshow

Summary: Instruction Hazards

Instruction Class	Integer Application	Floating-Point (FP) Application
Integer Arithmetic	50%	25%
FP-Arithmetic	0%	30%
Loads	17%	25%
Stores	8%	15%
Branches	25%	5%

	No-Forwarding	**Forwarding**	**Hazard**
R-Format	**1-3**	**1**	**Data**
Load	**1-3**	**1-2**	**Data, Structural**
Store	**1**	**1-2**	**Structural**

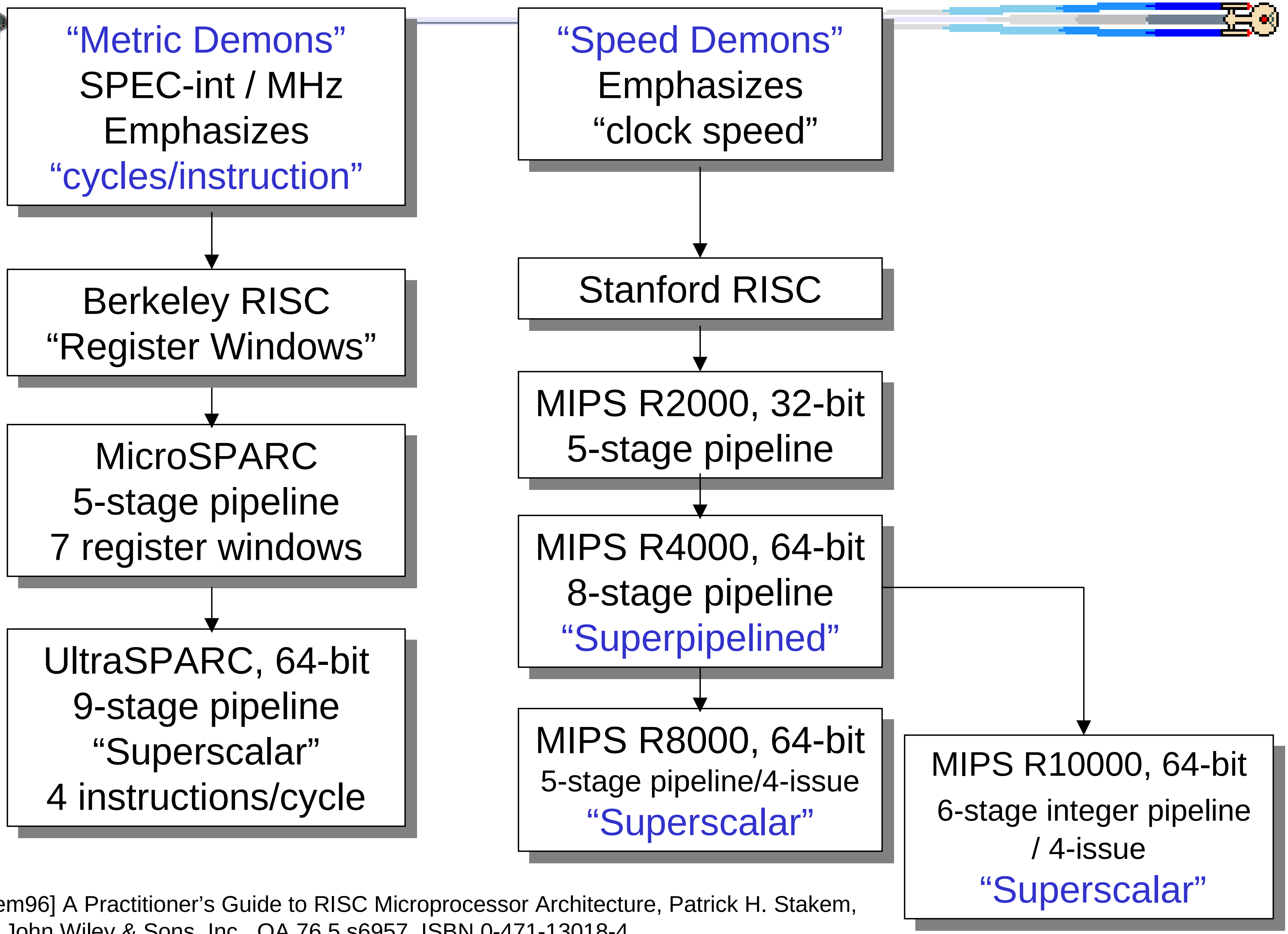
	No Delay Slot	**Delay Slot**	**Hazard**
Branch **(decision is made in the ID stage)**	**2**	**1**	**Control**
Branch **(decision is made in the EX stage)**	**3**	**1**	**Control**
Jump	**2**	**1**	

Structural Hazard: Instruction & Data memory combined.

Ref: http://www.sun.com/microelectronics/whitepapers/UltraSPARCtechnology/

RISC camps

[Skakem96]



“Metric Demons”
SPEC-int / MHz
Emphasizes
“cycles/instruction”

Berkeley RISC
“Register Windows”

MicroSPARC
5-stage pipeline
7 register windows

UltraSPARC, 64-bit
9-stage pipeline
“Superscalar”
4 instructions/cycle

“Speed Demons”
Emphasizes
“clock speed”

Stanford RISC

MIPS R2000, 32-bit
5-stage pipeline

MIPS R4000, 64-bit
8-stage pipeline
“Superpipelined”

MIPS R8000, 64-bit
5-stage pipeline/4-issue
“Superscalar”

MIPS R10000, 64-bit
6-stage integer pipeline
/ 4-issue
“Superscalar”

[Stakem96] A Practitioner’s Guide to RISC Microprocessor Architecture, Patrick H. Stakem, 1996, John Wiley & Sons, Inc., QA 76.5.s6957, ISBN 0-471-13018-4

Instruction Level Parallelism (ILP)

Superpipelined scheme

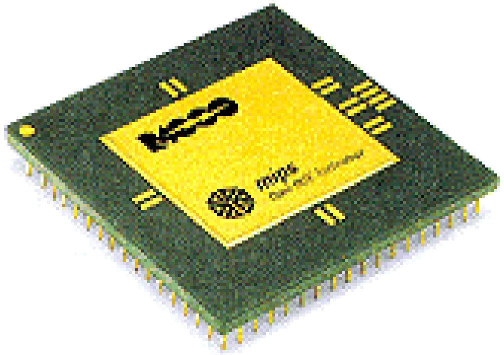
- uses a longer pipeline with more stages to reduce cycle time
- simple dependencies: structural, data, control pipeline hazards.
- requires higher clock speeds
- require little additional logic to baseline processor
- Branches cause a latency of 3 internal clocks and loads a 2-cycle latency.
- However, superpipelining increases performance, because each stage can run at twice the system clock.

Instruction Level Parallelism (ILP)

•Superscalar scheme

- multiple execution units by duplicating the functional units (ALUs)
- combinatorial dependance problem
  - ✓ instructions can only be issued only if they are independent
- require sophisticated & complex logic (i.e. instruction Scheduler)

R4400 processor



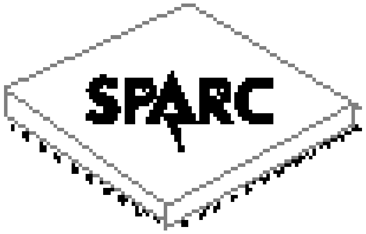
100MHz offers
85 MIPS, 16 Mflops,
70 SPECmarks

Both R4000 and R4400 use:

- True 64bit architecture
- On–chip TLB for virtual–to–physical address translation
- 8–stage superpipelined architecture
- large integrated caches

150MHz R4400 offers
136 MIPS, 24 Mflops,
100 SPECmarks.

- UltraSPARC is a high-volume processor — the R4400 is a low volume processor
- UltraSPARC enjoys the installed base of the SPARC product line — the R4400 has a small installed base
- UltraSPARC is binary compatible with other SPARC processors — the R4400 has narrow application support
- UltraSPARC-I is network and multimedia ready — the R4400 does not provide multimedia support
- UltraSPARC provides high performance and high bandwidth — the R4400 provides comparatively low integer and floating point performance



Ref: http://sunsite.ics.forth.gr/sunsite/Sun/sun_microelectronics/UltraSparc/ultra_arch_versus.html

µP Comparisons

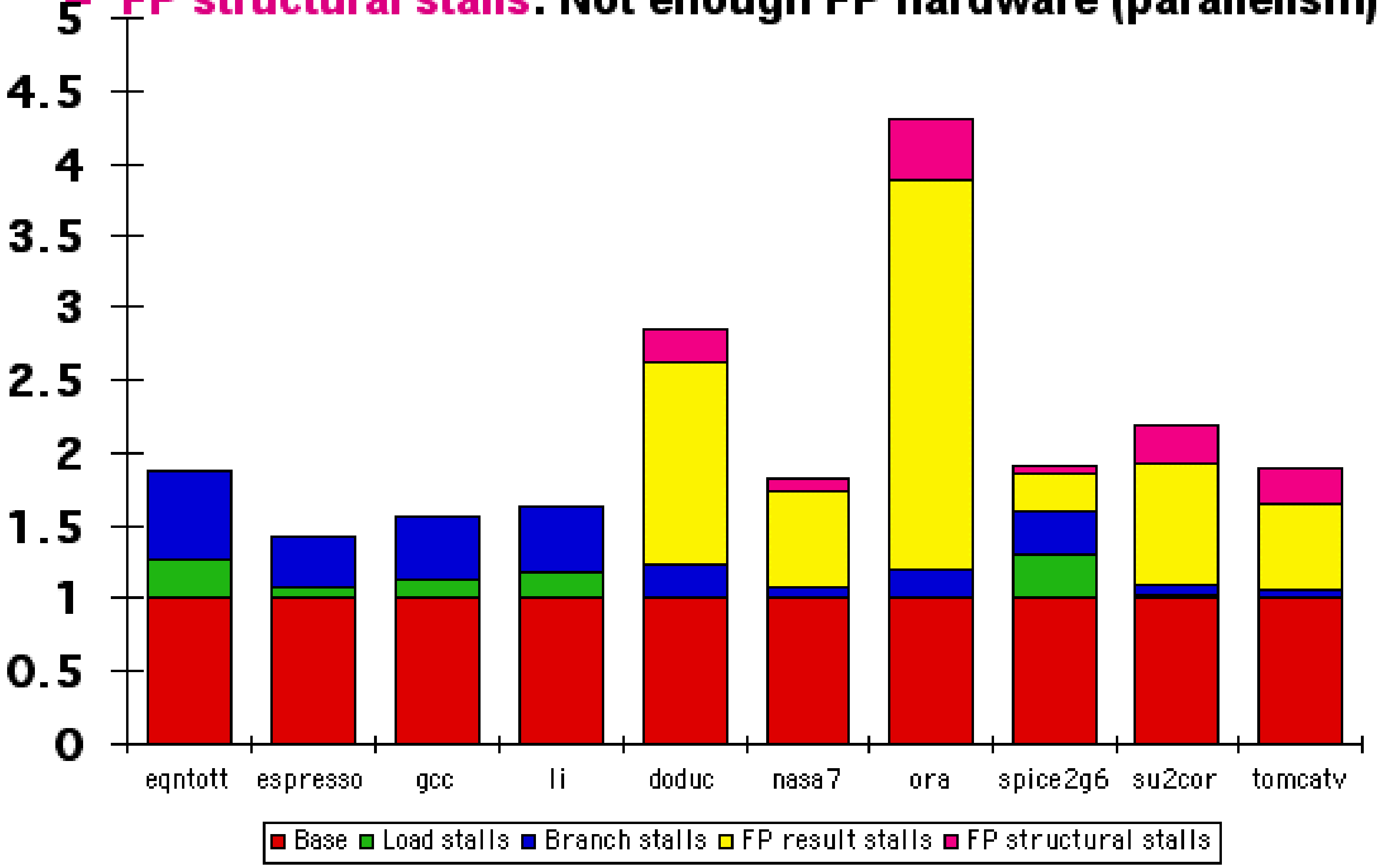
	MIPS R4400	UltraSparc I
Clock	250 MHz	167 MHz
Bus speed	50/66/75	83 Mhz
Pipeline	8-stage	9-stage
Superscalar	1-issue	4-issue
Branch prediction	no	yes
TLB	48 even/odd	64-Inst/64-Data
L1 I/D-cache	16k/16k	16k/16k
Associativity	1-way (direct)	2-way
L2 cache	1 Mb	0.5Mb
CMOS technology	0.35µ	0.5µ 4 layers
Fabrication Vendor	NEC, IDT, Toshiba	Fujitsu
Year	1993	1995
Voltage	3.3 volts	3.3 volts
Transistors	2.2 million	3.8-5.2 million
SpecInt92/fp92	175/178	250/350
SpecInt95/fp95	5.07/?	6.26/9.06
Cost:	$1250	$1395

