

# Digital Design & Computer Architecture

Sarah Harris & David Harris

## Chapter 8: Memory Systems

Cache veri erişiminde  
hız arttırımına neden  
olur

direct map

fully associative

set associative

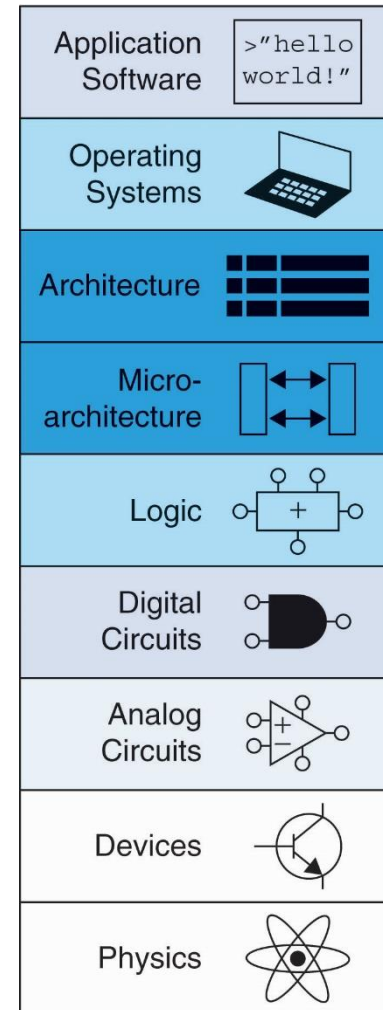
herhangi bir  
memory adresi  
doğrudan  
doğruya belirli bir  
cache satırına  
mapleniyor

→ cache blok  
geniřlięi



# Chapter 8 :: Topics

- Introduction
- Memory System Performance Analysis
- Caches
- Virtual Memory



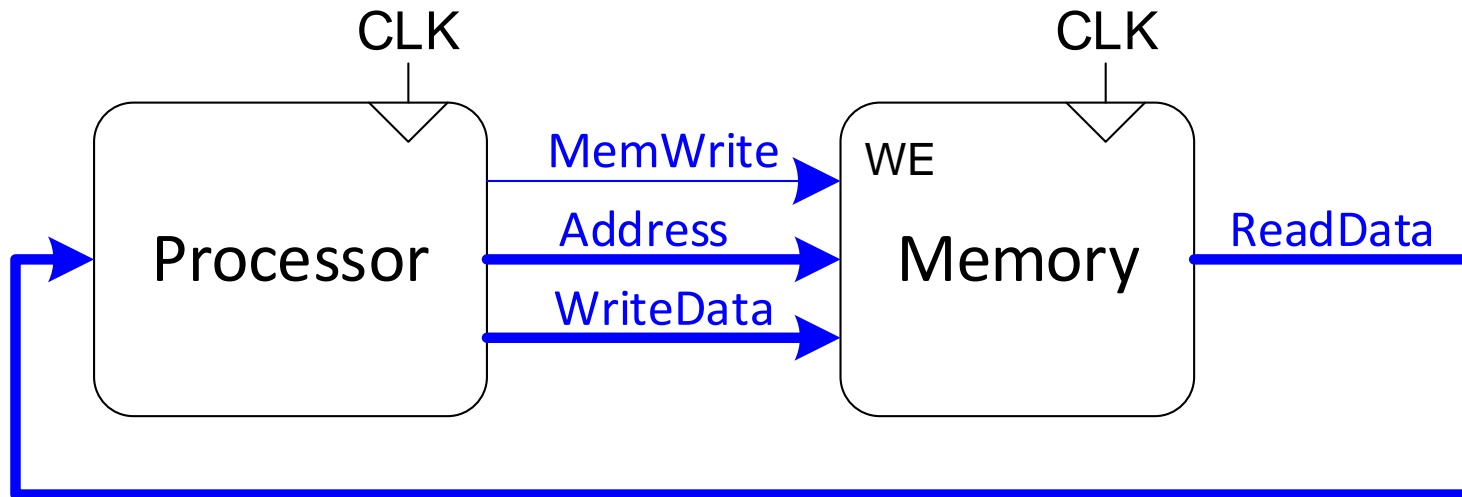




# Introduction

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

## Processor / Memory Interface:



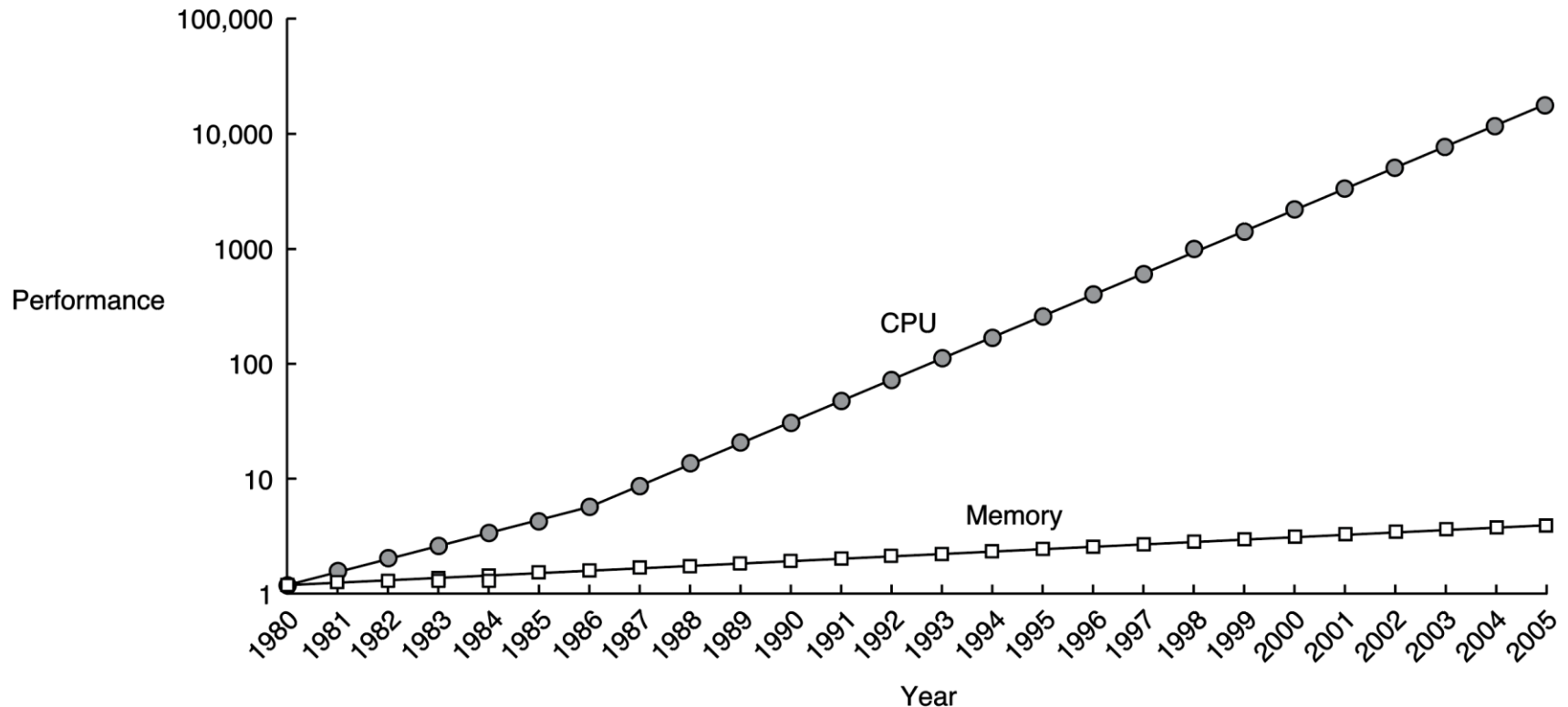






# Processor-Memory Gap

- In prior chapters, assumed access memory in 1 clock cycle – but hasn't been true since the 1980's.







# Memory System Challenge

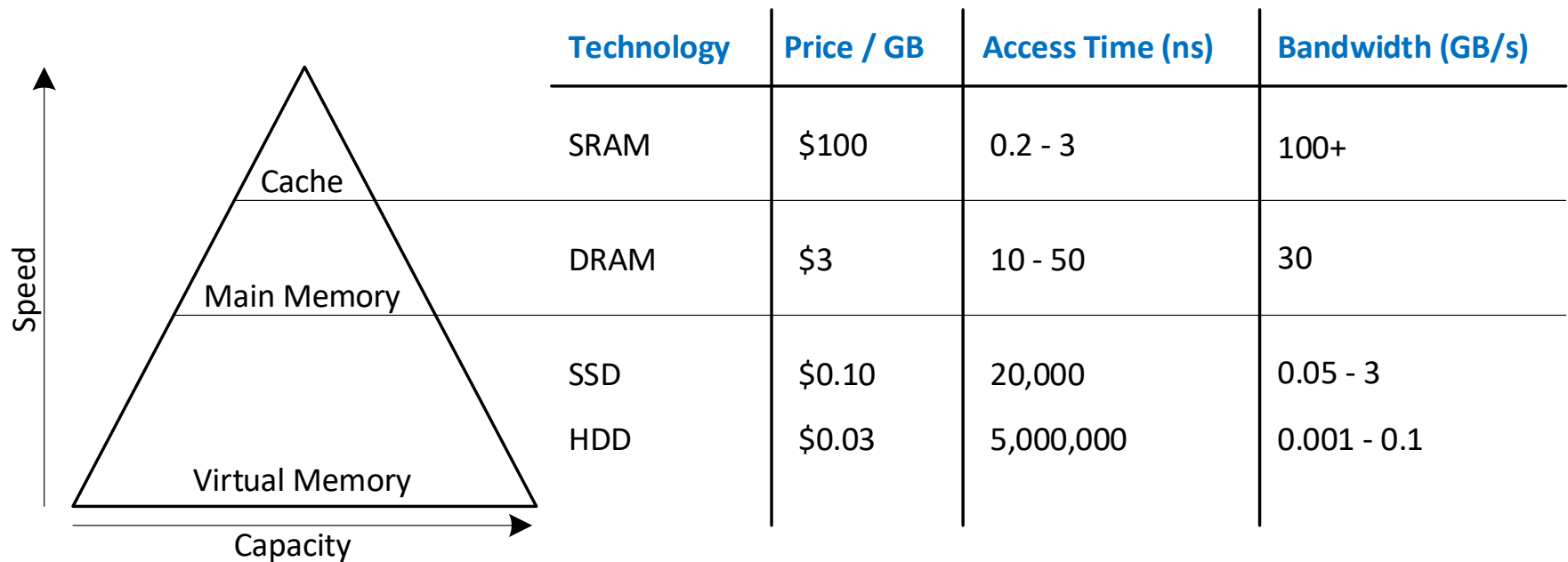
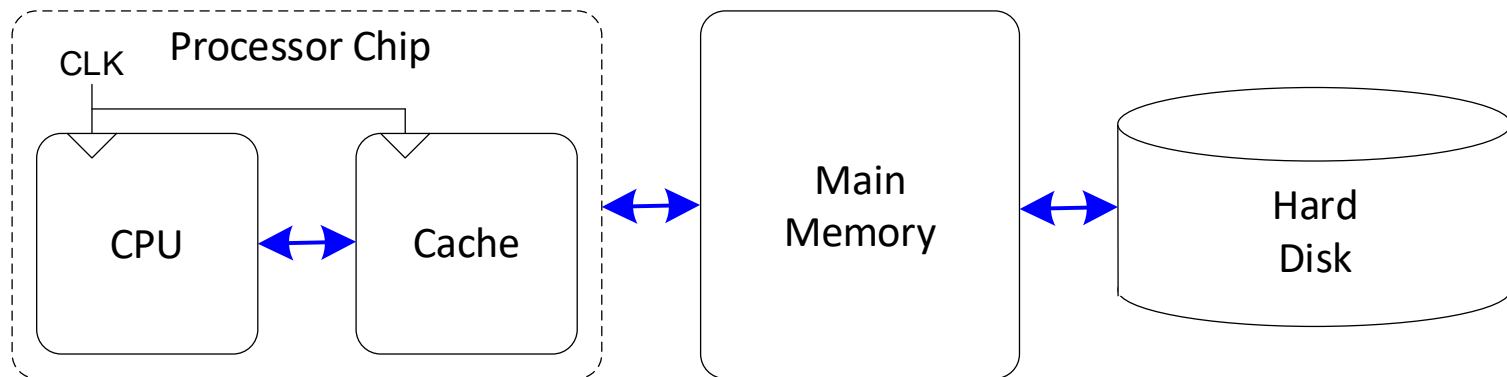
- Make memory system appear as fast as processor
- Use hierarchy of memories
- Ideal memory:
  - **Fast**
  - **Cheap** (inexpensive)
  - **Large** (capacity)
- **But can only choose two!**







# Memory Hierarchy









# Locality

Exploit locality to make memory accesses fast:

- **Temporal Locality:**

- Locality in time
- If data used 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
- If data used recently, likely to use nearby data soon
- **How to exploit:** when access data, bring nearby data into higher levels of memory hierarchy too





## Chapter 8: Memory Systems

# Memory Performance







# Memory Performance

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

**Hit Rate**  $= \# \text{ hits} / \# \text{ memory accesses}$   
 $= 1 - \text{Miss Rate}$

**Miss Rate**  $= \# \text{ misses} / \# \text{ memory accesses}$   
 $= 1 - \text{Hit Rate}$

- **Average memory access time (AMAT):** average time 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 loads and stores
- 1,250 of these data values in cache
- Rest supplied by other levels of memory hierarchy
- What are the **cache hit and miss rates**?





