Chapter 8 :: Memory Systems

Digital Design and Computer Architecture

David Money Harris and Sarah L. Harris

Lectured by Jeong-Gun Lee @ Hallym

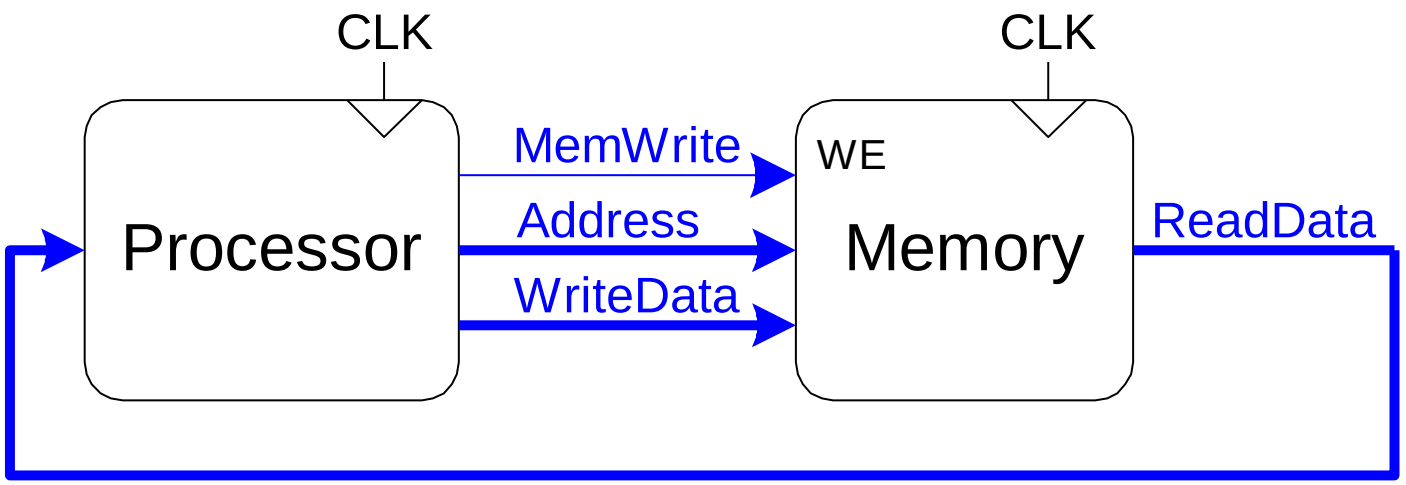
Chapter 8 :: Topics

- Introduction
- Memory System Performance Analysis
- Caches
- Virtual Memory
- Memory-Mapped I/O
- Summary

Introduction

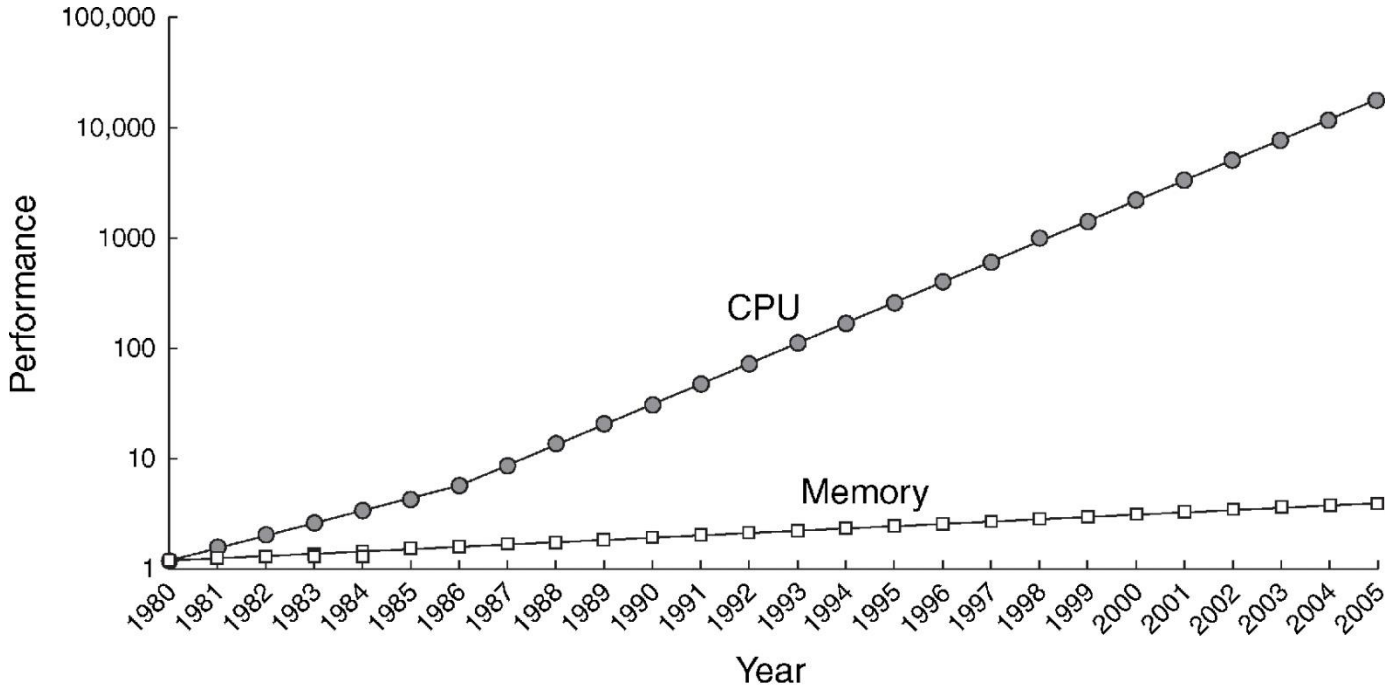
- Computer performance depends on:
  - Processor performance
  - Memory system performance

Memory Interface



Introduction

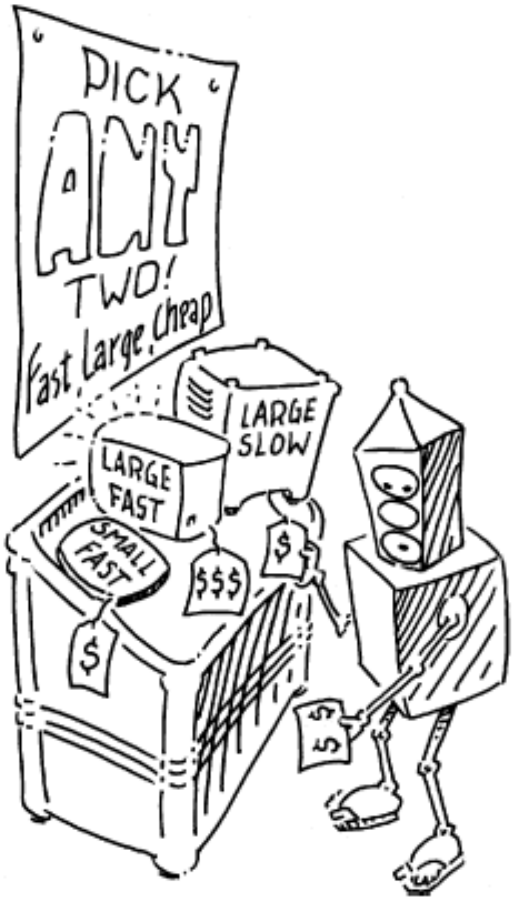
- Up until now, assumed memory could be accessed in 1 clock cycle
- But that hasn't been true since the 80's



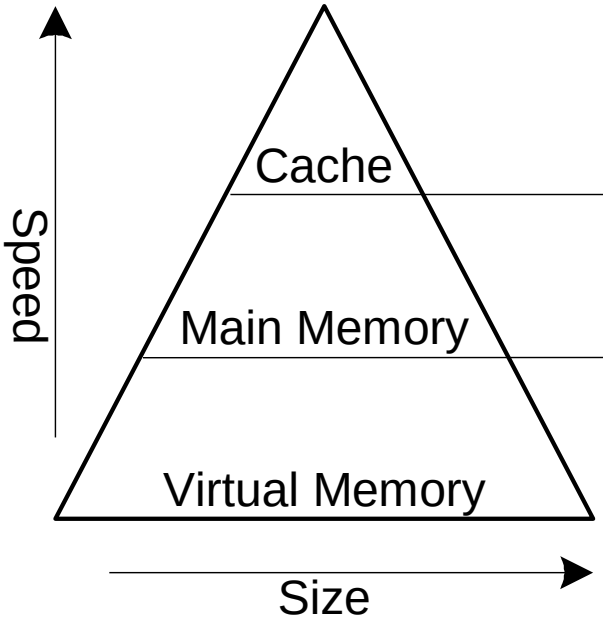
Memory System Challenge

- Make memory system appear as fast as processor
- Use a hierarchy of memories
- Ideal memory:
  - Fast
  - Cheap (inexpensive)
  - Large (capacity)

But can only choose two!



