

CSE 213

Computer Architecture

Lecture 1: History of Computer

Military Institute of Science and Technology

Computer System Organization

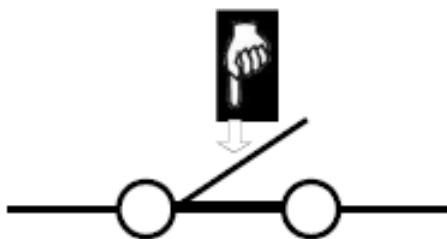
The most amazing and likely to be most long-lived invention of the 1800's was...

THE ELECTRIC SWITCH

Basic Building Blocks: A switch



A switch is a simple device that can act as a conductor or isolator

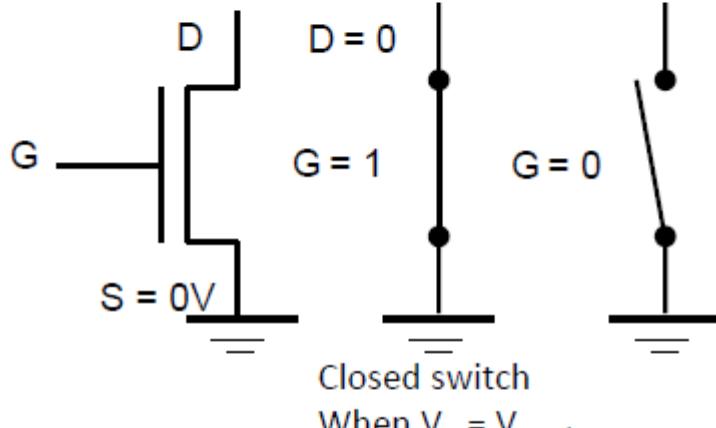


Can be used for amazing things...

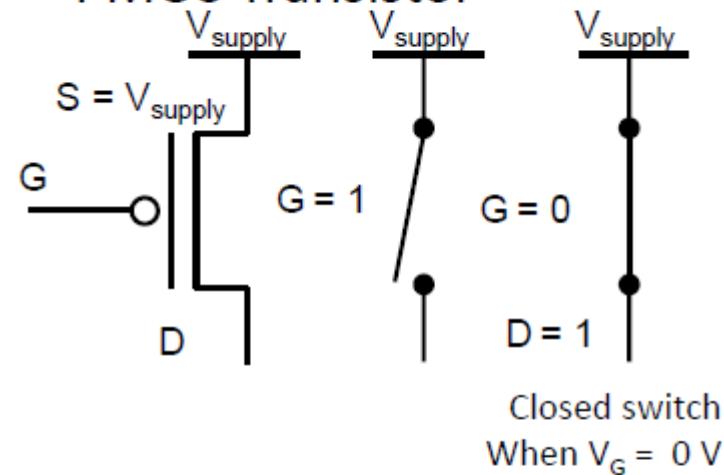


NMOS and PMOS Transistors

- NMOS Transistor



- PMOS Transistor

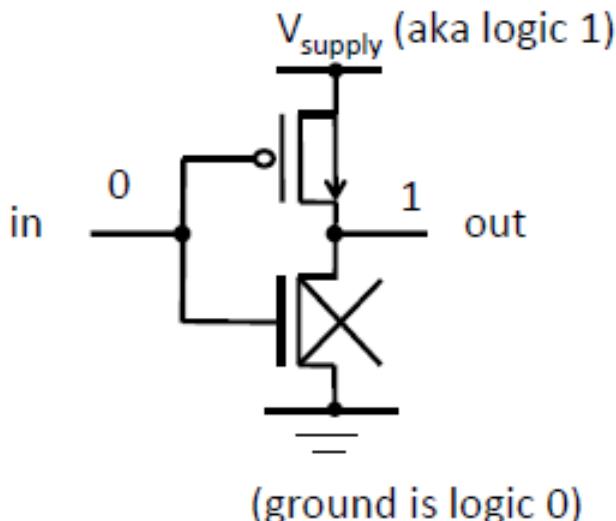


- Connect source to drain when gate = 1
- N-channel transistor

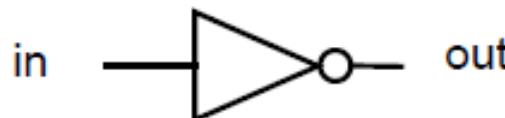
- Connect source to drain when gate = 0
- P-channel transistor

V_S : voltage at the source
 V_D : voltage at the drain
 V_{supply} : max voltage (aka a logical 1)
— (ground): min voltage (aka a logical 0)

Inverter



- Function: NOT
- Called an inverter
- Symbol:



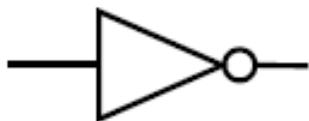
Truth table

In	Out
0	1
1	0

- Useful for taking the inverse of an input
- CMOS: complementary-symmetry metal–oxide–semiconductor

Building Functions

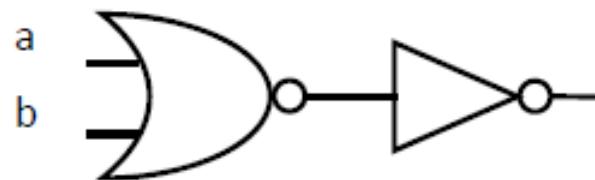
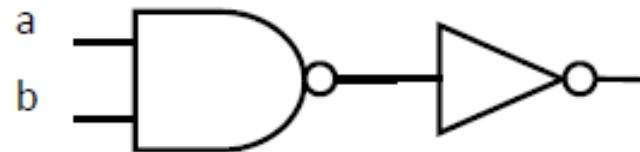
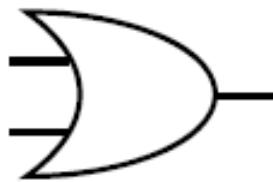
NOT:



AND:



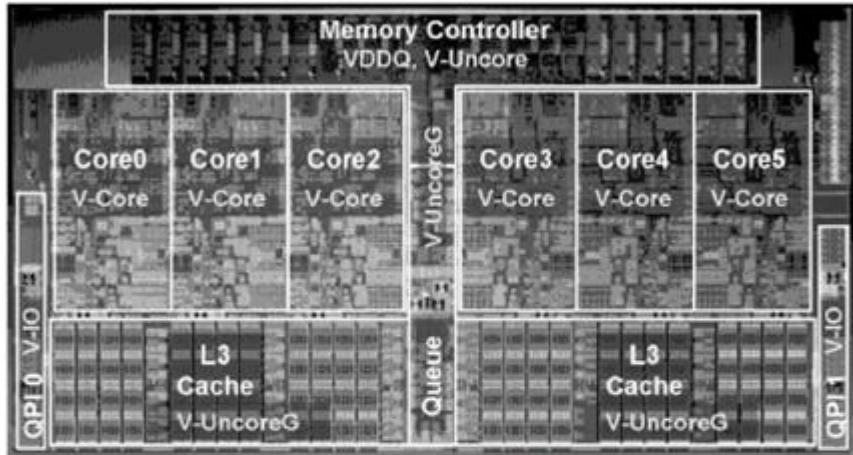
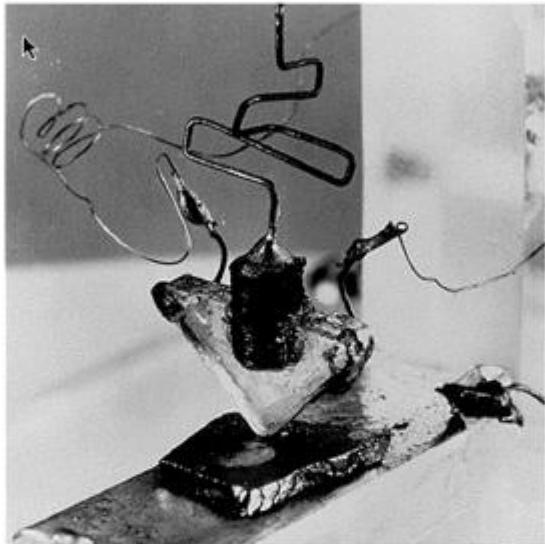
OR:



NAND and NOR are universal

- Can implement any function with NAND or just NOR gates
- useful for manufacturing

Then and Now



http://www.theregister.co.uk/2010/02/03/intel_westmere_ep_preview/

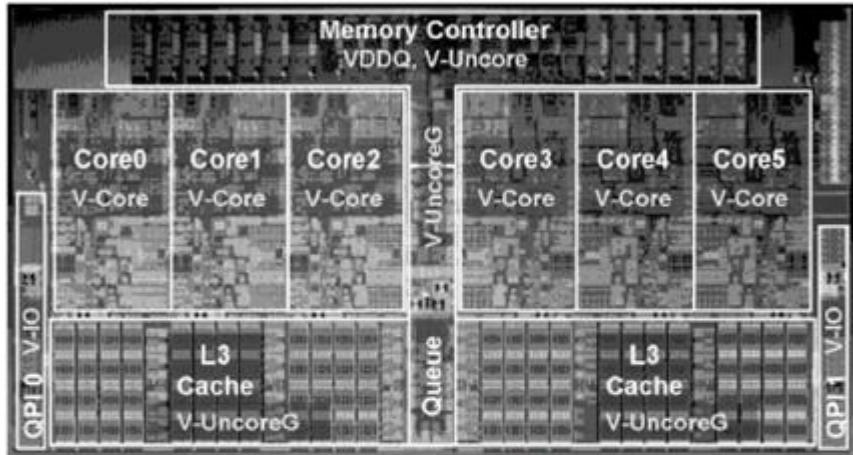
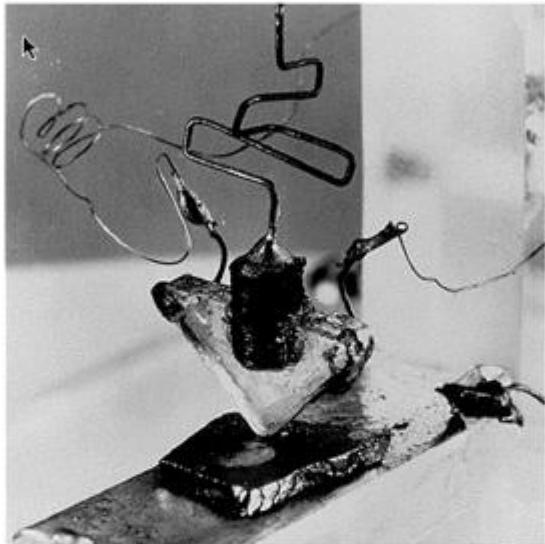
The first transistor

- on a workbench at AT&T Bell Labs in 1947
- Bardeen, Brattain, and Shockley

• An Intel Westmere

- 1.17 billion transistors
- 240 square millimeters
- 32 nanometer: transistor gate width
- Six processing cores
- Release date: January 2010

Then and Now



http://www.theregister.co.uk/2010/02/03/intel_westmere_ep_preview/

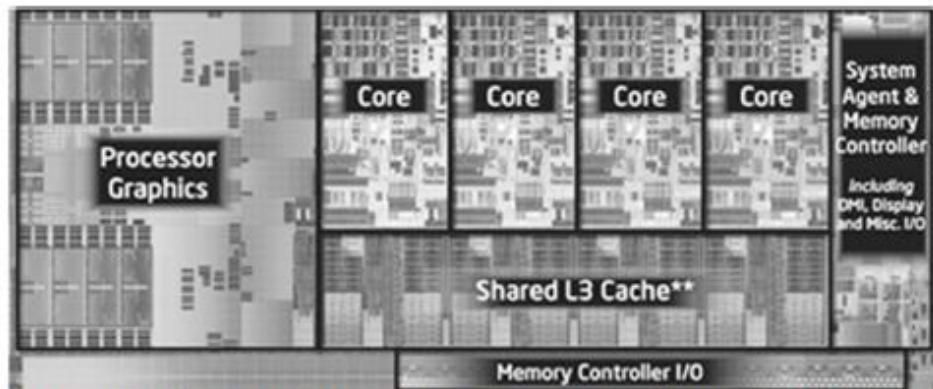
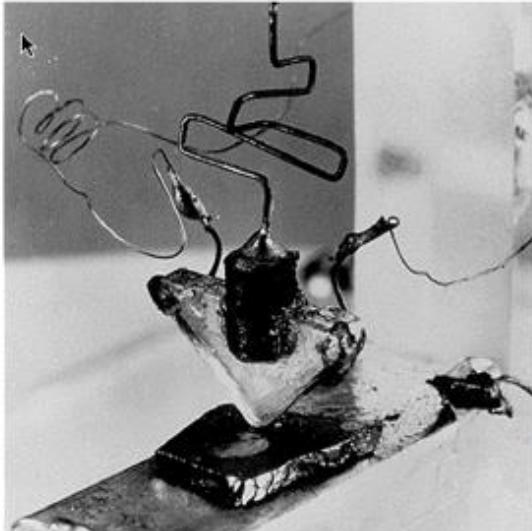
The first transistor

- on a workbench at AT&T Bell Labs in 1947
- Bardeen, Brattain, and Shockley

• An Intel Westmere

- 1.17 billion transistors
- 240 square millimeters
- 32 nanometer: transistor gate width
- Six processing cores
- Release date: January 2010

Then and Now



<http://forwardthinking.pcmag.com/none/296972-intel-releases-ivy-bridge-first-processor-with-tri-gate-transistor>

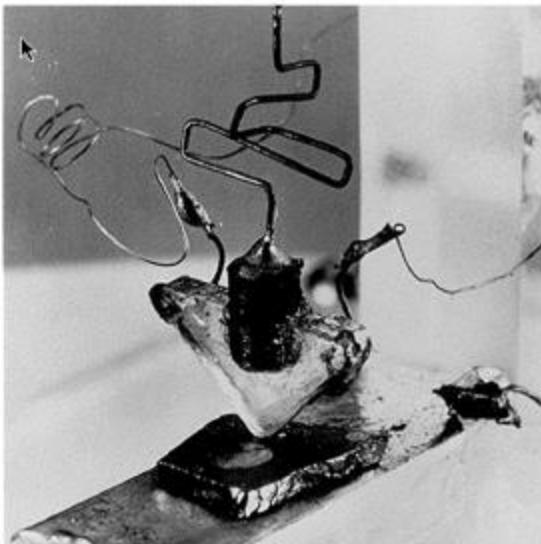
The first transistor

- on a workbench at AT&T Bell Labs in 1947
- Bardeen, Brattain, and Shockley

• An Intel Ivy Bridge

- 1.4 billion transistors
- 160 square millimeters
- 22 nanometer: transistor gate width
- Up to eight processing cores
- Release date: April 2012

Then and Now



<http://www.anandtech.com/show/6386/samsung-galaxy-note-2-review-t-mobile-3>

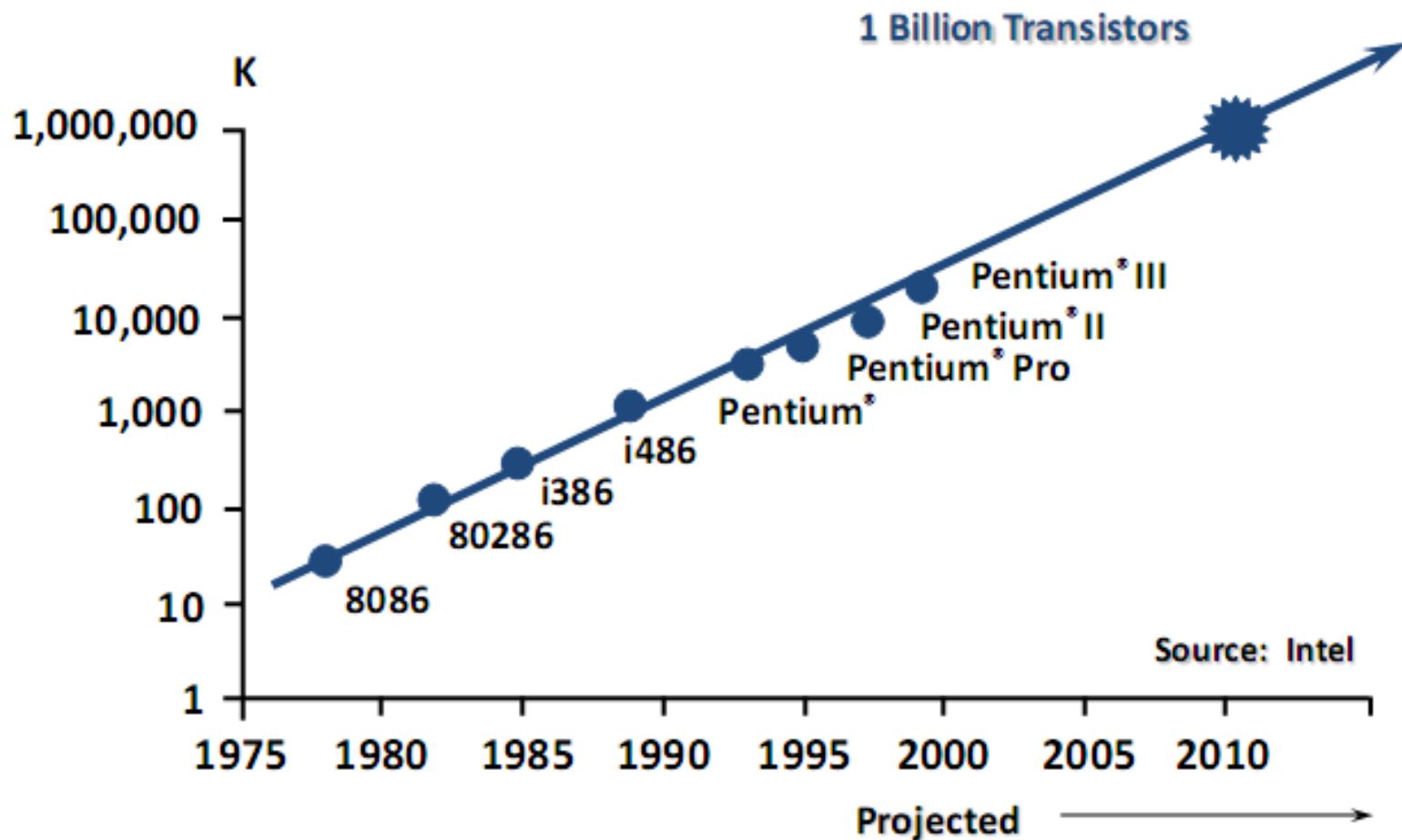
The first transistor

- on a workbench at AT&T Bell Labs in 1947
- Bardeen, Brattain, and Shockley

• Samsung Galaxy Note II

- Eynos 4412 System on a Chip (SoC)
- ARM Cortex-A9 processing core
- 32 nanometer: transistor gate width
- Four processing cores
- Release date: November 2012

Transistor Counts



Moore's Law

The number of transistors integrated on a single die will double every 24 months...