Ref: http://sunsite.ics.forth.gr/sunsite/Sun/sun_microelectronics/UltraSparc/ultra_arch_versus.html
http://www.mips.com/Documentation/R4400_Overview.pdf
http://www.spec.org/osg/cpu95/results/res96q3/cpu95-960624-00962.html **and** http://www.eecs.umich.edu/~tnm/theses/mikeu.pdf

R4000: no dynamic branch prediction

R4000 Performance

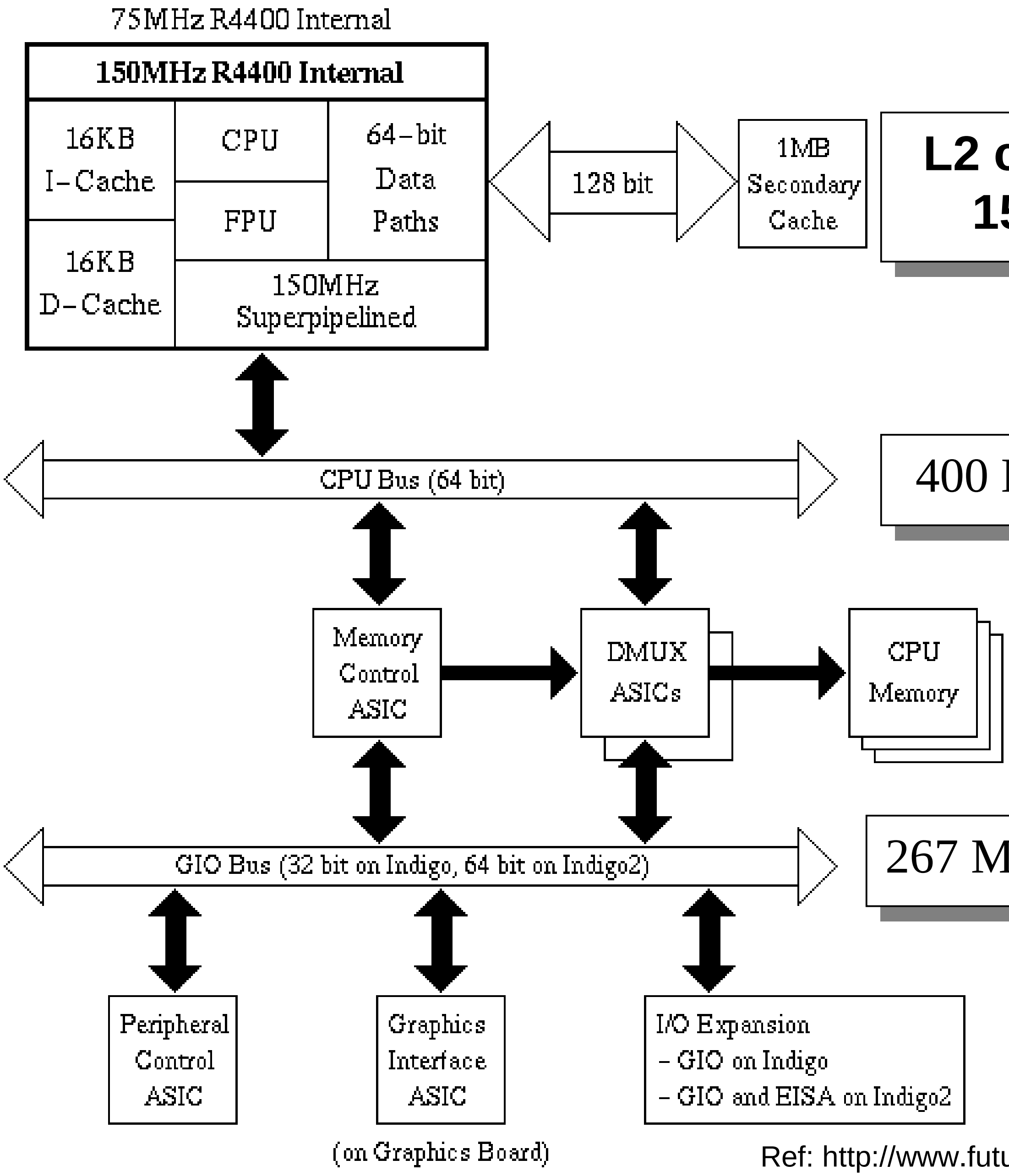
- Not ideal CPI of 1:
  - Load stalls (1 or 2 clock cycles)
  - Branch stalls (2 cycles + unfilled slots)
  - FP result stalls: RAW data hazard (latency)
  - FP structural stalls: Not enough FP hardware (parallelism)



Differences Between the MIPS R4400 and UltraSPARC-I

- The MIPS R4400 uses an 8-stage pipeline architecture, and is less efficient than the superscalar, pipelined UltraSPARC-I.
- Although it is an integrated processor, the R4400 requires several other modules in order to incorporate it into a system.
- External secondary caches (L2) must be designed around the processor, and multiprocessor and graphics support are not provided.
- The highly integrated UltraSPARC-I, utilizing on-chip caches, an advanced processor design and UPA architecture, requires little to complete its chip set, significantly easing its integration into systems.

R4400 Bus



L2 cache
15ns

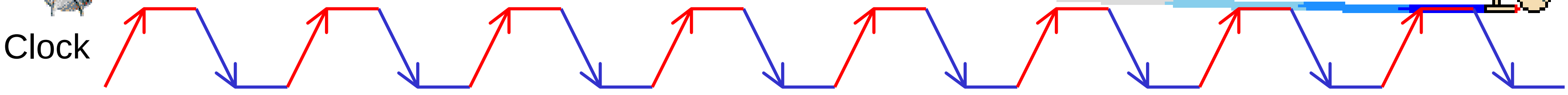
400 MB/sec peak

267 MB/sec sustained

Ref: http://www.futuretech.vuurwerk.nl/r4k150upgrade.html

R4000 Pipeline

[Heinrich93]

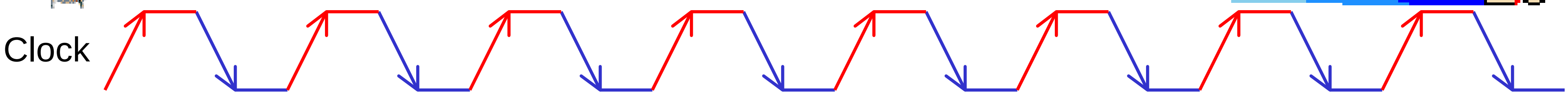


Phase	Φ_1	Φ_2	Φ_1	Φ_2	Φ_1	Φ_2	Φ_1	Φ_2	Φ_1	Φ_2	Φ_1	Φ_2	Φ_1	Φ_2	Φ_1	Φ_2
Stage	**IF**		**IS**		**RF**		**EX**		**DF**		**DS**		**TC**		**WB**	

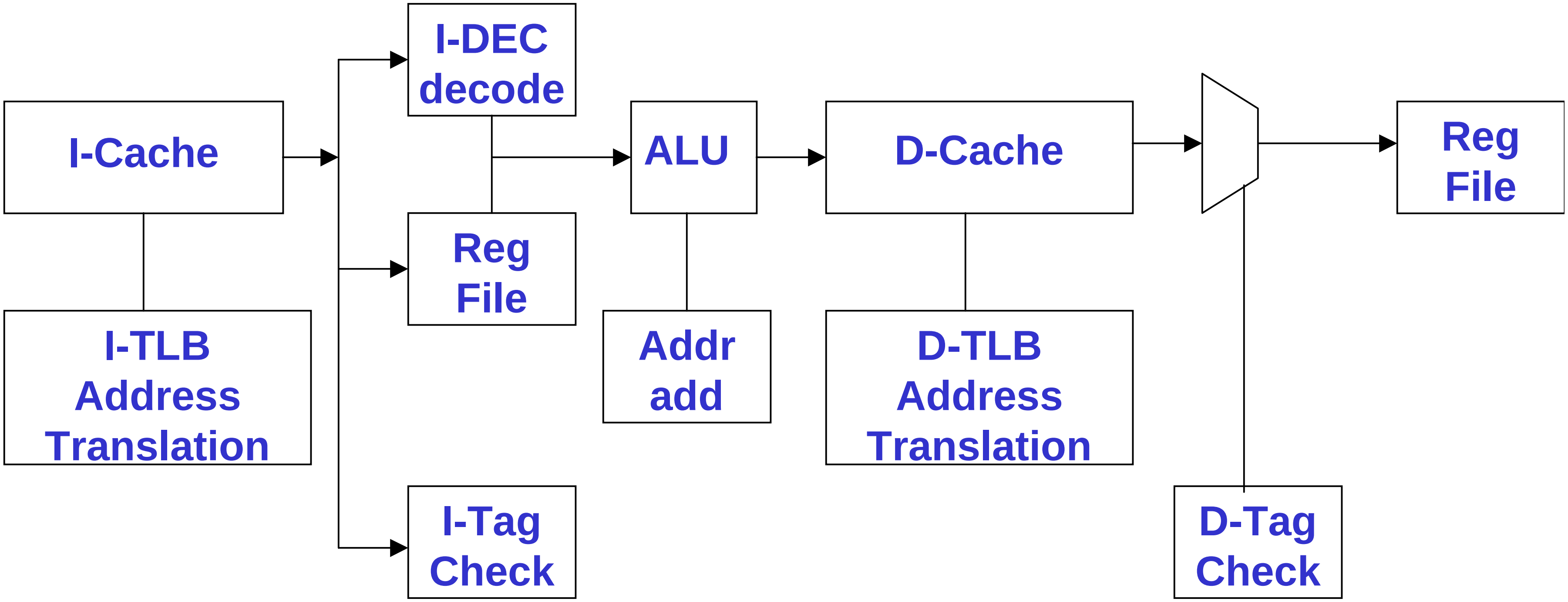
- IF - Instruction Fetch, first half
- IS - Instruction fetch, Second half
- RF - Register Fetch
- EX - Execution (branch compare)
- DF - Data Fetch, first half
- DS - Data fetch, Second half
- TC - Tag Check
- WB - Write Back