# 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** (average memory access time) of the program from Example 1?







# Gene Amdahl, 1922 -

- **Amdahl's Law:** the effort spent increasing the performance of a subsystem is wasted unless the subsystem affects a large percentage of overall performance
- Co-founded 3 companies, including one called Amdahl Corporation in 1970







# Chapter 8: Memory Systems

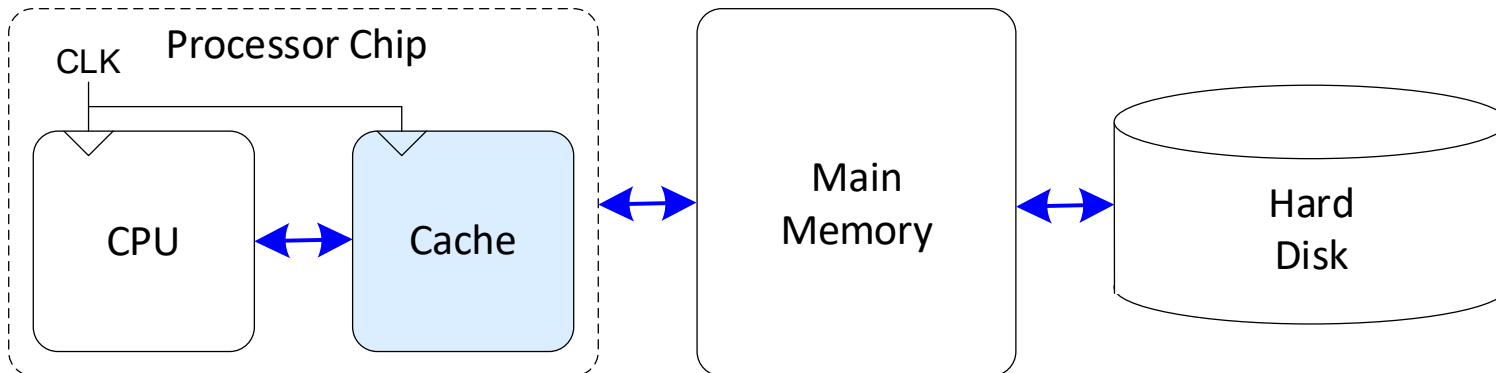
## **Caches**





# Cache

- Highest level in memory hierarchy
- Fast (typically  $\sim 1$  cycle access time)
- Ideally supplies most data to processor
- Usually holds most recently accessed data









# Cache Design Questions

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

We focus on data loads, but stores follow the same principles.





# What data is held in the cache?

- Ideally, cache anticipates needed data and puts it in cache
- But impossible to predict future
- Use past to predict future – temporal and spatial locality:
  - **Temporal locality:** copy newly accessed data into cache
  - **Spatial locality:** copy neighboring data into cache too







# Cache Terminology

- **Capacity ( $C$ ):**
  - number of data bytes in cache
- **Block size ( $b$ ):**
  - bytes of data brought into cache at once
- **Number of blocks ( $B = C/b$ ):**
  - number of blocks in cache:  $B = C/b$
- **Degree of associativity ( $N$ ):**
  - number of blocks in a set
- **Number of sets ( $S = B/N$ ):**
  - each memory address maps to exactly one cache set





# How is data found?

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





# Example Cache Parameters

- **$C = 8$**  words (capacity)
- **$b = 1$**  word (block size)
- So,  **$B = 8$**  (# of blocks)

