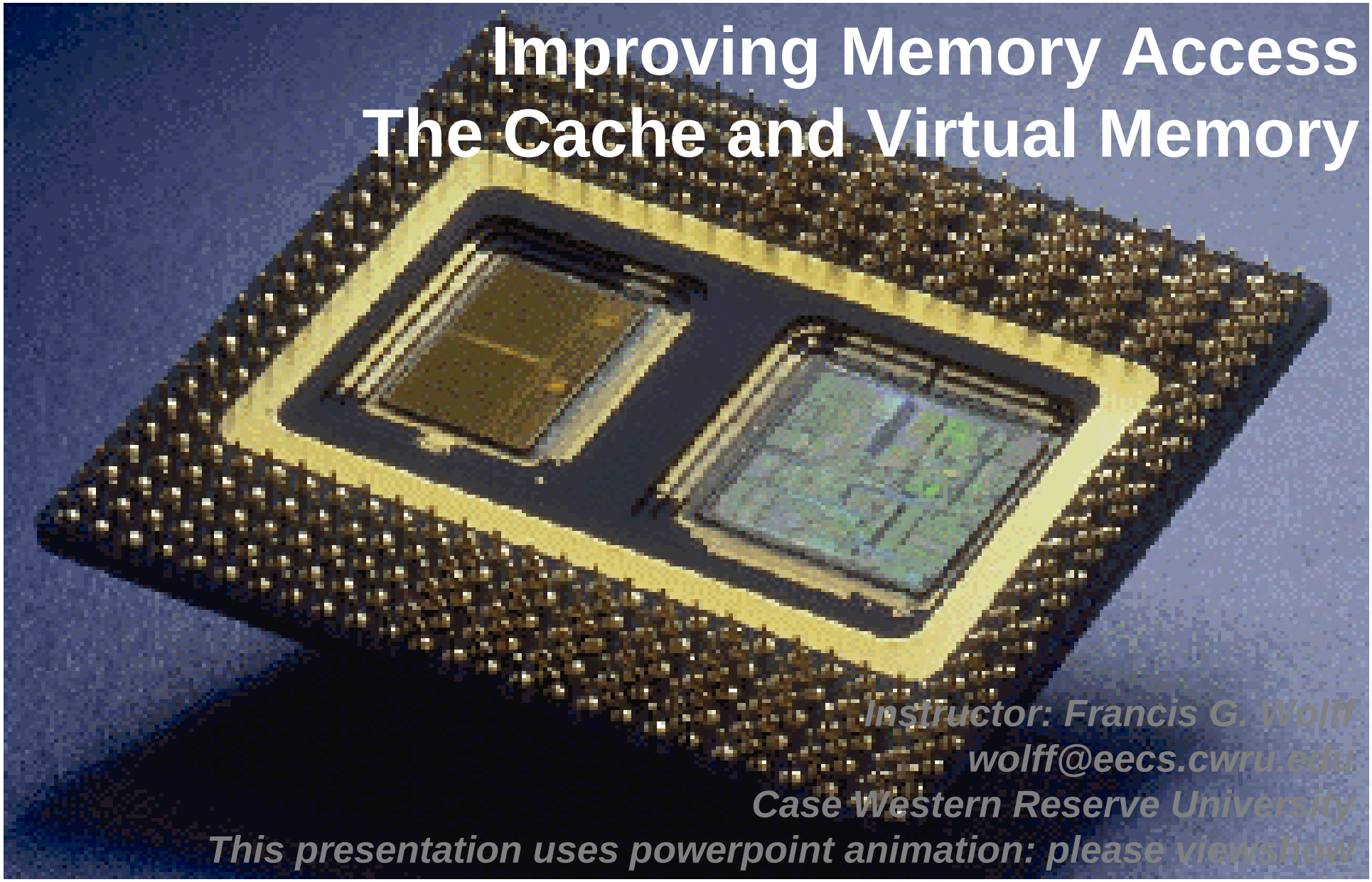
EECS 322 Computer Architecture



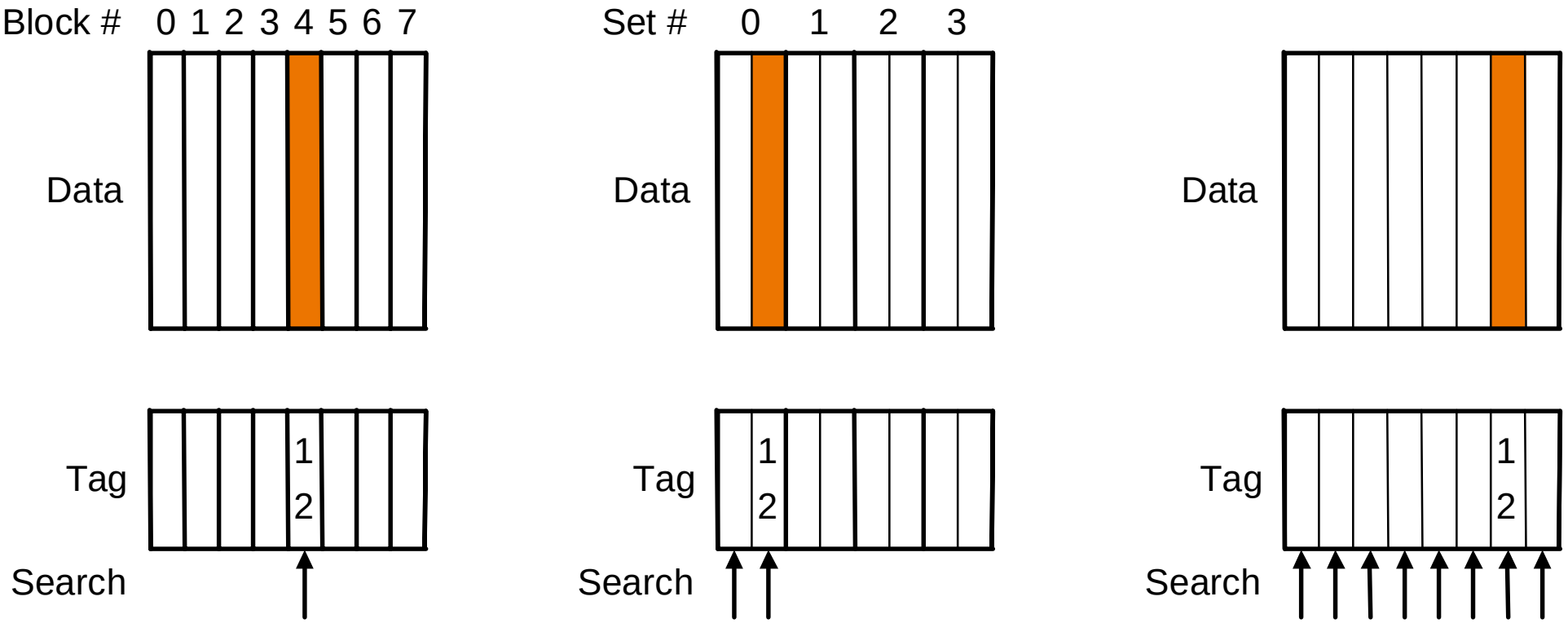
Improving Memory Access
The Cache and Virtual Memory

Instructor: Francis G. Wolff
wolff@eecs.cwru.edu
Case Western Reserve University

This presentation uses powerpoint animation: please viewshow

Cache associativity

Figure 7.15



Direct-mapped cache
(a.k.a. 1-way set associative cache)

2-way set associative cache

Fully associative cache

Cache architectures

Figure 7.16

Two-way set associative

Set	Tag	Data	Tag	Data
0				
1				
2				
3				

One-way set associative
(direct mapped)

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

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

Direct Mapped Cache: Example

Tag | Index | Byte offset | Byte address

lw	$1, 000 00 00($0)	Miss: valid	lw $1,0($0)
lw	$2, 010 00 00($0)	Miss: tag	lw $2,32($0)