[Heinrich93] MIPS R4000 User’s Manual, Joseph Heinrich, Prentice-Hall, 1993, QA76.8.m523h45, ISBN 0-13-105925-4

R4400 SuperPipeline



Phase	Φ_1	Φ_2	Φ_1	Φ_2	Φ_1	Φ_2	Φ_1	Φ_2	Φ_1	Φ_2	Φ_1	Φ_2	Φ_1	Φ_2	Φ_1	Φ_2
Stage	IF		IS		RF		EX		DF		DS		TC		WB	



R4000 Pipeline stages: IF & IS

IF - instruction Fetch, First half

- PC: Branch logic selects an instruction address and
- instruction catch fetch begins
- I-TLB: instruction translation lookaside buffer begins the virtual-to-physical address translation

IS - instruction fetch, second half

- Complete instruction catch fetch and
- the virtual-to-physical address translation

R4000 Pipeline stages: RF

RF - register fetch

- I-DEC: instruction decoder decodes the instruction and checks for interlock conditions

- instruction cache tag is checked against the page frame number (PFN) obtained from the ITLB.

- Any required operands are fetched from the register file

R4000 Pipeline stages: EX

EX - execution

- Register-to-register instructions: The ALU performs arithmetic or logical operation

- Load & Store instructions: the ALU calculates the data virtual address (i.e. offset + base register).

- Branch instructions: The ALU determines whether the branch condition is true & calculates the virtual target address.

R4000 Pipeline stages: DF & DS

DF - data fetch, first half

- Register-to-Register: No operations are performed during DF, DS, and TC stages

- Load & Store instructions: The data cache fetch and the data virtual-to-physical translation begins

- Branch instructions: address translation and TLB update begins

DS - data fetch, second half

- Load & Store: completion of data cache fetch & data virtual-to-physical translation. The shifter aligns data to its word or doubleword boundary

- branch: completion of instruction address translation and TLB update

R4000 Pipeline stages: TC & WB

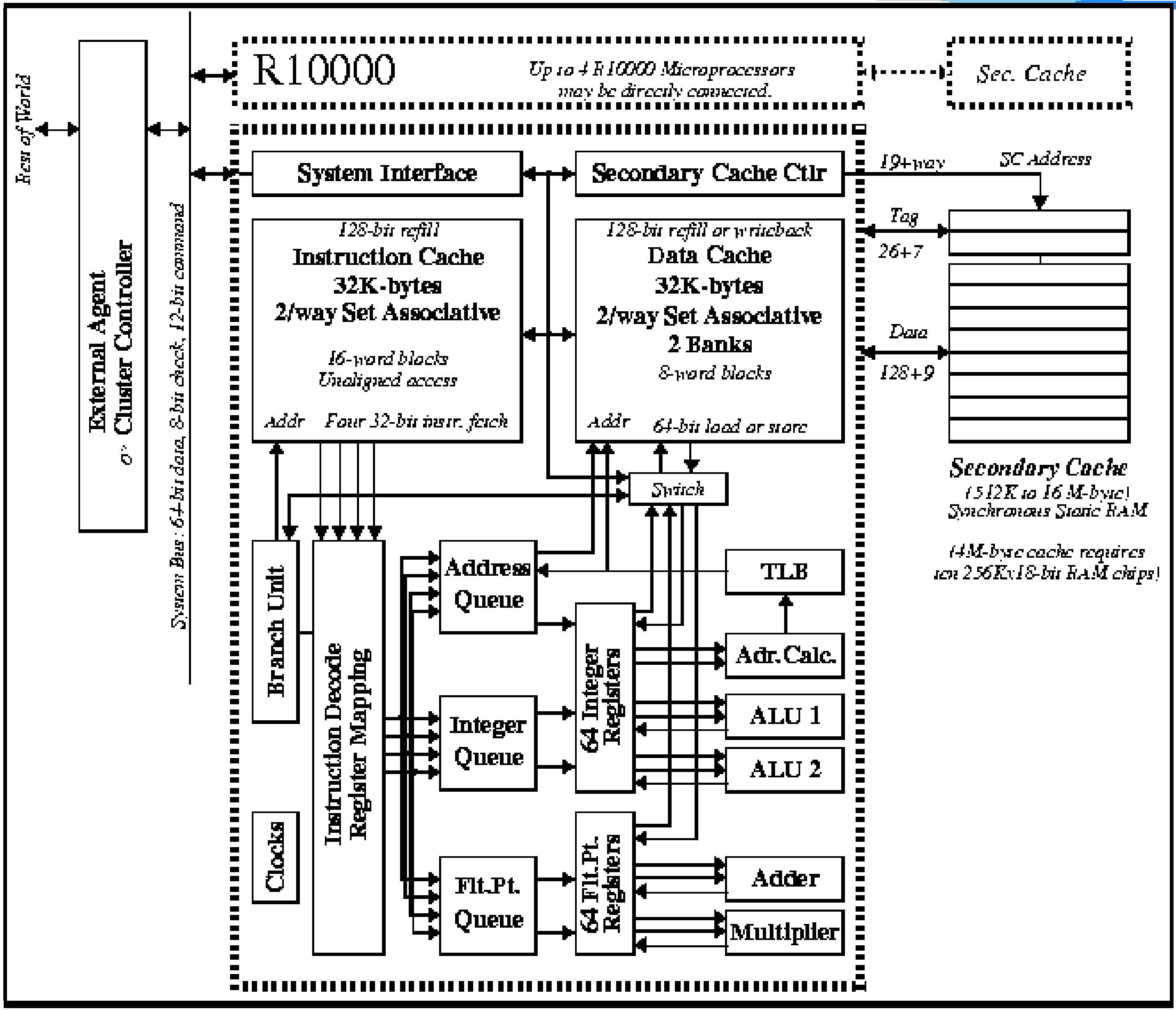
TC - Tag check

- Load & Store instructions: the cache performs the tag check.
- Hit or Miss: physical address from TLB is checked against the tag check to determine if there is a hit or miss.

WB - write back

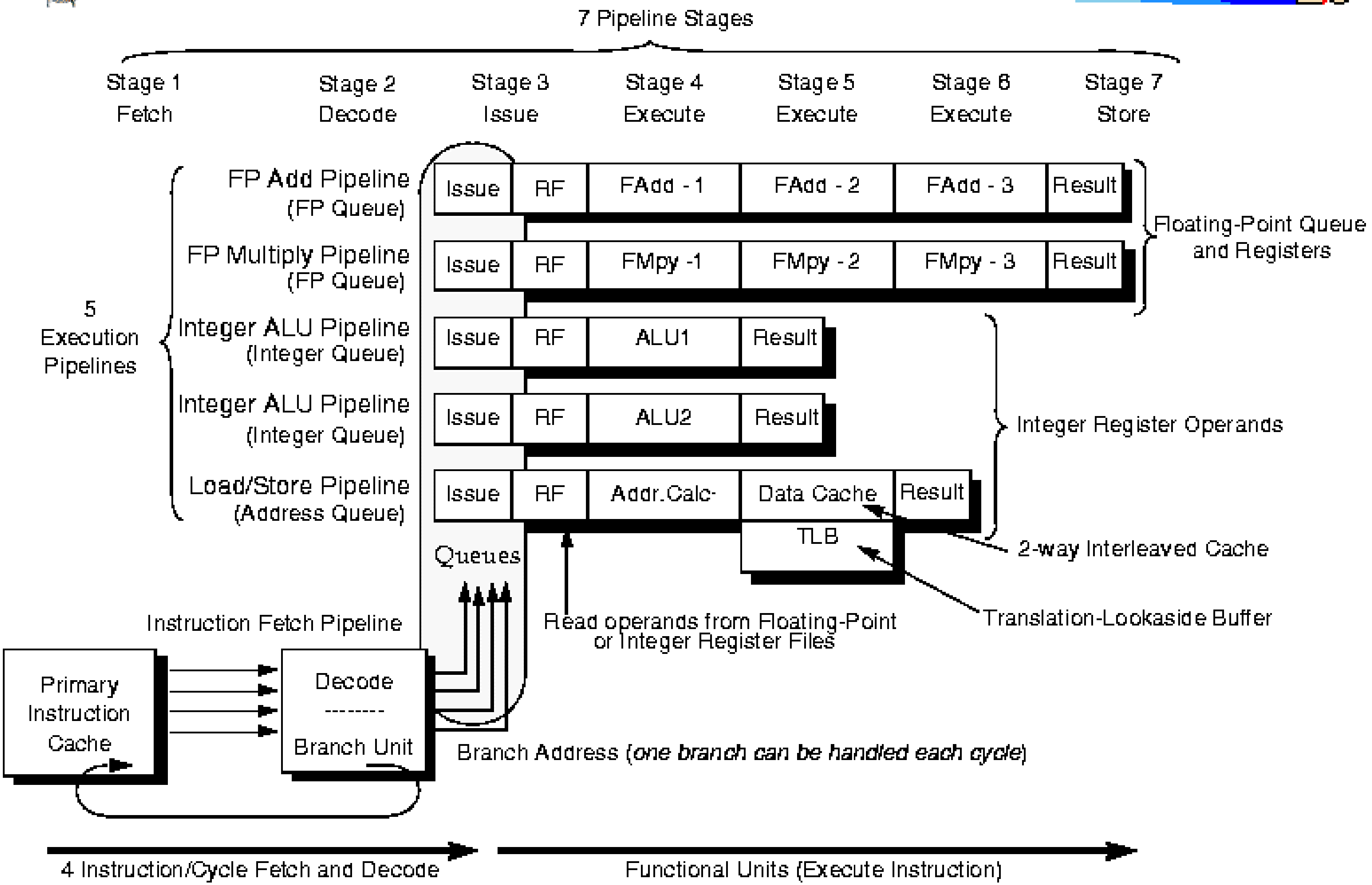
- Register-to-register & load: the instruction result is written back to the register file
- Branch: no operation