Ridiculously small, but will illustrate organizations







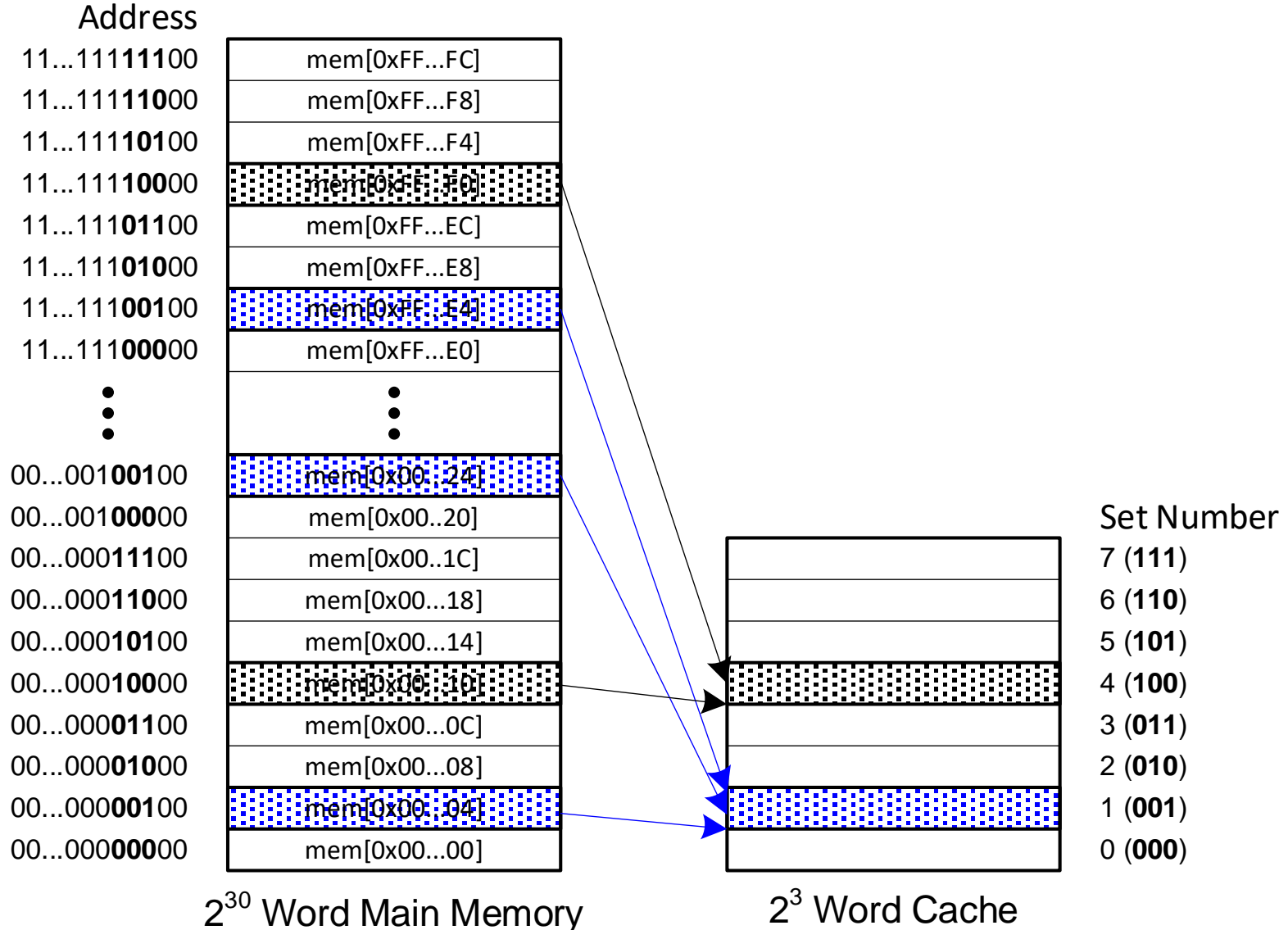
## Chapter 8: Memory Systems

# Direct-Mapped Caches





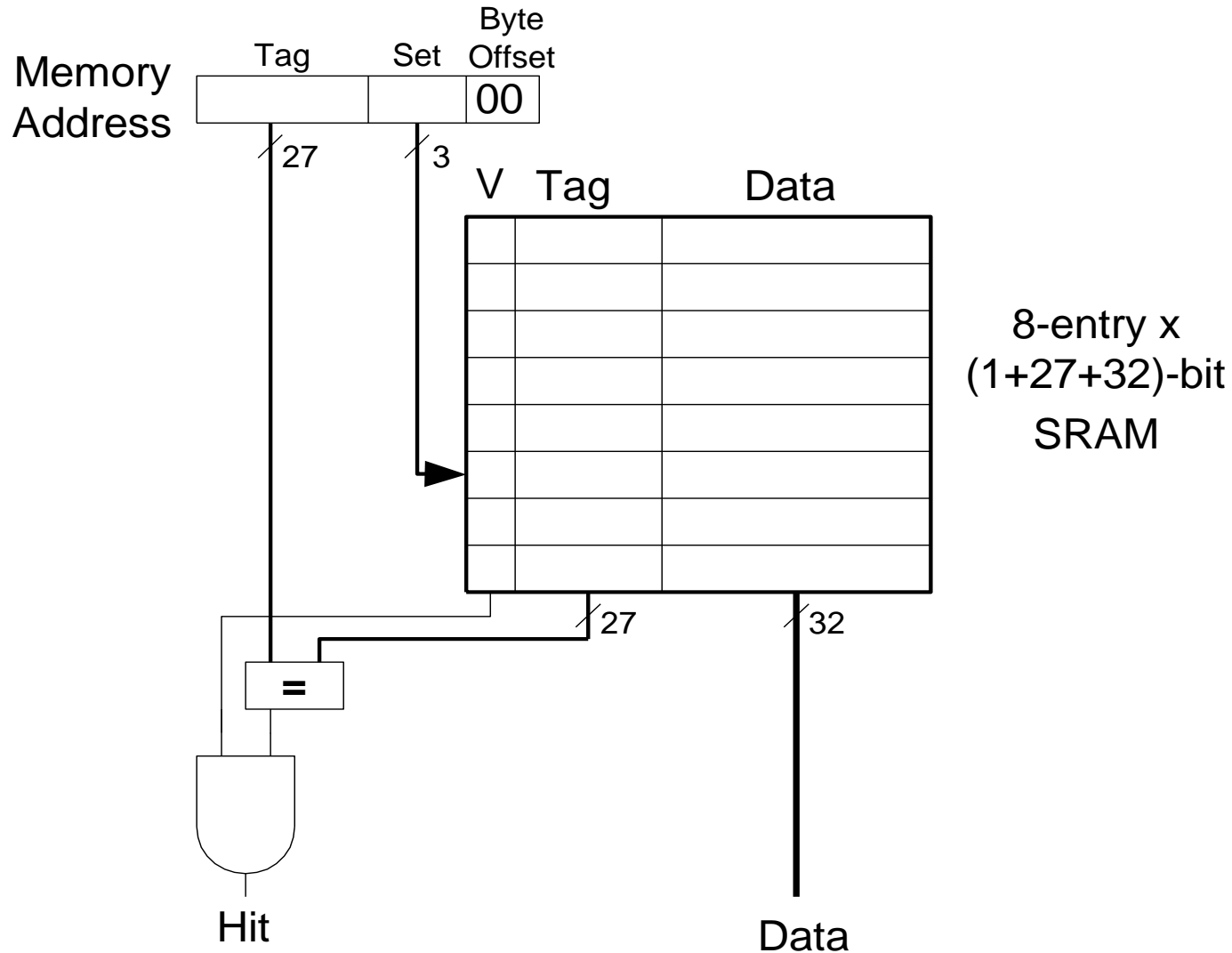
# Direct Mapped Cache







# Direct Mapped Cache Hardware

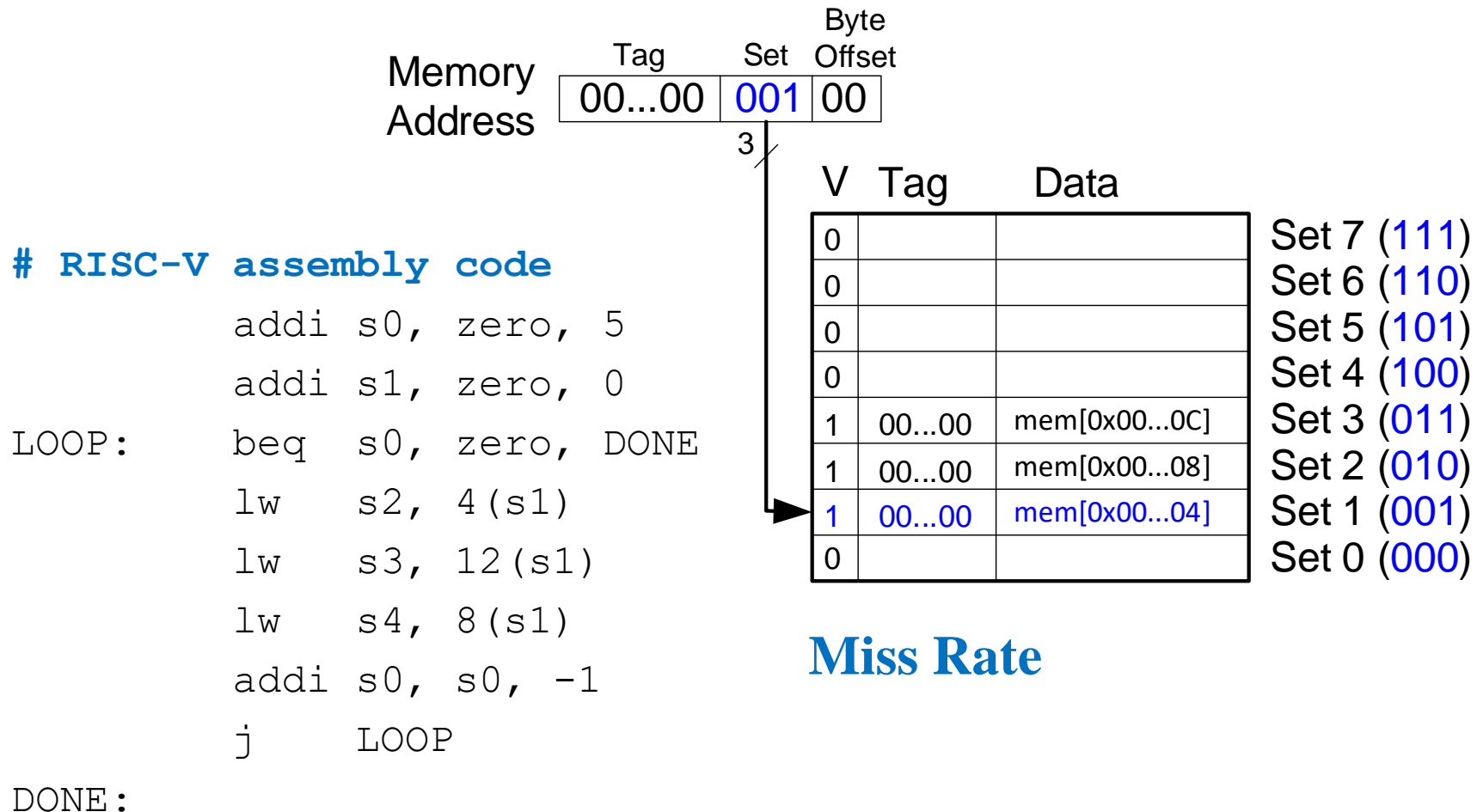








# Direct Mapped Cache Performance







# Direct Mapped Cache: Conflict Miss



## # RISC-V assembly code

```
    addi s0, zero, 5
    addi s1, zero, 0
LOOP: beq  s0, zero, DONE
      lw   s2, 0x4(s1)
      lw   s4, 0x24(s1)
      addi s0, s0, -1
      j    LOOP
DONE:
```

V	Tag	Data	
0			Set 7 (111)
0			Set 6 (110)
0			Set 5 (101)
0			Set 4 (100)
0			Set 3 (011)
0			Set 2 (010)
1	00...00	mem[0x00...04] mem[0x00...24]	Set 1 (001)
0			Set 0 (000)

Miss Rate







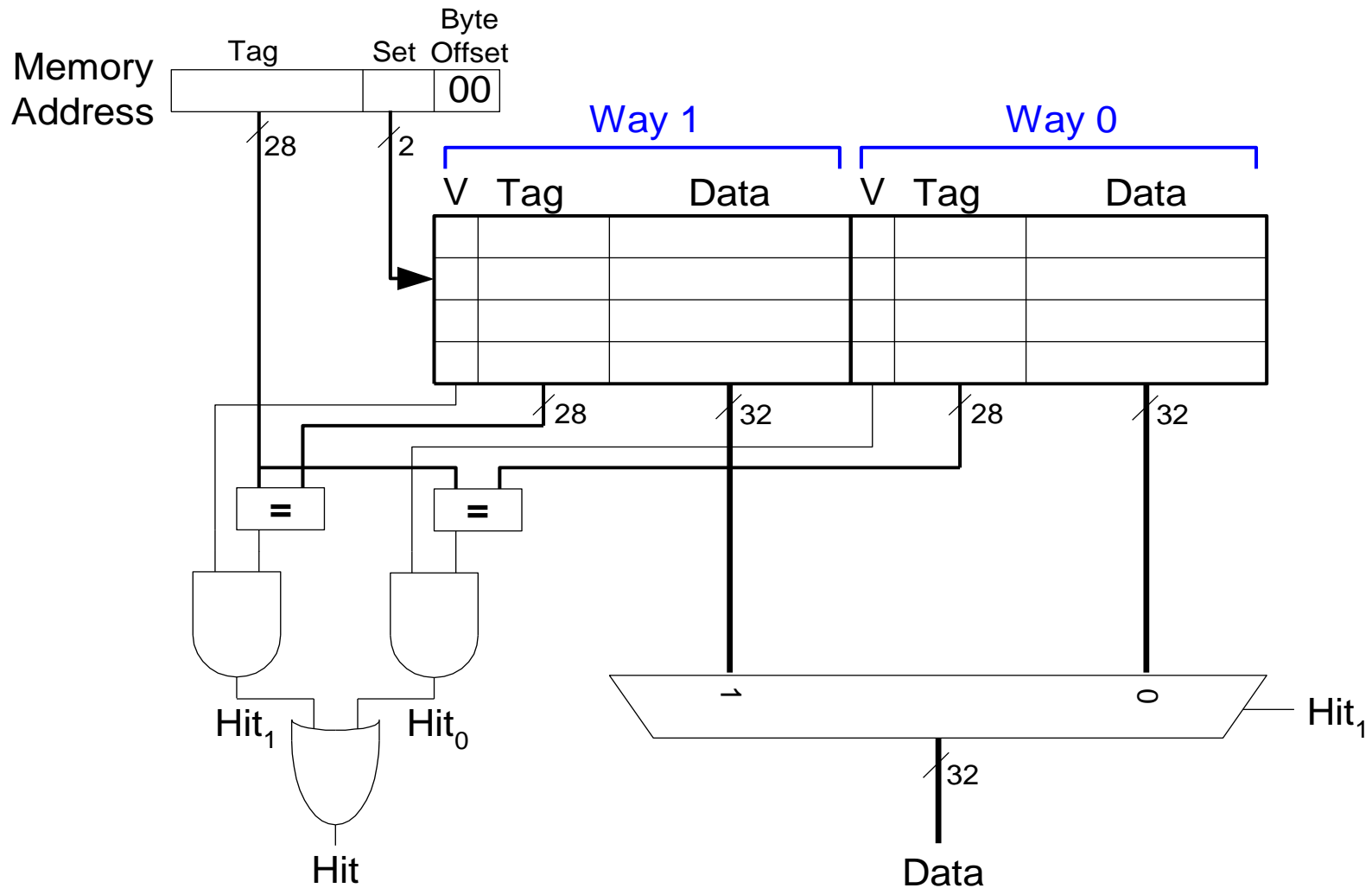
## Chapter 8: Memory Systems

# **Associative Caches**





# N-Way Set Associative Cache







# N-Way Set Assoc. Cache Performance

## # RISC-V assembly code

```
    addi s0, zero, 5
    addi s1, zero, 0
LOOP: beq  s0, zero, DONE
      lw   s2, 0x4(s1)
      lw   s4, 0x24(s1)
      addi s0, s0, -1
      j    LOOP
DONE:
```