lw	$3, 000 00 00($0)	Miss: tag	lw $3,0($0)
lw	$4, 001 10 00($0)	Miss: valid	lw $4,24($0)
lw	$5, 010 00 00($0)	Miss: tag	lw $5,32($0)

5/5 Misses

Index	Valid	Tag	Data
00	Y	010	Memory[010 00 00]
01	N		
10	Y	001	Memory[001 10 00]
11	N		

2-Way Set Associative Cache: Example

Tag | Index | Byte offset | Byte address

lw	$1, 0000 0 00($0)	Miss: valid	lw	$1,0($0)
lw	$2, 0100 0 00($0)	Miss: tag	lw	$2,32($0)
lw	$3, 0000 0 00($0)	Hit!	lw	$3,0($0)
lw	$4, 0011 0 00($0)	Miss: tag	lw	$4,24($0)
lw	$5, 0100 0 00($0)	Miss: tag	lw	$5,32($0)
		4/5 Misses		

Index	Valid	Tag	Data
0	Y	0100	Memory[0100 0 00]
	Y	0011	Memory[0011 0 00]
1	N		
	N		

Fully Associative Cache: Example

Tag | Index | Byte offset | Byte address

lw	$1, 00000 00($0)	Miss: valid	lw $1,0($0)
lw	$2, 01000 00($0)	Miss	lw $2,32($0)
lw	$3, 00000 00($0)	Hit!	lw $3,0($0)
lw	$4, 00110 00($0)	Miss	lw $4,24($0)
lw	$5, 01000 00($0)	Hit!	lw $5,32($0)
		3/5 Misses	