– Gordon Moore, Intel co-founder, 1965

Amazingly Visionary

1971 – 2300 transistors – 1MHz – 4004

1990 – 1M transistors – 50MHz – i486

2001 – 42M transistors – 2GHz – Xeon

2004 – 55M transistors – 3GHz – P4

2007 – 290M transistors – 3GHz – Core 2 Duo

2009 – 731M transistors – 2GHz – Nehalem

2012 – 1400M transistors – 2-3GHz – Ivy Bridge

Moore's Law

1965; Gordon Moore –
co-founder of Intel



Observed number of transistors that could be put on a single chip was doubling every year

The pace slowed to a doubling every 18 months in the 1970's but has sustained that rate ever since

Consequences of Moore's law:

The cost of computer logic and memory circuitry has fallen at a dramatic rate

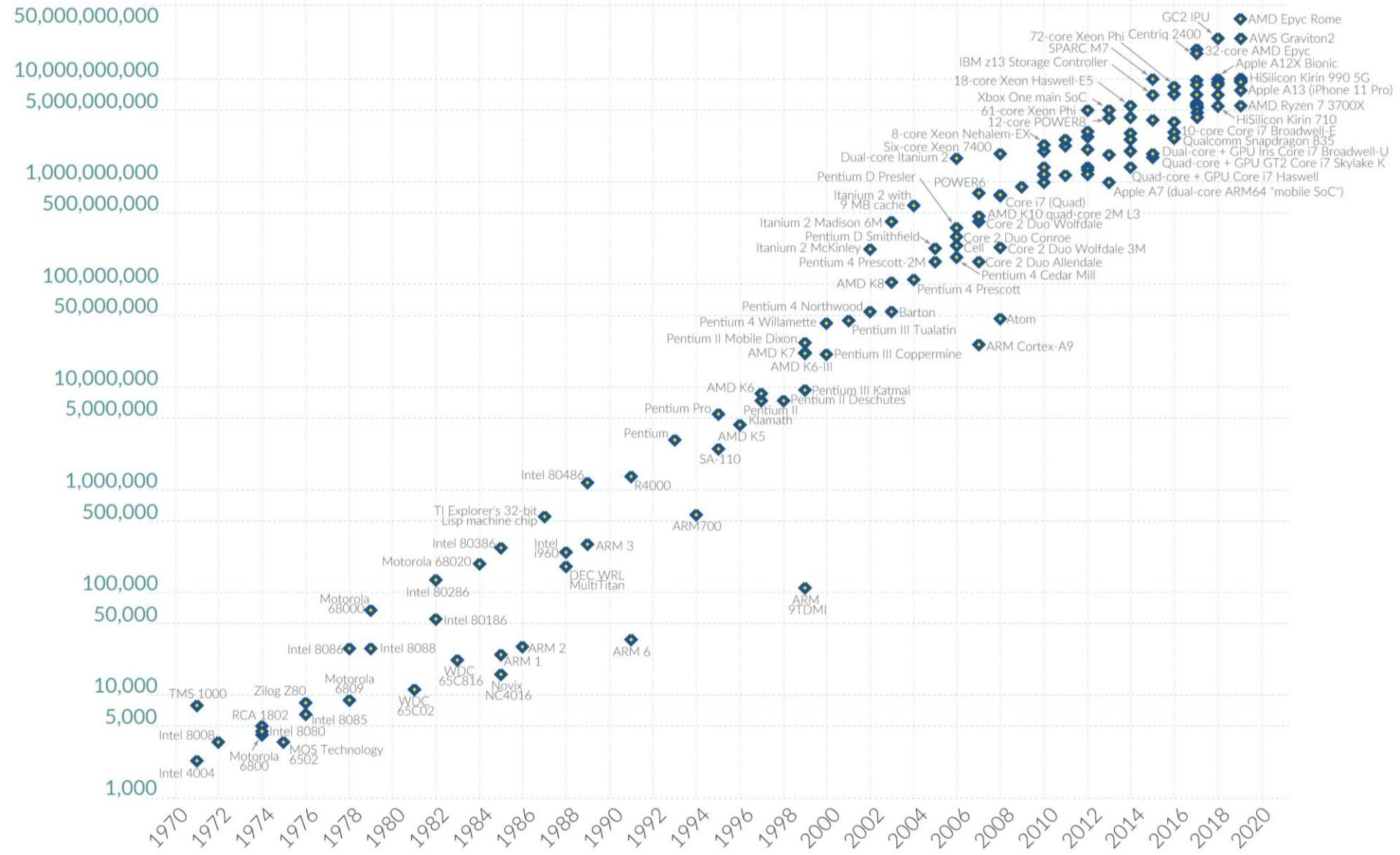
The electrical path length is shortened, increasing operating speed

Computer becomes smaller and is more convenient to use in a variety of environments

Reduction in power and cooling requirements

Fewer interchip connections

Transistor count



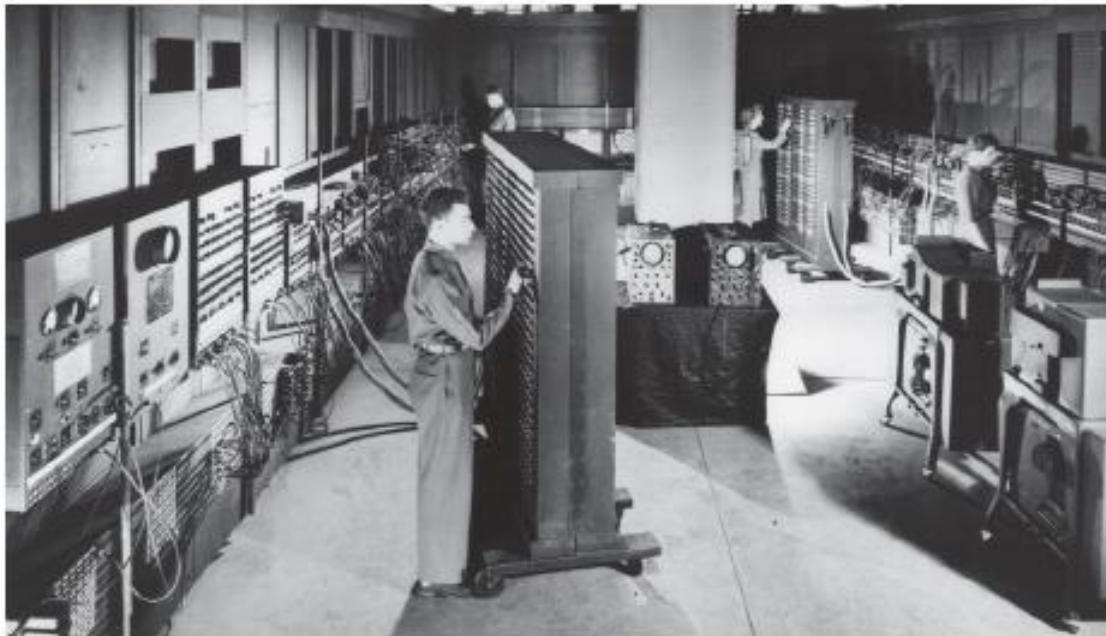
Why large scale integration?

- Less die area, compactness
- More functions with less cost
- Less power consumption
- Less testing requirements at the system level
- Higher reliability, due to high quality on the chip interconnec
- Higher speed, due to reduced interconnect length
- Significant cost savings

Computer Architecture

A bit of History

The first electronic computers



ENIAC (Electronic Numerical Integrator and Calculator) –
The world's first electronic computer

3

ENIAC

- ENIAC provided conditional jumps and was programmable, clearly distinguishing it from earlier calculators
- Programming was done manually by plugging cables and setting switches, and data was entered on punched cards. Programming for typical calculations required from half an hour to a whole day
- ENIAC was a general-purpose machine, limited primarily by a small amount of storage and tedious programming

ENIAC

- J. Presper Eckert and John Mauchly at the Moore School of the University of Pennsylvania
- Funded by the United States Army
- Became operational during World War II but was not publicly disclosed until 1946

ENIAC

Weighed
30
tons

Occupied
1500
square
feet
of
floor
space

Contained
more
than
18,000
vacuum
tubes

140 kW
Power
consumption

Capable
of
5000
additions
per
second

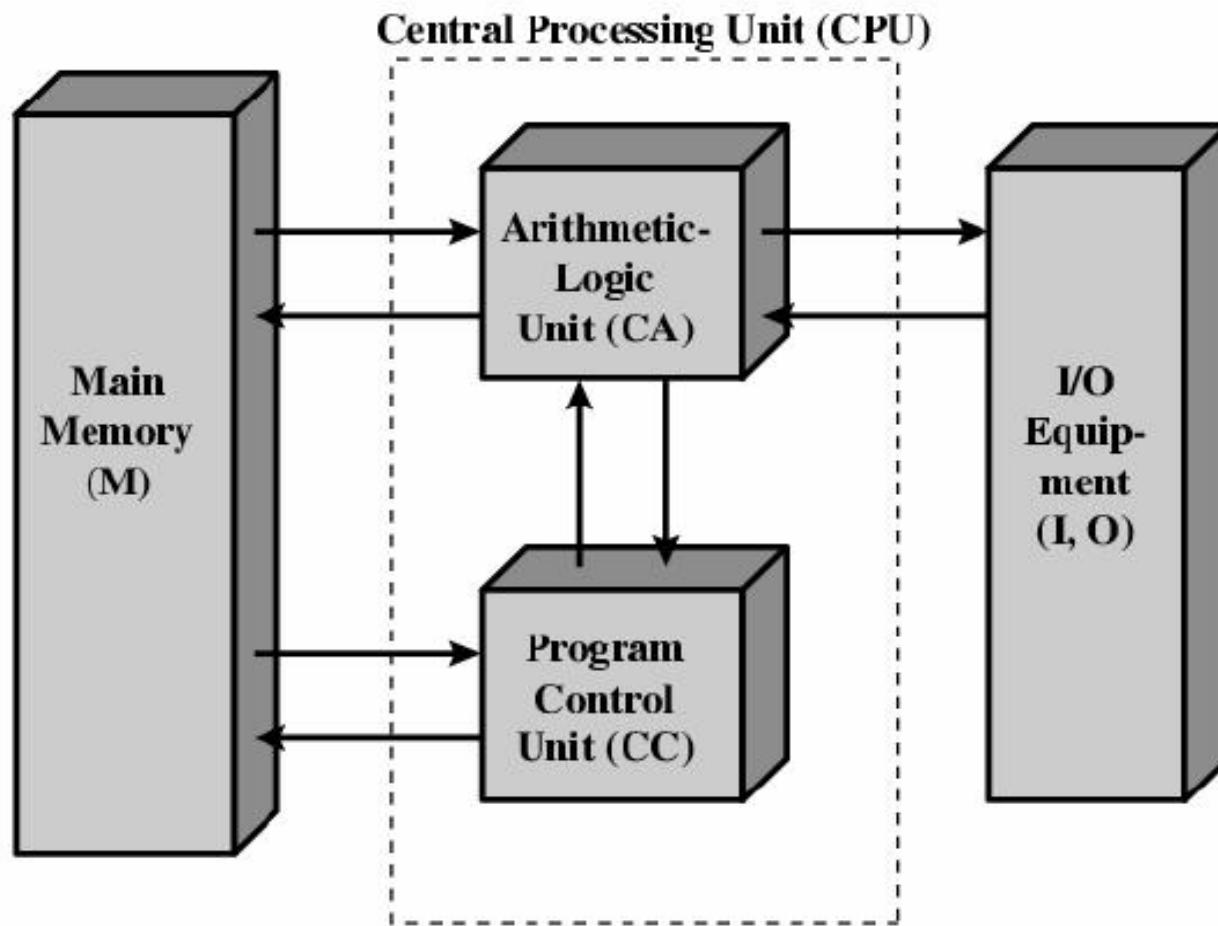
Decimal
rather
than
binary
machine

Memory
consisted
of 20
accumulators,
each
capable
of
holding
a
10 digit
number

Major
drawback
was the need
for manual
programming
by setting
switches
and
plugging/
unplugging
cables

The von Neumann Model

- The invention of stored program computers has been ascribed to a mathematician, John von Neumann, who was a contemporary of Mauchley and Eckert.
- Stored-program computers have become known as **von Neumann Architecture** systems.



IAS (Princeton) computer model by Von Neumann's group.

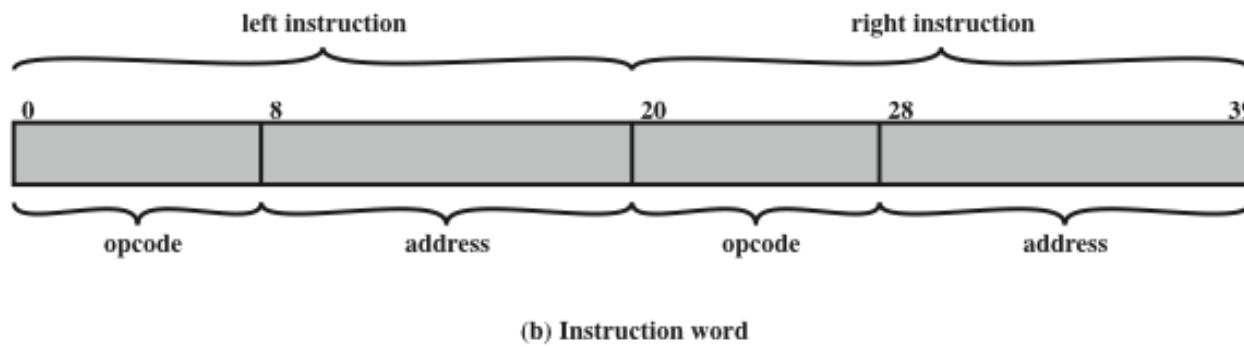
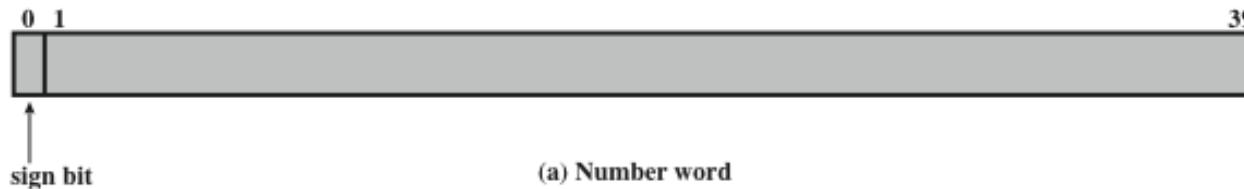
Institute for Advanced Studies(IAS)



IAS Memory Formats

- The memory of the IAS consists of 1000 storage locations (called *words*) of 40 bits each

- Both data and instructions are stored there
- Numbers are represented in binary form and each instruction is a binary code





Registers

Memory buffer register (MBR)

- Contains a word to be stored in memory or sent to the I/O unit
- Or is used to receive a word from memory or from the I/O unit

Memory address register (MAR)

- Specifies the address in memory of the word to be written from or read into the MBR

Instruction register (IR)

- Contains the 8-bit opcode instruction being executed

Instruction buffer register (IBR)