## Miss Rate

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

V	Tag	Data	V	Tag	Data	V	Tag	Data	V	Tag	Data	V	Tag	Data	V	Tag	Data	V	Tag	Data

**Reduces conflict misses**

**Expensive to build**





# Chapter 8: Memory Systems

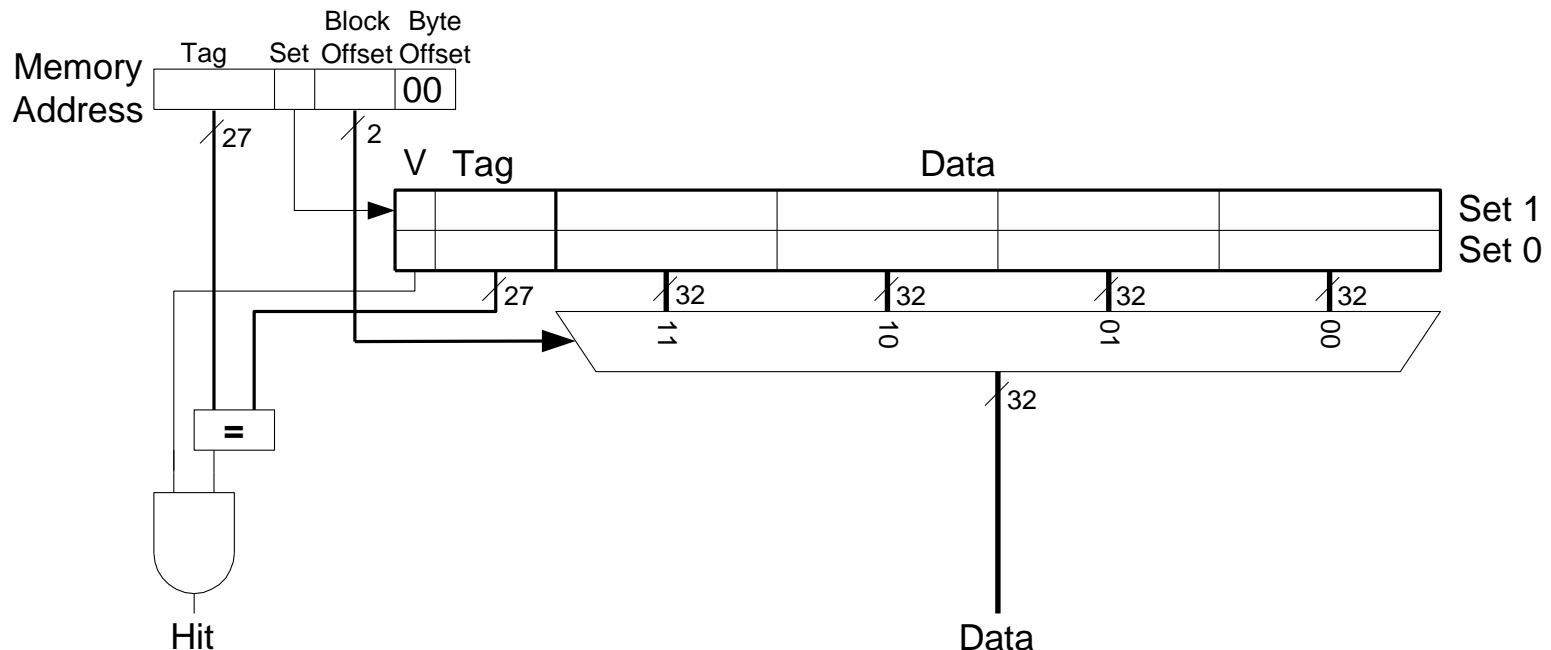
## **Spatial Locality**





# Spatial Locality

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

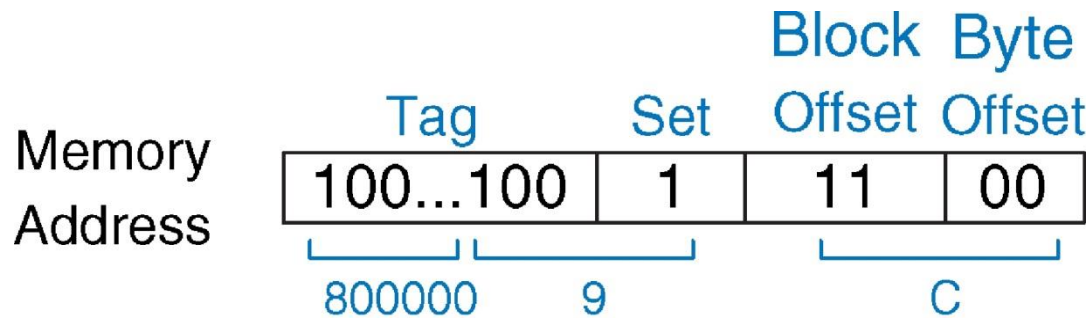




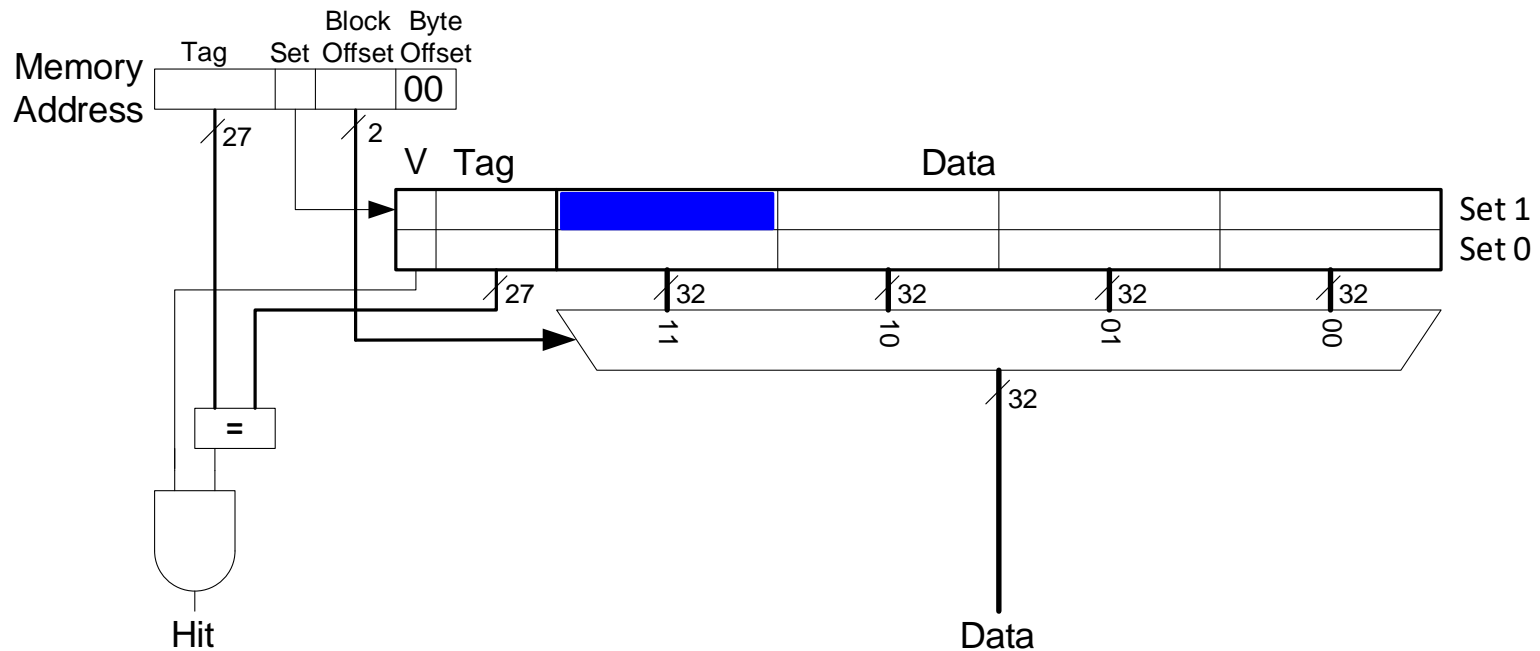




# Cache with Larger Block Size



© 2007 Elsevier, Inc. All rights reserved







# Cache Perf. with Spatial Locality

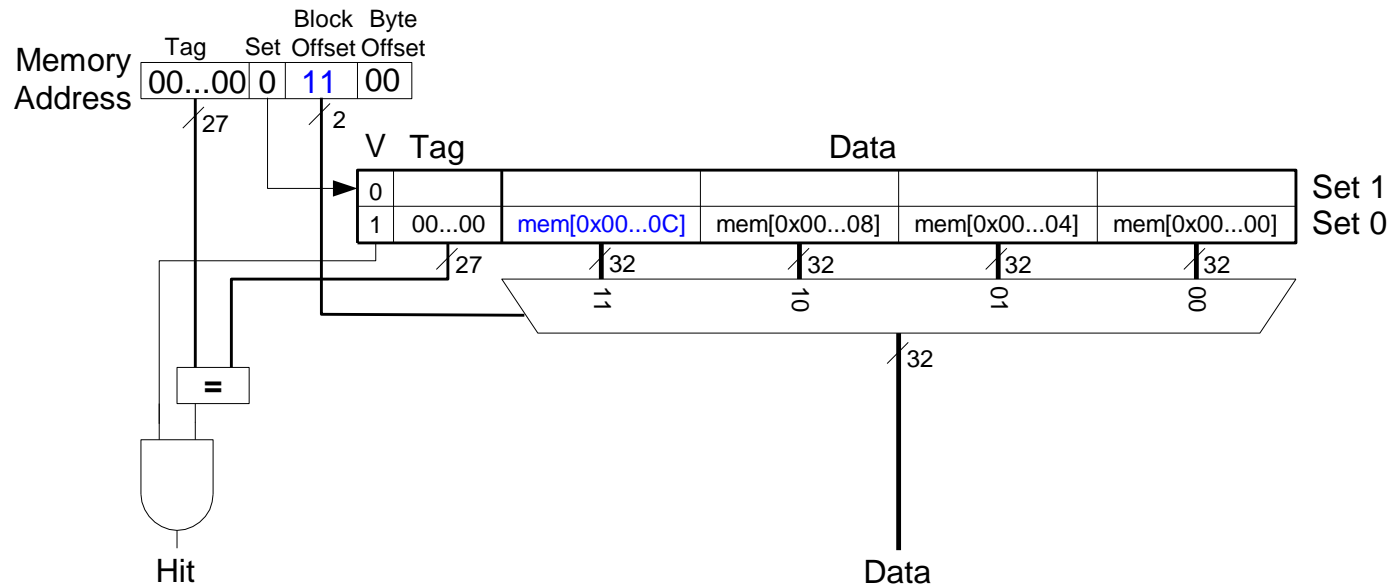
```

    addi s0, zero, 5
    addi s1, zero, 0
LOOP: beq  s0, zero, DONE
      lw   s2, 4(s1)
      lw   s3, 12(s1)
      lw   s4, 8(s1)
      addi s0, s0, -1
      j    LOOP

```

**Miss Rate**

DONE :









# Types of Misses

- **Compulsory:** first time data accessed
- **Capacity:** cache too small to hold all data of interest
- **Conflict:** data of interest maps to same location in cache

**Miss penalty:** time it takes to retrieve a block from lower level of hierarchy





# Cache Organization Recap

- **Capacity:**  $C$
- **Block size:**  $b$
- **Number of blocks in cache:**  $B = C/b$
- **Number of blocks in a set:**  $N$
- **Number of sets:**  $S = B/N$

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





## Chapter 8: Memory Systems

# Cache Replacement Policy







# Replacement Policy

- Cache is too small to hold all data of interest at once
- If cache full: program accesses data X and evicts data Y
- **Capacity miss** when access Y again
- How to choose Y to minimize chance of needing it again?
  - **Least recently used (LRU) replacement**: the least recently used block in a set evicted