R10000 superscalar architecture



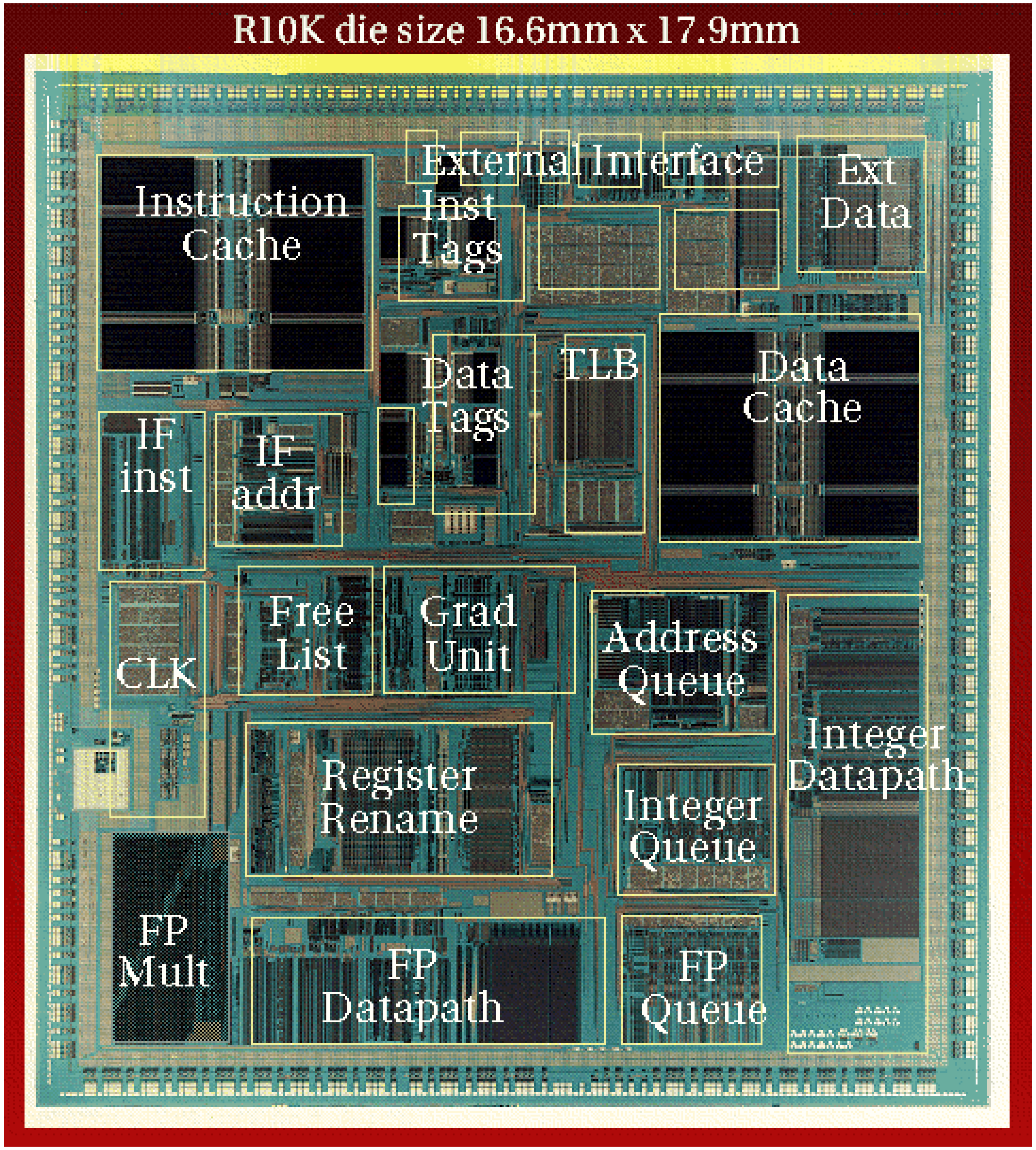
Ref: http://www.sgi.com/processors/r10k/tech_info/Tech_Brief.html

R10000 - superscalar



Ref: http://www.sgi.com/processors/r10k/manual/T5.HW.Ch01.intro_AFrame_16.gif

R10000 die



R10000

SPECint95 base 14.1
SPECint95 peak 14.7 SPECfp95 base 22.6
SPECfp95 peak 24.5

200 MHz Clock

I/D-cache: 32k/32k

TLB: 64 entries
Virtual Page Sizes: 16k-16M

0.35µ 4-layer CMOS technology

17 mm x18 mm chip

contains about **6.7** million transistors

including about 4.4 million transistors in its primary caches.

Ref: http://www.byte.com/art/9801/sec4/art4.htm

Principle of Locality

- Principle of Locality

  states that programs access a relatively small portion of their address space at any instance of time

- Two types of locality
  - Temporal locality (locality in time)

    If an item is referenced, then

    the same item will tend to be referenced soon

    “the tendency to reuse recently accessed data items”

  - Spatial locality (locality in space)

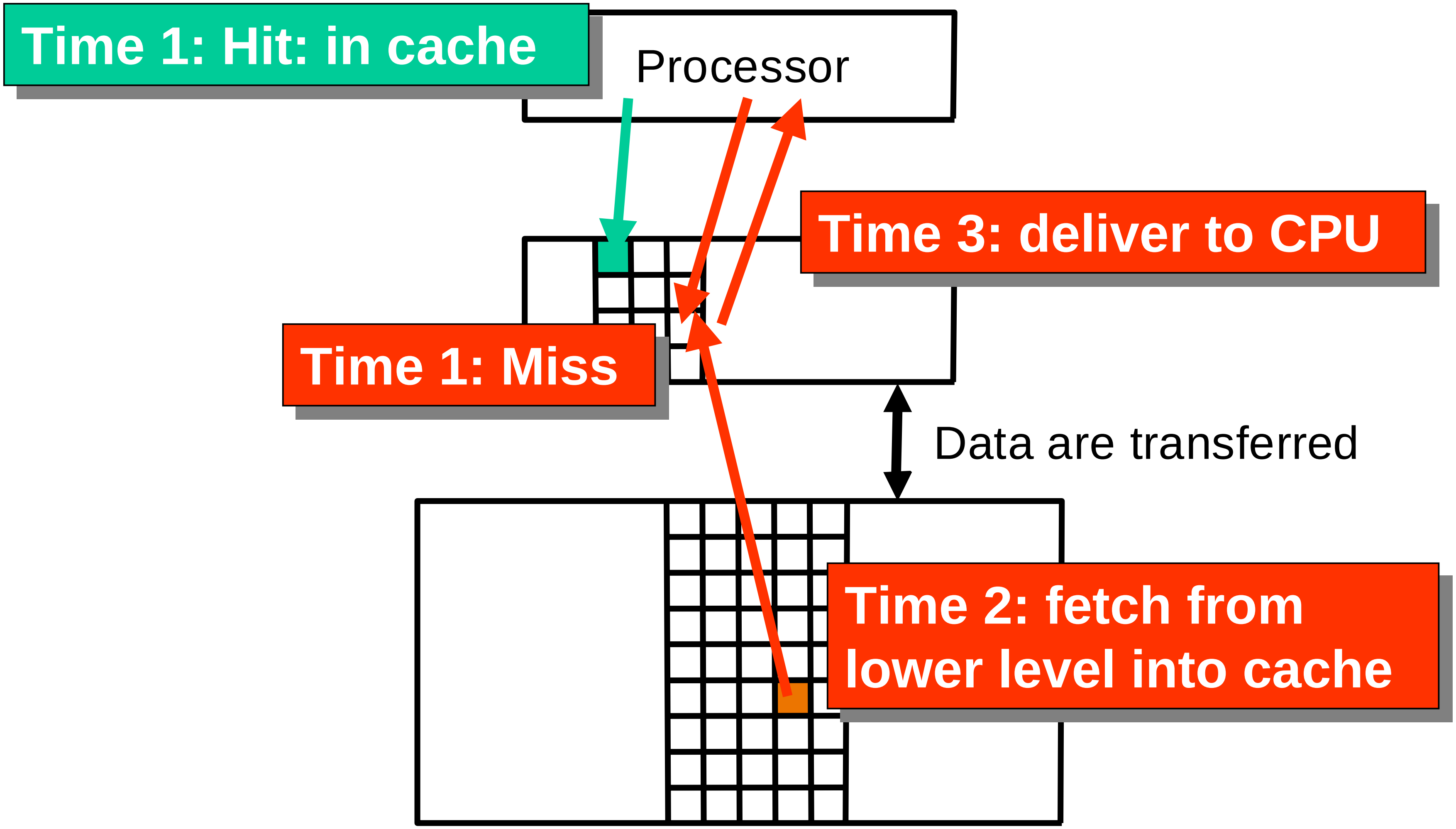
    If an item is referenced, then

    nearby items will be referenced soon

    “the tendency to reference nearby data items”

Figure 7.2

Cache Example



Time 1: Hit: in cache

Processor

Time 3: deliver to CPU

Time 1: Miss

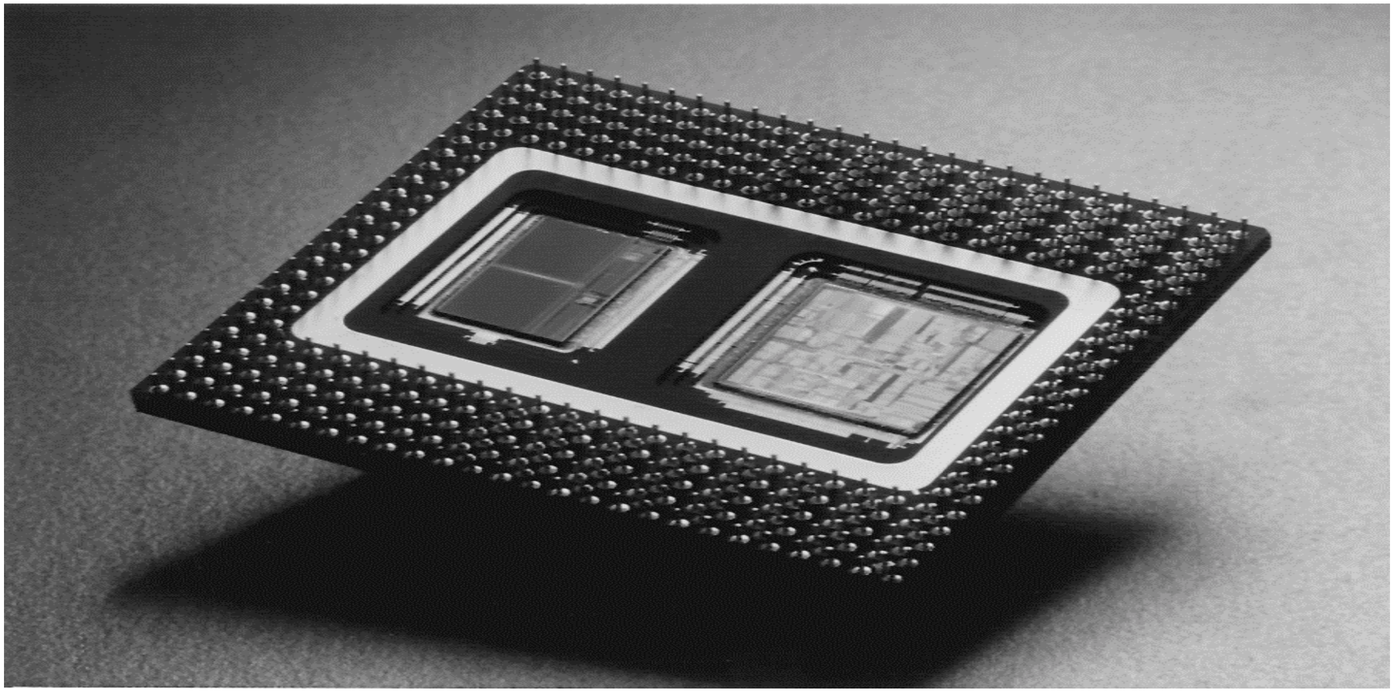
Data are transferred

Time 2: fetch from lower level into cache

Hit time = **Time 1**

Miss penalty = **Time 2 + Time 3**

Modern Systems: Pentium Pro and PowerPC



Characteristic	Intel Pentium Pro	PowerPC 604
Cache organization	Split instruction and data caches	Split intruction 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