- Employed to temporarily hold the right-hand instruction from a word in memory

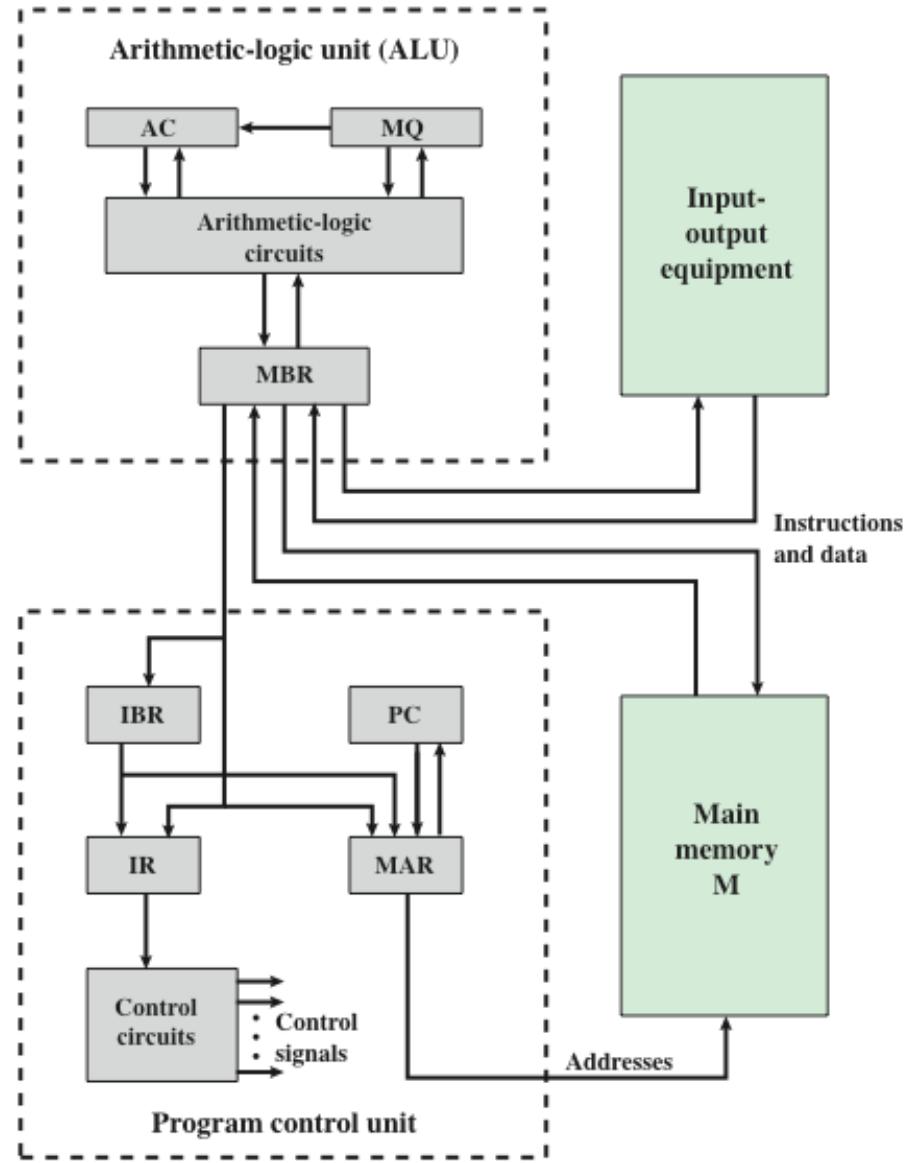
Program counter (PC)

- Contains the address of the next instruction pair to be fetched from memory

Accumulator (AC) and multiplier quotient (MQ)

- Employed to temporarily hold operands and results of ALU operations

Structure of IAS Computer





Commercial Computers

UNIVAC



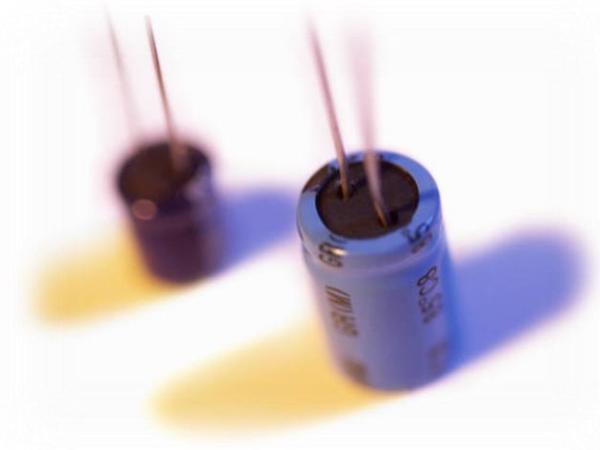
- 1947 – Eckert and Mauchly formed the Eckert-Mauchly Computer Corporation to manufacture computers commercially
- UNIVAC I (Universal Automatic Computer)
 - First successful commercial computer
 - Was intended for both scientific and commercial applications
 - Commissioned by the US Bureau of Census for 1950 calculations
- The Eckert-Mauchly Computer Corporation became part of the UNIVAC division of the Sperry-Rand Corporation
- UNIVAC II – delivered in the late 1950's
 - Had greater memory capacity and higher performance
- Backward compatible



History of Computers

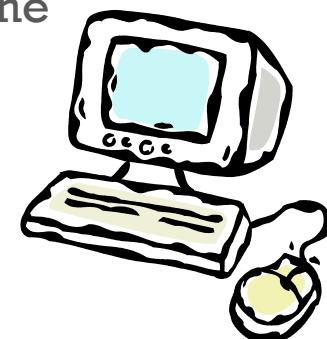
Second Generation: Transistors

- Smaller
- Cheaper
- Dissipates less heat than a vacuum tube
- Is a *solid state device* made from silicon
- Was invented at Bell Labs in 1947
- It was not until the late 1950's that fully transistorized computers were commercially available



Second Generation Computers

- Introduced:
 - More complex arithmetic and logic units and control units
 - The use of high-level programming languages
 - Provision of *system software* which provided the ability to:
 - load programs
 - move data to peripherals and libraries
 - perform common computations
- Appearance of the Digital Equipment Corporation (DEC) in 1957
- PDP-1 was DEC's first computer
- This began the mini-computer phenomenon that would become so prominent in the third generation



History of Computers

Third Generation: Integrated Circuits

- 1958 – the invention of the integrated circuit
- *Discrete component*
 - Single, self-contained transistor
 - Manufactured separately, packaged in their own containers, and soldered or wired together onto masonite-like circuit boards
 - Manufacturing process was expensive and cumbersome
- The two most important members of the third generation were the IBM System/360 and the DEC PDP-8



Microelectronics

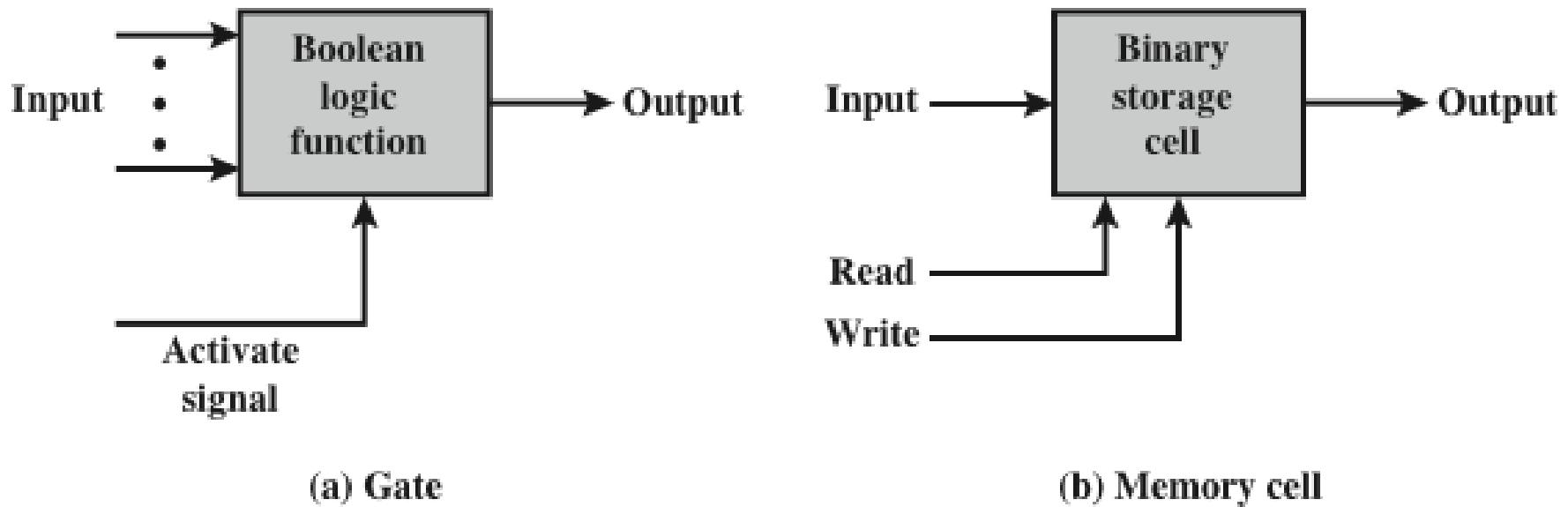


Figure 2.6 Fundamental Computer Elements

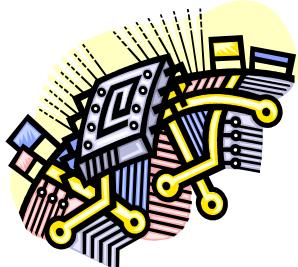


Integrated Circuits

- Data storage – provided by memory cells
- Data processing – provided by gates
- Data movement – the paths among components are used to move data from memory to memory and from memory through gates to memory
- Control – the paths among components can carry control signals
- A computer consists of gates, memory cells, and interconnections among these elements
- The gates and memory cells are constructed of simple digital electronic components
- Exploits the fact that such components as transistors, resistors, and conductors can be fabricated from a semiconductor such as silicon
- Many transistors can be produced at the same time on a single wafer of silicon
- Transistors can be connected with a processor metallization to form circuits

+

Later Generations



Semiconductor Memory
Microprocessors

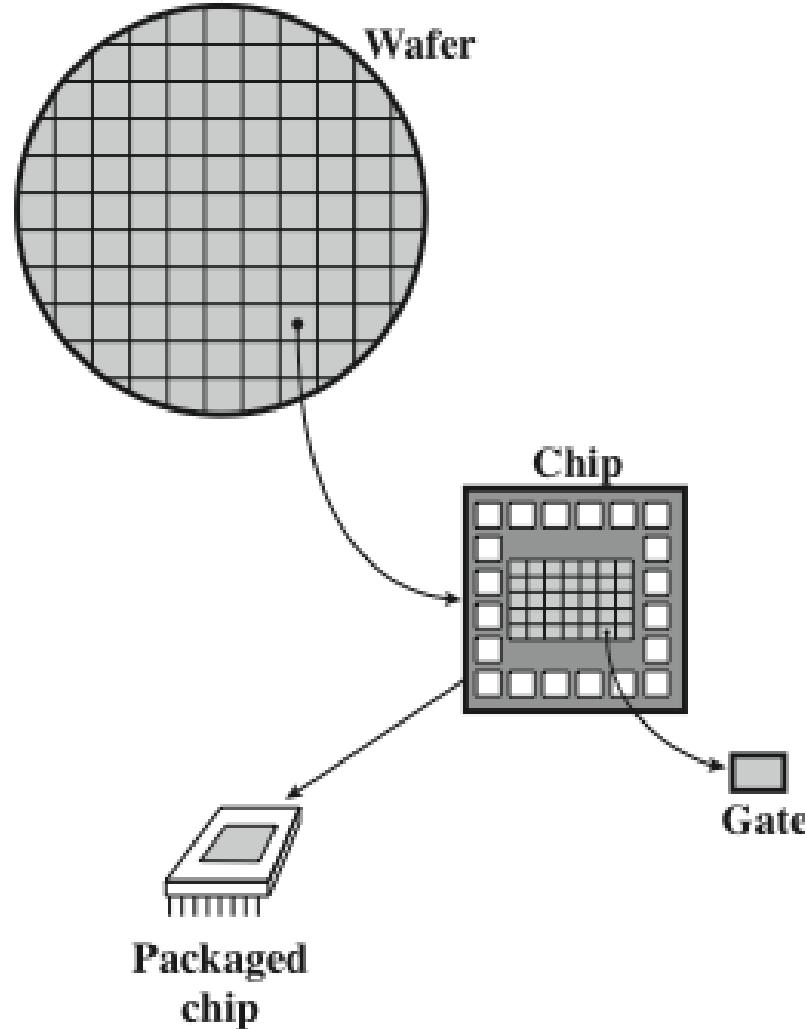
VLSI

Very Large
Scale
Integration

LSI
Large
Scale
Integration

ULSI
Ultra Large
Scale
Integration

Wafer, Chip, and Gate Relationship



Computer Generations

Generation	Approximate Dates	Technology	Typical Speed (operations per second)
1	1946–1957	Vacuum tube	40,000
2	1958–1964	Transistor	200,000
3	1965–1971	Small and medium scale integration	1,000,000
4	1972–1977	Large scale integration	10,000,000
5	1978–1991	Very large scale integration	100,000,000
6	1991-	Ultra large scale integration	1,000,000,000

Computer Architecture Vs Computer Organization

Computer Architecture

Computer Organization

- Attributes of a system visible to the programmer
- Have a direct impact on the logical execution of a program

Computer
Architecture

Architectural
attributes include:

- Instruction set, number of bits used to represent various data types, I/O mechanisms, techniques for addressing memory

Organizational
attributes include:

Computer
Organization

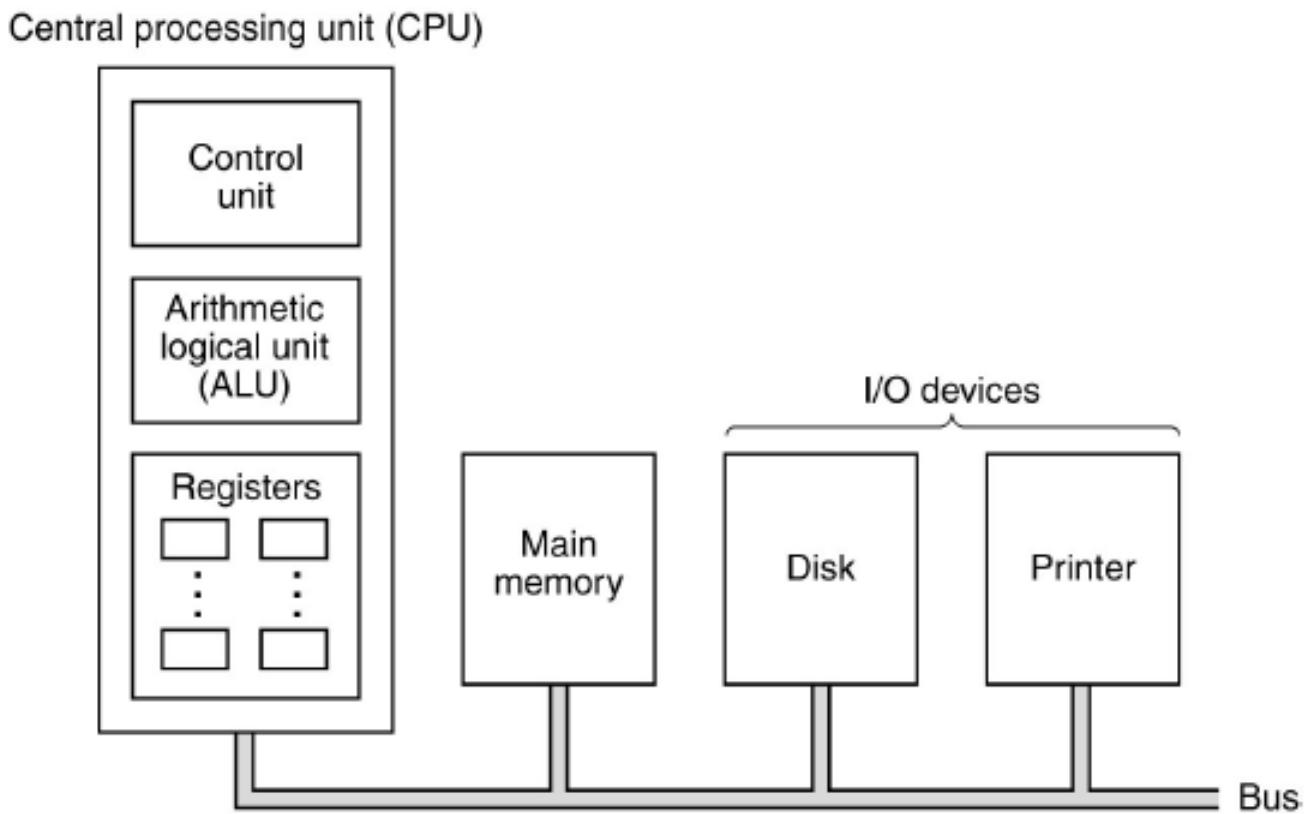
- Hardware details transparent to the programmer, control signals, interfaces between the computer and peripherals, memory technology used