# LRU Replacement

## # RISC-V assembly

```
lw s1, 0x04(zero)
```

```
lw s2, 0x24(zero)
```

```
lw s3, 0x54(zero)
```

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





# Chapter 8: Memory Systems

## **Cache Summary**







# Cache Summary

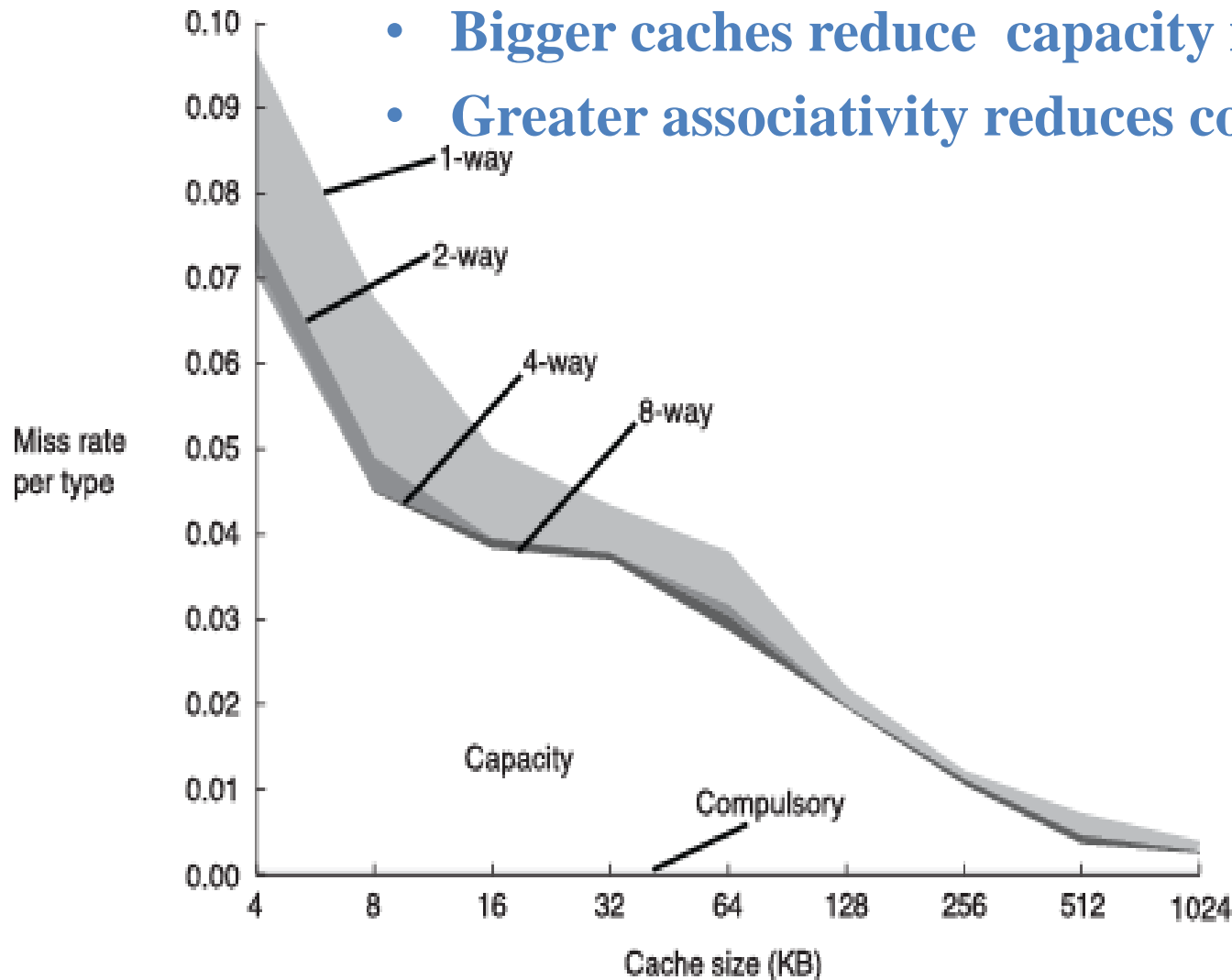
- **What data is held in the cache?**
  - Recently used data (temporal locality)
  - Nearby data (spatial locality)
- **How is data found?**
  - Set is determined by address of data
  - Word within block also determined by address
  - In associative caches, data could be in one of several ways
- **What data is replaced?**
  - Least-recently used way in the set





# Miss Rate Trends

- Bigger caches reduce capacity misses
- Greater associativity reduces conflict misses



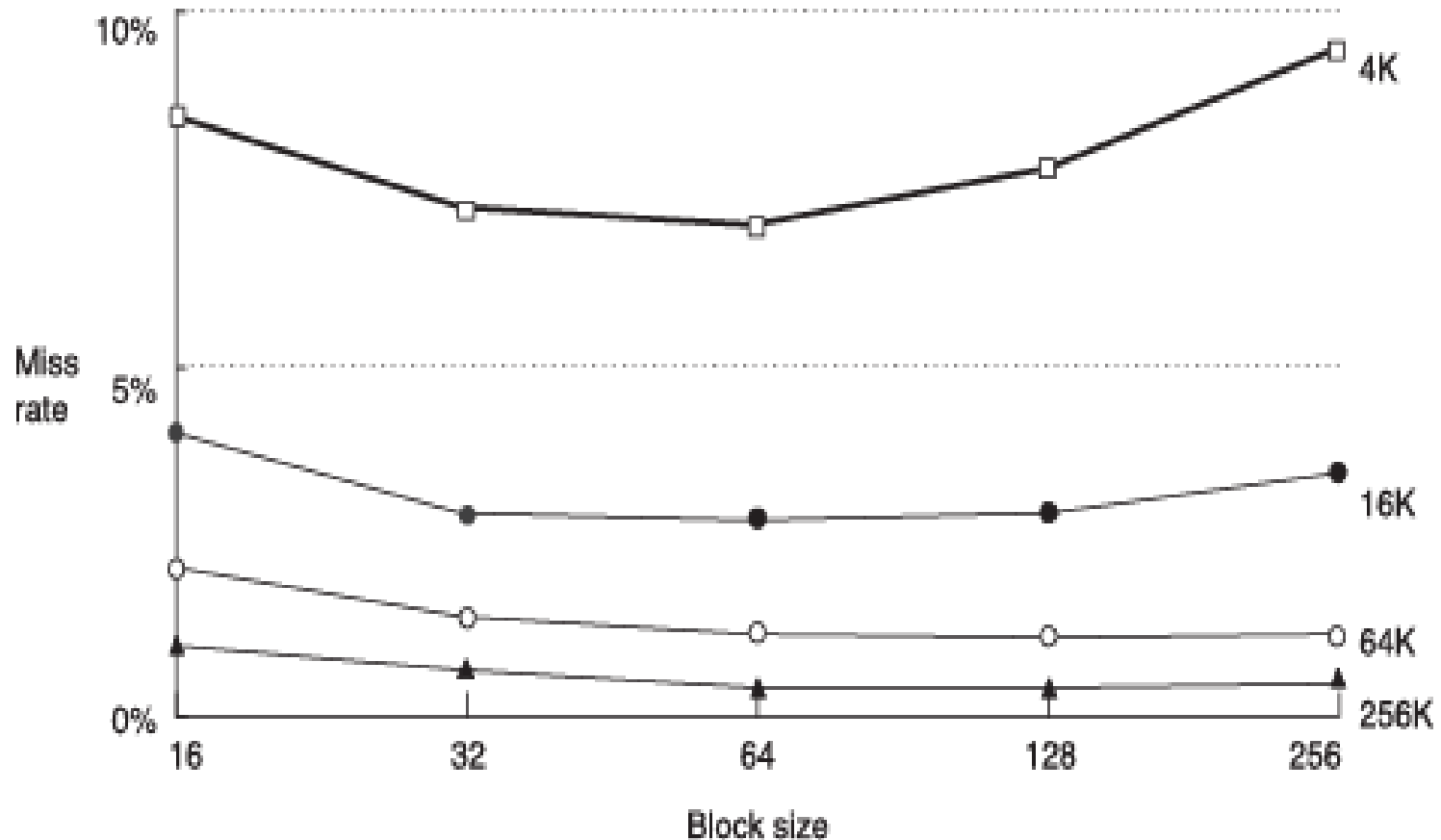
Adapted from Patterson & Hennessy, *Computer Architecture: A Quantitative Approach*, 2011







# Miss Rate Trends



- Bigger blocks reduce compulsory misses
- Bigger blocks increase conflict misses





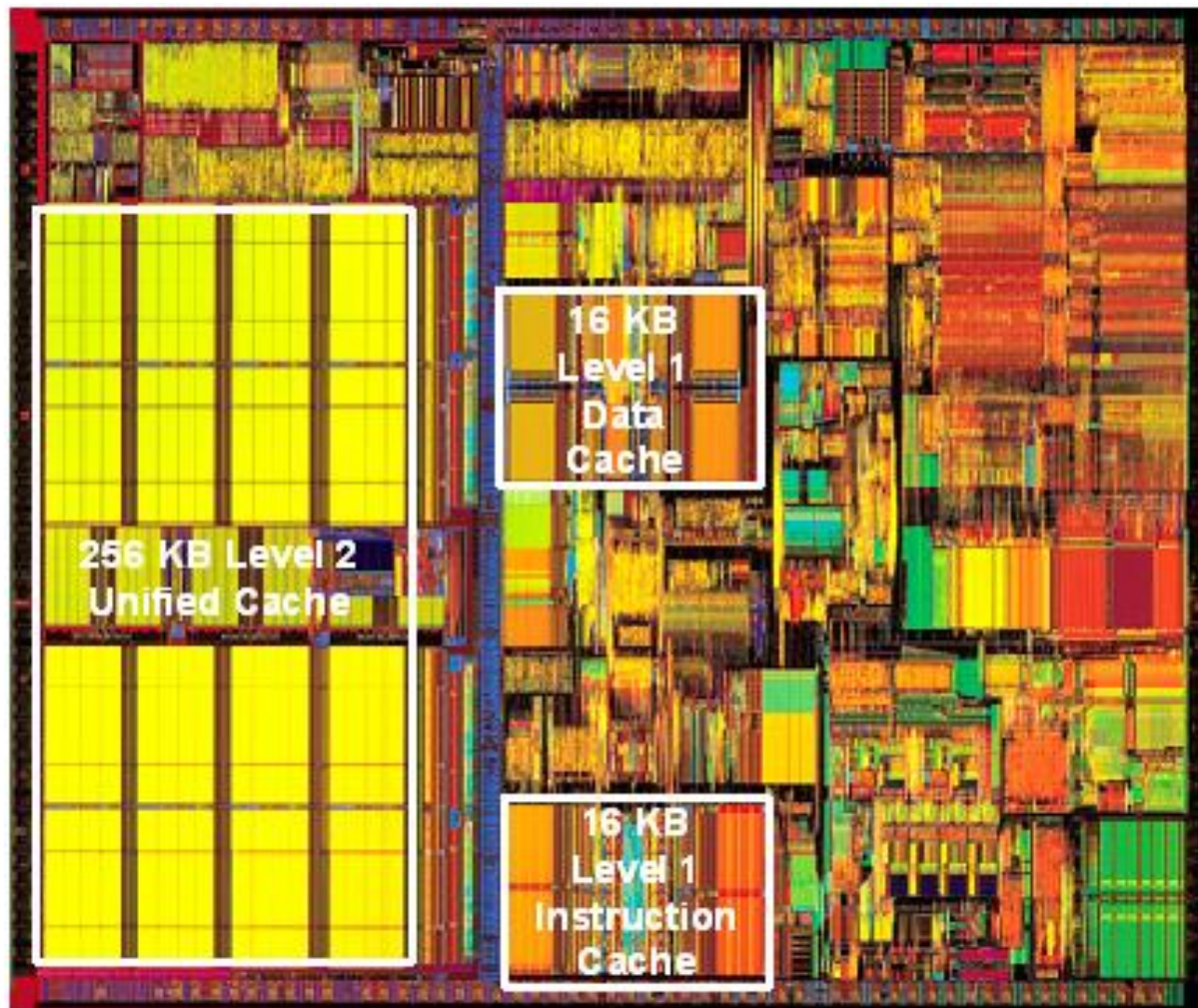
# Multilevel Caches

- Larger caches have lower miss rates, longer access times
- Expand memory hierarchy to multiple levels of caches
- Level 1: small and fast (e.g. 16 KB, 1 cycle)
- Level 2: larger and slower (e.g. 256 KB, 2-6 cycles)
- Most modern PCs have L1, L2, and L3 cache





# Intel Pentium III Die



© Intel Corp.