Memory Hierarchy



Technology	cost / GB	Access time
SRAM	~ $10,000	~ 1 ns
DRAM	~ $100	~ 100 ns
Hard Disk	~ $1	~ 10,000,000 ns

Locality

- Exploit locality to make memory accesses fast
- **Temporal Locality:**
  - Locality in time (e.g., if looked at a Webpage recently, likely to look at it again soon)
  - If used data recently, likely to use it again soon
  - **How to exploit:** keep recently accessed data in higher levels of memory hierarchy
- **Spatial Locality:**
  - Locality in space (e.g., if read one page of book recently, likely to read nearby pages soon)
  - If used data recently, likely to use nearby data soon
  - **How to exploit:** when access data, bring nearby data into higher levels of memory hierarchy too

Memory Performance

- **Hit:** is found in that level of memory hierarchy
- **Miss:** is not found (must go to next level)

Hit Rate = # hits / # memory accesses
= 1 – Miss Rate

Miss Rate = # misses / # memory accesses
= 1 – Hit Rate

- **Average memory access time (AMAT):** average time it takes for processor to access data

AMAT = $t_{\text{cache}} + MR_{\text{cache}}[t_{MM} + MR_{MM}(t_{VM})]$

Memory Performance Example 1

- A program has 2,000 load and store instructions
- 1,250 of these data values found in cache
- The rest are supplied by other levels of memory hierarchy
- **What are the miss and hit rates for the cache?**

Memory Performance Example 1

- A program has 2,000 load and store instructions
- 1,250 of these data values found in cache
- The rest are supplied by other levels of memory hierarchy
- **What are the miss and hit rates for the cache?**

Hit Rate = 1250/2000 = 0.625

Miss Rate = 750/2000 = 0.375 = 1 – Hit Rate

Memory Performance Example 2

- Suppose processor has 2 levels of hierarchy: cache and main memory
- t_{cache} = 1 cycle, t_{MM} = 100 cycles
- **What is the AMAT of the program from Example 1?**

Memory Performance Example 2

- Suppose processor has 2 levels of hierarchy: cache and main memory
- $t_{\text{cache}} = 1$ cycle, $t_{MM} = 100$ cycles
- **What is the AMAT of the program from Example 1 when using this memory system?**

$$\begin{aligned} \text{AMAT} &= t_{\text{cache}} + MR_{\text{cache}}(t_{MM}) \\ &= [1 + 0.375(100)] \text{ cycles} \\ &= 38.5 \text{ cycles} \end{aligned}$$

Gene Amdahl, 1922 -

- **Amdahl's Law:** the effort spent on increasing the performance of a subsystem is wasted unless the subsystem affects a large percentage of the overall performance
- Cofounded three companies, including one called Amdahl Corporation in 1970

Cache

A safe place to hide things

- Speeds up memory access time
- Is fast (usually ~ 1 cycle access time)
- Holds most recently used data

- Cache design questions:
  - What data is held in the cache?
  - How is data found?
  - What data is replaced?

Cache

A safe place to hide things

- Holds most recently accessed data
- Fast (usually ~ 1 cycle access time)
- Highest level in memory hierarchy
- Ideally supplies most of the data to the processor

Cache Design Questions

- What data is held in the cache?
- How is data found?
- What data is replaced?

We'll focus on data loads, but stores follow same principles

Cache Terminology

- Cache parameters:
  - Capacity (*C*): the number of data bits a cache stores
  - Number of sets (*S*): each memory address maps to exactly one set in a cache
  - Block size (*b*): size of data brought into cache at once
  - Number of blocks (*B*): number of blocks in cache
  - Degree of associativity (*N*): number of blocks in a set (also, number of places a single address could be found in cache)

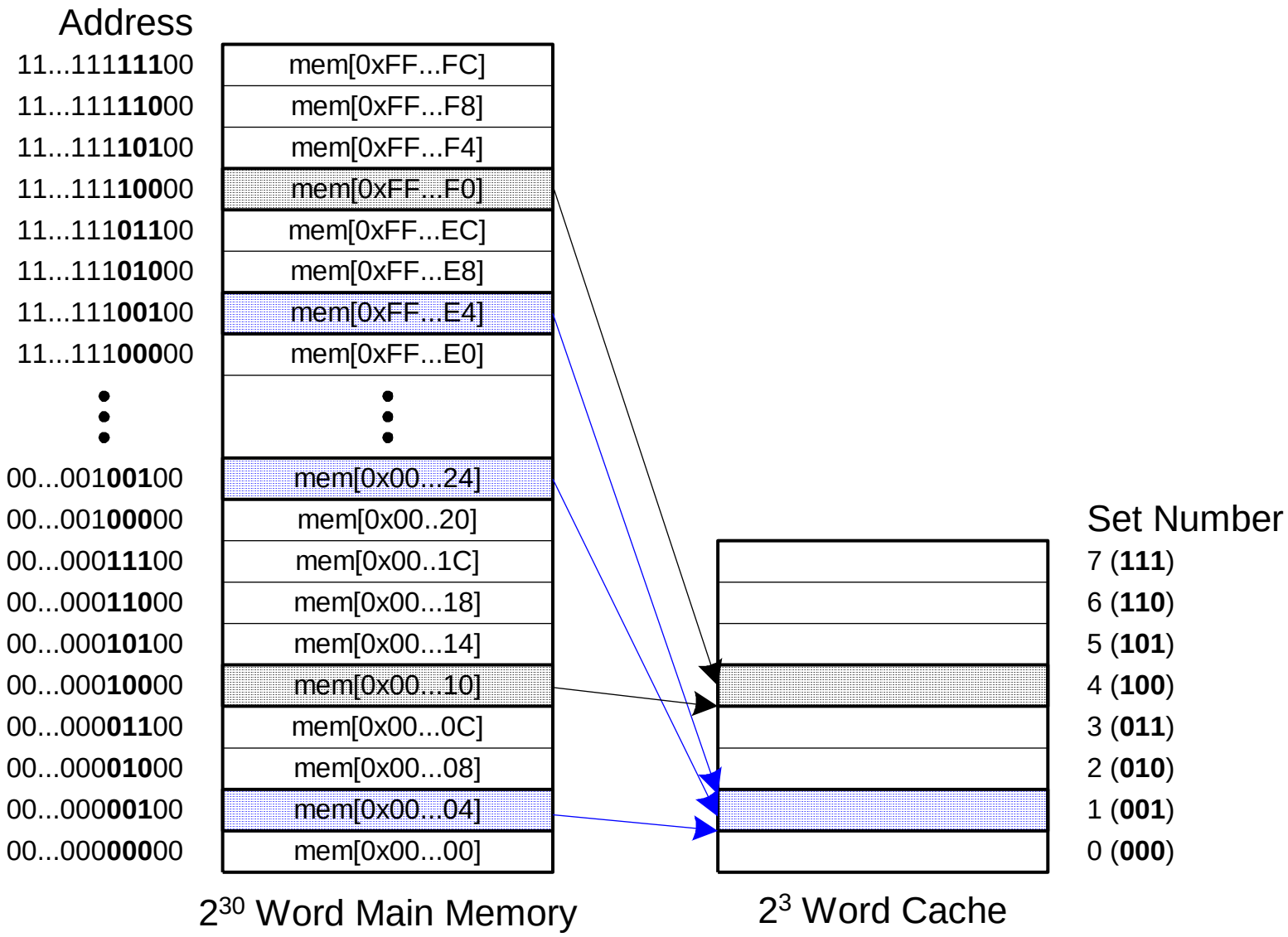
What data is held in the cache?

- Ideally, cache anticipates data needed by processor and holds it in cache
- But impossible to predict future
- So, use past to predict future – temporal and spatial locality:
  - **Temporal locality:** if processor accesses data not held in cache, copy data from lower level of hierarchy into cache. Next time, data is available in cache.
  - **Spatial locality:** copy neighboring data into cache too. Block size = number of bytes copied into cache at once.