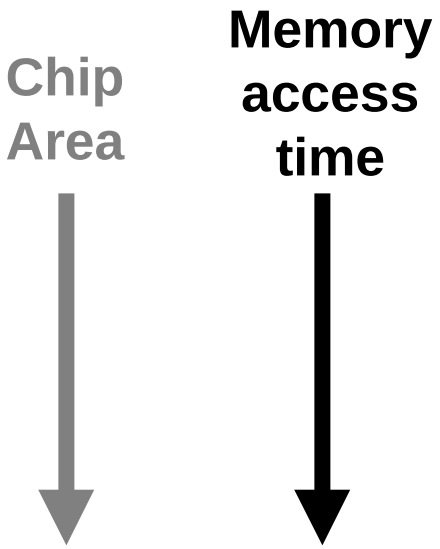
Valid	Tag	Data
Y	00000	Memory[00000 00]
Y	01000	Memory[01000 00]
Y	00110	Memory[00110 00]
N		

Cache comparisons: page 571 Example

Example (page 571): 5 loads

```
lw    $1,0($0)
lw    $2,32($0)
lw    $3,0($0)
lw    $4,24($0)
lw    $5,32($0)
```

Cache architecture	misses
Direct Mapped	5 out of 5
2-Way Set associative	4 out of 5
Fully associative	3 out of 5



Decreasing the number of misses, speeds up the access time

Bits in 64 KB Data Cache Summary

Cache Type	Index Size	Tag Size	Set Size	Over-head	Miss Rate
Temporial Direct Mapped	14 bits	16 bits	49 bits	1.5%	5.4%
Spatial Direct Mapped	12 bits	16 bits	145 bits	1.13%	1.9%
2 - Way Set Associative	11 bits	17 bits	311 bits	1.21%	1.5%
4 - Way Set Associative	10 bits	18 bits	585 bits	1.14%	1.5%
Fully Associative	0 bits	28 bits	157 bits	1.23%	

Decreasing miss penalty with multilevel caches

The effective CPI with only L1 to M cache

Total CPI = Base CPI + (Memory Stall cycles per Instruction)

= 1.0 + 5% × 100ns = 6.0

The effective CPI with L1 and L2 caches

Total CPI = Base CPI + (L1 to L2 penalty) + (L2 to M penalty)

= 1.0 + (5% × 10ns) + (2% × 100ns) = 3.5

$$\text{CPU Speedup} = \frac{6.0}{3.5} = 1.7$$

Direct Mapping Cache addresses

$$\text{Cache Index} = (\text{Block Address}) \ \% \ (\text{Cache Size})$$

$$\text{Cache Index} = \left\lfloor \frac{\text{Byte address}}{\textit{Data}\ \text{Bytes per cache block}} \right\rfloor \% \ (\text{Cache Size})$$

$$\text{Block address} = \left\lfloor \frac{\text{Byte address}}{\textit{Data}\ \text{Bytes per cache block}} \right\rfloor$$

$$\text{Cache Size in Blocks} = \frac{\text{Cache Size in Bytes}}{\text{Bytes per Cache Block}}$$

N-Way Set-Associative Mapping Cache addresses

An N-way set associative cache consists of a number of sets, where each set consists on N blocks

$$\textbf{Cache Index} = \textit{Set \#} = \textbf{(Block Address) \% (Cache Size)}$$

$$\textbf{Cache Index} = \left\lfloor \frac{\textbf{Byte address}}{\textit{Data}\ \textbf{Bytes per cache block}} \right\rfloor \textbf{\% (Cache Size)}$$

$$\textbf{Block address} = \left\lfloor \frac{\textbf{Byte address}}{\textit{Data}\ \textbf{Bytes per cache block}} \right\rfloor$$

$$\textbf{Cache Size in } N-\textit{way Sets} = \frac{\textbf{Cache Size in Bytes}}{N \times \textbf{Data Bytes per Cache Block}}$$

Direct Mapped Cache

Suppose we have the have the following byte addresses:

0, 32, 0, 24, 32

Given a direct mapped cache consisting of 16 data bytes

where Data bytes per cache block = 4

where Cache size = 16 bytes/(4 bytes per block) = 4

The cache block addresses are: 0, 8, 0, 6, 8

The cache index addresses are: 0, 0, 0, 2, 0

For example, 24,

Block address =floor(byte address / data bytes per block)

= floor(24/4)=6

Cache Index = 6 % Cache size = 6 % 4 = 2

2-Way Set associative cache

Again, suppose we have the have the following byte addresses:

0, 32, 0, 24, 32

Given a 2-way set associative cache of 16 data bytes

where Data bytes per cache block = 4

where Cache size = 16 bytes/(2 × 4 bytes per block) = 2

The 2-way cache block addresses are: 0, 8, 0, 6, 8

The 2-way cache index addresses are: 0, 0, 0, 0, 0

Note: Direct cache index addresses were: 0, 0, 0, 2, 0

For example, 24,

Block address =floor(byte address / data bytes per block)

= floor(24/4) = 6

Cache Index = 6 % Cache size = 6 % 2 = 0

Fully associative cache

Again, suppose we have the have the following byte addresses:

0, 32, 0, 24, 32

Given a 2-way set associative cache of 16 data bytes

where Data bytes per cache block = 4

where Cache size = 16 bytes/(4 bytes per block) = 4

The fully cache block addresses are: None!