# Chapter 8: Memory Systems

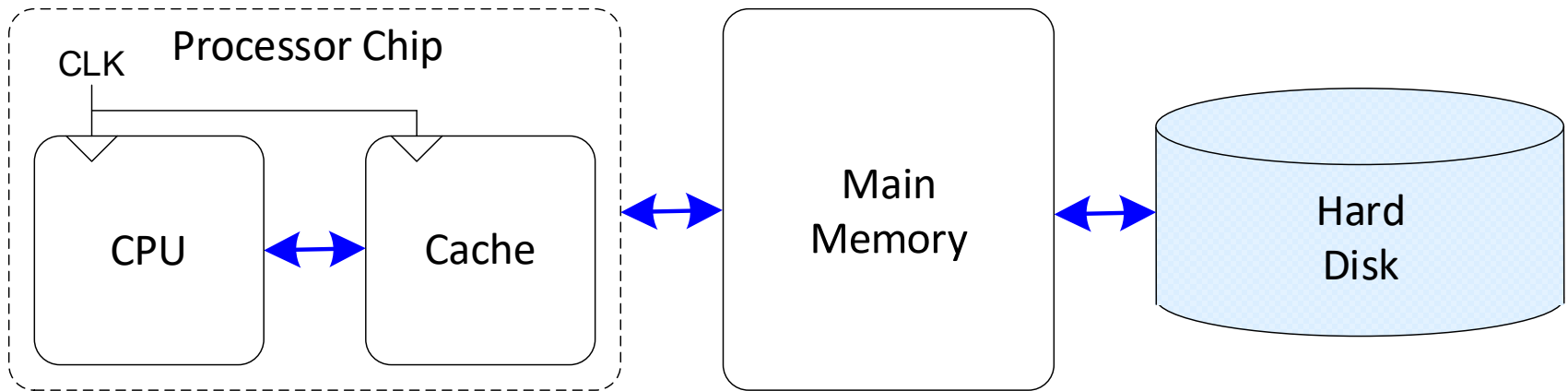
## **Virtual Memory**





# Virtual Memory

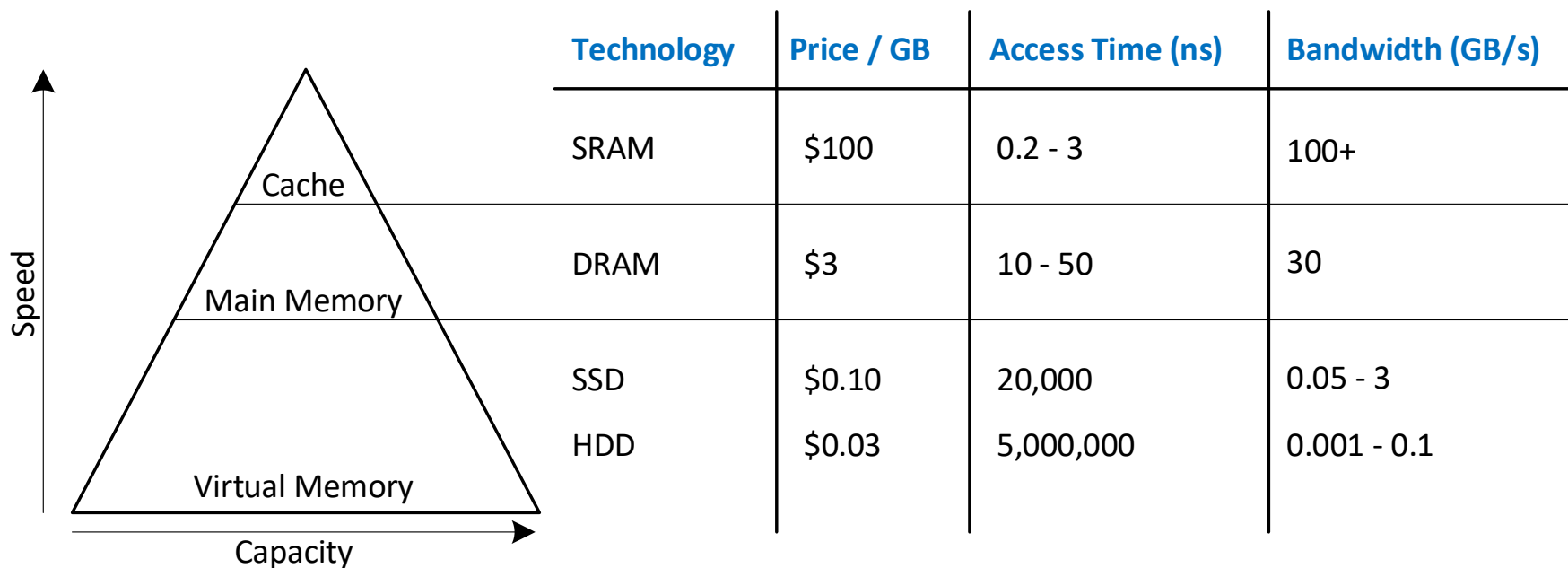
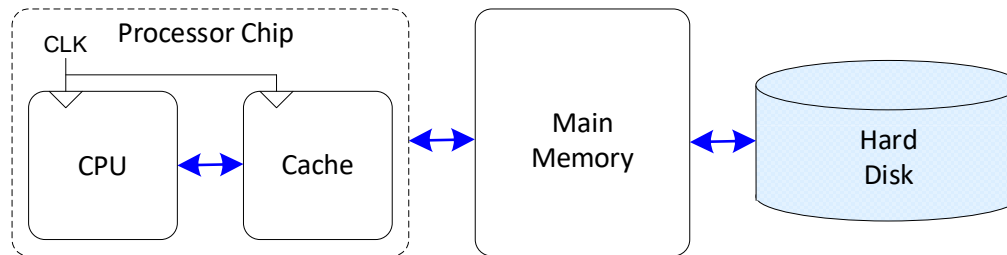
- Gives the illusion of bigger memory
- Main memory (DRAM) acts as cache for hard disk







# Memory Hierarchy



- **Physical Memory:** DRAM (Main Memory)
- **Virtual Memory:** Hard drive
  - Slow, Large, Cheap

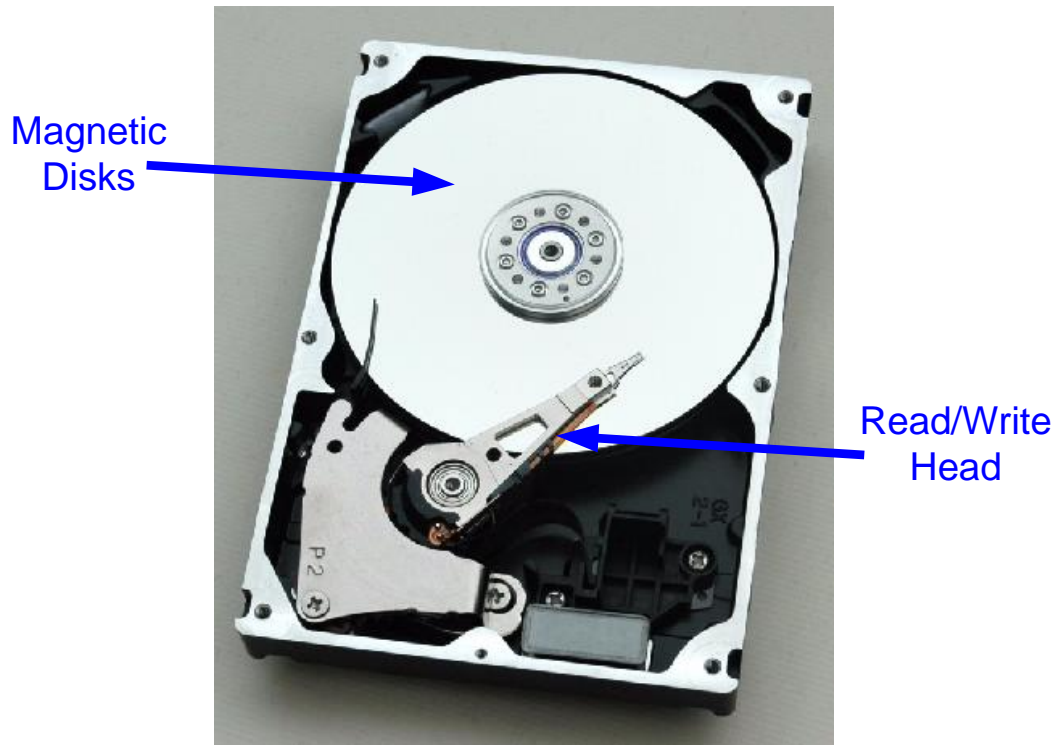






# Memory Hierarchy

## Hard Disk Drive



Takes milliseconds to *seek* correct location on disk

## Solid State Drive



Arshane88 / CC BY-SA 4.0 / Wikimedia Commons





# Virtual Memory

- **Virtual addresses**
  - Programs use virtual addresses
  - Entire virtual address space stored on a hard drive
  - Subset of virtual address data in DRAM
  - CPU translates virtual addresses into *physical addresses* (DRAM addresses)
  - Data not in DRAM fetched from hard drive