Cache Terminology

A hit if the data requested by the CPU is in the upper level

Hit rate or **Hit ratio**

is the fraction of accesses found in the upper level

Hit time

is the time required to access data in the upper level
= <detection time for hit or miss> + <hit access time>

A miss if the data is not found in the upper level

Miss rate or **(1 – hit rate)**

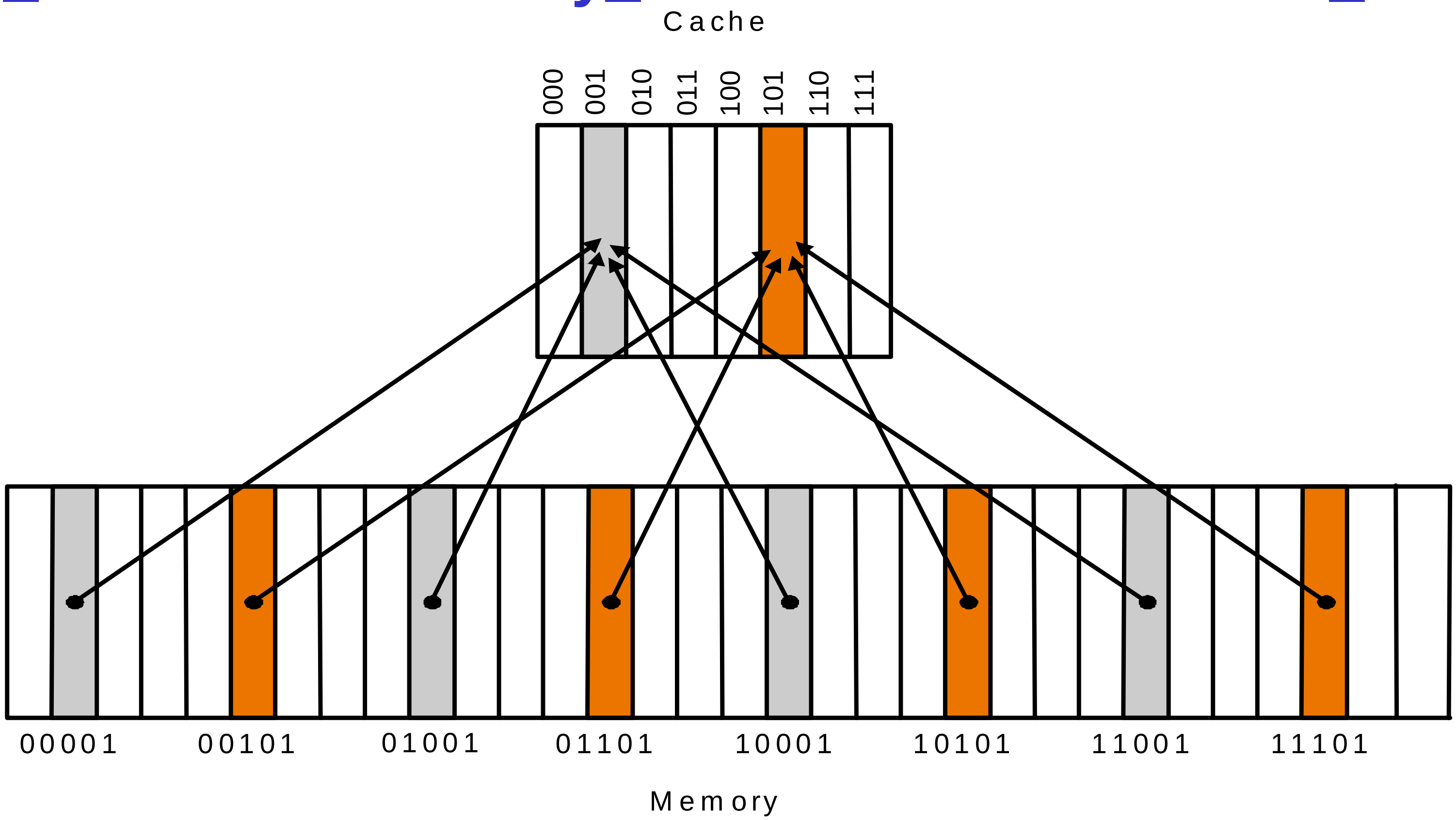
is the fraction of accesses not found in the upper level

Miss penalty

is the time required to access data in the lower level
= <lower access time>+<reload processor time>

Direct Mapped Cache

- **Direct Mapped:** assign the cache location based on the address of the word in memory
- **cache_address = memory_address modulo cache_size;**



Observe there is a Many-to-1 memory to cache relationship

Direct Mapped Cache: Data Structure

There is a Many-to-1 relationship between memory and cache

How do we know whether the data in the cache corresponds to the requested word?

tags

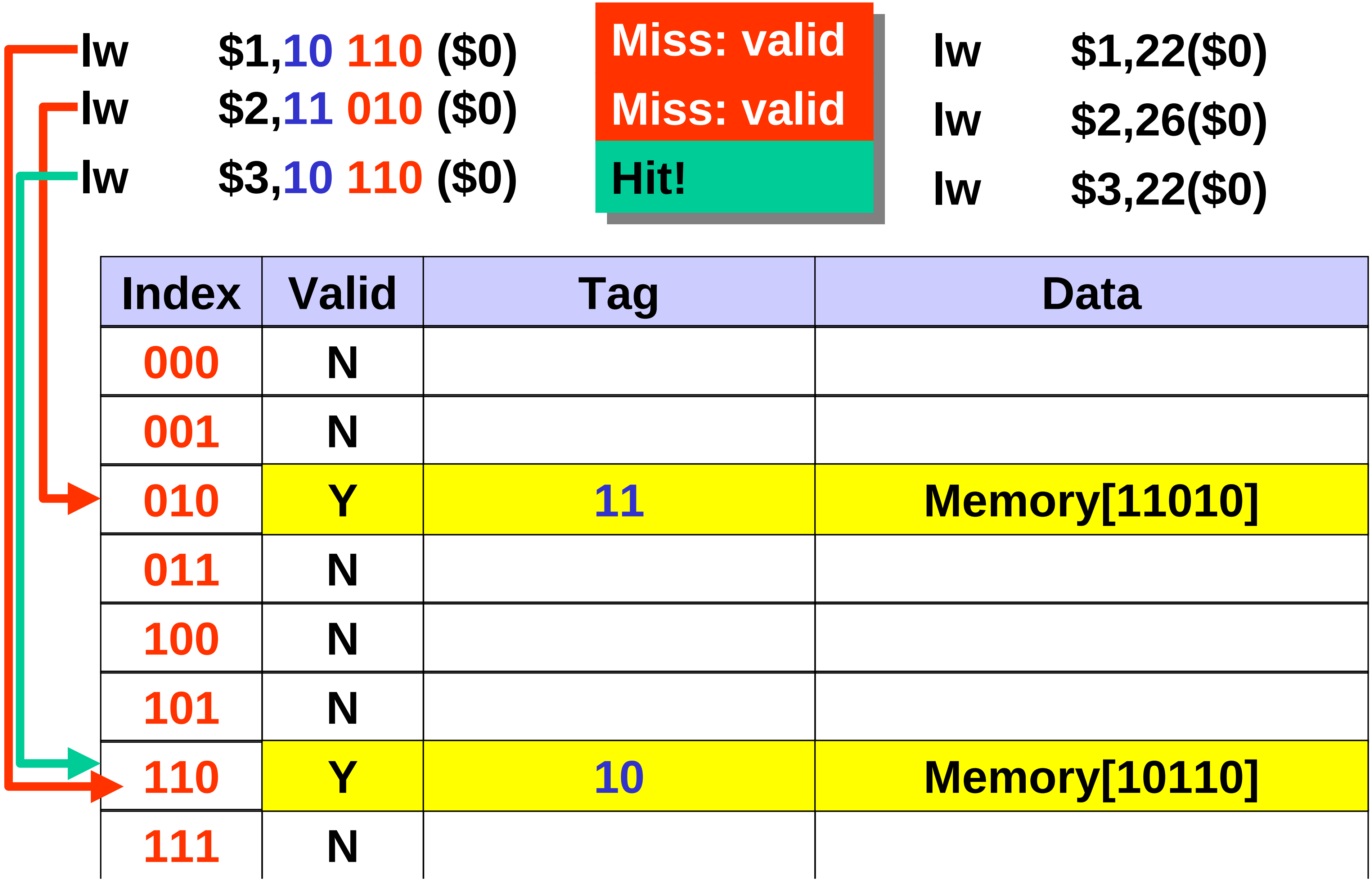
- contain the address information required to identify whether a word in the cache corresponds to the requested word.
- tags need only to contain the upper portion of the memory address (often referred to as a page address)

valid bit

- indicates whether an entry contains a valid address

Direct Mapped Cache: Temporal Example

Figure 7.6



lw $1,10 110 ($0)

lw $2,11 010 ($0)

lw $3,10 110 ($0)

Miss: valid

Miss: valid

Hit!

lw $1,22($0)

lw $2,26($0)

lw $3,22($0)

Index	Valid	Tag	Data
000	N		
001	N		
010	Y	11	Memory[11010]
011	N		
100	N		
101	N		
110	Y	10	Memory[10110]
111	N		

Direct Mapped Cache: Worst case, always miss!