The 2-way cache index addresses are: NONE!

Bits in a Temporal Direct Mapped Cache

How many total bits are required for a cache with 64KB (= 2^{16} KiloBytes) of data and one word (=4 bytes =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

Bits in a Spatial Direct Mapped Cache

How many total bits are required for a cache with 64KB (= 2^{16} KiloBytes) of data and 4 - one word (=4*4=16 bytes =4*32=128 bits) blocks assuming a 32 bit byte memory address?

Cache index width = $\log_2$ words

= $\log_2 2^{16}/16 = \log_2 2^{12}$ words = 12 bits

Block address width = <byte address width> – $\log_2$ blocksize

= 32 – 4 = 28 bits

Tag size = <block address width> – <cache index width>

= 28 – 12 = 16 bits

Cache block size = <valid size>+<tag size>+<block data size>

= 1 bit + 16 bits + 128 bits = 145 bits

Total size = <Cache word size> × <Cache block size>

= 2^{12} words × 145 bits = 593920 bits / (1024*8) = 72.5 Kbyte

= 72.5 KB/64 KB = 1.13 times overhead

Bits in a 2-Way Set Associative Cache

How many total bits are required for a cache with 64KB (= 2^{16} KiloBytes) of data and 4 - one word (=4*4=16 bytes =4*32=128 bits) blocks assuming a 32 bit byte memory address?

Cache index width = $\log_2$ words

= $\log_2 2^{16}/(2\times16)$ = $\log_2 2^{11}$ words = 11 bits

Block address width = <byte address width> – $\log_2$ blocksize

= 32 – 4 = 28 bits (same!)

Tag size = <block address width> – <cache index width>

= 28 – 11 = 17 bits

Cache Set size = <valid size>+<tag size>+<block data size>

= 1 bit + 2×(17 bits + 128 bits) = 311 bits

Total size = <Cache word size> × <Cache block size>

= 2^{11} words × 311 bits = 636928 bits / (1024*8) = 77.75 Kbyte

= 72.5 KB/64 KB = 1.21 times overhead

Bits in a 4-Way Set Associative Cache

How many total bits are required for a cache with 64KB (= 2^{16} KiloBytes) of data and 4 - one word (=4*4=16 bytes =4*32=128 bits) blocks assuming a 32 bit byte memory address?

Cache index width = $\log_2$ words

= $\log_2 2^{16}/(4\times16)$ = $\log_2 2^{10}$ words = 10 bits

Block address width = <byte address width> – $\log_2$ blocksize

= 32 – 4 = 28 bits (same!)

Tag size = <block address width> – <cache index width>

= 28 – 10 = 18 bits

Cache Set size = <valid size>+<tag size>+<block data size>

= 1 bit + 4×(18 bits + 128 bits) = 585 bits

Total size = <Cache word size> × <Cache block size>

= 2^{10} words × 585 bits = 599040 bits / (1024*8) = 73.125 Kbyte

= 73.125 KB/64 KB = 1.14 times overhead

Bits in a Fully Associative Cache

How many total bits are required for a cache with 64KB (= 2^{16} KiloBytes) of data and 4-one word (=4*4=16 bytes =4*32=128 bits) blocks assuming a 32 bit byte memory address?

Cache index width = 0

Block address width = <byte address width> – $\log_2$ blocksize
= 32 – 4 = 28 bits (same!)