- The operational units and their interconnections that realize the architectural specifications

Computer Architecture

Basic Components

Central Processing Unit (CPU) based CO

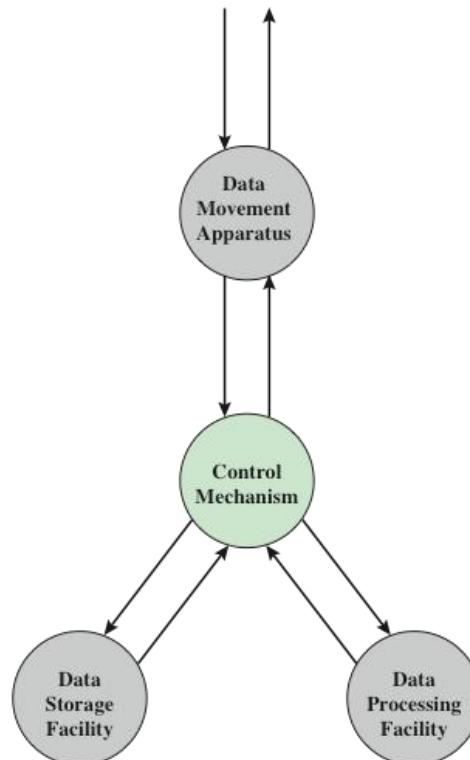


The organization of a simple computer with
one CPU and two I/O devices

Function

- A computer can perform four basic functions:
 - a) Data movement
 - b) Data storage
 - c) Data processing
 - d) Control

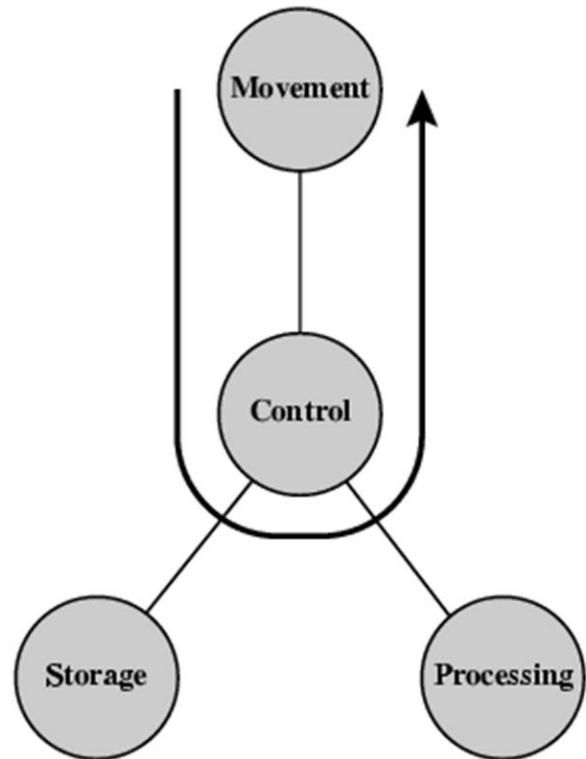
Operating Environment
(source and destination of data)



Operation

(a) Data movement:

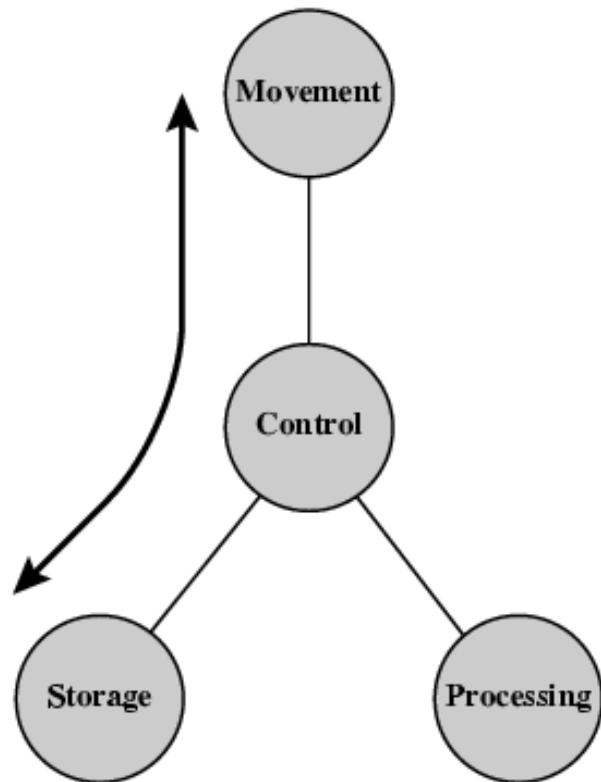
The computer must be able to **move data** between itself and the outside world. The computer's operating environment consists of devices that serve as either sources or destinations of data. When data are received from or delivered to a device that is directly connected to the computer, the process is known as *input–output (I/O)*, and the device is referred to as a *peripheral*. When data are moved over longer distances, to or from a remote device, the process is known as *data communications*.



Operation

(b) Data Storage:

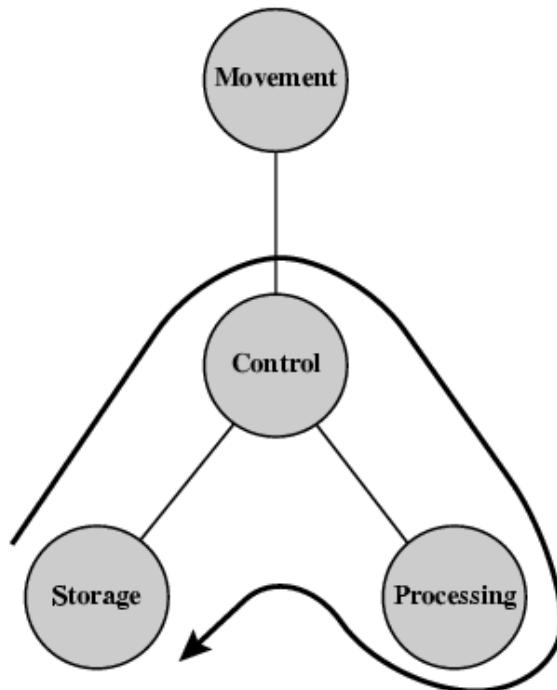
It is also essential that a computer **store data**. Even if the computer is processing data on the fly (i.e., data come in and get processed, and the results go out immediately), the computer must temporarily store at least those pieces of data that are being worked on at any given moment. Thus, there is at least a short-term data storage function. Equally important, the computer performs a long-term data storage function. Files of data are stored on the computer for subsequent retrieval and update.



Operation

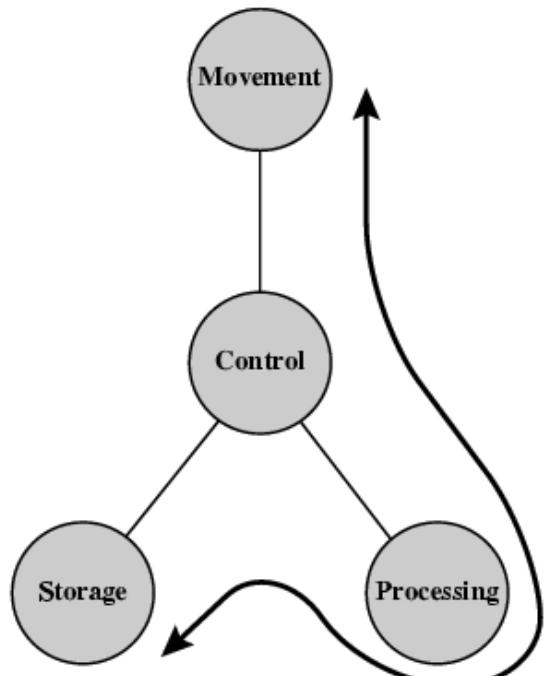
(c) Data Processing:

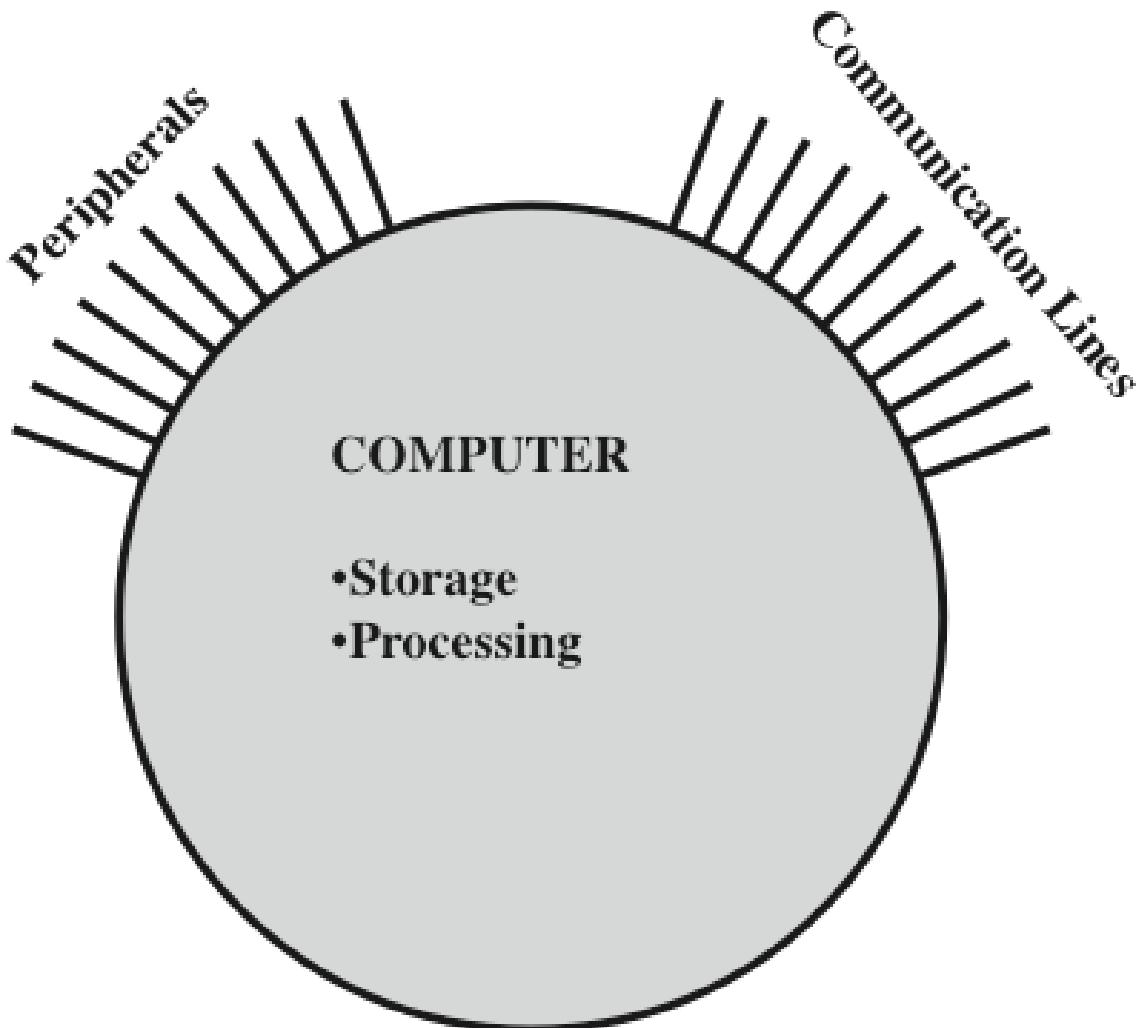
The computer, of course, must be able to **process data**. The data may take a wide variety of forms, and the range of processing requirements is broad. However, we shall see that there are only a few fundamental methods or types of data processing.



Operation

(d) **Control:** There must be **control** of these three functions. Ultimately, this control is exercised by the individual(s) who provides the computer with instructions. Within the computer, a control unit manages the computer's resources and orchestrates the performance of its functional parts in response to those instructions.





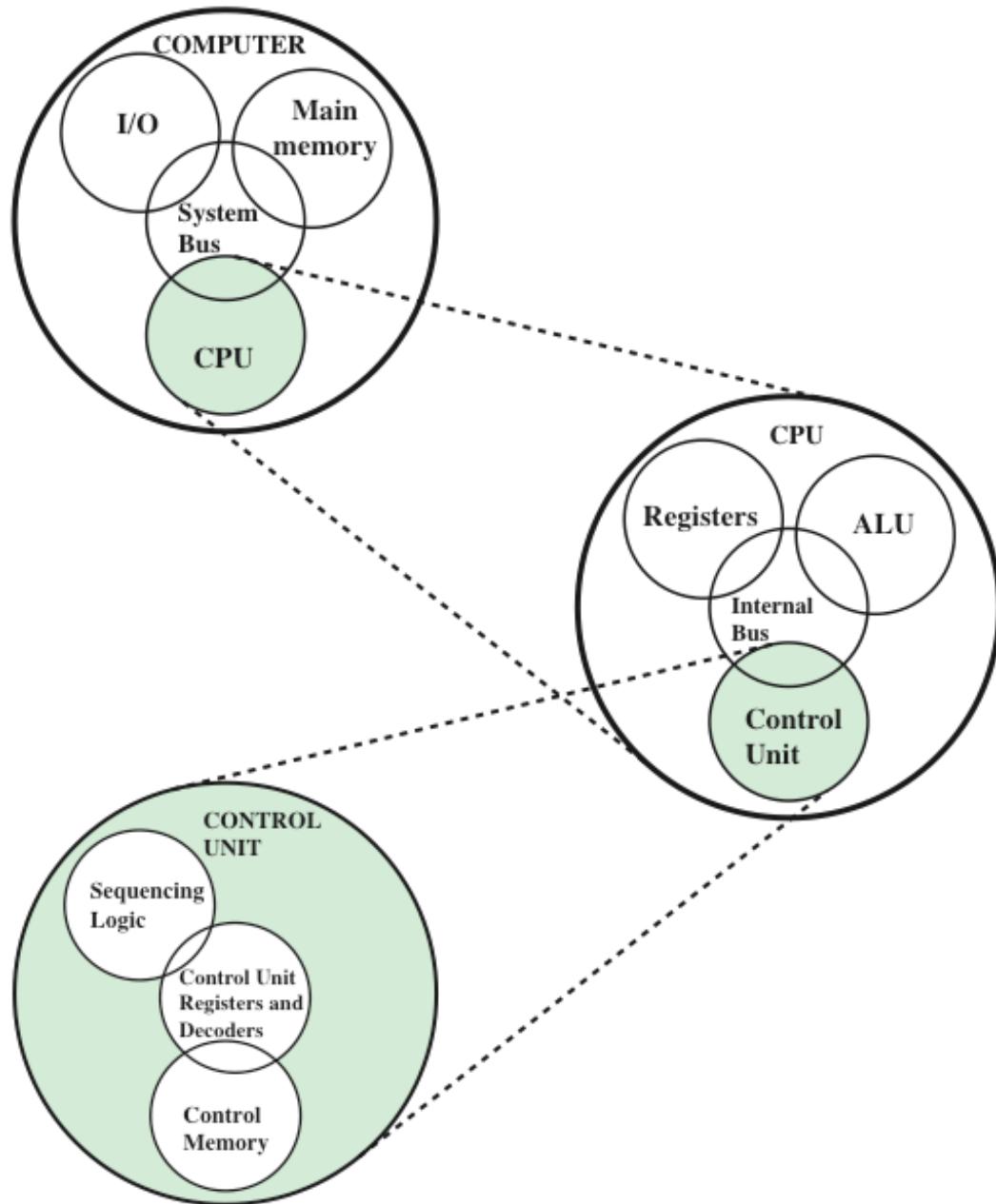
The Computer

Structure and Function

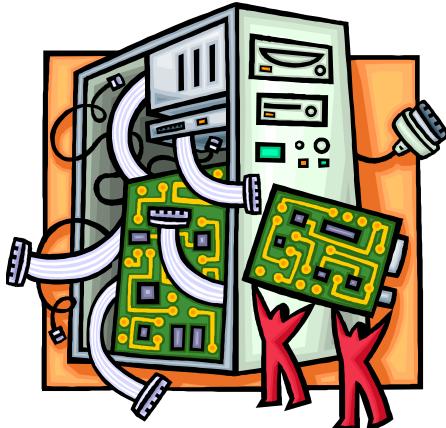
- Hierarchical system
 - Set of interrelated subsystems
- Hierarchical nature of complex systems is essential to both their design and their description
- Designer need only deal with a particular level of the system at a time
 - Concerned with structure and function at each level
- Structure
 - The way in which components relate to each other
- Function
 - The operation of individual components as part of the structure