# Virtual Memory

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

**Physical memory acts as cache for virtual memory**





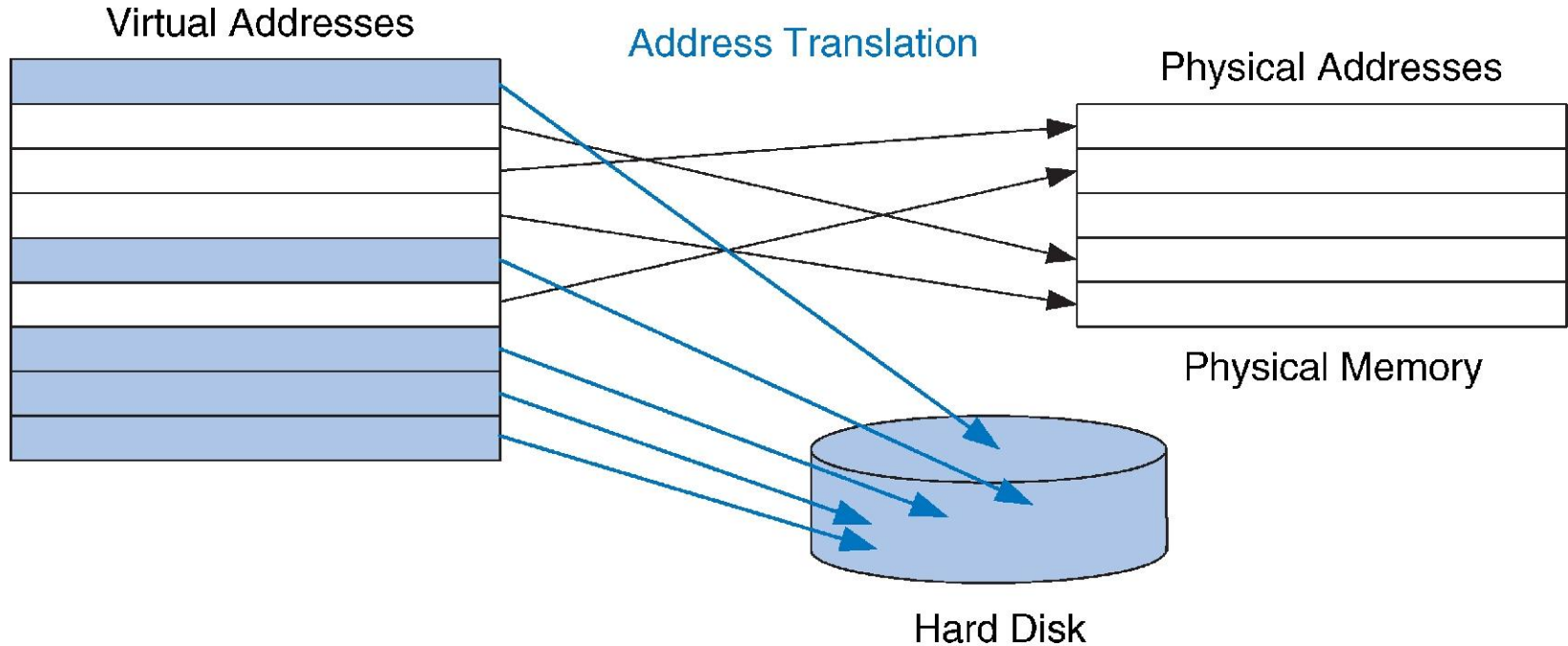
# Virtual Memory Definitions

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





# Virtual Memory Definitions



© 2007 Elsevier, Inc. All rights reserved

**Most accesses hit** in physical memory

But programs have the **large capacity** of virtual memory







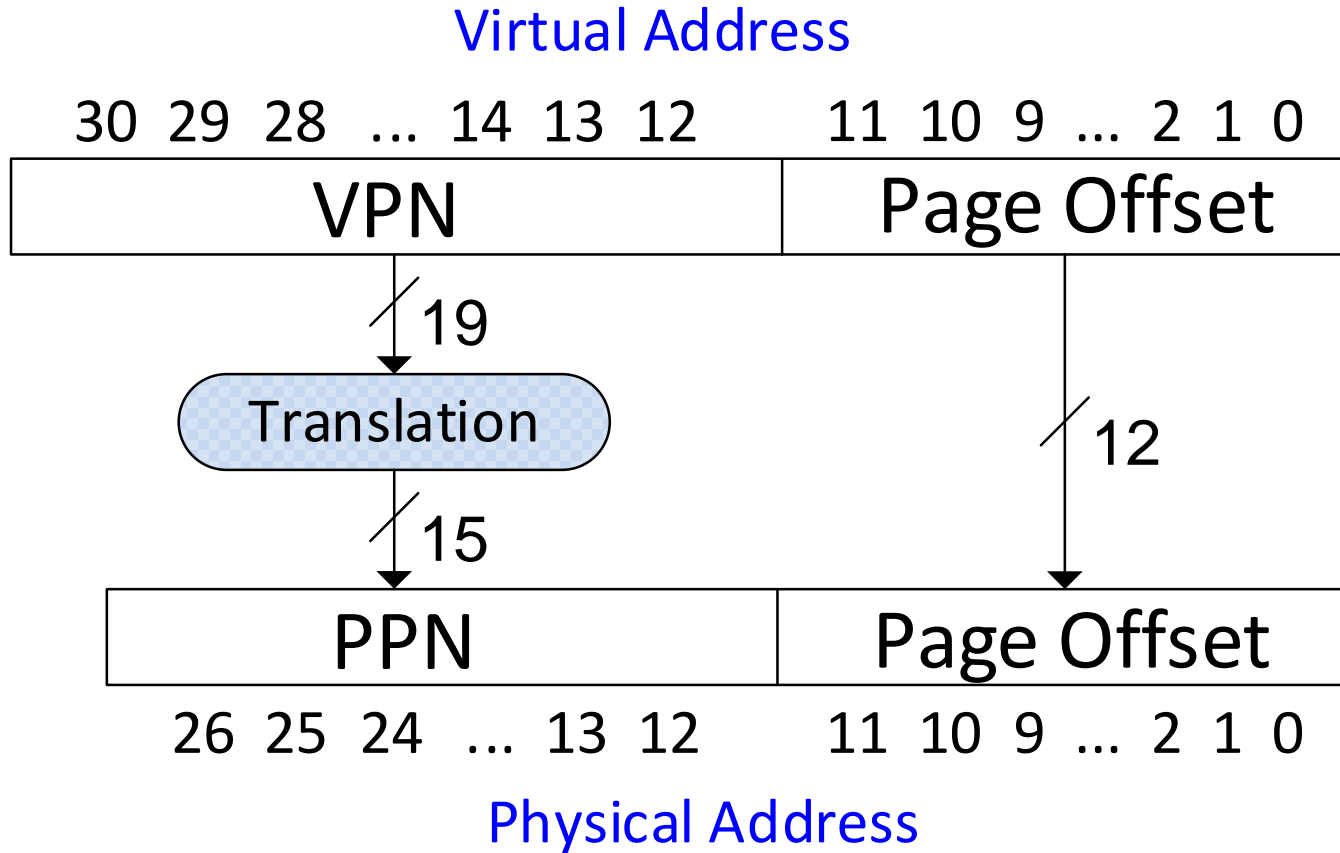
## Chapter 8: Memory Systems

# Address Translation





# Address Translation







# Virtual Memory Example

- **System:**
  - Virtual memory size: 2 GB =  $2^{31}$  bytes
  - Physical memory size: 128 MB =  $2^{27}$  bytes
  - Page size: 4 KB =  $2^{12}$  bytes

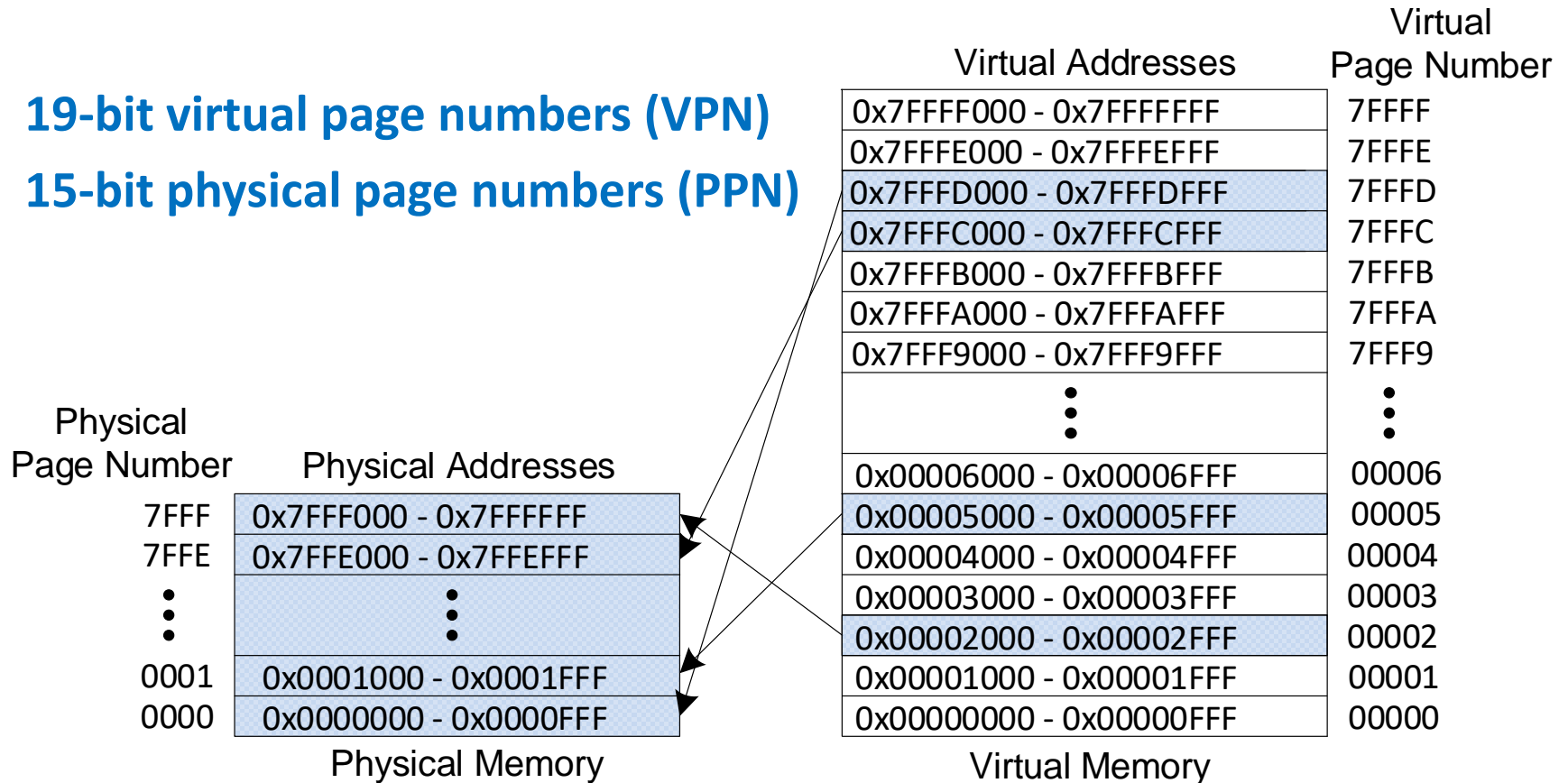






# Virtual Memory Example

- 19-bit virtual page numbers (VPN)
- 15-bit physical page numbers (PPN)

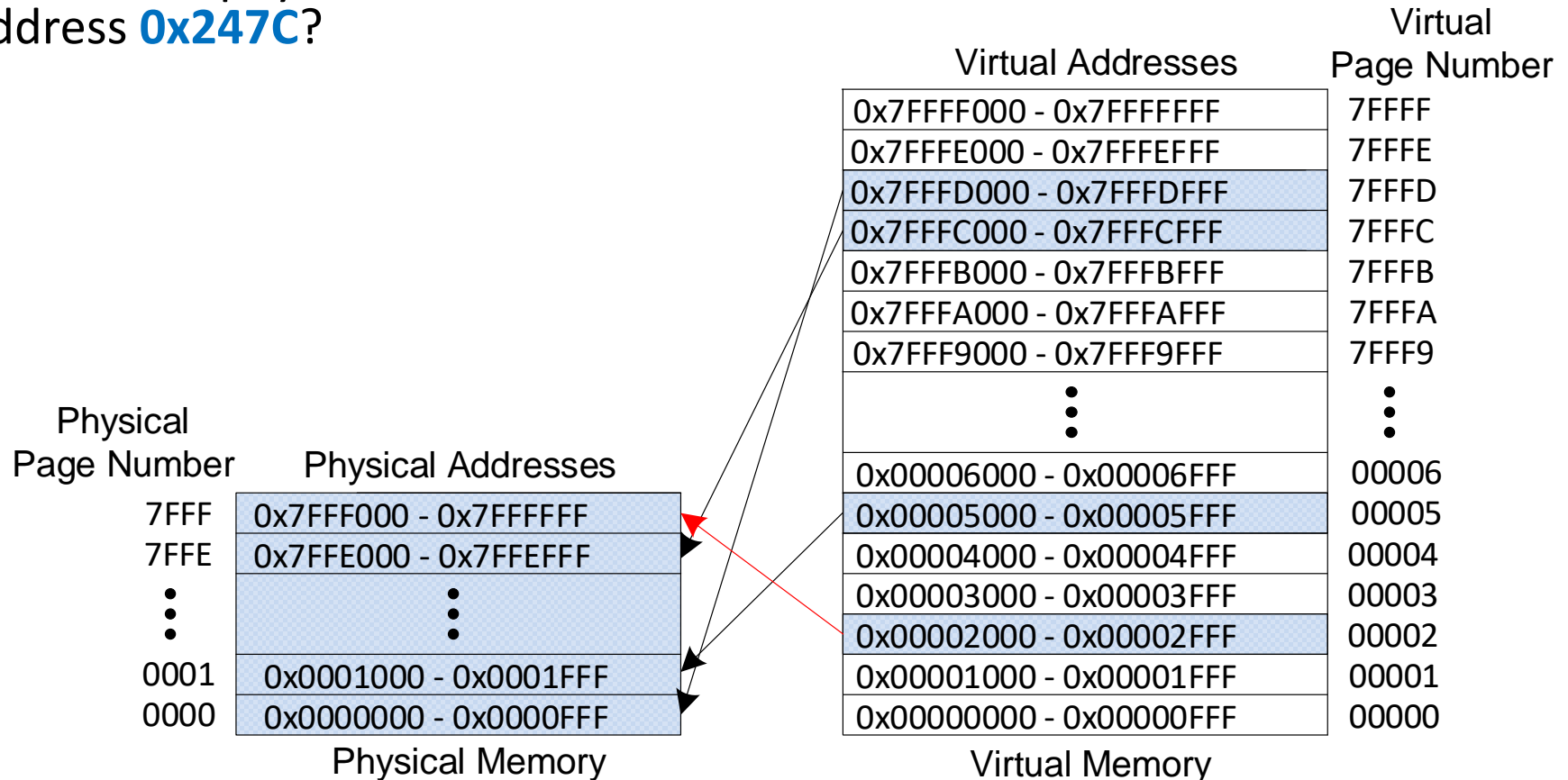






# Virtual Memory Example

What is the physical address of virtual address **0x247C**?