Tag size = <block address width> – <cache index width>
= 28 – 0 = 28 bits

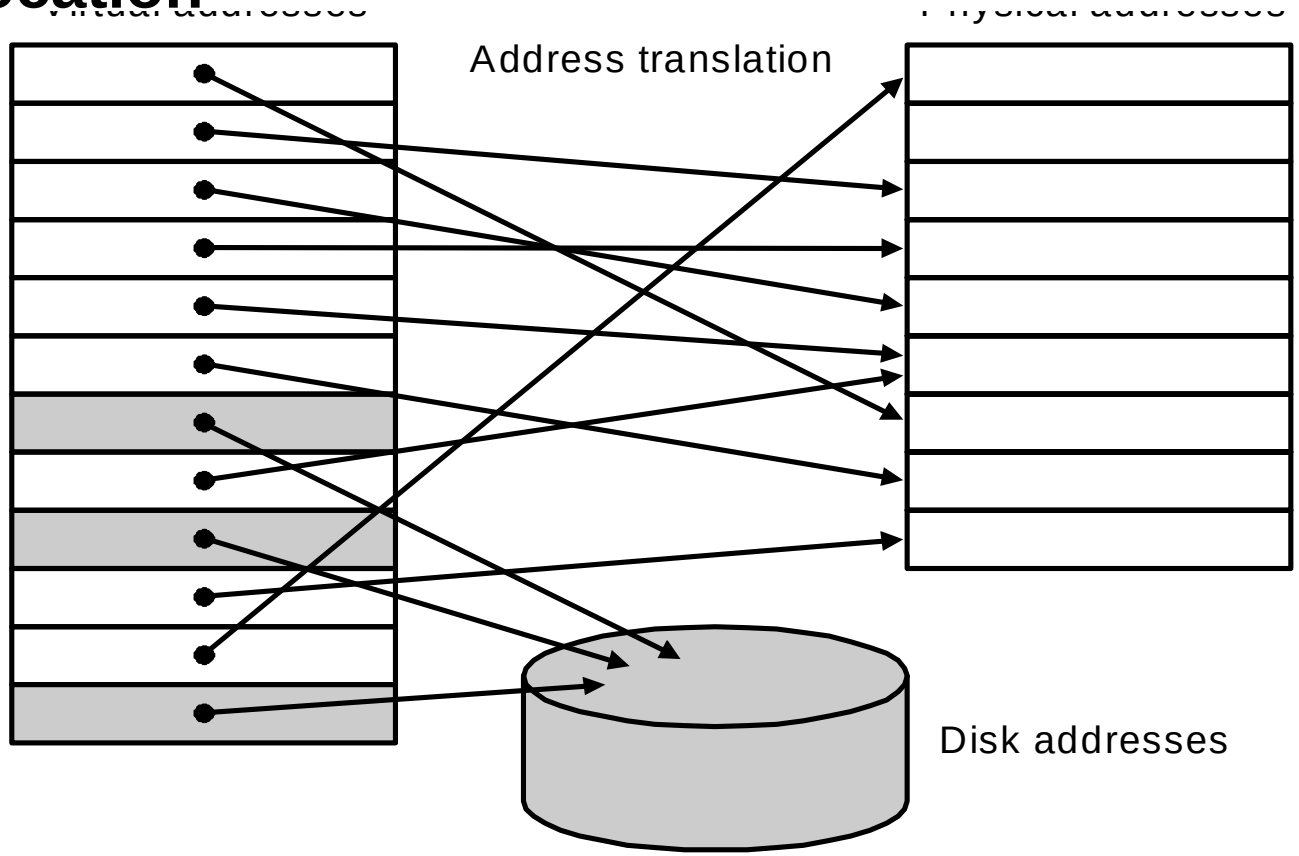
Cache Set size = <valid size>+<tag size>+<block data size>
= 1 bit + 28 bits + 128 bits = 157 bits

Total size = <Cache word size> × <Cache block size>
= 2^{12} words × 157 bits = 643072 bits / (1024*8) = 78.5 Kbyte
= 78.5 KB/64 KB = 1.23 times overhead