Figure 7.6

lw $1,10 110 ($0)

lw $2,11 110 ($0)

lw $3,00 110 ($0)

Miss: valid

Miss: tag

Miss: tag

lw $1,22($0)

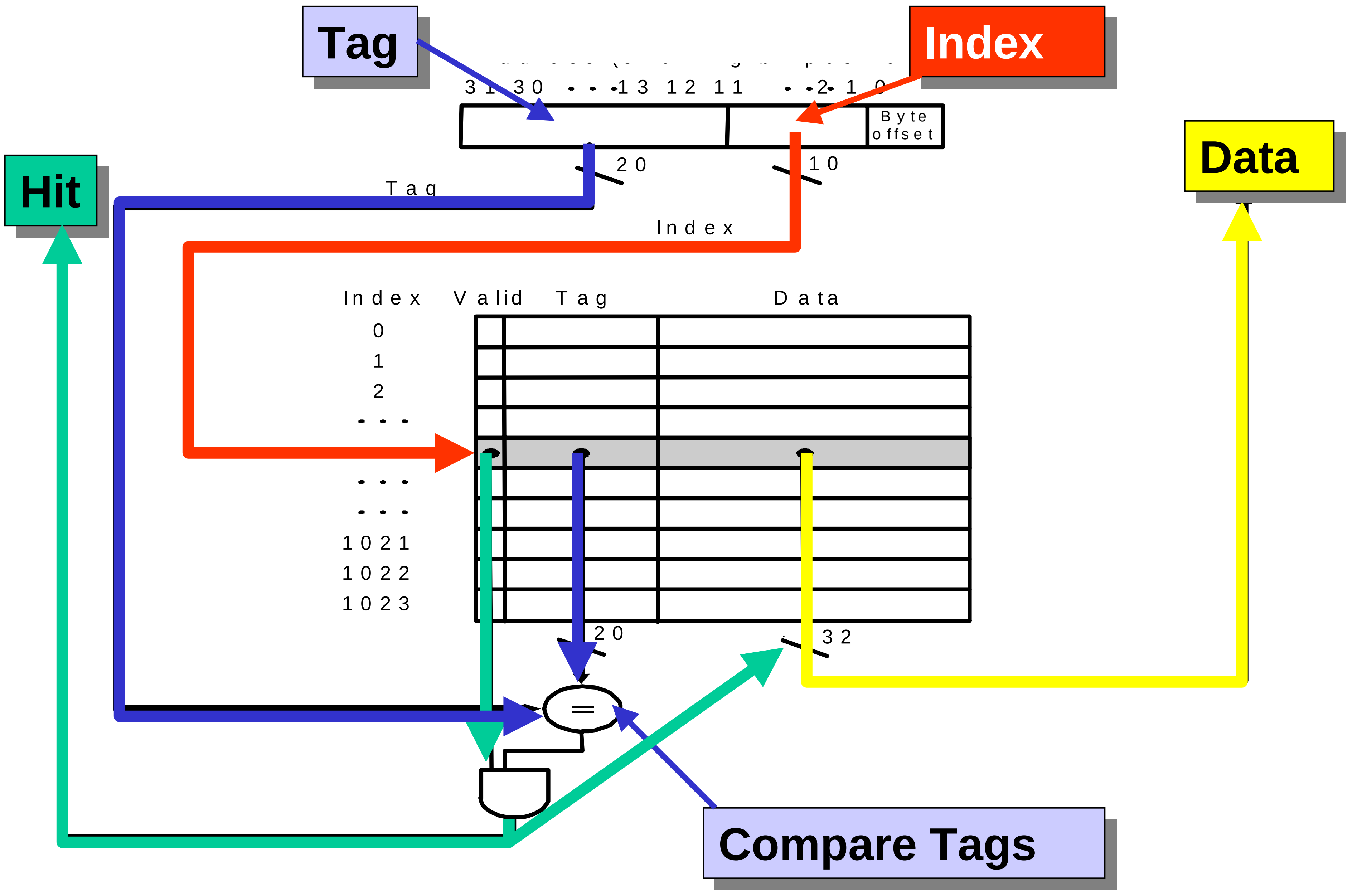
lw $2,30($0)

lw $3,6($0)

Index	Valid	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	00	Memory[00110]
111	N		

Direct Mapped Cache: Mips Architecture

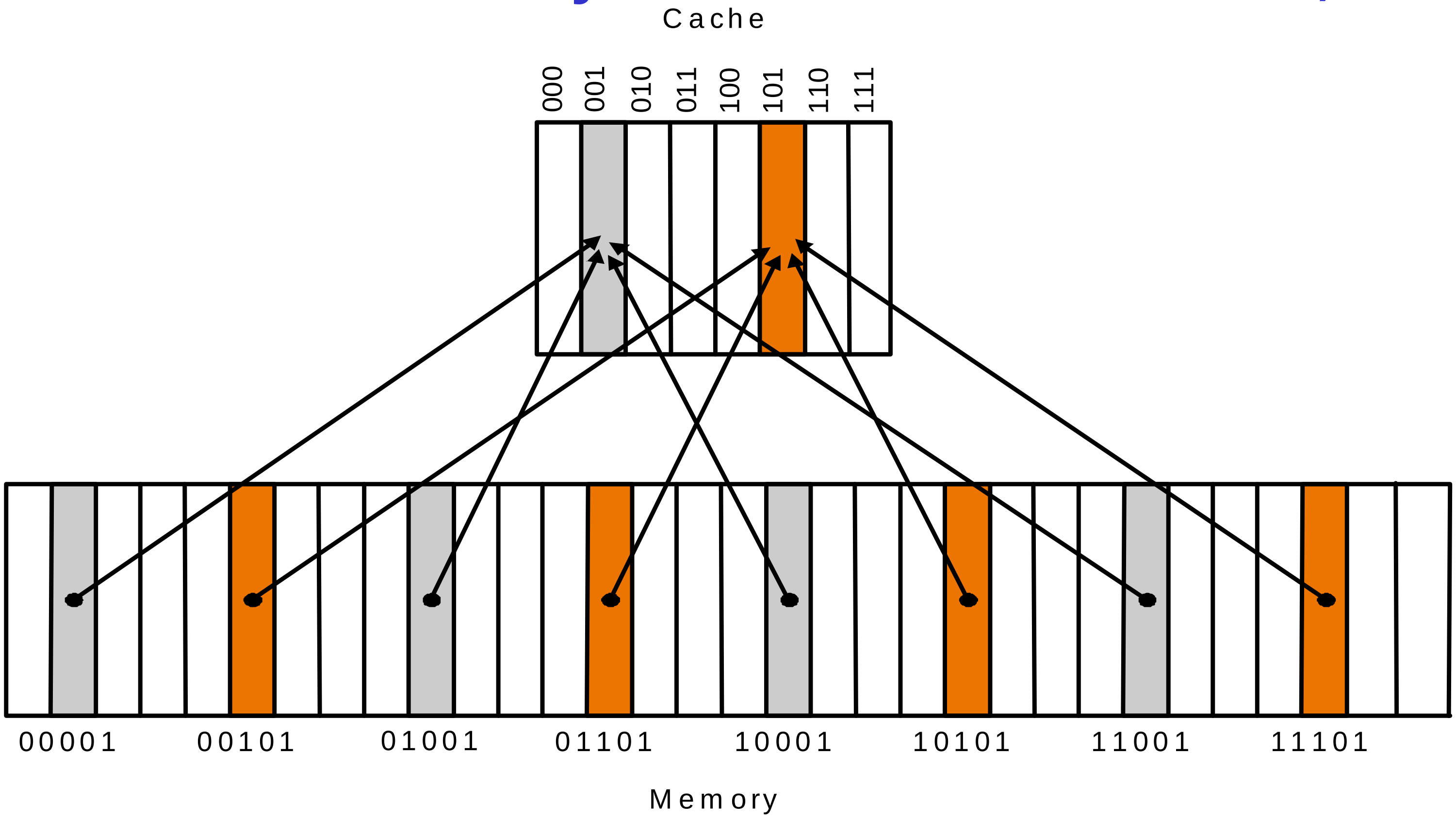
Figure 7.7



Direct Mapped Cache

- **Direct Mapped:** assign the cache location based on the address of the word in memory

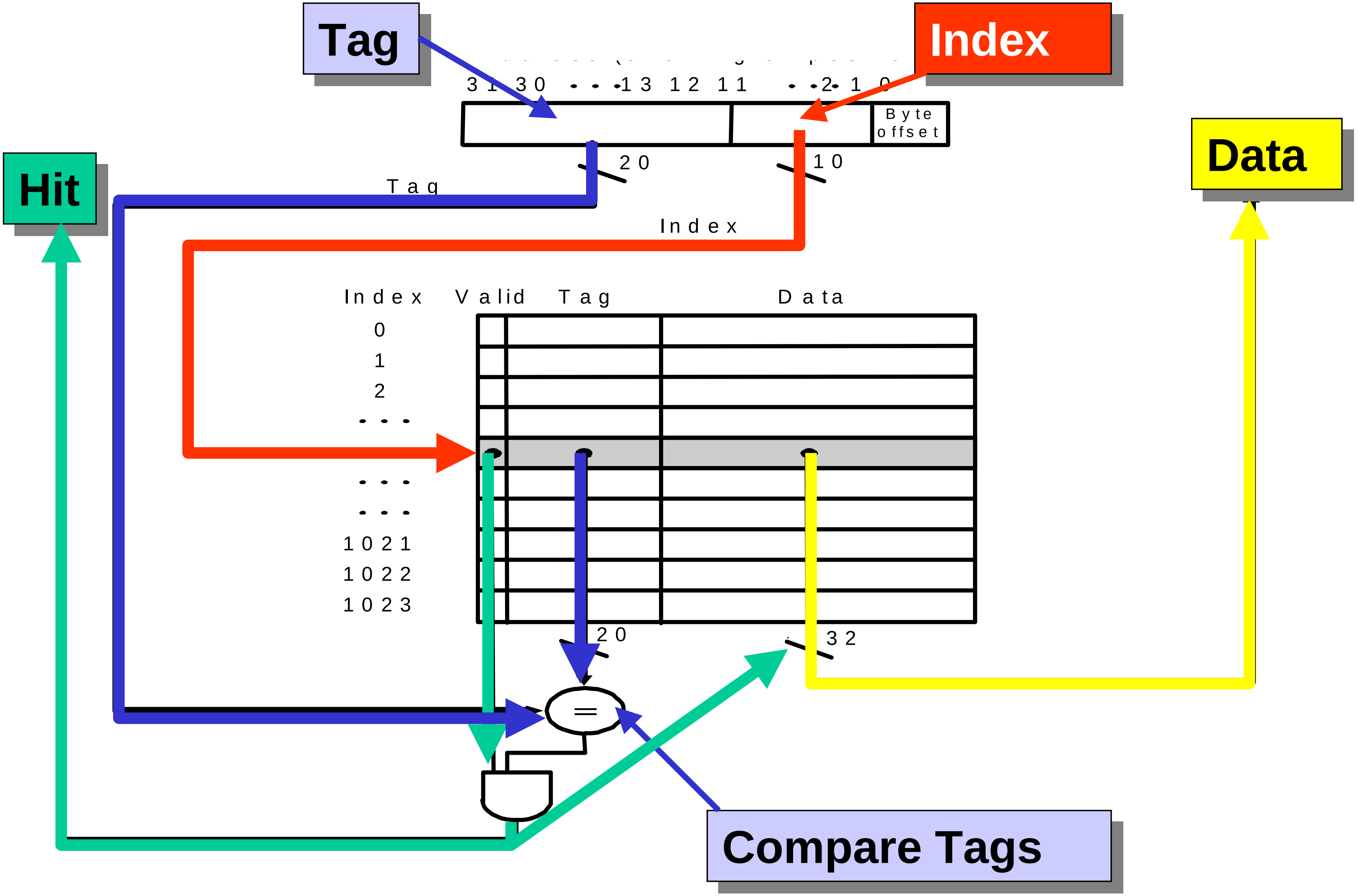
- **cache_address = memory_address % cache_size;**



Observe there is a Many-to-1 memory to cache relationship

Direct Mapped Cache: Mips Architecture

Figure 7.7



Bits in a Direct Mapped Cache

How many total bits are required for a direct mapped cache
with 64KB (= 2^{16} KiloBytes) of data
and one word (=32 bit) blocks
assuming a 32 bit byte memory address?