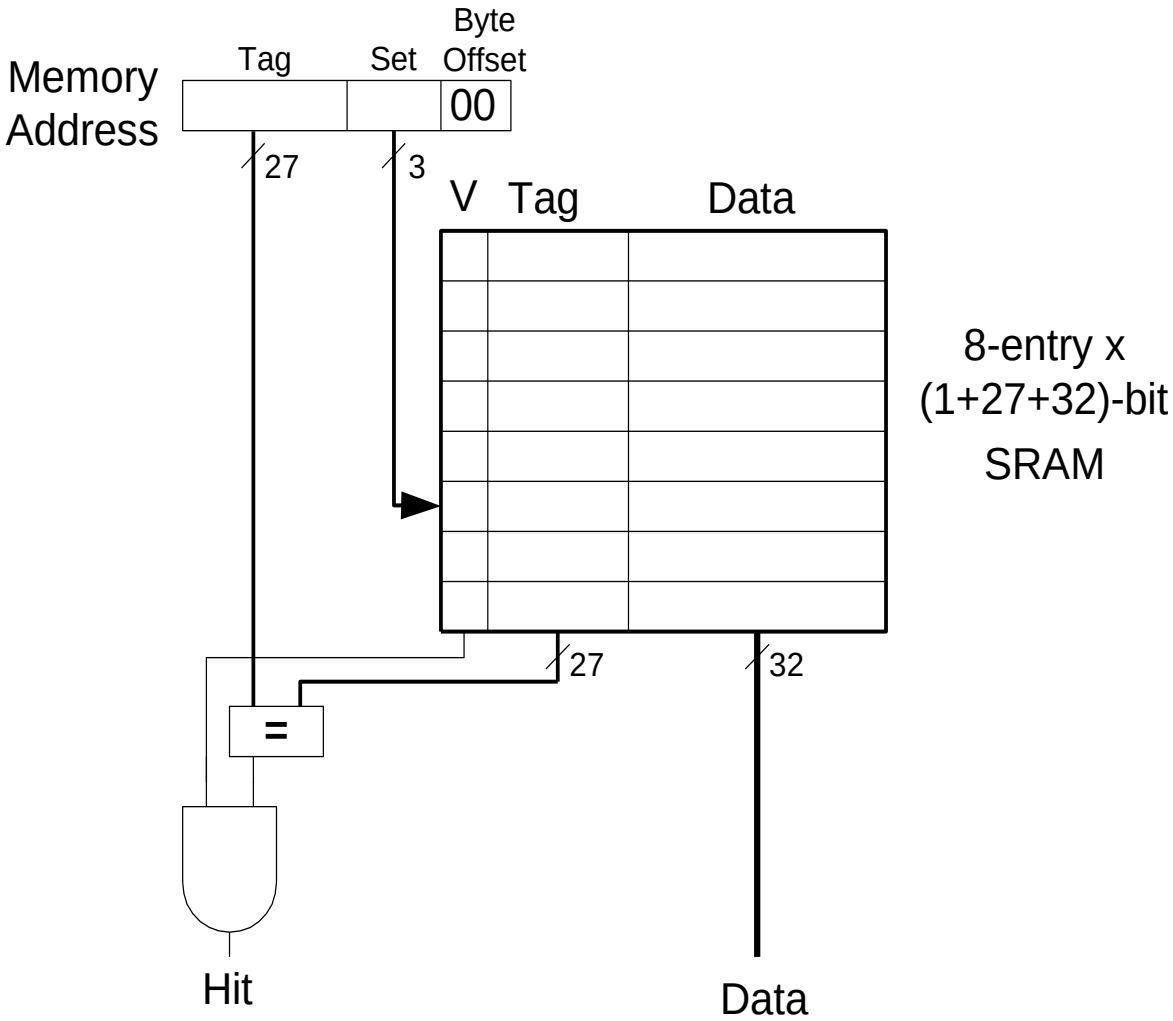
How is data found?

- Cache organized into *S* sets
- Each memory address maps to exactly one set
- Caches categorized by number of blocks in a set:
  - **Direct mapped:** 1 block per set
  - **N-way set associative:** N blocks per set
  - **Fully associative:** all cache blocks are in a single set
- Examine each organization for a cache with:
  - Capacity (*C* = 8 words)
  - Block size (*b* = 1 word)
  - So, number of blocks (*B* = 8)

Direct Mapped Cache

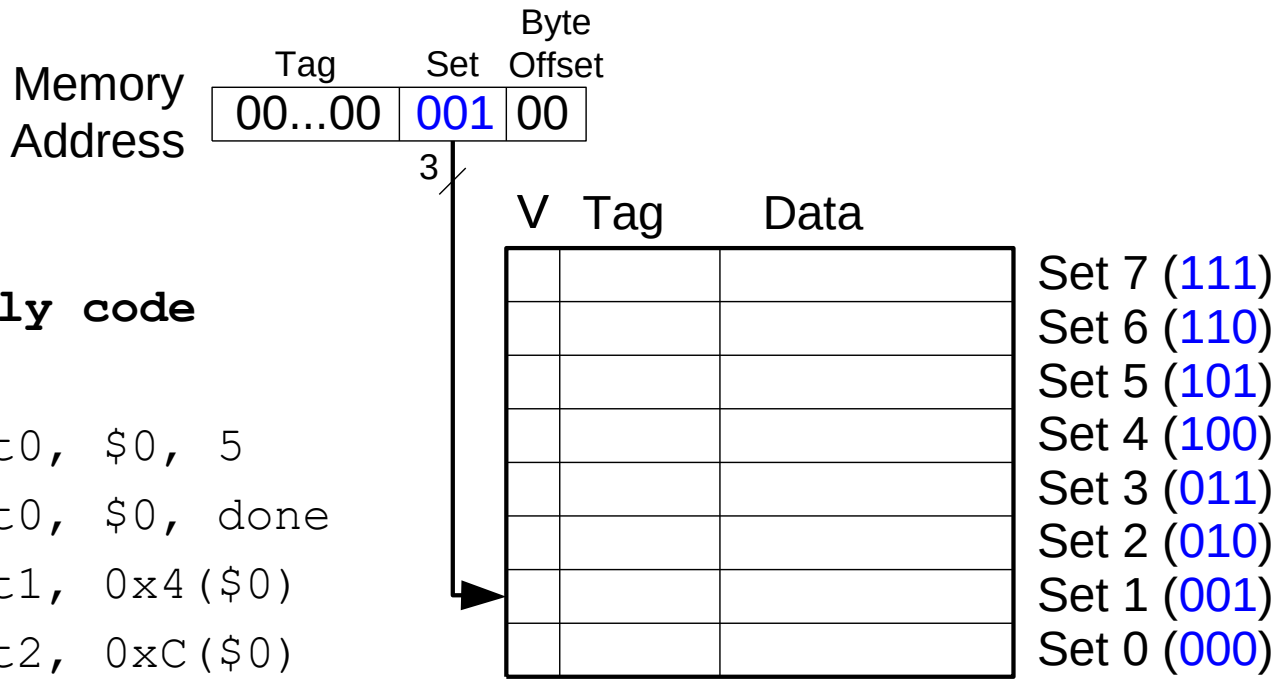


Direct Mapped Cache Hardware



Direct Mapped Cache Performance

```
# MIPS assembly code

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0xC($0)
        lw   $t3, 0x8($0)
        addi $t0, $t0, -1
        j    loop
done:
```