Virtual Memory

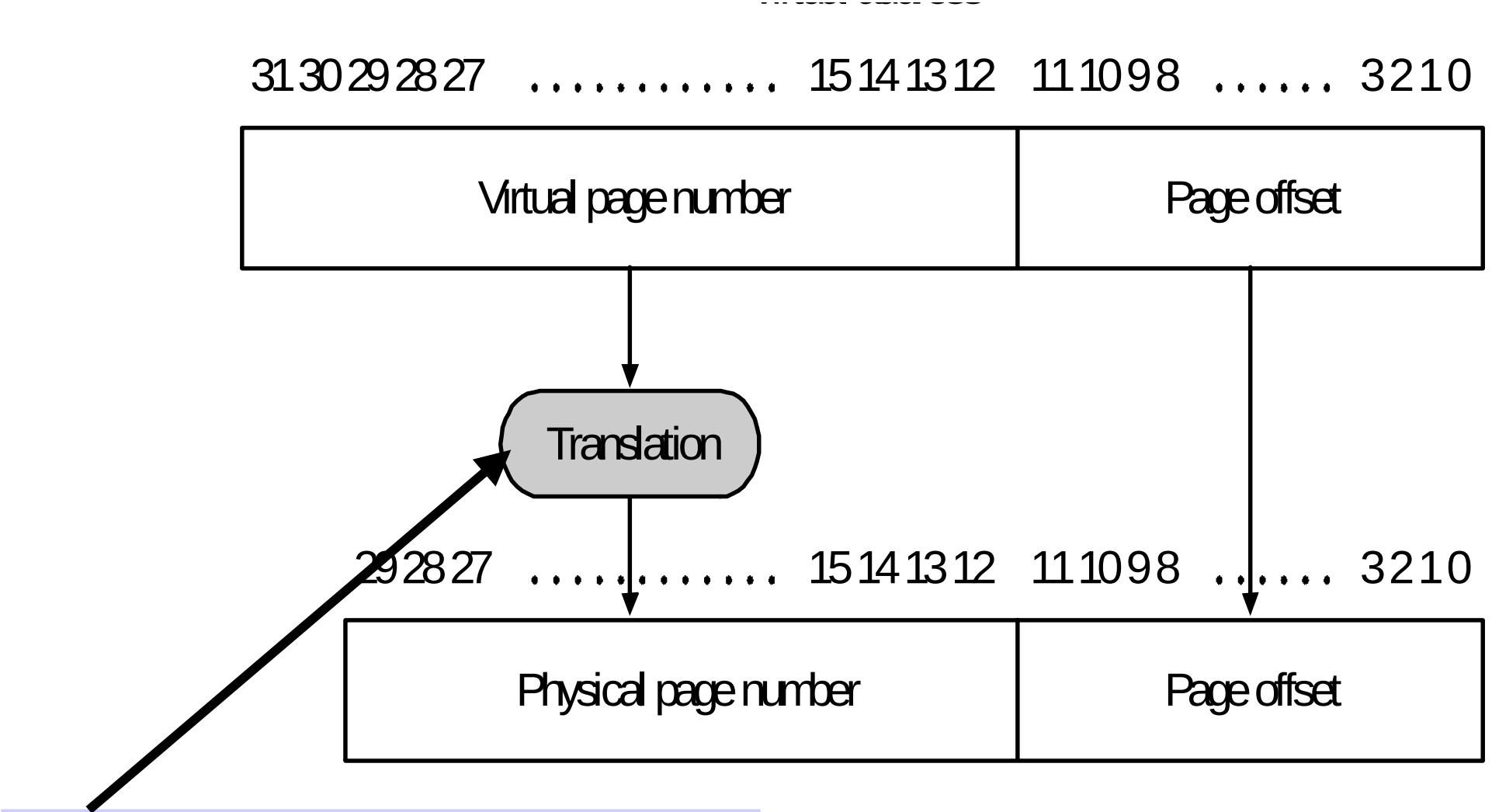
Figure 7.20

- **Main memory can act as a cache for the secondary storage (disk) Advantages:**
  - **illusion of having more physical memory**
  - **program relocation**
  - **protection**