# Chapter 8: Memory Systems

## **Page Table**





# How to Perform Translation

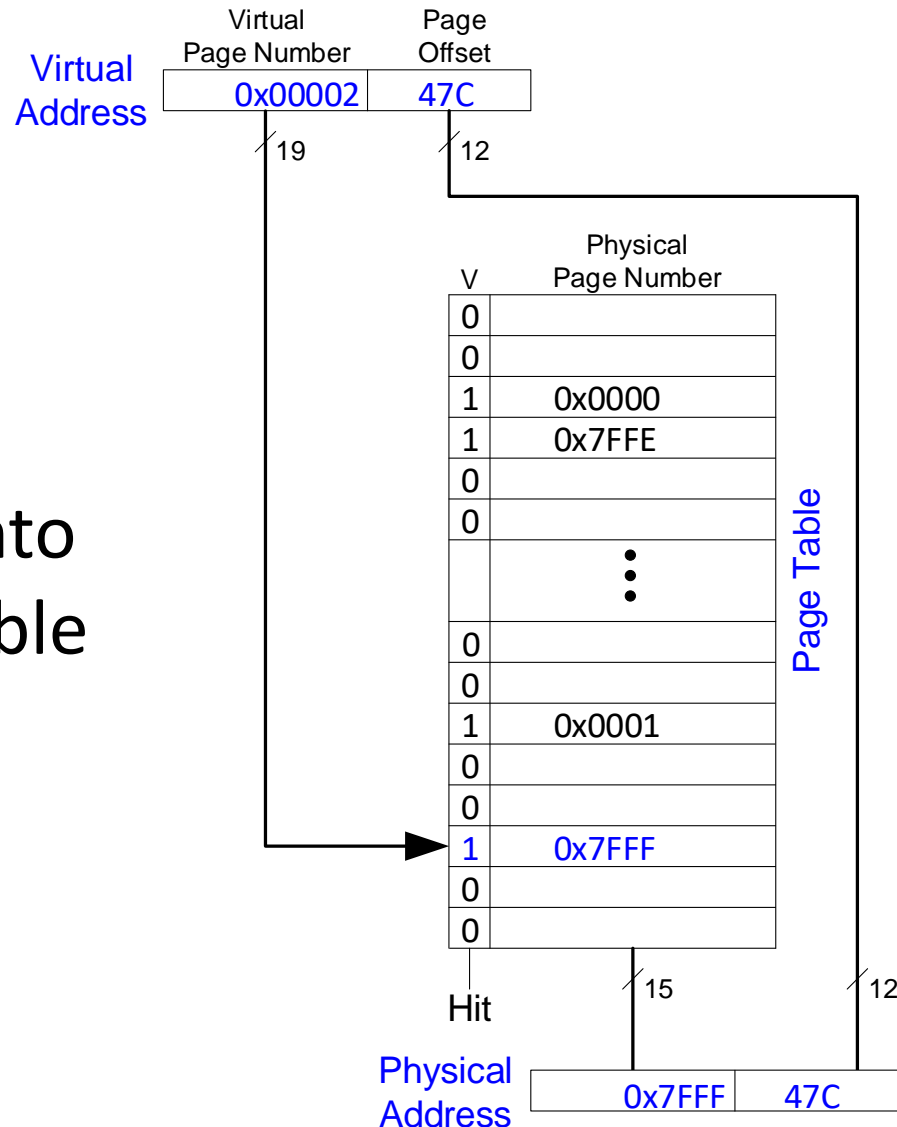
- **Page table**
  - Entry for each virtual page
  - Entry fields:
    - **Valid bit:** 1 if page in physical memory
    - **Physical page number:** where the page is located





# Page Table Example

**VPN** is  
**index** into  
page table









# Page Table Example 1

What is the physical address of virtual address **0x5F20**?

V	Physical Page Number	Virtual Page Number
0		7FFFF
0		7FFFE
1	0x0000	7FFFD
1	0x7FFE	7FFFC
0		7FFFB
0		7FFFA
	⋮	⋮
0		00007
0		00006
1	0x0001	00005
0		00004
0		00003
1	0x7FFF	00002
0		00001
0		00000

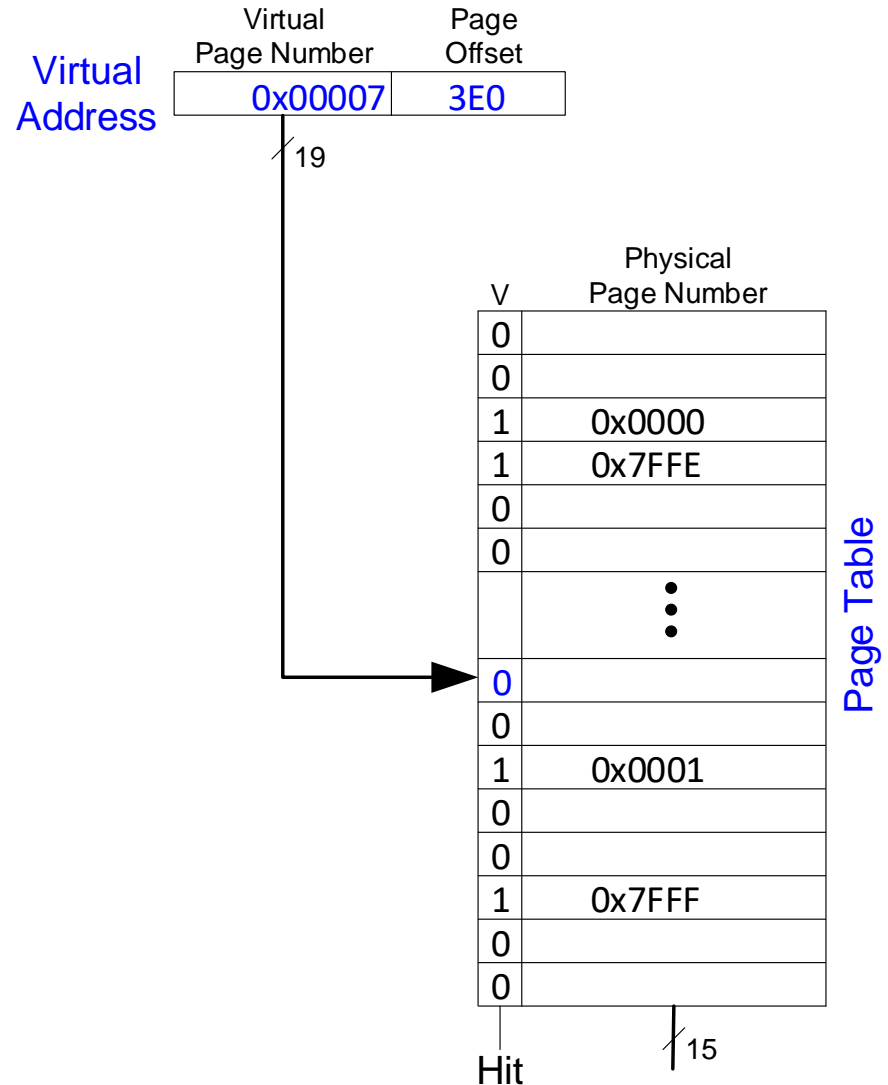
Page Table





# Page Table Example 2

What is the physical address of virtual address **0x73E4**?







# Page Table Challenges

- Page table is **large**
  - usually located in physical memory
- Load/store requires **2 main memory accesses**:
  - one for translation (page table read)
  - one to access data (after translation)
- Cuts memory performance in half
  - *Unless we get clever...*







## Chapter 8: Memory Systems

# **Translation Lookaside Buffer (TLB)**





# Translation Lookaside Buffer (TLB)

- Small cache of most recent translations
- Reduces number of memory accesses for *most* loads/stores from 2 to **1**







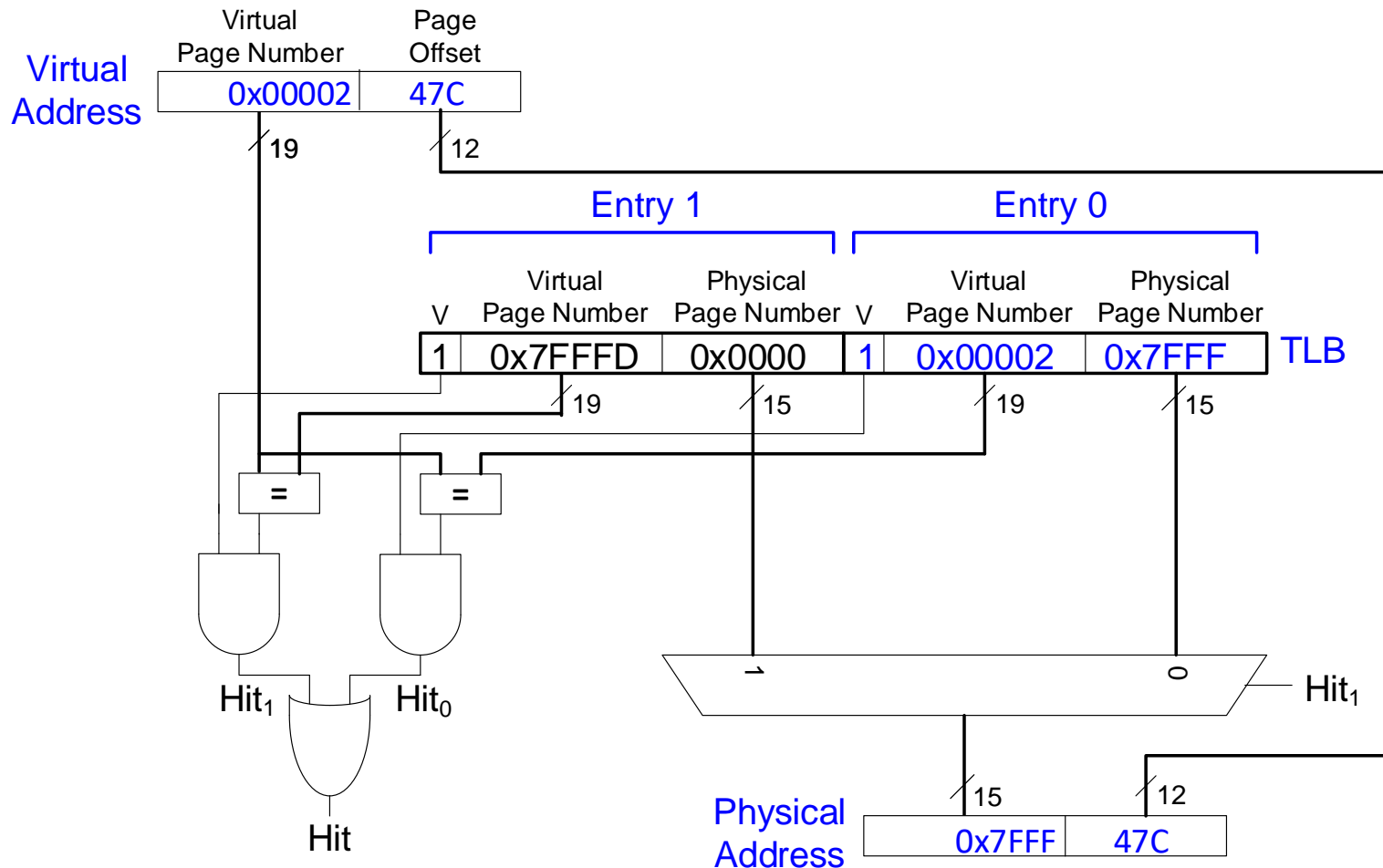
# TLB

- **Page table accesses:** high temporal locality
  - Large page size, so consecutive loads/stores likely to access same page
- **TLB**
  - **Small:** accessed in  $< 1$  cycle
  - Typically **16 - 512 entries**
  - **Fully associative**
  - **$> 99\%$**  hit rates typical
  - **Reduces number of memory accesses** for most loads/stores from 2 to 1





# Example: 2-entry TLB







# Chapter 8: Memory Systems

## **Virtual Memory Summary**







# Memory Protection

- **Multiple processes** (programs) run at once
- Each process has its **own page table**
- Each process can use **entire virtual address space**
- A process can only access a **subset of physical pages**: those mapped in its own page table





# Virtual Memory Summary

- Virtual memory increases **capacity**
- A subset of virtual pages in physical memory
- **Page table** maps virtual pages to physical pages – address translation
- A **TLB** speeds up address translation
- Different page tables for different programs provides **memory protection**





# About these Notes

**Digital Design and Computer Architecture Lecture Notes**

**© 2021 Sarah Harris and David Harris**

**These notes may be used and modified for educational and/or non-commercial purposes so long as the source is attributed.**