Structure



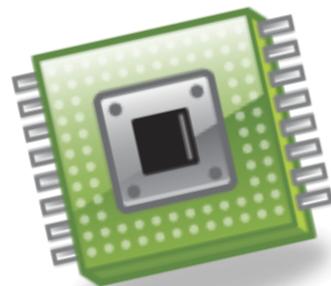
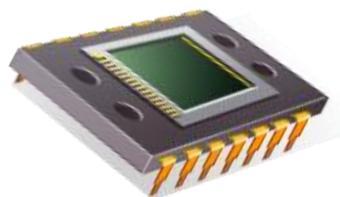
There are four main structural components of the computer:



- ◆ CPU – controls the operation of the computer and performs its data processing functions
- ◆ Main Memory – stores data
- ◆ I/O – moves data between the computer and its external environment
- ◆ System Interconnection – some mechanism that provides for communication among CPU, main memory, and I/O

CPU

Major structural components:



- Control Unit
 - Controls the operation of the CPU and hence the computer
- Arithmetic and Logic Unit (ALU)
 - Performs the computer's data processing function
- Registers
 - Provide storage internal to the CPU
- CPU Interconnection
 - Some mechanism that provides for communication among the control unit, ALU, and registers

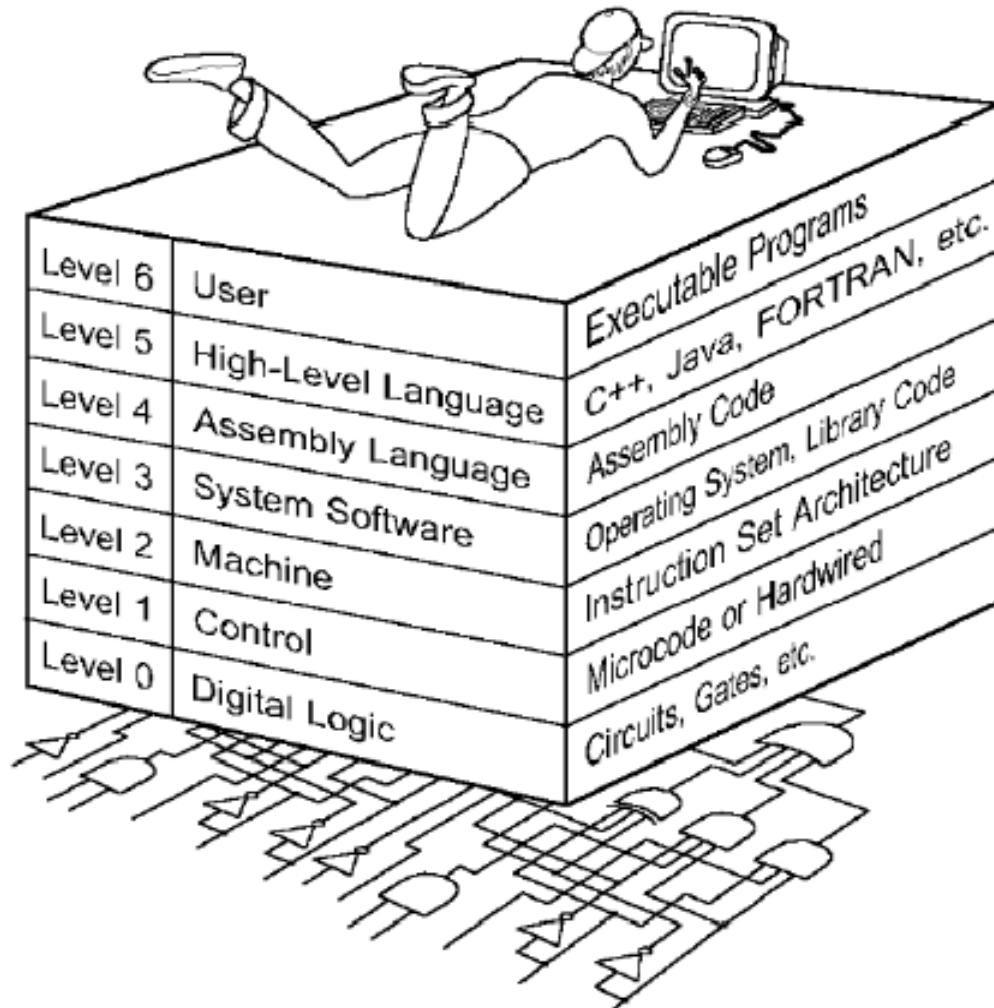
Computer Architecture

Below your program

Below your program

- We constantly interact with these computers
 - E.g., via apps on iPhone or via a Word Processor
- How does an application interact with the hardware?

Computer Level Hierarchy



Language of the hardware

- The hardware understands *on* or *off*. Hence, the language of the hardware needs two symbols 0 and 1.
- Commonly referred to as binary digits or bits.
- 2 letters do not limit what computers can do (just like the finite letters in our languages)

Language of the hardware

- Instructions are collections of bits that the computer understands
- What our programs instruct the computer to do are the instructions:
- Hundreds/thousands/millions of lines of code (instructions) are hidden beneath our apps and programs

Hierarchical Layers of Program Code

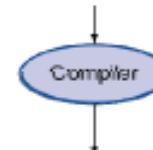
- **High-level language**
 - Level of abstraction closer to problem domain
 - Provides for productivity and portability
 - **Assembly language**
 - Textual representation of instructions
 - **Hardware representation**
 - Binary digits (bits)
 - Encoded instructions and data

High-level
language
program
(in C)

```

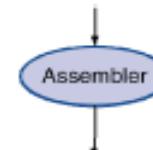
swap(int v[], int k)
{int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}

```



Assembly language program (for MPS)

```
swap:  
    muli $2, $5,4  
    add $2, $4,$2  
    lw $15, 0($2)  
    lw $16, 4($2)  
    sw $16, 0($2)  
    sw $15, 4($2)  
    jr $31
```



Binary machine
language
program
(for MIPS)

More on Evolution

+ Semiconductor Memory

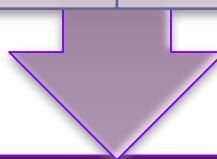
In 1970 Fairchild produced the first relatively capacious semiconductor memory

Chip was about the size
of a single core

Could hold 256 bits of
memory

Non-destructive

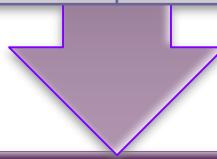
Much faster than core



In 1974 the price per bit of semiconductor memory dropped below the price per bit of core memory

There has been a continuing and rapid decline in
memory cost accompanied by a corresponding
increase in physical memory density

Developments in memory and processor
technologies changed the nature of computers in
less than a decade



Since 1970 semiconductor memory has been through 13 generations

Each generation has provided four times the storage density of the previous generation, accompanied
by declining cost per bit and declining access time

Microprocessors

- The density of elements on processor chips continued to rise
 - More and more elements were placed on each chip so that fewer and fewer chips were needed to construct a single computer processor
- 1971 Intel developed 4004
 - First chip to contain all of the components of a CPU on a single chip
 - Birth of microprocessor
- 1972 Intel developed 8008
 - First 8-bit microprocessor
- 1974 Intel developed 8080
 - First general purpose microprocessor
 - Faster, has a richer instruction set, has a large addressing capability



Evolution of Intel Microprocessors



	4004	8008	8080	8086	8088
Introduced	1971	1972	1974	1978	1979
Clock speeds	108 kHz	108 kHz	2 MHz	5 MHz, 8 MHz, 10 MHz	5 MHz, 8 MHz
Bus width	4 bits	8 bits	8 bits	16 bits	8 bits
Number of transistors	2,300	3,500	6,000	29,000	29,000
Feature size (μm)	10		6	3	6
Addressable memory	640 Bytes	16 KB	64 KB	1 MB	1 MB

a. 1970s Processors

	80286	386TM DX	386TM SX	486TM DX CPU
Introduced	1982	1985	1988	1989
Clock speeds	6 MHz - 12.5 MHz	16 MHz - 33 MHz	16 MHz - 33 MHz	25 MHz - 50 MHz
Bus width	16 bits	32 bits	16 bits	32 bits
Number of transistors	134,000	275,000	275,000	1.2 million
Feature size (μm)	1.5	1	1	0.8 - 1
Addressable memory	16 MB	4 GB	16 MB	4 GB
Virtual memory	1 GB	64 TB	64 TB	64 TB
Cache	—	—	—	8 kB

b. 1980s Processors

Evolution of Intel Microprocessors



	486TM SX	Pentium	Pentium Pro	Pentium II
Introduced	1991	1993	1995	1997
Clock speeds	16 MHz - 33 MHz	60 MHz - 166 MHz,	150 MHz - 200 MHz	200 MHz - 300 MHz
Bus width	32 bits	32 bits	64 bits	64 bits
Number of transistors	1.185 million	3.1 million	5.5 million	7.5 million
Feature size (μm)	1	0.8	0.6	0.35
Addressable memory	4 GB	4 GB	64 GB	64 GB
Virtual memory	64 TB	64 TB	64 TB	64 TB
Cache	8 kB	8 kB	512 kB L1 and 1 MB L2	512 kB L2

c. 1990s Processors

	Pentium III	Pentium 4	Core 2 Duo	Core i7 EE 990
Introduced	1999	2000	2006	2011
Clock speeds	450 - 660 MHz	1.3 - 1.8 GHz	1.06 - 1.2 GHz	3.5 GHz
Bus width	64 bits	64 bits	64 bits	64 bits
Number of transistors	9.5 million	42 million	167 million	1170 million
Feature size (nm)	250	180	65	32
Addressable memory	64 GB	64 GB	64 GB	64 GB
Virtual memory	64 TB	64 TB	64 TB	64 TB
Cache	512 kB L2	256 kB L2	2 MB L2	1.5 MB L2/12 MB L3

d. Recent Processors



Microprocessor Speed

Techniques built into contemporary processors include:

Pipelining

- Processor moves data or instructions into a conceptual pipe with all stages of the pipe processing simultaneously

Branch prediction

- Processor looks ahead in the instruction code fetched from memory and predicts which branches, or groups of instructions, are likely to be processed next

Data flow analysis

- Processor analyzes which instructions are dependent on each other's results, or data, to create an optimized schedule of instructions

Speculative execution

- Using branch prediction and data flow analysis, some processors speculatively execute instructions ahead of their actual appearance in the program execution, holding the results in temporary locations, keeping execution engines as busy as possible



Performance Balance

- Adjust the organization and architecture to compensate for the mismatch among the capabilities of the various components
- Architectural examples include:

Change the DRAM interface to make it more efficient by including a cache or other buffering scheme on the DRAM chip

Increase the number of bits that are retrieved at one time by making DRAMs “wider” rather than “deeper” and by using wide bus data paths

Reduce the frequency of memory access by incorporating increasingly complex and efficient cache structures between the processor and main memory

Increase the interconnect bandwidth between processors and memory by using higher speed buses and a hierarchy of buses to buffer and structure data flow



Improvements in Chip Organization and Architecture

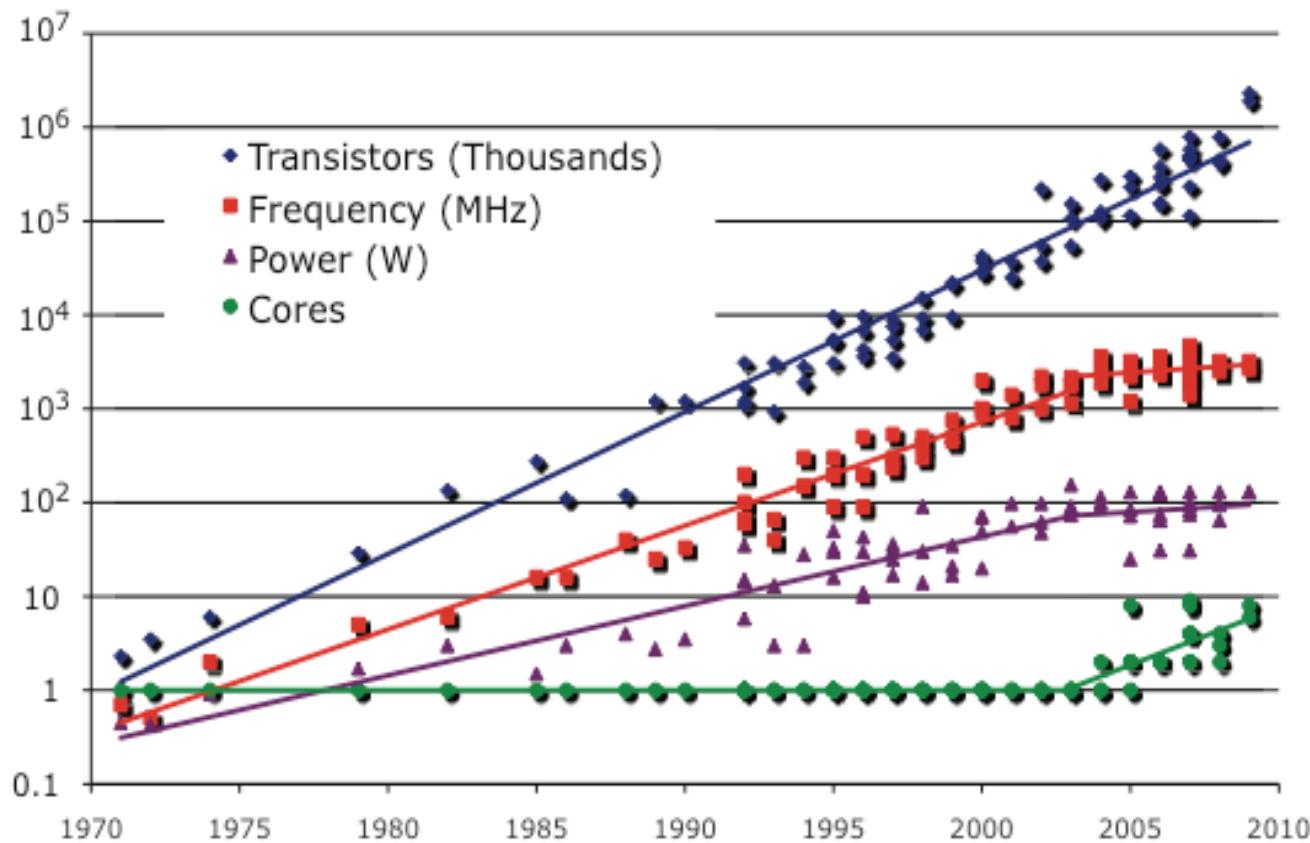
- **Increase hardware speed of processor**
 - Fundamentally due to shrinking logic gate size
 - More gates, packed more tightly, increasing clock rate
 - Propagation time for signals reduced
- **Increase size and speed of caches**
 - Dedicating part of processor chip
 - Cache access times drop significantly
- **Change processor organization and architecture**
 - Increase effective speed of instruction execution
 - Parallelism



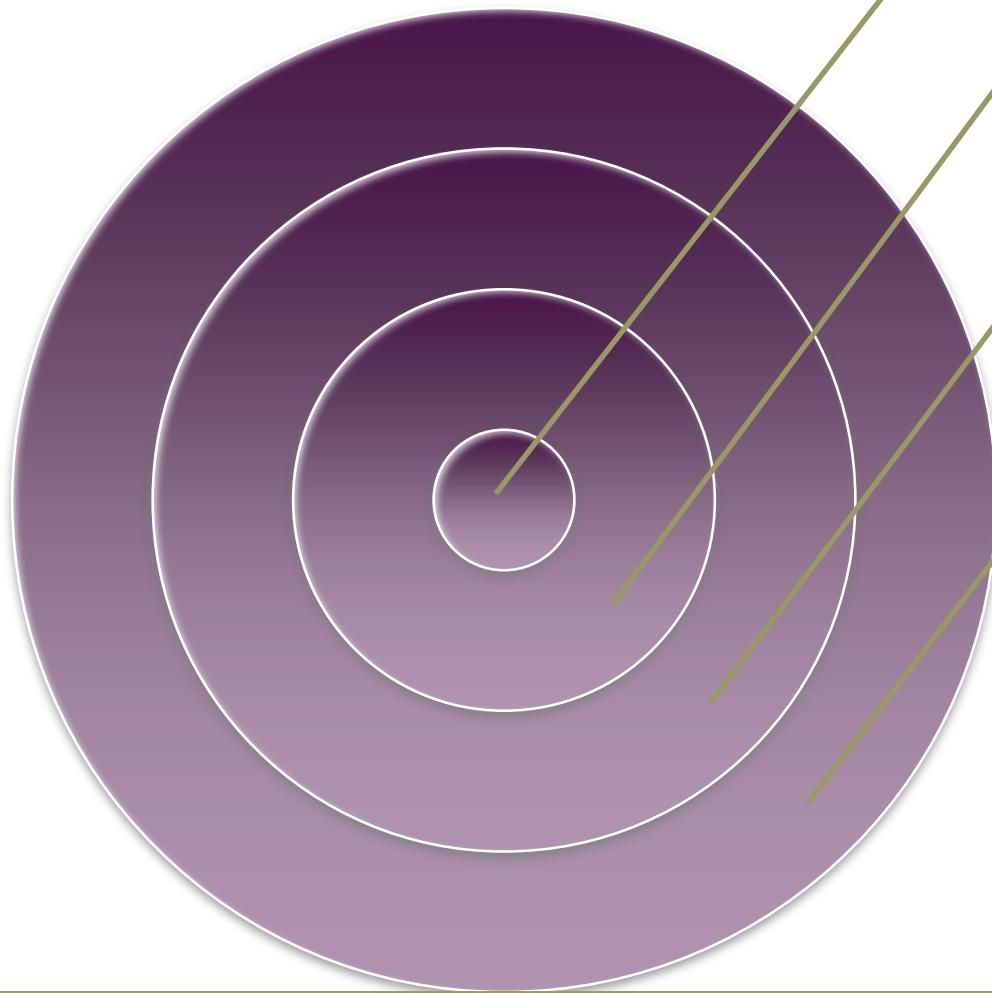
Problems with Clock Speed and Login Density