Pages: virtual memory blocks

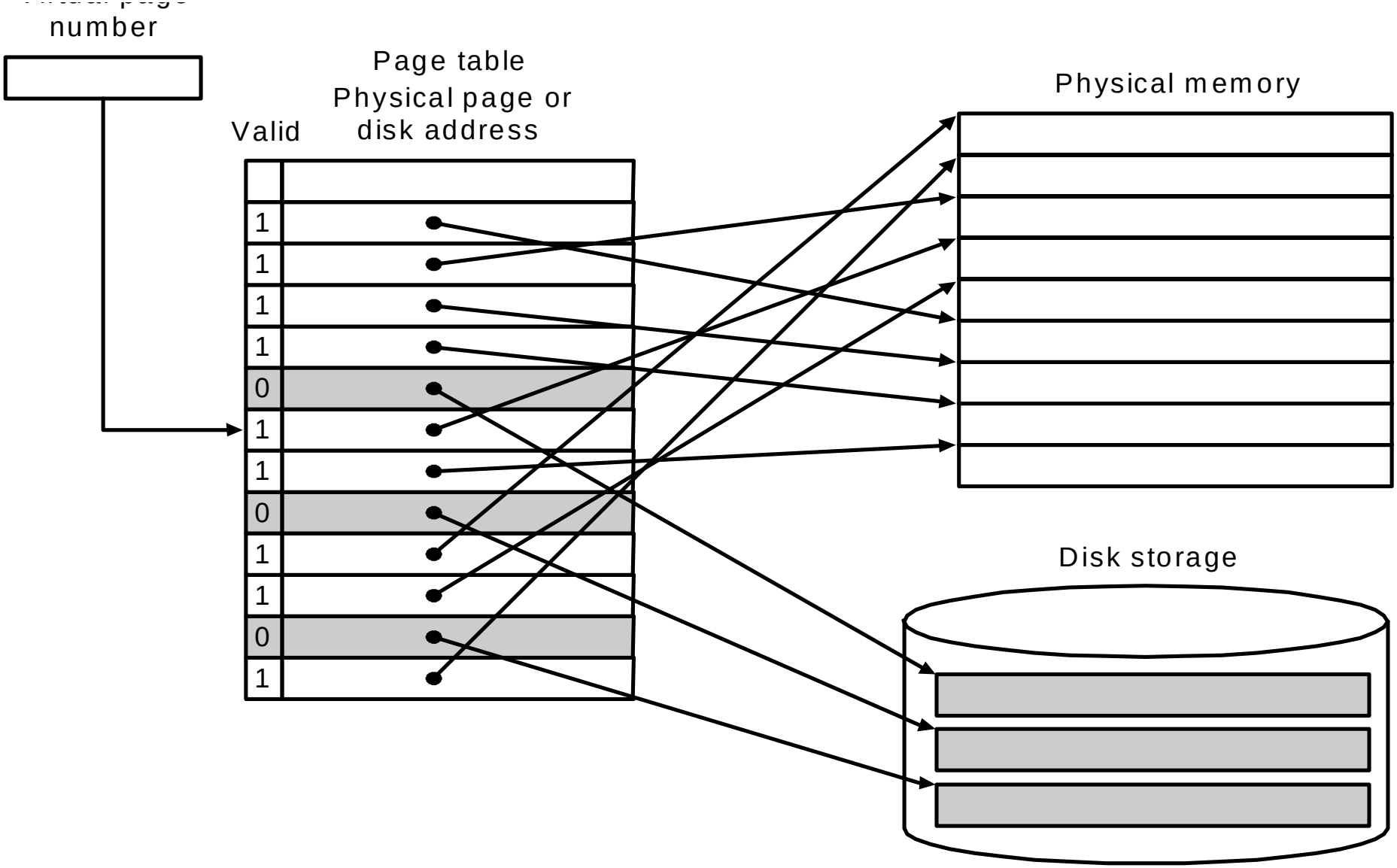
Figure 7.21



The automatic transmission of a car: Hardware/Software does the shifting

Page Tables

Figure 7.23



Virtual Example: 32 Byte pages

lw $2, 00100 0 00($0)
lw $2, 00101 0 00($0)

C-Miss P-Fault
C-Miss P- Hit!

lw $1,32($0)
lw $2,40($0)

Index	Valid	Tag	Data
0	Y	00101	Memory[000101 0 00]
1	N		

Virtual Page Number	Page offset
001	01000

Page Index	Valid	Physical Page number
0	N	
1	Y	111
...	N	

Memory

Disk

Page Table Size

Suppose we have

- 32 bit virtual address (=2^{32} bytes)
- 4096 bytes per page (=2^{12} bytes)
- 4 bytes per page table entry (=2^{2} bytes)

What is the total page table size?

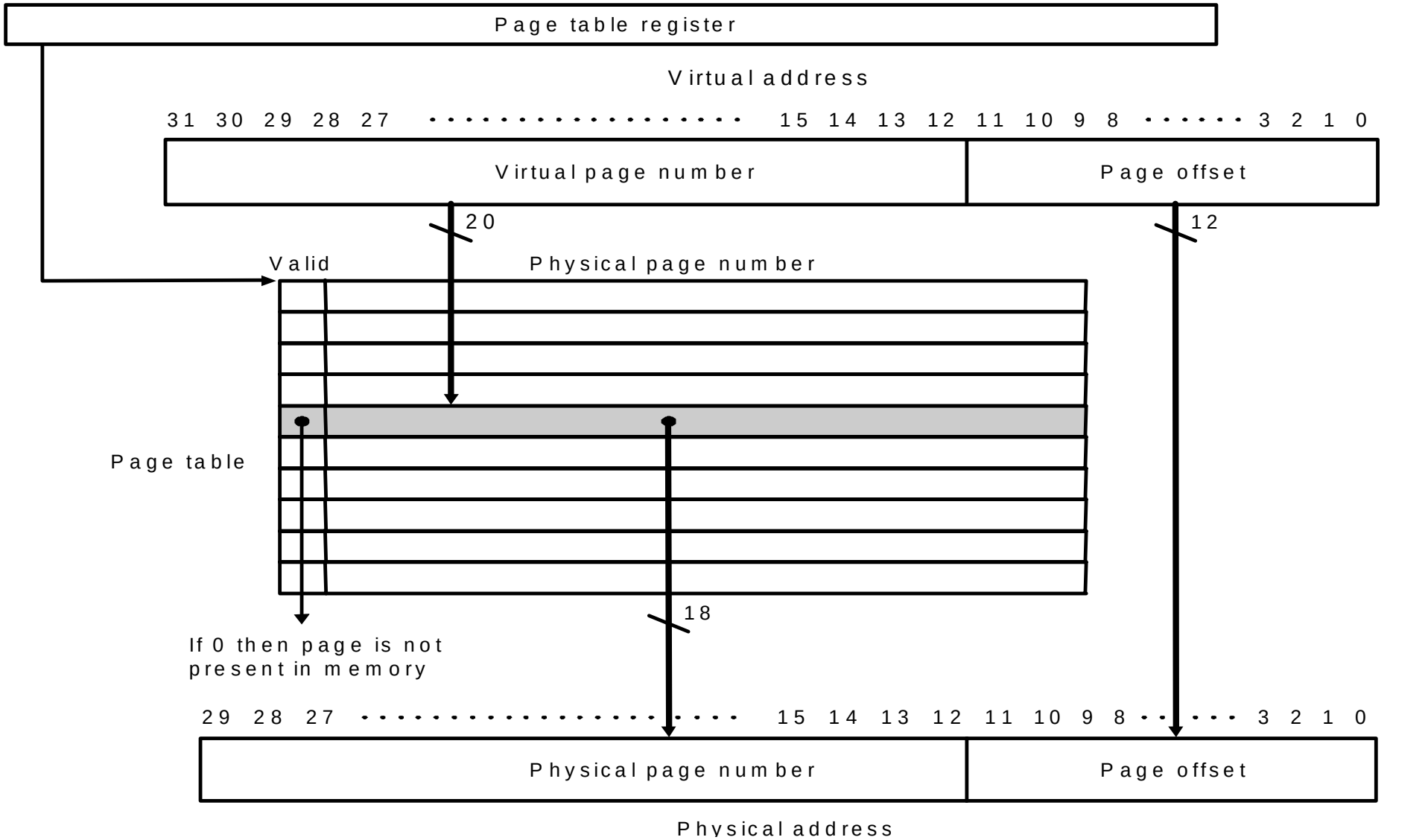
$$\text{Number of page table entries} = \frac{2^{32}}{2^{12}} = 2^{20} \text{ entries}$$