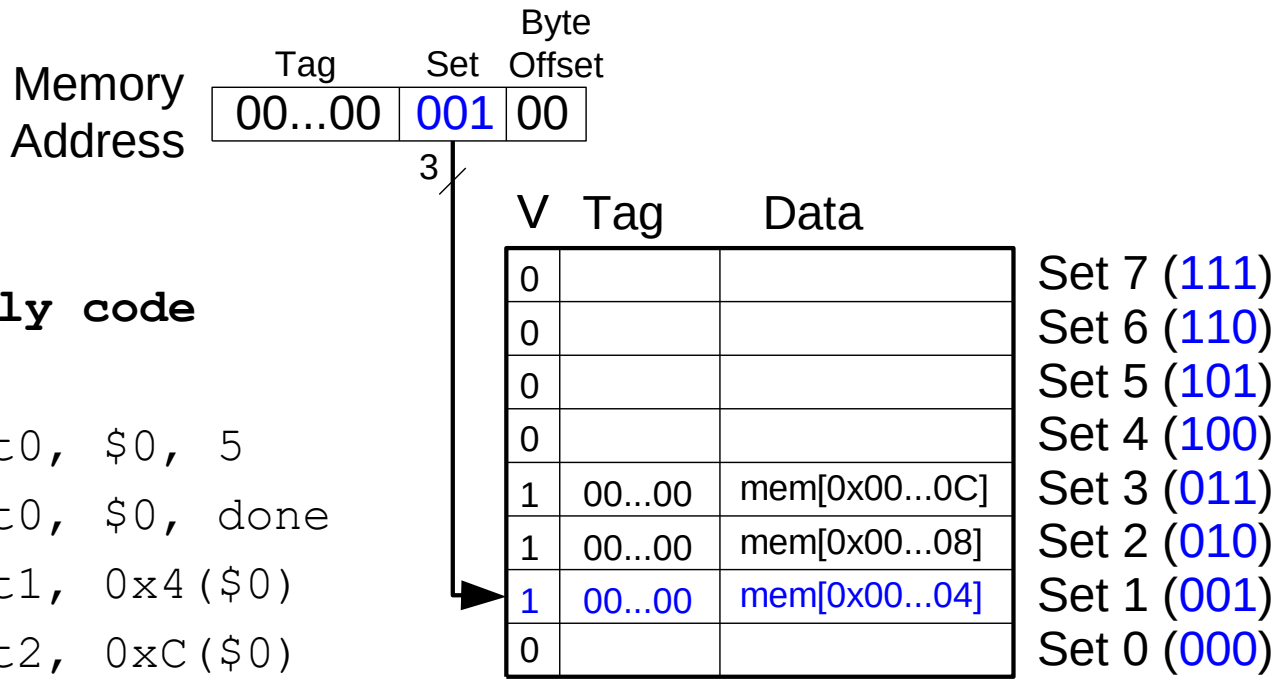


Miss Rate = 3/15

= 20%

Direct Mapped Cache Performance

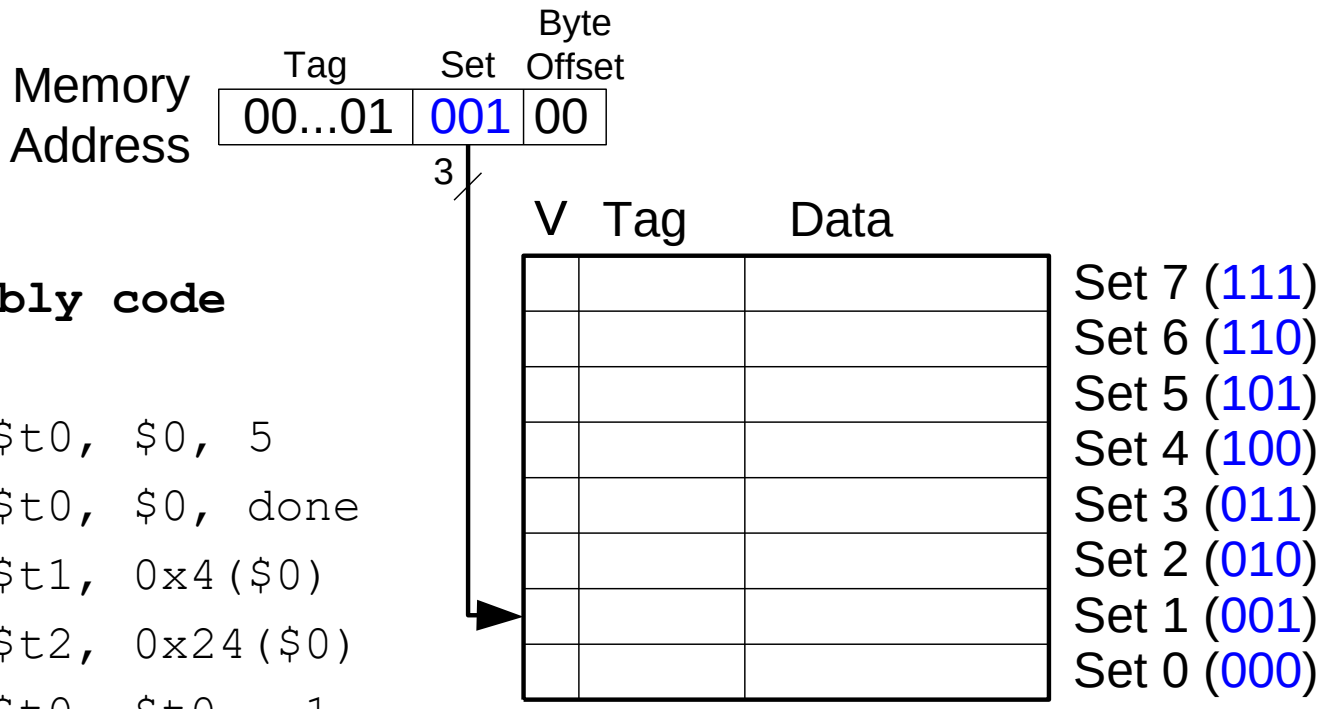
```
# MIPS assembly code

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0xC($0)
        lw   $t3, 0x8($0)
        addi $t0, $t0, -1
        j    loop
done:
```