- Power
 - Power density increases with density of logic and clock speed
 - Dissipating heat
- RC delay
 - Speed at which electrons flow on a chip between transistors is limited by resistance and capacitance of metal wires connecting them
 - Delay increases as RC product increases
 - Wire interconnects thinner, increasing resistance
 - Wires closer together, increasing capacitance
- Memory latency
 - Memory speeds lag processor speeds

Processor Trends



Multicore



The use of multiple processors on the same chip provides the potential to increase performance without increasing the clock rate

Strategy is to use two simpler processors on the chip rather than one more complex processor

With two processors larger caches are justified

As caches became larger it made performance sense to create two and then three levels of cache on a chip



Many Integrated Core (MIC) Graphics Processing Unit (GPU)

MIC

- Leap in performance as well as the challenges in developing software to exploit such a large number of cores
- The multicore and MIC strategy involves a homogeneous collection of general purpose processors on a single chip

GPU

- Core designed to perform parallel operations on graphics data
- Traditionally found on a plug-in graphics card, it is used to encode and render 2D and 3D graphics as well as process video
- Used as vector processors for a variety of applications that require repetitive computations

Overview

- Results of decades of design effort on complex instruction set computers (CISCs)
- Excellent example of CISC design
- Incorporates the sophisticated design principles once found only on mainframes and supercomputers
- An alternative approach to processor design is the reduced instruction set computer (RISC)
- The ARM architecture is used in a wide variety of embedded systems and is one of the most powerful and best designed RISC based systems on the market
- In terms of market share Intel is ranked as the number one maker of microprocessors for non-embedded systems

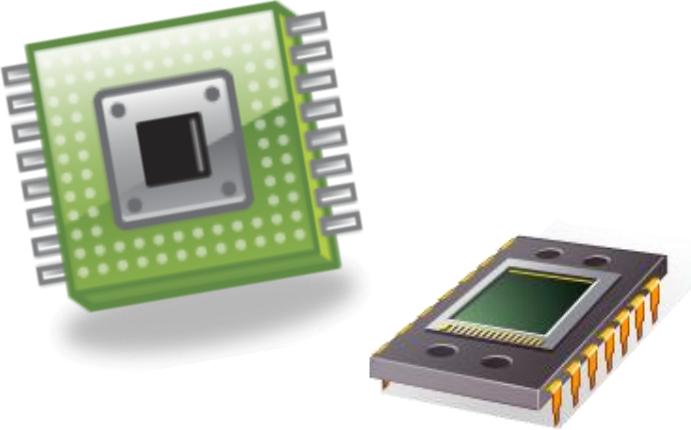
ARM

Intel

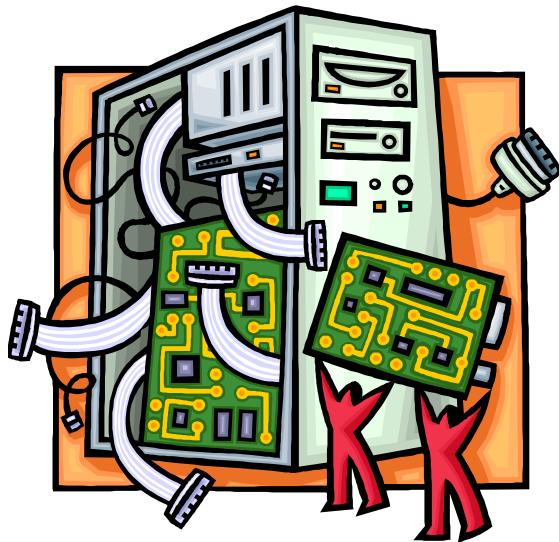
x86 Architecture

CISC

RISC



x86 Evolution



8080

- First general purpose microprocessor
- 8-bit machine with an 8-bit data path to memory
- Used in the first personal computer (Altair)

8086

- 16-bit machine
- Used an instruction cache, or queue
- First appearance of the x86 architecture

8088

- used in IBM's first personal computer

80286

- Enabled addressing a 16-MByte memory instead of just 1 MByte

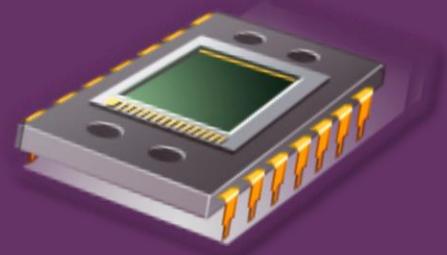
80386

- Intel's first 32-bit machine
- First Intel processor to support multitasking

80486

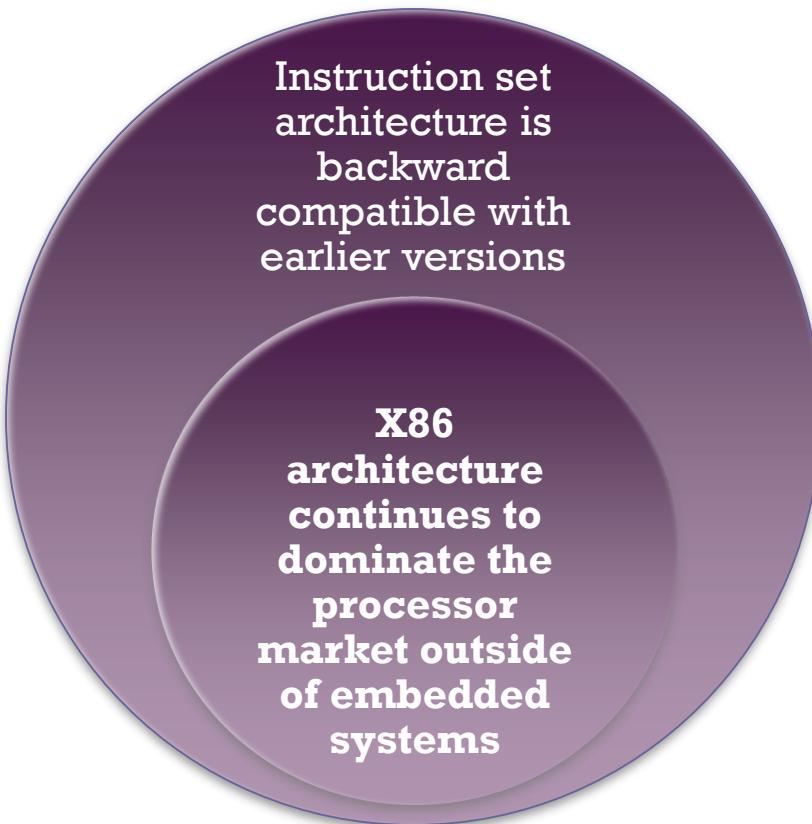
- More sophisticated cache technology and instruction pipelining
- Built-in math coprocessor

x86 Evolution - Pentium



Pentium	Pentium Pro	Pentium II	Pentium III	Pentium 4
<ul style="list-style-type: none">• Superscalar• Multiple instructions executed in parallel	<ul style="list-style-type: none">• Increased superscalar organization• Aggressive register renaming• Branch prediction• Data flow analysis• Speculative execution	<ul style="list-style-type: none">• MMX technology• Designed specifically to process video, audio, and graphics data	<ul style="list-style-type: none">• Additional floating-point instructions to support 3D graphics software	<ul style="list-style-type: none">• Includes additional floating-point and other enhancements for multimedia

x86 Evolution (continued)



■ Core

- First Intel x86 microprocessor with a dual core, referring to the implementation of two processors on a single chip

■ Core 2

- Extends the architecture to 64 bits
- Recent Core offerings have up to 10 processors per chip

Embedded system:

“A combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a dedicated function. In many cases, embedded systems are part of a larger system or product, as in the case of an antilock braking system in a car.”

Embedded

Systems





Examples of Embedded Systems and Their Markets

Market	Embedded Device
Automotive	Ignition system Engine control Brake system
Consumer electronics	Digital and analog televisions Set-top boxes (DVDs, VCRs, Cable boxes) Personal digital assistants (PDAs) Kitchen appliances (refrigerators, toasters, microwave ovens) Automobiles Toys/games Telephones/cell phones/pagers Cameras Global positioning systems
Industrial control	Robotics and controls systems for manufacturing Sensors
Medical	Infusion pumps Dialysis machines Prosthetic devices Cardiac monitors
Office automation	Fax machine Photocopier Printers Monitors Scanners



Embedded Systems

Requirements and Constraints

Small to large systems, implying different cost constraints and different needs for optimization and reuse

Different models of computation ranging from discrete event systems to hybrid systems

Different application characteristics resulting in static versus dynamic loads, slow to fast speed, compute versus interface intensive tasks, and/or combinations thereof

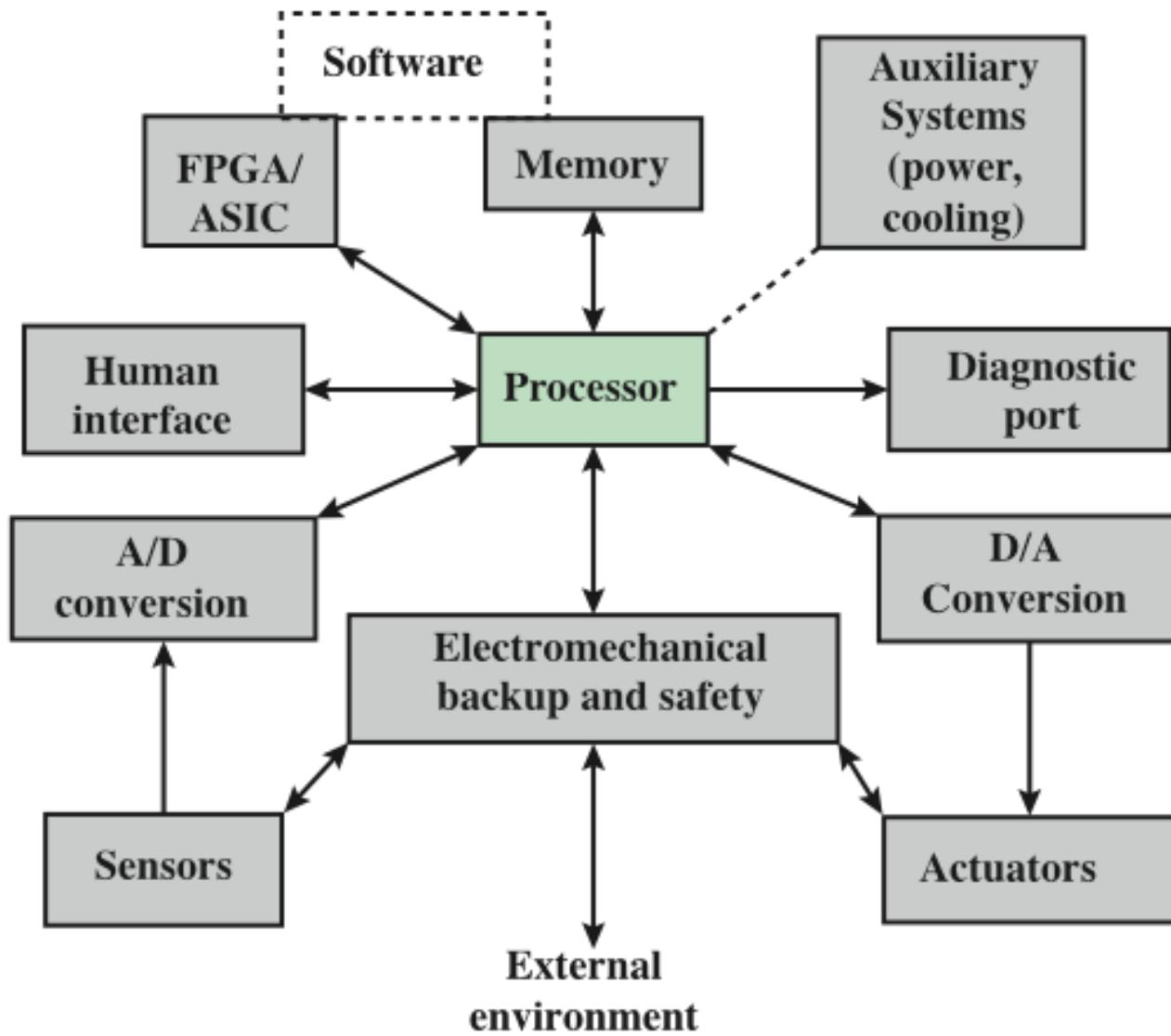


Relaxed to very strict requirements and combinations of different quality requirements with respect to safety, reliability, real-time and flexibility

Short to long life times

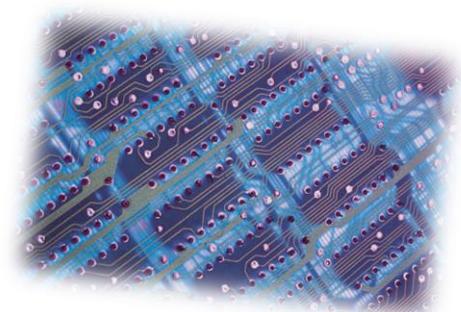
Different environmental conditions in terms of radiation, vibrations, and humidity

Possible Organization of an Embedded System



Acorn RISC Machine (ARM)

- Family of RISC-based microprocessors and microcontrollers
- Designs microprocessor and multicore architectures and licenses them to manufacturers
- Chips are high-speed processors that are known for their small die size and low power requirements
- Widely used in PDAs and other handheld devices
- Chips are the processors in iPod and iPhone devices
- Most widely used embedded processor architecture
- Most widely used processor architecture of any kind



+

E
v
o
l
u
t
i
o
n

Family	Notable Features	Cache	Typical MIPS @ MHz
ARM1	32-bit RISC	None	
ARM2	Multiply and swap instructions; Integrated memory management unit, graphics and I/O processor	None	7 MIPS @ 12 MHz
ARM3	First use of processor cache	4 KB unified	12 MIPS @ 25 MHz
ARM6	First to support 32-bit addresses; floating-point unit	4 KB unified	28 MIPS @ 33 MHz
ARM7	Integrated SoC	8 KB unified	60 MIPS @ 60 MHz
ARM8	5-stage pipeline; static branch prediction	8 KB unified	84 MIPS @ 72 MHz
ARM9		16 KB/16 KB	300 MIPS @ 300 MHz
ARM9E	Enhanced DSP instructions	16 KB/16 KB	220 MIPS @ 200 MHz
ARM10E	6-stage pipeline	32 KB/32 KB	
ARM11	9-stage pipeline	Variable	740 MIPS @ 665 MHz
Cortex	13-stage superscalar pipeline	Variable	2000 MIPS @ 1 GHz
XScale	Applications processor; 7-stage pipeline	32 KB/32 KB L1 512 KB L2	1000 MIPS @ 1.25 GHz

DSP = digital signal processor

SoC = system on a chip

ARM Design Categories

- ARM processors are designed to meet the needs of three system categories:

- Secure applications
 - Smart cards, SIM cards, and payment terminals

- Embedded real-time systems
 - Systems for storage, automotive body and power-train, industrial, and networking applications

- Application platforms
 - Devices running open operating systems including Linux, Palm OS, Symbian OS, and Windows CE in wireless, consumer entertainment and digital imaging applications

CSE 213

Computer Architecture

Lecture 2: Computer Performance

Military Institute of Science and Technology

Performance

- *Performance is the key to understanding underlying motivation for the hardware and its organization*
- *Why is some hardware better than others for different programs?*
- *What factors of system performance are hardware related?
(e.g., do we need a new machine, or a new operating system?)*

Performance

- Why is performance important?
 - For purchasers: to choose between computers
 - For designers: to make the sales pitch
- Defining performance is not straightforward!
 - An analogy with airplanes shows the difficulty

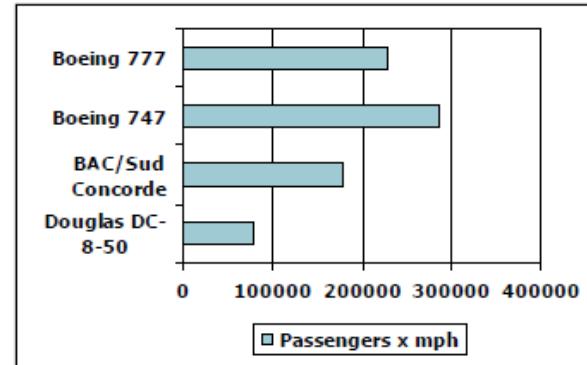
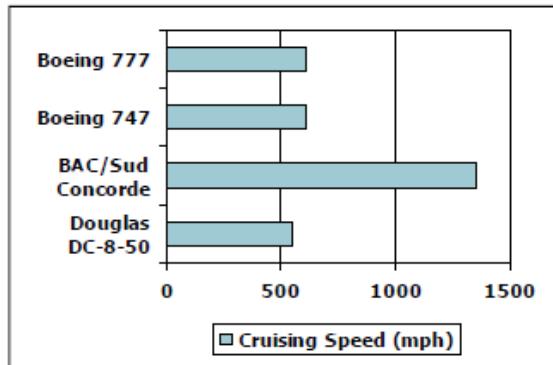
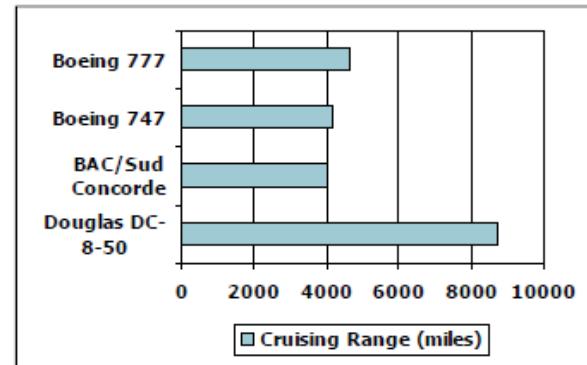
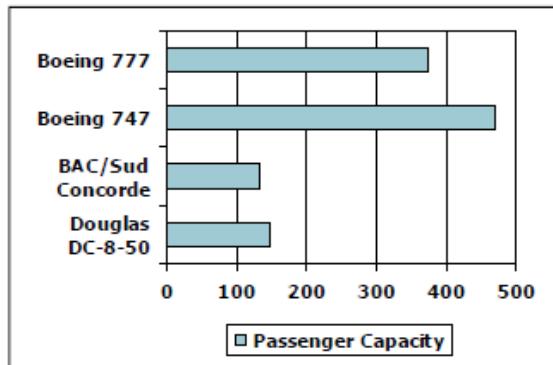
What do we measure? Define performance....