Cache index width = $\log_2$ words
= $\log_2 2^{16}/4$ = $\log_2 2^{14}$ words = 14 bits

Block address width = <byte address width> – $\log_2$ word
= 32 – 2 = 30 bits

Tag size = <block address width> – <cache index width>
= 30 – 14 = 16 bits

Cache block size = <valid size>+<tag size>+<block data size>
= 1 bit + 16 bits + 32 bits = 49 bits

Total size = <Cache word size> × <Cache block size>
= 2^{14} words × 49 bits = 784 × 2^{10} = 784 Kbits = 98 KB
= 98 KB/64 KB = 1.5 times overhead

The DECStation 3100 cache

write-through cache

Always write the data into both the cache and memory and then wait for memory.

DECStation uses a write-through cache

- 128 KB total cache size (=32K words)
  - = 64 KB instruction cache (=16K words)
  - + 64 KB data cache (=16K words)
- 10 processor clock cycles to write to memory

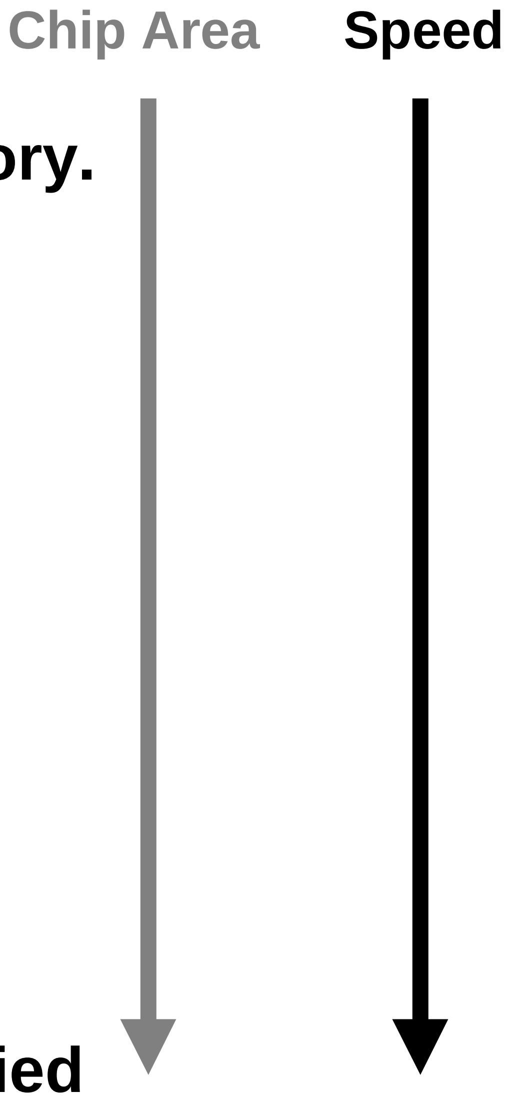
In a gcc benchmark, 13% of the instructions are stores.

- Thus, CPI of 1.2 becomes 1.2+13%x10 = 2.5
- Reduces the performance by more than a factor of 2!

Cache schemes

write-through cache

Always write the data into both the cache and memory and then wait for memory.



write buffer

write data into cache and write buffer.
If write buffer full processor must stall.

No amount of buffering can help
if writes are being generated faster
than the memory system can accept them.

write-back cache

Write data into the cache block and
only write to memory when block is modified
but complex to implement in hardware.

Hits vs. Misses

- Read hits
  - ✓ this is what we want!
- Read misses
  - ✓ stall the CPU, fetch block from memory, deliver to cache, and restart.
- Write hits
  - ✓ write-through: can replace data in cache and memory.
  - ✓ write-buffer: write data into cache and buffer.
  - ✓ write-back: write the data only into the cache.
- Write misses
  - ✓ read the entire block into the cache, then write the word.

The DECStation 3100 miss rates

Figure 7.9

- A split instruction and data cache increases the bandwidth

Benchmark Program	gcc	spice
Instruction miss rate	6.1%	1.2%
Data miss rate	2.1%	1.3%
Effective split miss rate	5.4%	1.2%
Combined miss rate	4.8%	

Why a lower miss rate?

Numerical programs tend to consist of a lot of small program loops

split cache has slightly worse miss rate

Spatial Locality

- Temporal only cache

  cache block contains only one word (No spatial locality).

- Spatial locality

  Cache block contains multiple words.

  - When a miss occurs, then fetch multiple words.
  - Advantage

    Hit ratio increases because there is a high probability that the adjacent words will be needed shortly.

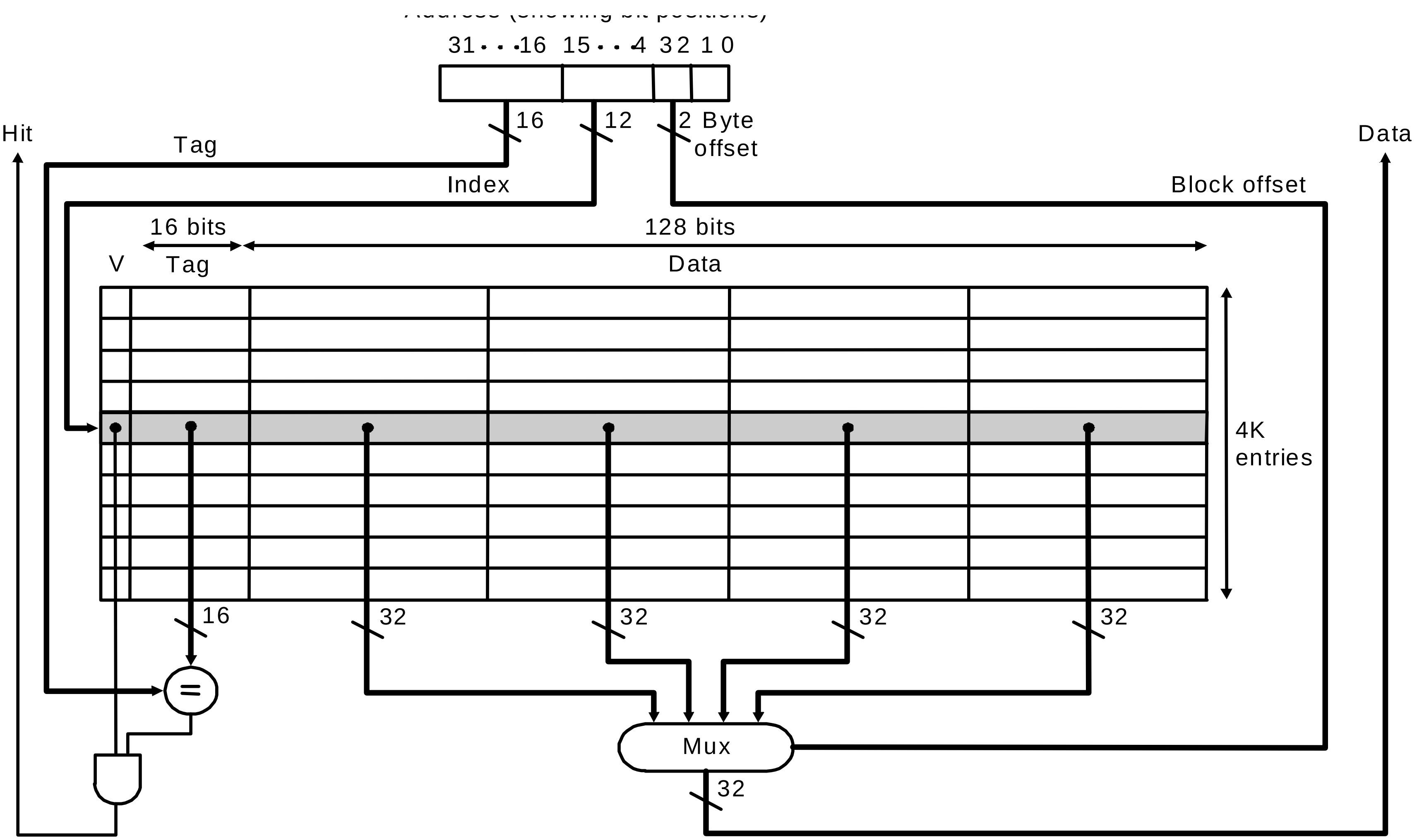
  - Disadvantage

    Miss penalty increases with block size

Spatial Locality: 64 KB cache, 4 words

Figure 7.10

- 64KB cache using four-word (16-byte word)
- 16 bit tag, 12 bit index, 2 bit block offset, 2 bit byte offset.



Performance

Figure 7.11

- Use split caches because there is more spatial locality in code:

Program Block size	gcc =1	gcc =4	spice =1	spice =4
Instruction miss rate	6.1%	2.0%	1.2%	0.3%
Data miss rate	2.1%	1.7%	1.3%	0.6%
Effective split miss rate	5.4%	1.9%	1.2%	0.4%
Combined miss rate	4.8%	4.8%		

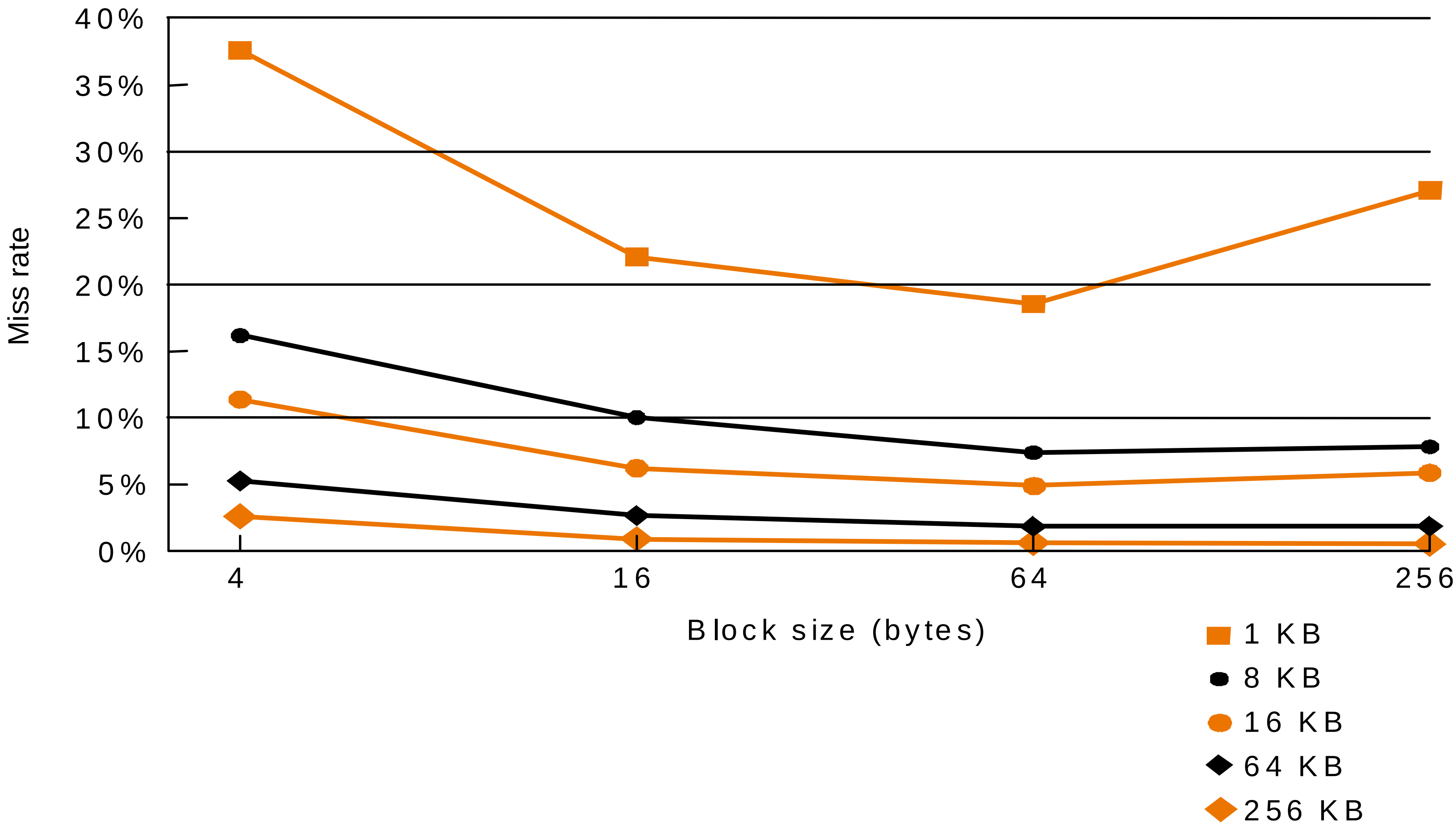
Temporal only split cache: has slightly worse miss rate