Miss Rate = 3/15

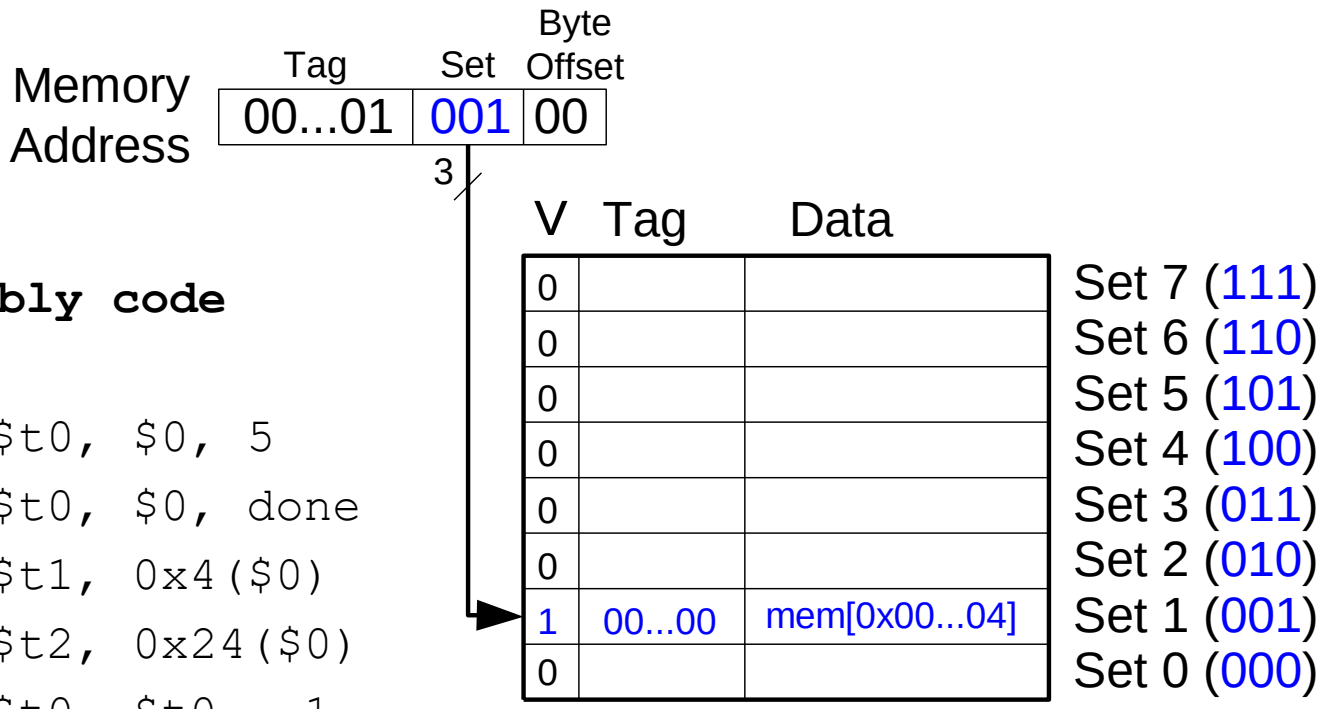
= 20%

Direct Mapped Cache: Conflict




```
# MIPS assembly code

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```

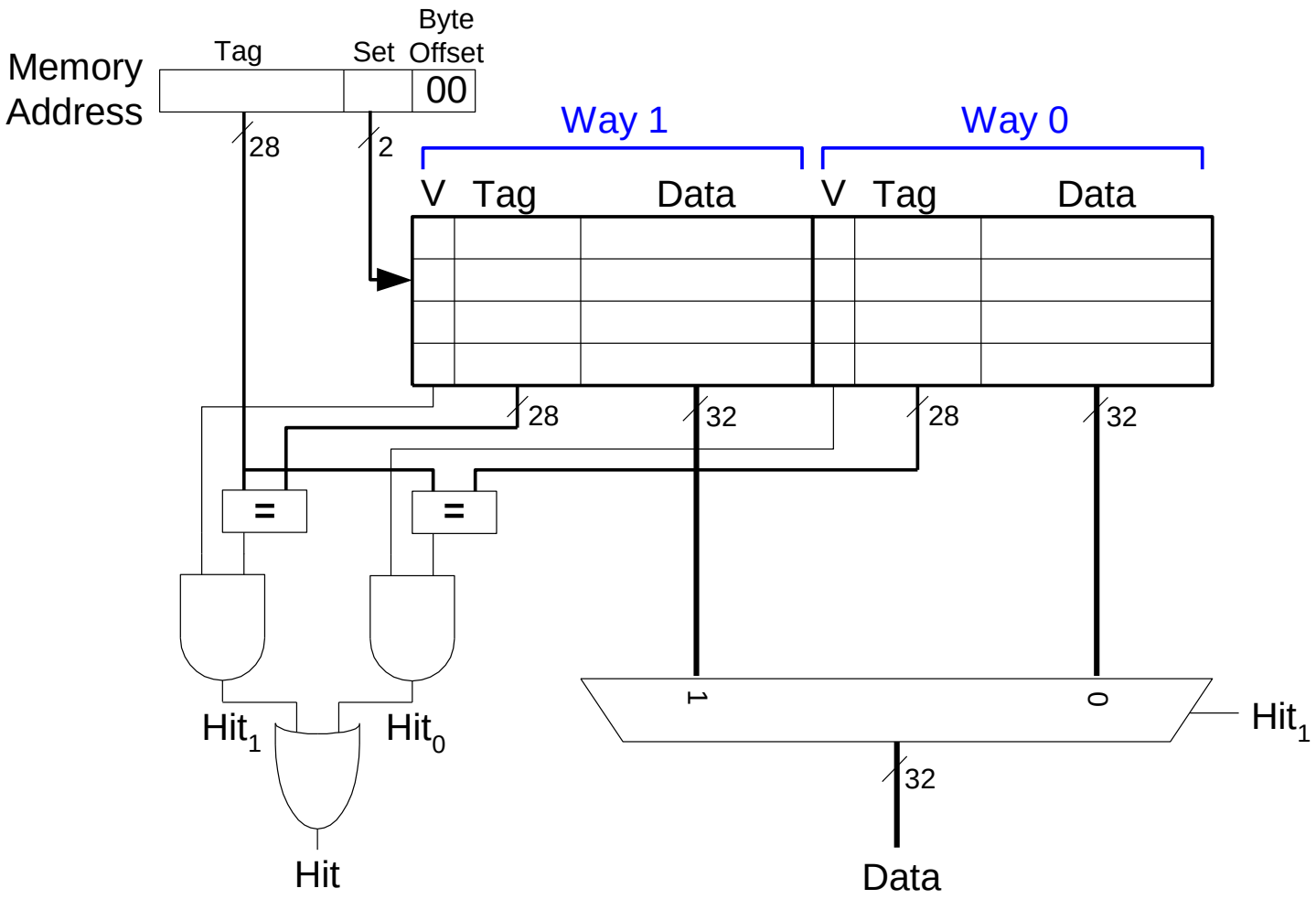
Direct Mapped Cache: Conflict




```
# MIPS assembly code

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```

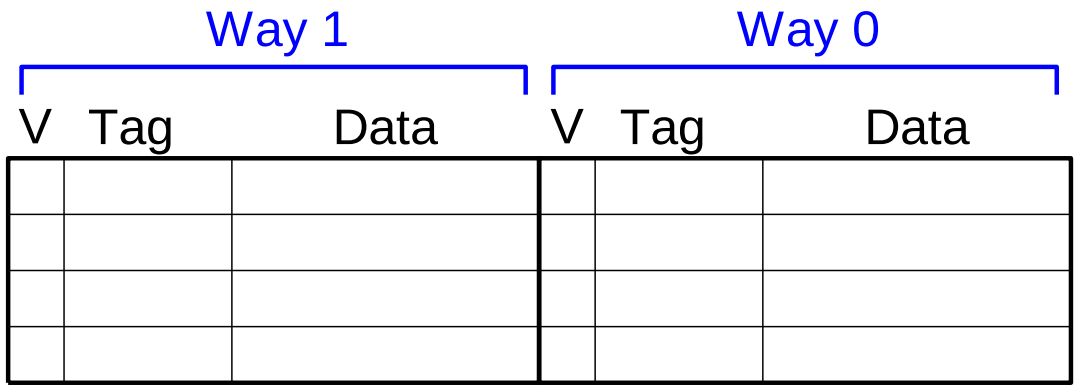
Miss Rate = 100%

N-Way Set Associative Cache



N-Way Set Associative Performance

```
# MIPS assembly code

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```



N-way Set Associative Performance

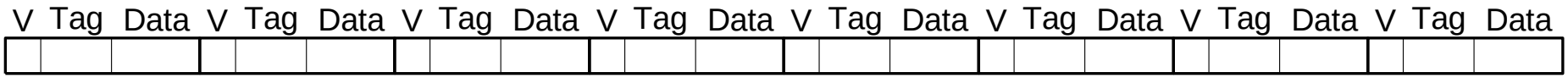
```
# MIPS assembly code

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```

Miss Rate = 2/10

= 20%

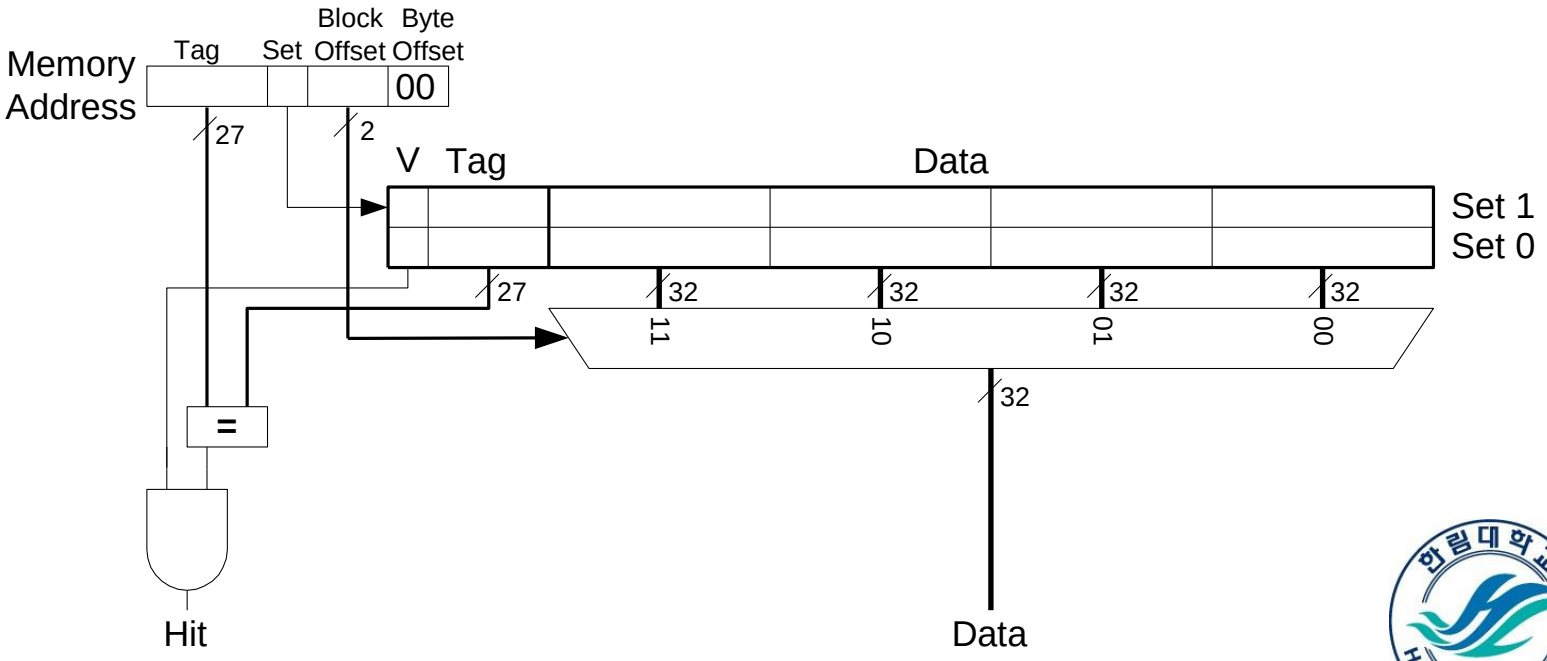
	Way 1			Way 0		
V	Tag	Data	V	Tag	Data	
0			0			Set 3
0			0			Set 2
1	00...10	mem[0x00...24]	1	00...00	mem[0x00...04]	Set 1
0			0			Set 0