Airplane	Passenger capacity	Cruising range (miles)	Cruising speed (m.p.h.)	Passenger throughput (passengers × m.p.h.)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286,700
BAC/Sud Concorde	132	4000	1350	178,200
Douglas DC-8-50	146	8720	544	79,424

- How much faster is the Concorde compared to the 747?
- How much bigger is the Boeing 747 than the Douglas DC-8?

Defining Performance

- Which airplane has the best performance?



Computer Performance: TIME, TIME, TIME!!!

- *Response Time (elapsed time, latency):*
 - How long does it take to complete (start to finish) *a task*?
 - Eg: how long must *I* wait for the database query?
- 
- Individual user concerns...

Individual is more interested in response time. As a user of a smart phone/laptop, the one that responds faster is better!

Response time (computer): the total time required by computer to complete a task including :

Disk access Memory access I/O activities OS overheads CPU exec. time etc

Computer Performance: TIME, TIME, TIME!!!

- *Throughput:*
 - Total work done per unit time.....(per hr,day etc)
 - how *many* jobs can the machine run at once?
 - what is the *average* execution rate?
 - how *much* work is getting done?
- 
- Systems manager concerns...

Response Time and Throughput

- *If we upgrade a machine with a new processor what do we increase?*
- How are response time and throughput affected by
 - Replacing the processor with a faster version?
 - Adding more processors?

Relative Performance

- Define Performance = $1/\text{Execution Time}$
- $\text{Performance}_x > \text{Performance}_y$
- $1/\text{Execution Time}_x > 1/\text{Execution Time}_y$
- $\text{Execution Time}_y > \text{Execution Time}_x$

$$\begin{aligned}\text{Performance}_x / \text{Performance}_y \\ = \text{Execution time}_y / \text{Execution time}_x = n\end{aligned}$$

Relative Performance

- Define Performance = 1/Execution Time
- “X is n time faster than Y”

$$\begin{aligned} \text{Performance}_X / \text{Performance}_Y \\ = \text{Execution time}_Y / \text{Execution time}_X = n \end{aligned}$$

- Example: time taken to run a program
 - 10s on A, 15s on B
 - $\text{Execution Time}_B / \text{Execution Time}_A$
 $= 15s / 10s = 1.5$
 - So A is 1.5 times faster than B

Execution Time

- *Elapsed Time*

- counts everything (*disk and memory accesses, waiting for I/O, running other programs, etc.*) from start to finish
- **a useful number, but often not good for comparison purposes**

Elapsed time = CPU time + wait time (I/O, other programs, etc.)

- *CPU time*

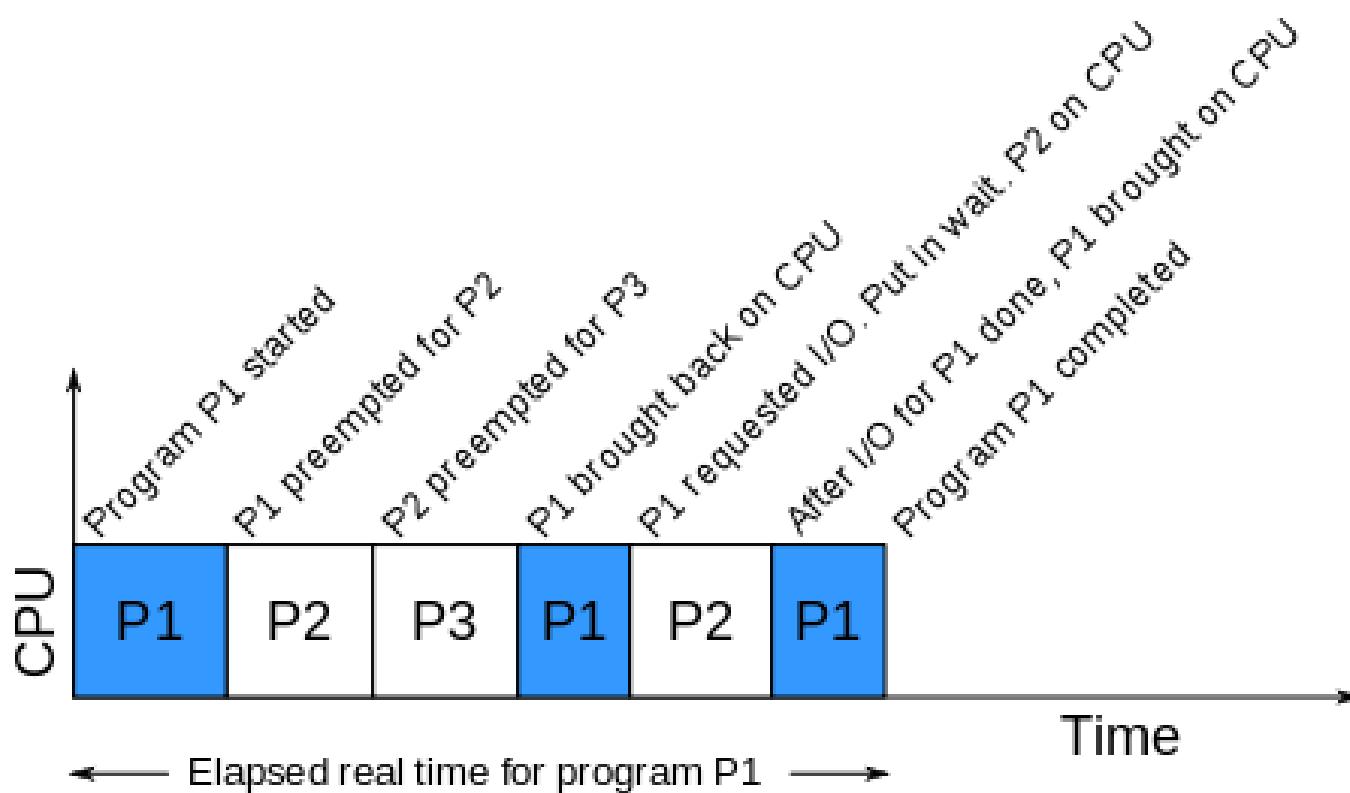
- **doesn't count waiting for I/O or time spent running other programs**
- can be divided into *user CPU time* and *system CPU time* (OS calls)

CPU time = user CPU time + system CPU time

- Our focus:

- *user CPU time* (*CPU execution time* or, simply, *execution time*)
 - **time spent executing the lines of code that are *in our program***
 - *For easier writing, user CPU time has been termed simply as CPU time in rest of the studies.*

Execution Time



CPU Clocking

Clock cycle. The speed of a computer processor, or CPU, is determined by the clock cycle, which is the amount of time between two pulses of an oscillator.

- Operation of digital hardware governed by a constant-rate clock
 - Clock period: duration of a clock cycle
 - e.g., $250\text{ps} = 0.25\text{ns} = 250 \times 10^{-12}\text{s}$
 - Clock frequency (rate): cycles per second
 - e.g., $4.0\text{ GHz} = 4000\text{ MHz} = 4.0 \times 10^9\text{Hz}$

CPU Time

CPU Time = CPU Clock Cycles \times Clock Cycle Time

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance improved by
 - Reducing number of clock cycles
 - Increasing clock rate
 - Hardware designer must often trade off clock rate against cycle count

Performance Equation - I

Example

- Our favorite program runs in 10 seconds on computer A, which has a 2Ghz. clock.
- We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program.
- *What clock rate should we tell the designer to target?*

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
 - Aim for 6s CPU time
 - Can do faster clock, but causes $1.2 \times$ clock cycles compared to A
- How fast must Computer B clock be i.e., what is the clock rate for Computer B?

$$\text{CPU Time}_B = \frac{\text{CPU Clock Cycles}_B}{\text{Clock Rate}_B}$$

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
 - Aim for 6s CPU time
 - Can do faster clock, but causes $1.2 \times$ clock cycles compared to A
- How fast must Computer B clock be?

$$\text{CPU Time}_B = \frac{\text{CPU Clock Cycles}_B}{\text{Clock Rate}_B}$$

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6s}$$

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
 - Aim for 6s CPU time
 - Can do faster clock, but causes $1.2 \times$ clock cycles
- How fast must Computer B clock be?

$$\text{CPU Time}_B = \frac{\text{CPU Clock Cycles}_B}{\text{Clock Rate}_B}$$

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6s}$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6s} = \frac{24 \times 10^9}{6s} = 4\text{GHz}$$

Instruction Count and CPI

Clock Cycles = Instruction Count \times Cycles per Instruction

CPU Time = Instruction Count \times CPI \times Clock Cycle Time

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count for a program
 - Determined by program, ISA and compiler
- Average cycles per instruction
 - Determined by CPU hardware
 - If different instructions have different CPI
 - Average CPI affected by instruction mix

Performance Equation - II

Factors Influencing Performance

- Execution time = **clock cycle time x number of instrs x avg CPI**
- **Clock cycle time:** manufacturing process (how fast is each transistor), how much work gets done in each pipeline stage (more on this later)
- **Number of instructions:** the quality of the compiler and the instruction set architecture
- **CPI:** the nature of each instruction and the quality of the architecture implementation

CPU Time Example

Suppose we have two implementations of the same instruction set architecture. Computer A has a clock cycle time of 250 ps and a CPI of 2.0 for some program, and computer B has a clock cycle time of 500 ps and a CPI of 1.2 for the same program. Which computer is faster for this program and by how much?

CPU Time Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$

CPU Time Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$

$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

A is faster...

$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$

$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

...by this much

Self Help

- Suppose we have two implementations of the same instruction set architecture (ISA). For some program:
 - machine A has a clock cycle time of 10 ns. and a CPI of 2.0
 - machine B has a clock cycle time of 20 ns. and a CPI of 1.2
- *Which machine is faster for this program, and by how much?*
- *If two machines have the same ISA, which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will always be identical?*

Average cycles per instruction (CPI):

- CPI_i be the number of cycles required for instruction type i , and I_i be the number of executed instructions of type i

$$CPI = \frac{\sum_{i=1}^n (CPI_i \times I_i)}{I_c}$$

The processor time T needed to execute a given program,

$$T = I_c \times CPI \times \tau$$

- τ = A constant cycle time
- I_c = Instruction count

CPI Example

- A compiler designer is trying to decide between two code sequences for a particular machine.
- Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C,

	CPI for each Instruction class		
	A	B	C
CPI	1	2	3

Code sequence	Instruction counts for each Instruction class		
	A	B	C
1	2	1	2
2	4	1	1

- *Which code sequence has the most instructions? Which sequence will be faster? How much? What is the CPI for each sequence?*

CPI Example

Which code sequence has the most instructions?

Sequence 1 executes $2 + 1 + 2 = 5$ instructions. Sequence 2 executes $4 + 1 + 1 = 6$ instructions. Therefore, sequence 1 executes fewer instructions.

We can use the equation for CPU clock cycles based on instruction count and CPI to find the total number of clock cycles for each sequence:

$$\text{CPU clock cycles} = \sum_{i=1}^n (\text{CPI}_i \times C_i)$$

Which sequence will be faster?

$$\text{CPU clock cycles}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10 \text{ cycles}$$

$$\text{CPU clock cycles}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9 \text{ cycles}$$

What is the CPI for each sequence

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = \frac{10}{5} = 2.0$$

$$\text{CPI}_2 = \frac{\text{CPU clock cycles}_2}{\text{Instruction count}_2} = \frac{9}{6} = 1.5$$

MIPS Rate:

A common measure of performance for a processor is the rate at which instructions are executed, expressed as millions of instructions per second (MIPS) aka the **MIPS rate**

$$\text{MIPS rate} = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

MIPS Example

EXAMPLE 2.2 Consider the execution of a program that results in the execution of 2 million instructions on a 400-MHz processor. The program consists of four major types of instructions. The instruction mix and the *CPI* for each instruction type are given below, based on the result of a program trace experiment:

Instruction Type	<i>CPI</i>	Instruction Mix (%)
Arithmetic and logic	1	60
Load/store with cache hit	2	18
Branch	4	12
Memory reference with cache miss	8	10

$$CPI = 0.6 + (2 \times 0.18) + (4 \times 0.12) + (8 \times 0.1) = 2.24$$

MIPS rate is $(400 \times 10^6)/(2.24 \times 10^6) \approx 178$

Self Help

- Two different compilers are being tested for a 500 MHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require 1, 2 and 3 cycles (respectively). Both compilers are used to produce code for a large piece of software.
- Compiler 1 generates code with 5 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.
- Compiler 2 generates code with 10 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.
- *Which sequence will be faster according to MIPS?*
- *Which sequence will be faster according to execution time?*

Another Example

A given application written in Java runs 15 seconds on a desktop processor. A new Java compiler is released that requires only 0.6 as many instructions as the old compiler. Unfortunately, it increases the CPI by 1.1. How fast can we expect the application to run using this new compiler? Pick the right answer from the three choices below

- a. $\frac{15 \times 0.6}{1.1} = 8.2 \text{ sec}$
- b. $15 \times 0.6 \times 1.1 = 9.9 \text{ sec}$
- c. $\frac{15 \times 1.1}{0.6} = 27.5 \text{ sec}$

MFLOPS

MFLOPS: Millions of floating-point operations per second

$$\text{MFLOPS rate} = \frac{\text{Number of executed floating - point operations in a program}}{\text{Execution time} \times 10^6}$$

List Performance Measure

- **Basic Measurement:**
 - Clock Speed
 - Instruction Execution Rate
 - MIPS
 - MFLOPS
- Calculating the Mean: calculating the mean value of a set of data points related to execution time.
 - Arithmetic Mean (AM)
 - Geometric mean (GM)
 - Harmonic mean (HM)

AM, GM, HM Formula

Arithmetic mean

$$AM = \frac{x_1 + \cdots + x_n}{n} = \frac{1}{n} \sum_{i=1}^n x_i$$

Geometric mean

$$GM = \sqrt[n]{x_1 \times \cdots \times x_n} = \left(\prod_{i=1}^n x_i \right)^{1/n} = e^{\left(\frac{1}{n} \sum_{i=1}^n \ln(x_i) \right)}$$

Harmonic mean

$$HM = \frac{n}{\left(\frac{1}{x_1} \right) + \cdots + \left(\frac{1}{x_n} \right)} = \frac{n}{\sum_{i=1}^n \left(\frac{1}{x_i} \right)} \quad x_i > 0$$

Basic example: How AM Performs?

- Suppose we have a set of n benchmark programs and record the execution times of each program on a given system as t_1, t_2, \dots, t_n .
- For simplicity, let us assume that each program executes the same number of operations Z
- The execution rate for each individual program is $R_i = Z/t_i$.
- We use the AM to calculate the average execution rate.

$$AM = \frac{1}{n} \sum_{i=1}^n R_i = \frac{1}{n} \sum_{i=1}^n \frac{Z}{t_i} = \frac{Z}{n} \sum_{i=1}^n \frac{1}{t_i}$$

- AM execution rate is proportional to the sum of the inverse execution times.