Spatial split cache: has lower miss rate

Cache Block size Performance

Figure 7.12

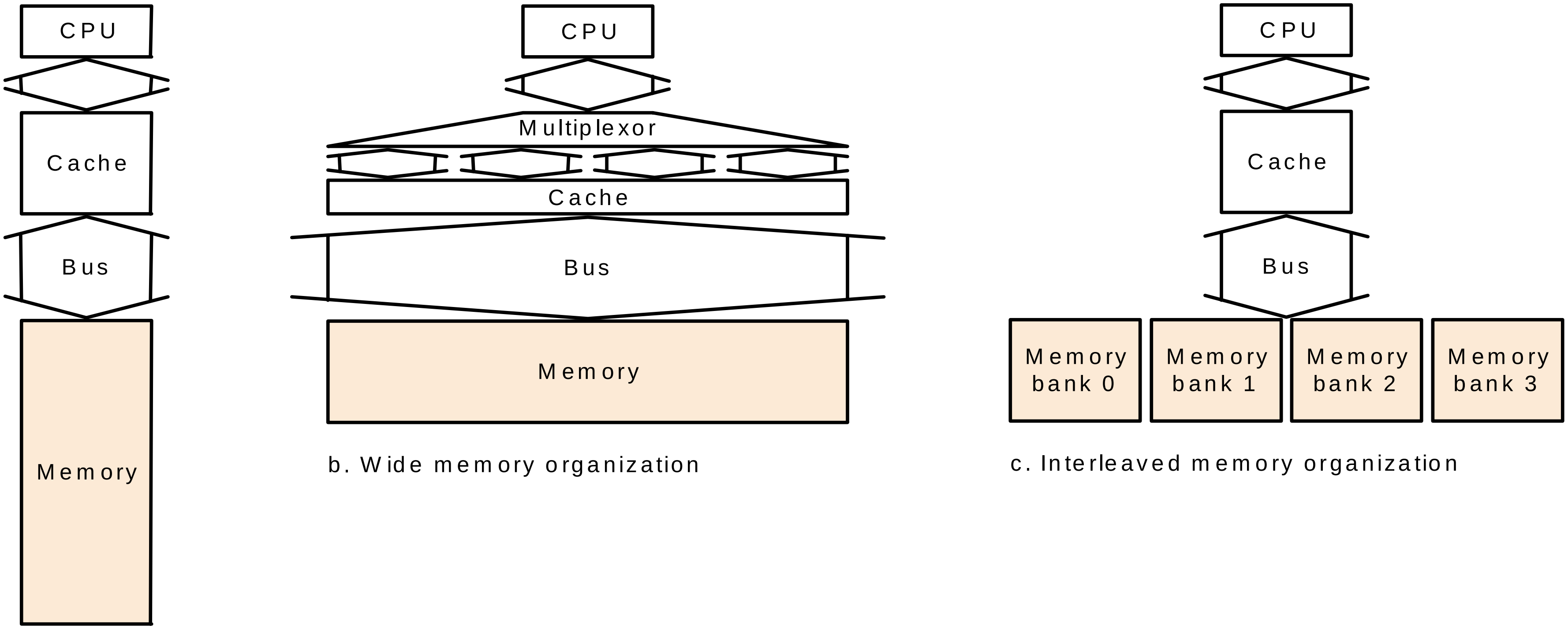
- **Increasing the block size tends to decrease miss rate:**



Designing the Memory System

Figure 7.13

- **Make reading multiple words easier by using banks of memory**



b. Wide memory organization

c. Interleaved memory organization

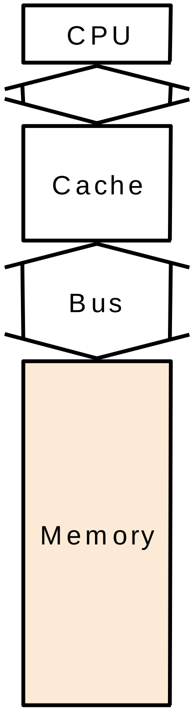
a. One-word-wide memory organization

1-word-wide memory organization

Figure 7.13

Suppose we have a system as follows

- 1-word-wide memory organization
- 1 cycle to send the address
- 15 cycles to access DRAM
- 1 cycle to send a word of data



a. One-word-wide memory organization

If we have a cache block of 4 words

Then the miss penalty is

=(1 address send) + 4×(15 DRAM reads)+4×(1 data send)

= 65 clocks per block read

Thus the number of bytes transferred per clock cycle

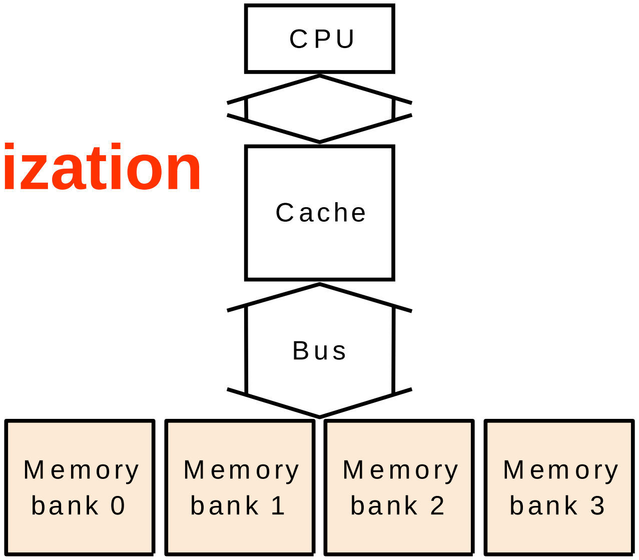
= 4 bytes/word x 4 words/65 clocks = 0.25 bytes/clock

Interleaved memory organization

Figure 7.13

Suppose we have a system as follows

- 4-bank memory interleaving organization
- 1 cycle to send the address
- 15 cycles to access DRAM
- 1 cycle to send a word of data



c. Interleaved memory organization

If we have a cache block of 4 words

Then the miss penalty is

= (1 address send) + 1×(15 DRAM reads)+ 4×(1 data send)

= 20 clocks per block read

Thus the number of bytes transferred per clock cycle

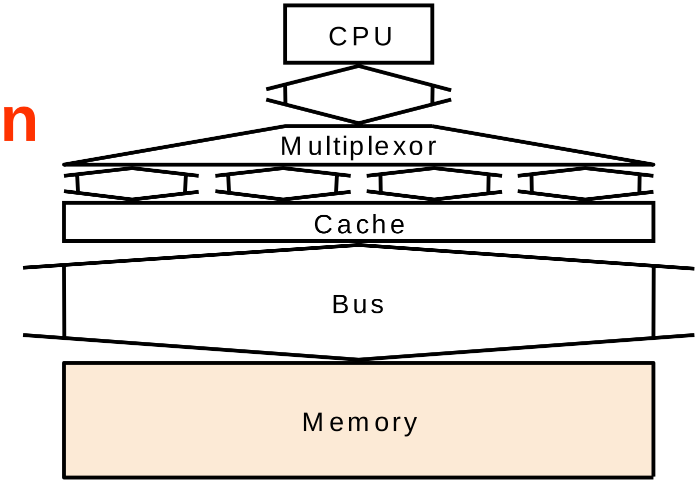
= 4 bytes/word x 4 words/17 clocks = 0.80 bytes/clock

we improved from 0.25 to 0.80 bytes/clock!

Wide bus: 4-word-wide memory organization Figure 7.13

Suppose we have a system as follows

- 4-word-wide memory organization
- 1 cycle to send the address
- 15 cycles to access DRAM
- 1 cycle to send a word of data



b. Wide memory organization

If we have a cache block of 4 words

Then the miss penalty is

= (1 address send) + 1×(15 DRAM reads)+ 1×(1 data send)

= 17 clocks per block read

Thus the number of bytes transferred per clock cycle

= 4 bytes/word x 4 words/17 clocks = 0.94 bytes/clock

we improved from 0.25 to 0.80 to 0.94 bytes/clock!

Memory organizations

Figure 7.13

Chip Area Speed

One word wide memory organization

Advantage

Easy to implement, low hardware overhead

Disadvantage

Slow: 0.25 bytes/clock transfer rate

Interleave memory organization

Advantage

Better: 0.80 bytes/clock transfer rate

Banks are valuable on writes: independently

Disadvantage

more complex bus hardware

Wide memory organization

Advantage

Fastest: 0.94 bytes/clock transfer rate

Disadvantage

Wider bus and increase in cache access time

Decreasing miss penalty with multilevel caches Page 576

Suppose we have a processor with

CPI	= 1.0
Clock Rate	= 500 Mhz = 2 ns
L1 Cache Miss rate	= 5%
DRAM	= 200 ns

How mach faster will the machine will be if we add a

L2 Cache	= 20 ns (hit time = miss penalty)
L1 Cache Miss rate	= 2%

$$\text{L to M Miss Penalty} = \frac{200\,\text{ns}}{2\,\text{ns per Clock Cycle}} = 100\,\text{Clock Cycles}$$

$$\text{L1 to L2 Miss Penalty} = \frac{20\,\text{ns}}{2\,\text{ns per Clock Cycle}} = 10\,\text{Clock Cycles}$$