Fully Associative Cache

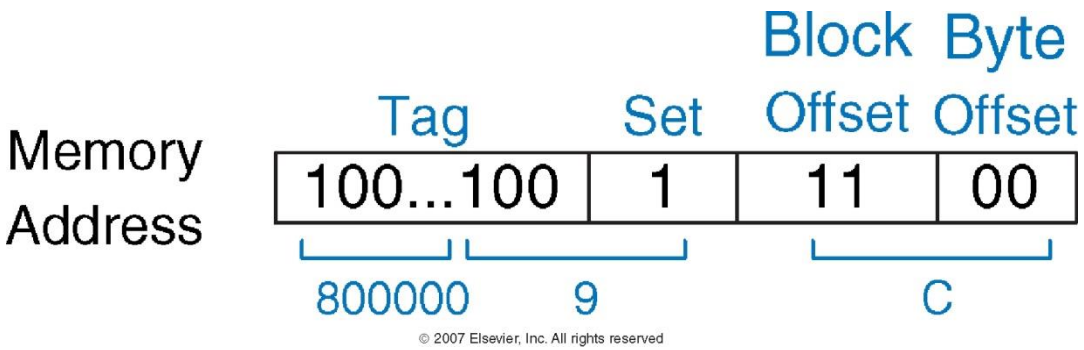


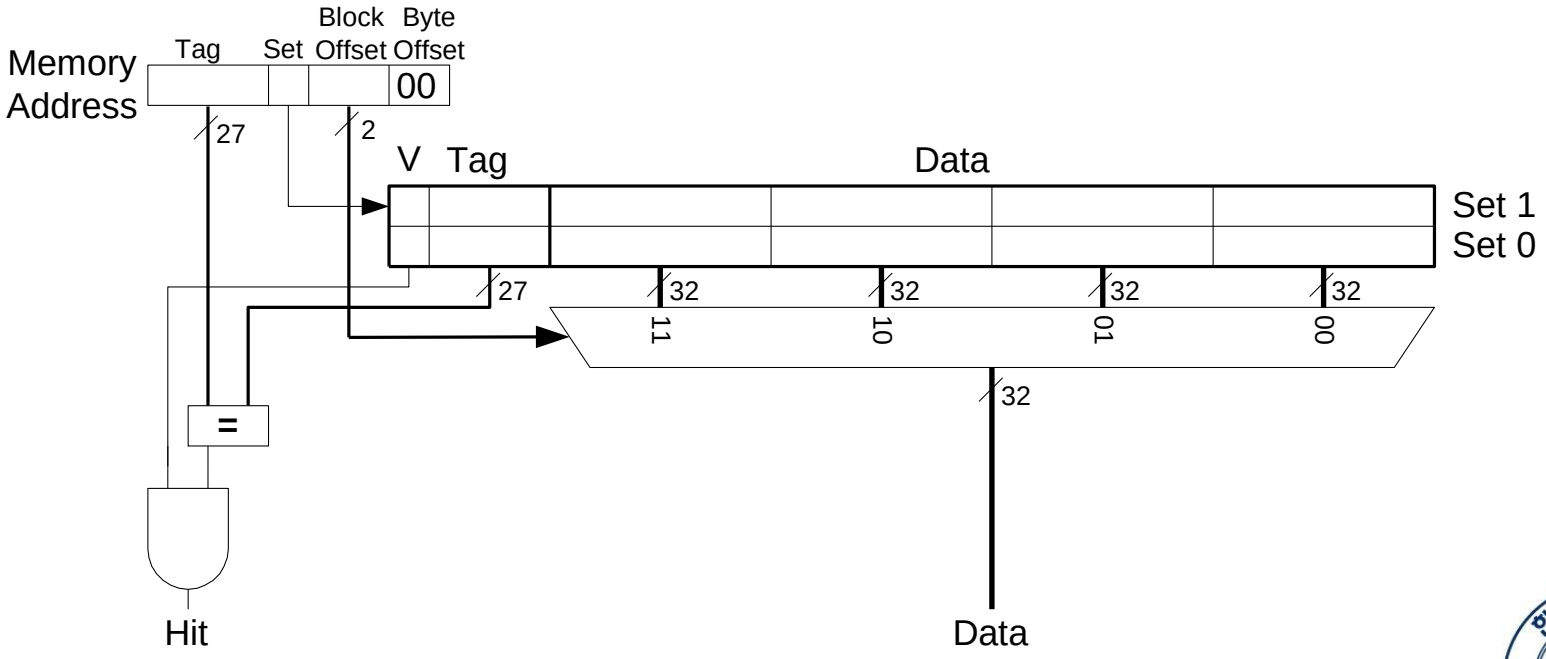
Spatial Locality?

- Increase block size:
  - Block size, b = 4 words
  - C = 8 words
  - Direct mapped (1 block per set)
  - Number of blocks, $B = C/b = 8/4 = 2$



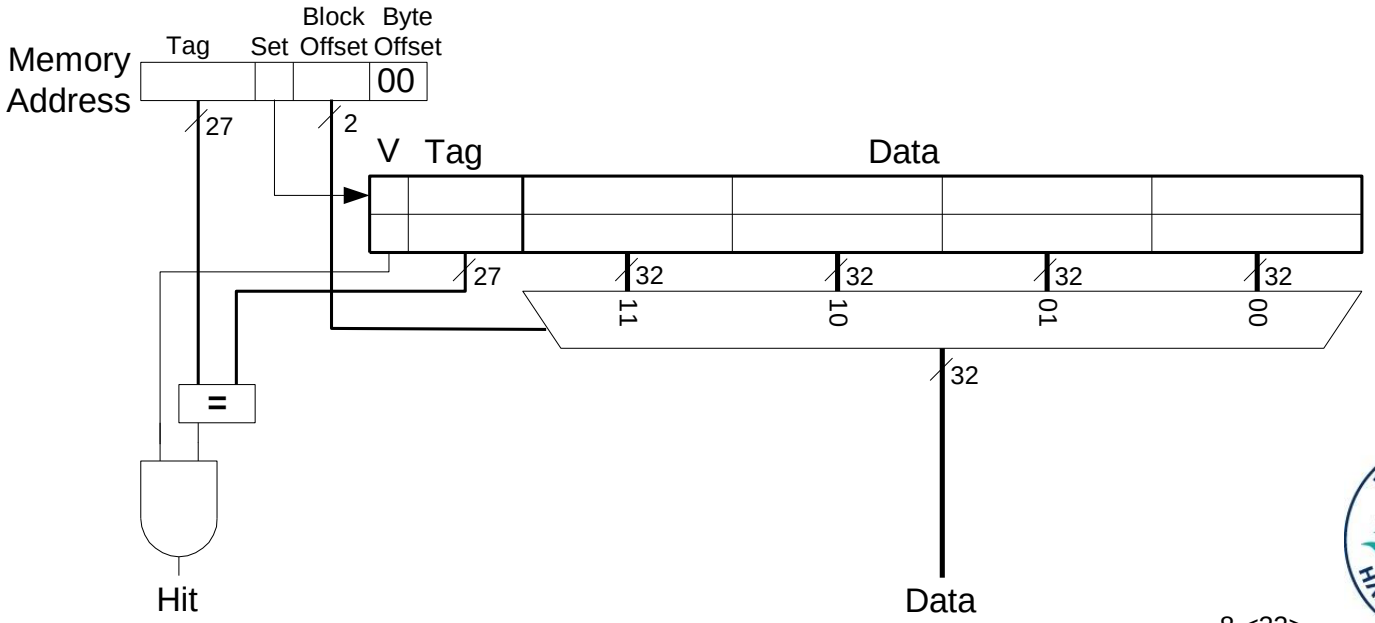
Cache with Larger Block Size





Direct Mapped Cache Performance

```
        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0xC($0)
        lw   $t3, 0x8($0)
        addi $t0, $t0, -1
        j    loop
done:
```

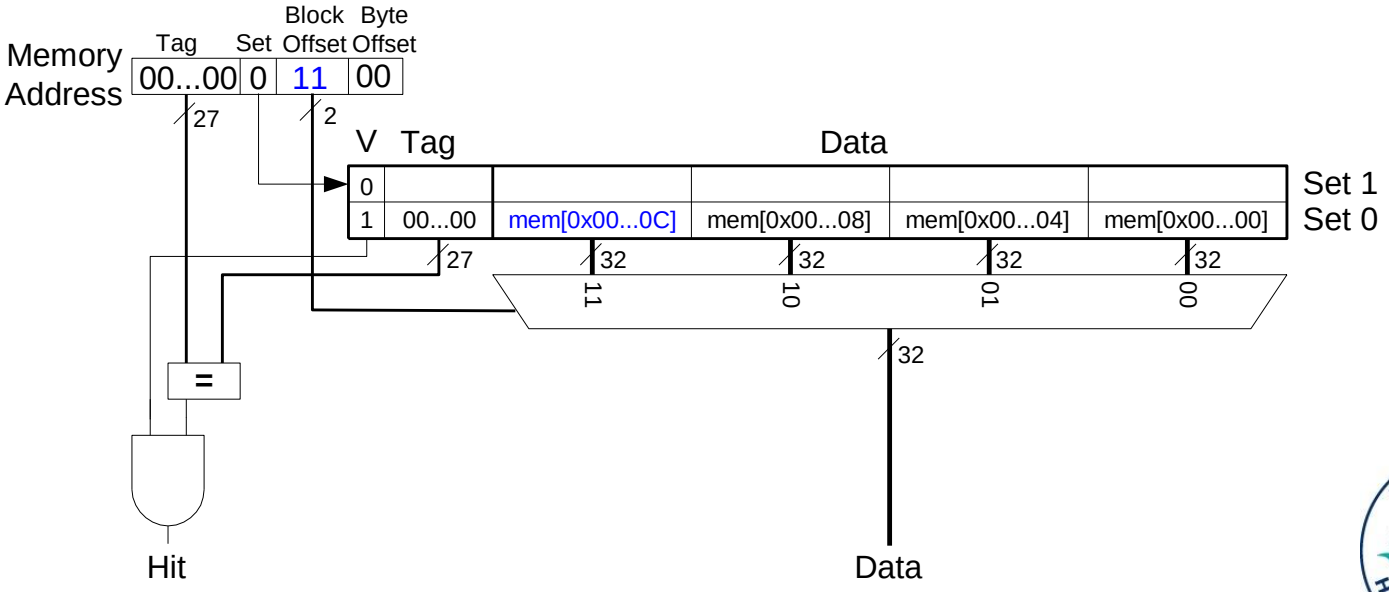


Direct Mapped Cache Performance

```
        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0xC($0)
        lw   $t3, 0x8($0)
        addi $t0, $t0, -1
        j    loop
done:
```

Miss Rate = 1/15

= 6.67%



Cache Organization Recap

- Capacity: C (in bytes)
- Block size: b (in bytes)
- Number of blocks in a set: N

Organization	Number of Ways (N)	Number of Sets (S)
Direct Mapped	1	B
N-Way Set Associative	$1 < N < B$	B / N
Fully Associative	B	1

What data is replaced?