$$\text{Size of page table} = 2^{20} \text{ entries} \times 2^{2} \text{ bytes} = 2^{22} \text{ bytes} = 4\,\text{MB}$$

4 Megabytes just for page tables!! Too Big

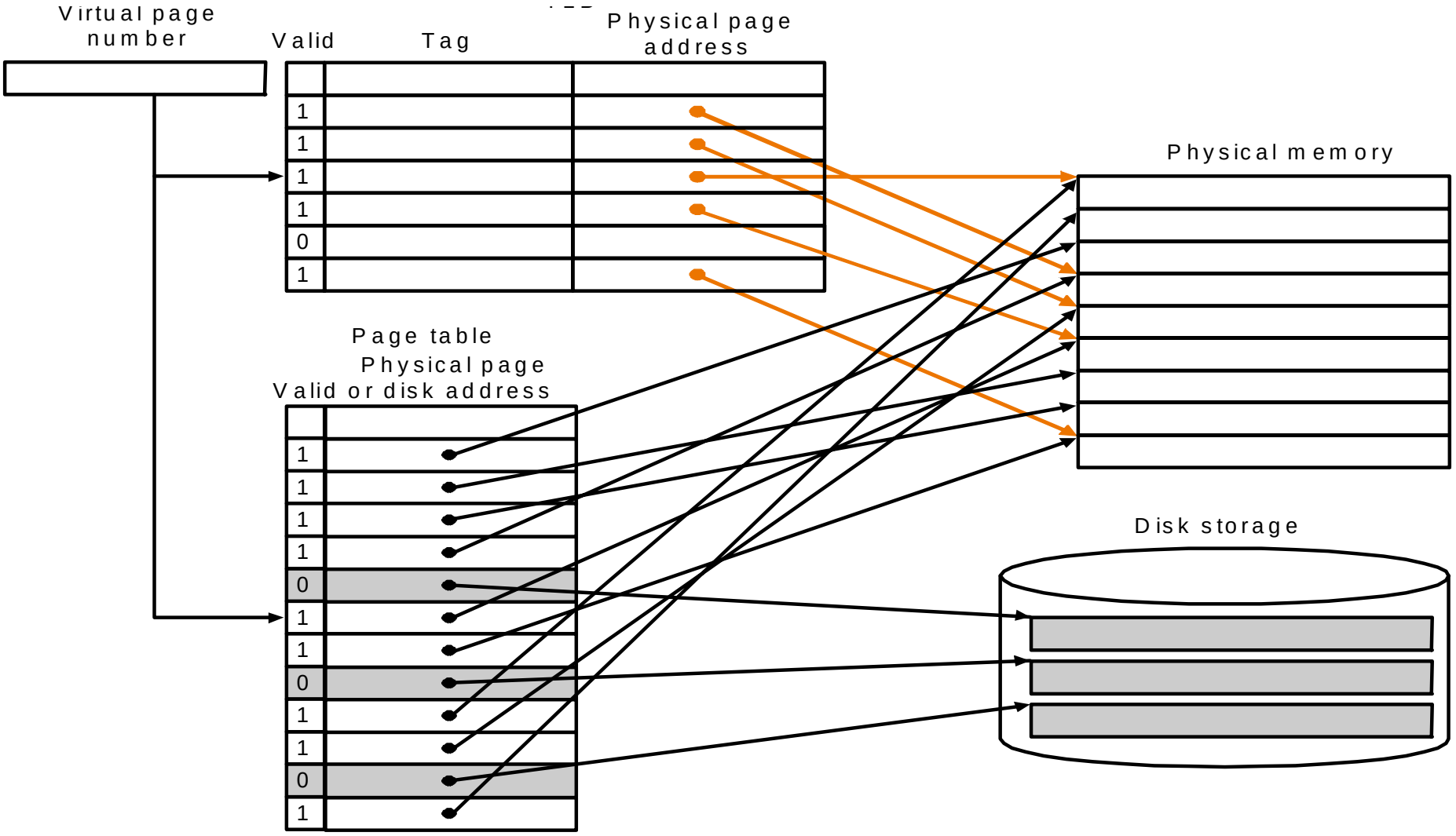
Page Tables

Figure 7.22



Making Address Translation Fast

- **A cache for address translations: translation lookaside buffer**



TLBs and caches