- Least recently used (LRU) replacement: the least recently used block is evicted when the cache is full.

LRU Replacement

```
# MIPS assembly

lw $t0, 0x04($0)
lw $t1, 0x24($0)
lw $t2, 0x54($0)
```

(a)

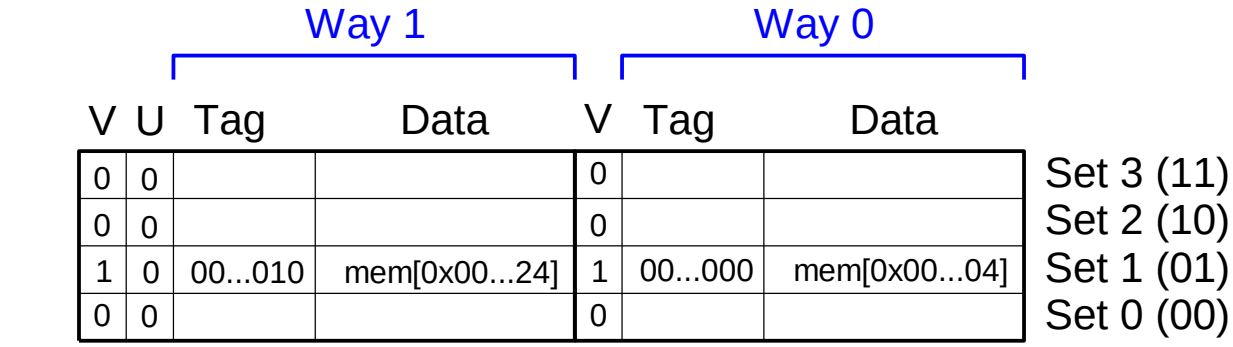
V	U	Tag	Data	V	Tag	Data	Set Number
							3 (**11**)
							2 (**10**)
							1 (**01**)
							0 (**00**)

(b)

V	U	Tag	Data	V	Tag	Data	Set Number
							3 (**11**)
							2 (**10**)
							1 (**01**)
							0 (**00**)

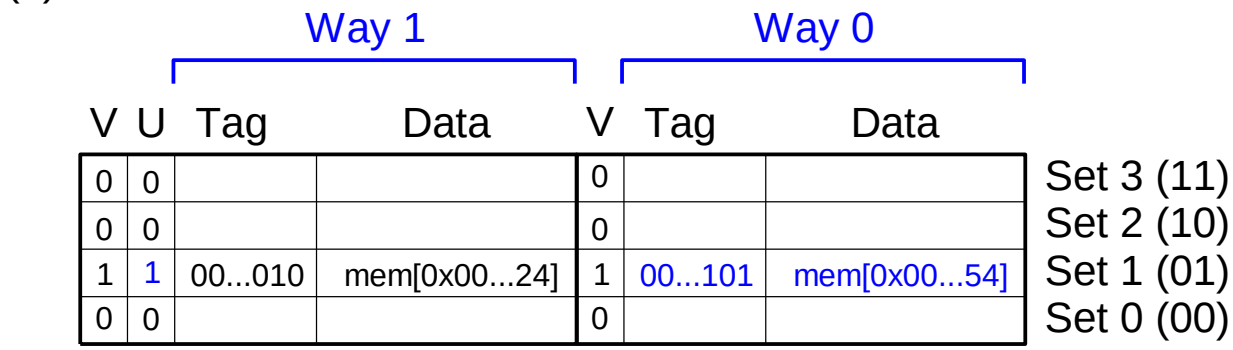
LRU Replacement

```
# MIPS assembly

lw $t0, 0x04($0)
lw $t1, 0x24($0)
lw $t2, 0x54($0)
```



	Way 1				Way 0		
	V	U	Tag	Data	V	Tag	Data
Set 3 (11)	0	0			0		
Set 2 (10)	0	0			0		
Set 1 (01)	1	0	00...010	mem[0x00...24]	1	00...000	mem[0x00...04]
Set 0 (00)	0	0			0		

(a)



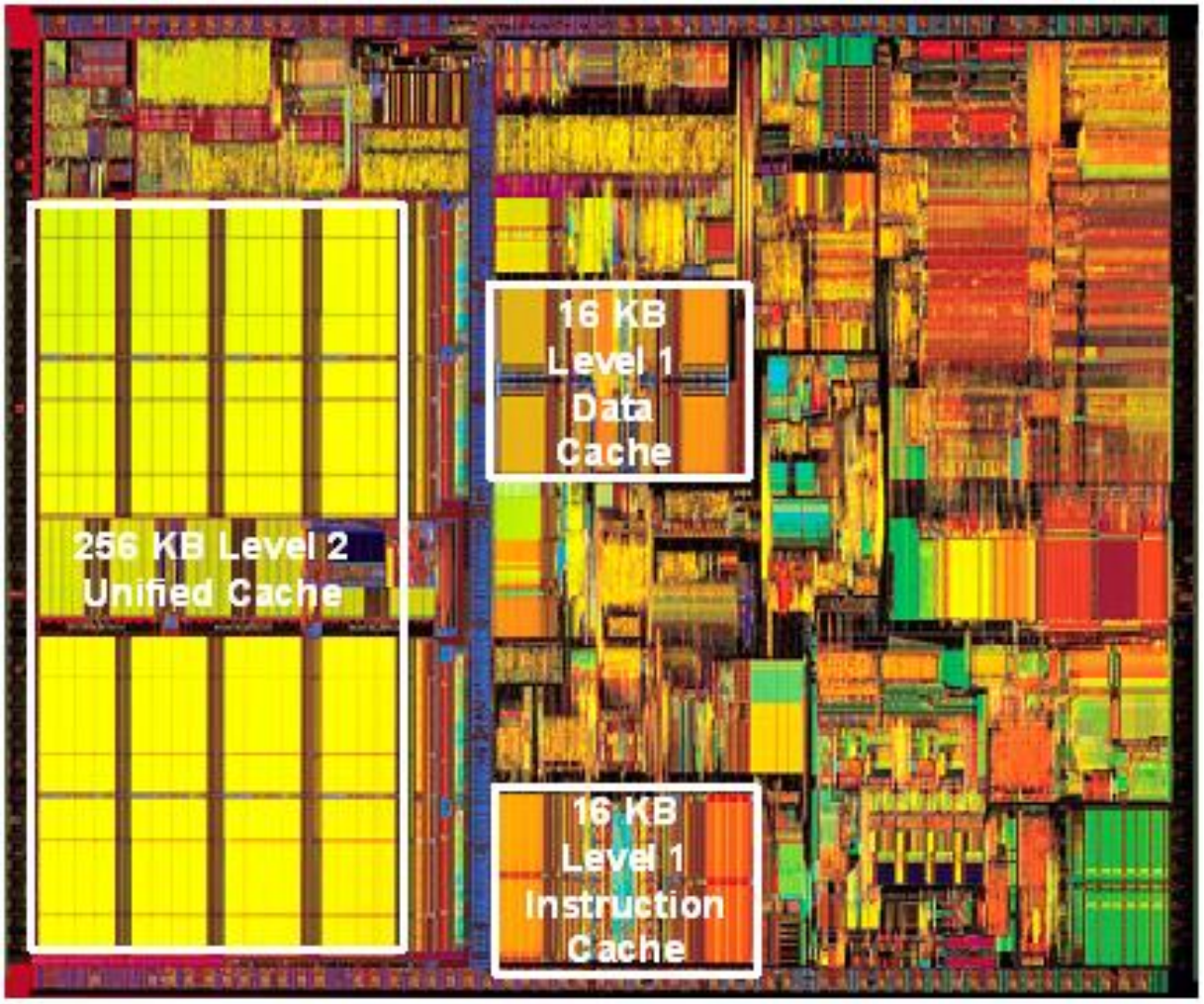
	Way 1				Way 0		
	V	U	Tag	Data	V	Tag	Data
Set 3 (11)	0	0			0		
Set 2 (10)	0	0			0		
Set 1 (01)	1	1	00...010	mem[0x00...24]	1	00...101	mem[0x00...54]
Set 0 (00)	0	0			0		

(b)

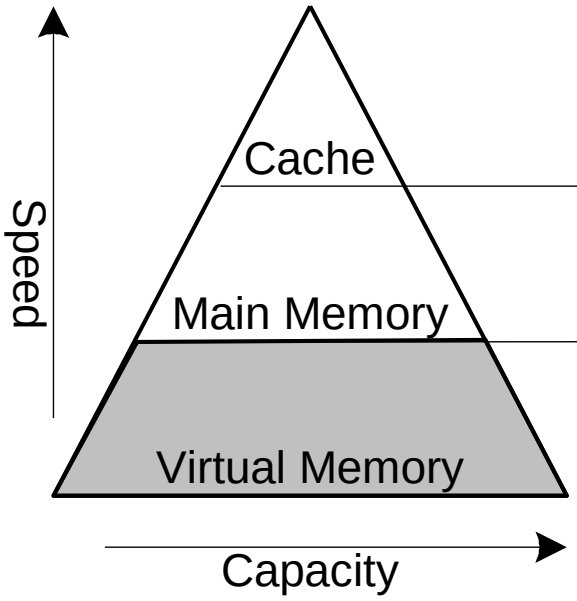
Caching Summary

- Temporal and spatial locality
- LRU replacement
- Cache Parameters:
  - ***C*** = capacity
  - ***b*** = block size
  - ***B*** = # blocks (= *C*/*b*)
  - ***S*** = # sets
  - ***N*** = # blocks in a set (# of ways)

Intel Pentium III Die



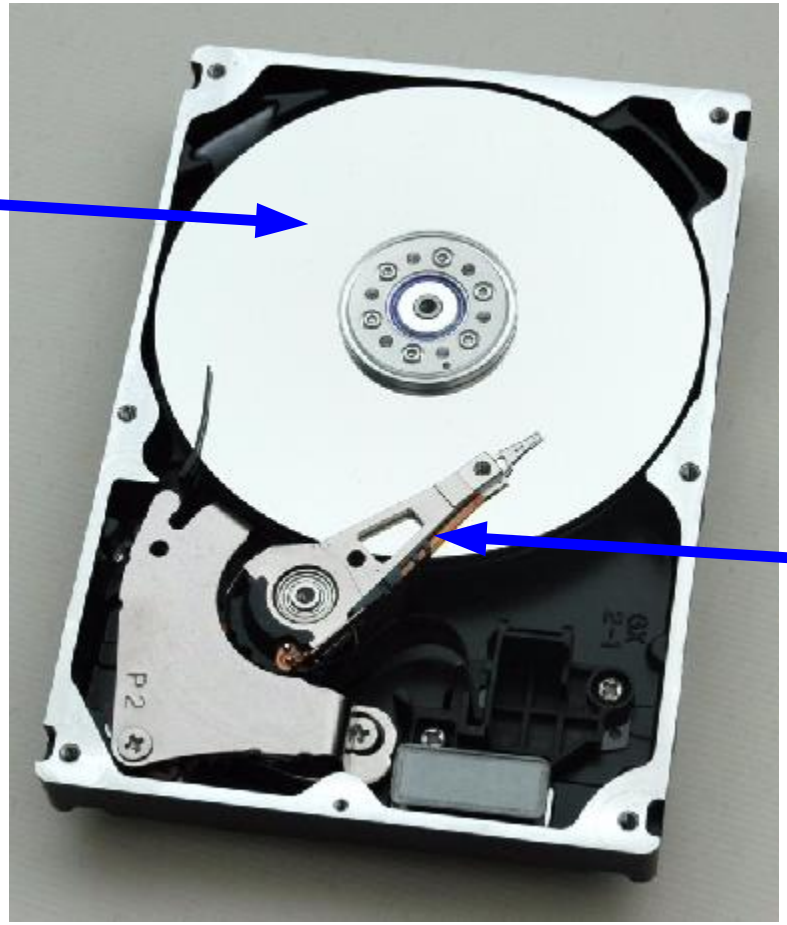
The Memory Hierarchy



Technology	cost / GB	Access time
SRAM	~ $10,000	~ 1 ns
DRAM	~ $100	~ 100 ns
Hard Disk	~ $1	~ 10,000,000 ns

- **Physical Memory:** DRAM
- **Virtual Memory:** Hard disk
  - Slow, Large, Cheap

The Hard Disk



Takes milliseconds to *seek* correct location on disk

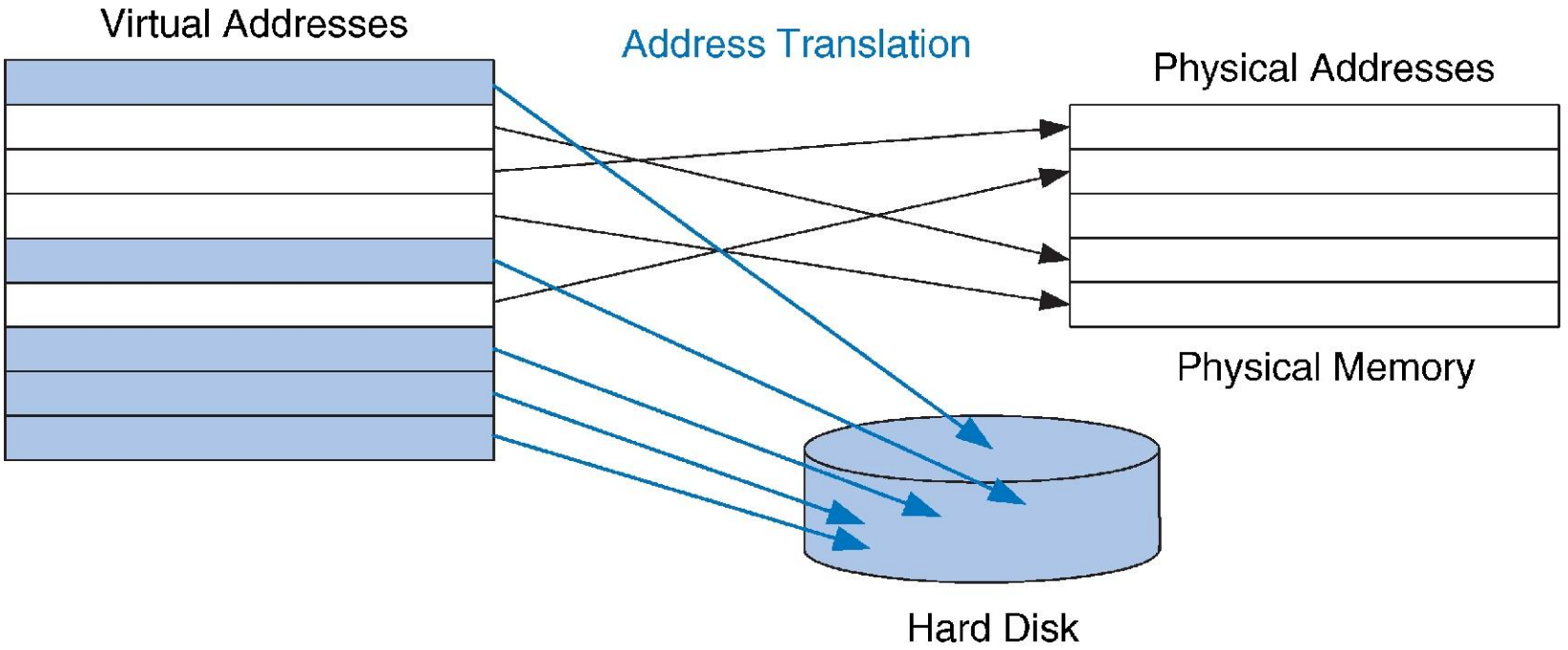
Virtual Memory Overview

- Illusion of larger memory (without the high cost of DRAM)
- Physical memory (DRAM) acts as cache for virtual memory (hard disk)
- Only subset of virtual memory in physical memory
- Processor generates virtual addresses but can only access physical memory
- Must translate virtual memory addresses to physical addresses

Virtual Memory Overview

- **Page size:** amount of memory transferred from hard disk to DRAM
- **Address translation:** determining the physical address from the virtual address
- **Page table:** lookup table used to translate virtual addresses to physical addresses

Virtual and Physical Addresses



Most accesses hit in physical memory, but benefit of large capacity of virtual memory

Cache/Virtual Memory Analogues

Physical memory cache for virtual memory

Cache	Virtual Memory
Block	Page
Block Size	Page Size
Block Offset	Page Offset
Miss	Page Fault
Tag	Virtual Page Number