Basic example: How AM Performs?

The HM yields the following result:

$$HM = \frac{n}{\sum_{i=1}^n \left(\frac{1}{R_i} \right)} = \frac{n}{\sum_{i=1}^n \left(\frac{1}{Z/t_i} \right)} = \frac{nZ}{\sum_{i=1}^n t_i}$$

The HM is inversely proportional to the total execution time, which is the desired property.

Benchmarks and Spec

$$\mathbf{A} = \mathbf{B} + \mathbf{C}$$

- In a complex instruction set computer (CISC), this instruction can be compiled into one processor instruction:
`add mem(B), mem(C), mem(A)`
- On a RISC machine, the compilation would look something like this:

```
load mem(B), reg(1);
load mem(C), reg(2);
add reg(1), reg(2), reg(3);
store reg(3), mem (A)
```

Both machines may execute the original high-level language instruction in about the same time.

- CISC machine is rated at **1 MIPS**,
- RISC machine is rated at **4 MIPS**.

Benchmarks and Spec

- Best Performance determined by running a real application
 - use programs typical of expected workload
 - or, typical of expected class of applications
 - e.g., compilers/editors, scientific applications, graphics, etc.
- Benchmark suites
 - Each vendor announces a SPEC rating for their system
 - a measure of execution time for a fixed collection of programs
 - is a function of a specific CPU, memory system, IO system, operating system, compiler enables easy comparison of different systems

SPEC (System Performance Evaluation Corporation)

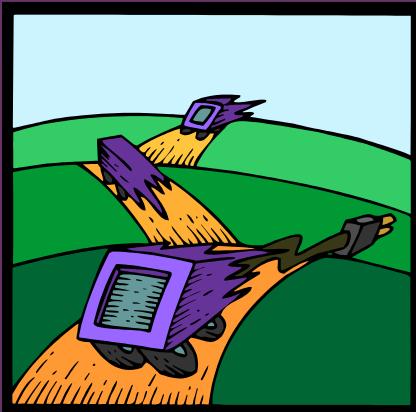
- Sponsored by industry but independent and self-managed – trusted by code developers and machine vendors
- Clear guides for testing, see www.spec.org
- Regular updates (benchmarks are dropped and new ones added periodically according to relevance)
- Specialized benchmarks for particular classes of applications

SPEC CPU

- The best known of the SPEC benchmark suites is SPEC CPU2006.
- The 2006 version includes 12 integer and 17 floating-point applications
 - The SPEC rating specifies how much faster a system is, compared to a baseline machine – a system with SPEC rating 600 is 1.5 times faster than a system with SPEC rating 400



Amdahl's Law



- Gene Amdahl [AMDA67]
- Deals with the potential speedup of a program using multiple processors compared to a single processor
- Illustrates the problems facing industry in the development of multi-core machines
 - Software must be adapted to a highly parallel execution environment to exploit the power of parallel processing
- Can be generalized to evaluate and design technical improvement in a computer system

Amdahl's law

Amdahl's law states that in parallelization

- f is the proportion of a system or program that can be made parallel
- $1-f$ is the proportion that remains serial

$$\begin{aligned}\text{Speedup} &= \frac{\text{Time to execute program on a single processor}}{\text{Time to execute program on } N \text{ parallel processors}} \\ &= \frac{T(1 - f) + Tf}{T(1 - f) + \frac{Tf}{N}} = \frac{1}{(1 - f) + \frac{f}{N}}\end{aligned}$$

Two important conclusions can be drawn:

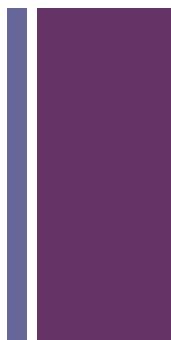
1. When f is small, the use of parallel processors has little effect.
2. As N approaches infinity, speedup is bound by $1/(1 - f)$, so that there are diminishing returns for using more processors.

Amdahl's law

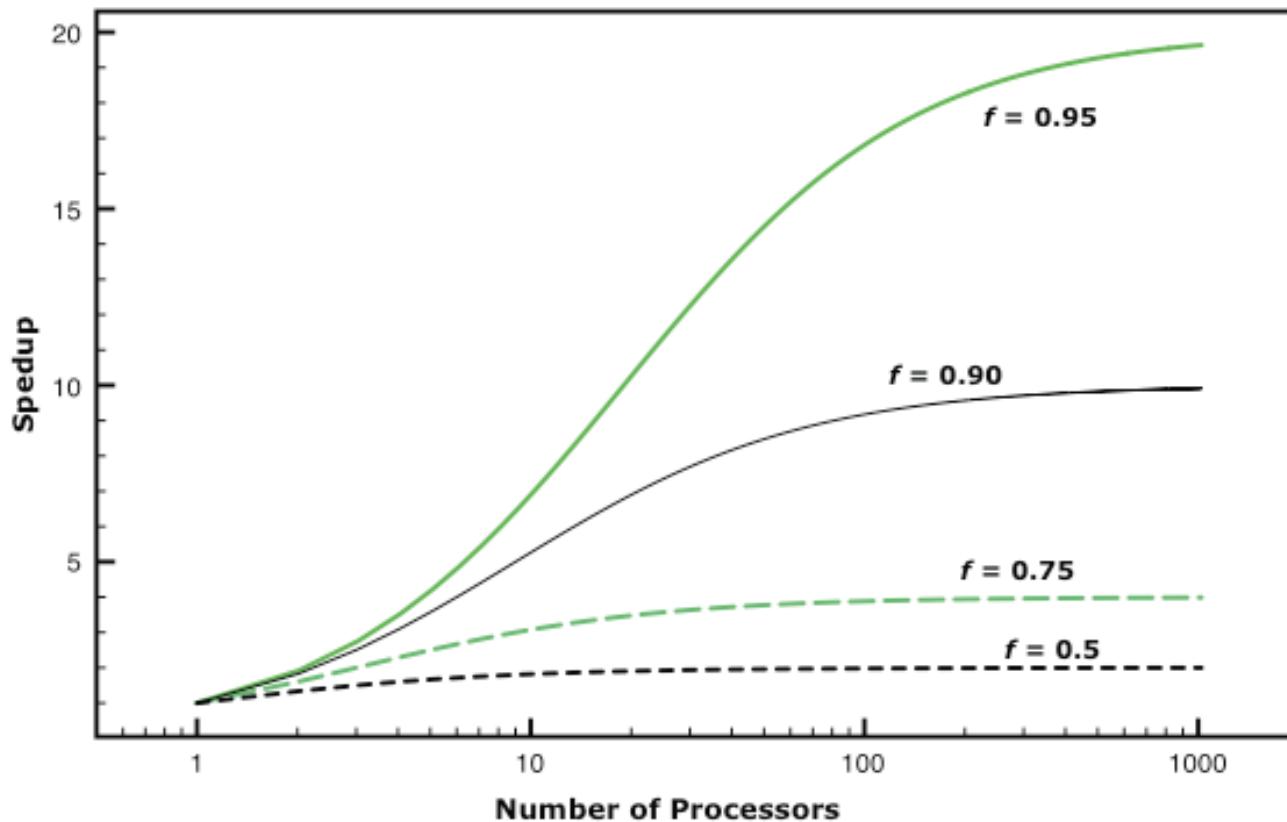
If a program needs 20 hours using a single processor core, and a particular part of the program which takes one hour to execute cannot be parallelized, If there are no limitations of using processors, then calculate the **minimum execution time** of the program.

Answer:

while the remaining 19 hours ($f = 0.95$) of execution time can be parallelized, then regardless of how many processors are devoted to a parallelized execution of this program, the minimum execution time cannot be less than that critical one hour. Hence, the theoretical speedup is limited to at most 20 times ($1/(1 - f) = 20$).



Amdahl's Law





Little's Law

- Fundamental and simple relation with broad applications
- Can be applied to almost any system that is statistically in steady state, and in which there is no leakage
- Queuing system
 - If server is idle an item is served immediately, otherwise an arriving item joins a queue
 - There can be a single queue for a single server or for multiple servers, or multiples queues with one being for each of multiple servers



Little's Law

- Average number of items in a queuing system equals the average rate at which items arrive multiplied by the time that an item spends in the system
 - Relationship requires very few assumptions
 - Because of its simplicity and generality it is extremely useful

Number of items in the system (L) = the rate items enter and leave the system (A / arrival rate / departure rate / throughput / λ) x the average amount of time items spend in the system (w/ lead time)

$$L = A \times W$$

Little's Law

You are estimating number of threads required by your server to execute clients requests efficiently and initially you starts 4 threads on the server. Request arrival rate on your server is 4 request/sec and each request takes fixed amount of time to complete with following time descriptions. The arrival rate is fixed and all new request arrivals have fixed service time.

Request_1= 0.2 sec; Request_2 =0.9 sec; Request_3=0.6 sec
Request_4=0.5 sec

What improvement factor should you think for the maximization of your thread uses?

Little's Law

Since each request would be assigned to each thread so no waits are there so Average Response time is $(0.2+0.9+0.6+0.5)/4=0.55$

Request arrival rate is 4.

Applying [Little's law](#) to estimate required threads on server to serve requests is now as follows:

Required threads on server =Request arrival rate * average response time= $4*(0.55)=2.2$

So, there should be 2 threads only for request arrivals of 4 req/sec with given service times,

and in this case any 2 requests must wait on each cycle of arrivals because we have only 2 resources (threads) and arrival rate is 4 requests/sec. So any 2 requests must wait and this results in increased response time. And two 2 threads on the server will always remain idle.

Summary

- Performance is specific to a particular program
 - total execution time is a consistent summary of performance
- For a given architecture performance increases come from:
 - increases in clock rate (without adverse CPI affects)
 - improvements in processor organization that lower CPI
 - compiler enhancements that lower CPI and/or instruction count

Thank you 😊

CSE 213

Computer Architecture

Lecture 3

A Top-Level View of Computer Function and Interconnection

Military Institute of Science and Technology

Outline

- Computer Components
- Computer Functions
- Interconnection Structures
- Bus Interconnection

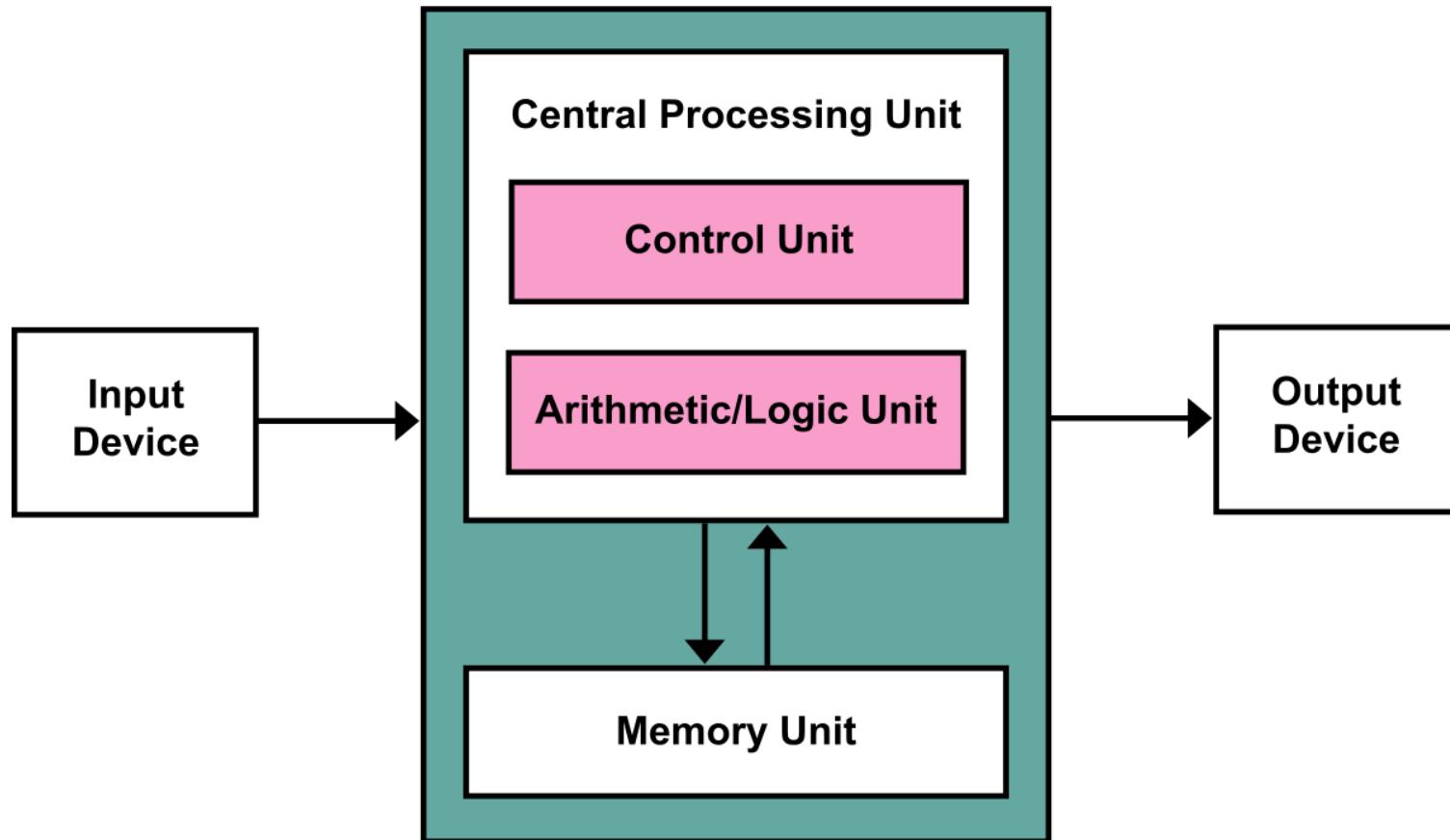


Computer Components

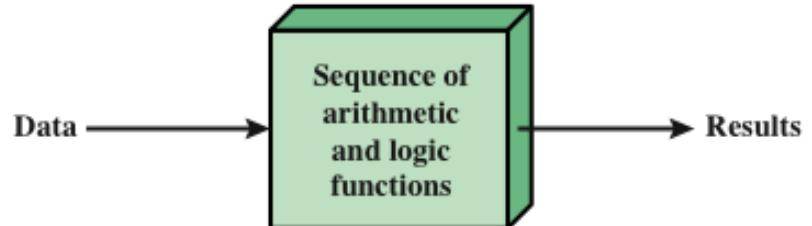
- Contemporary computer designs are based on concepts developed by John von Neumann at the Institute for Advanced Studies, Princeton, referred to as the *von Neumann architecture* and is based on three key concepts:
 - Data and instructions are stored in a single read-write memory
 - The contents of this memory are addressable by location, without regard to the type of data contained there
 - Execution occurs in a sequential fashion (unless explicitly modified) from one instruction to the next
- *Hardwired program*
 - If there is a particular computation to be performed, a configuration of logic components designed specifically for that computation could be constructed. The process of connecting the various components in the desired configuration as a form of programming. The resulting “program” is in the form of hardware and is termed a hardwired program.

+

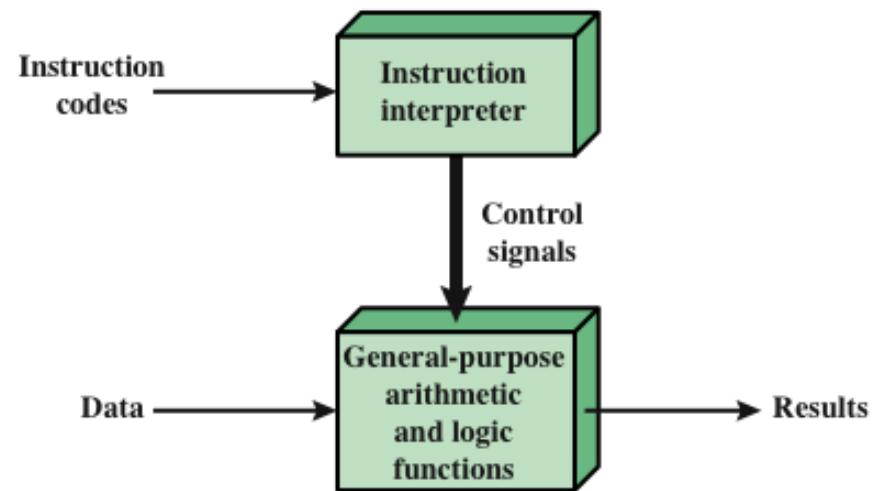
von Neumann architecture



Hardware and Software Approaches



(a) Programming in hardware



(b) Programming in software

Figure 3.1 Hardware and Software Approaches

Software

- A sequence of codes or instructions
- Part of the hardware interprets each instruction and generates control signals
- Provide a new sequence of codes for each new program instead of rewiring the hardware

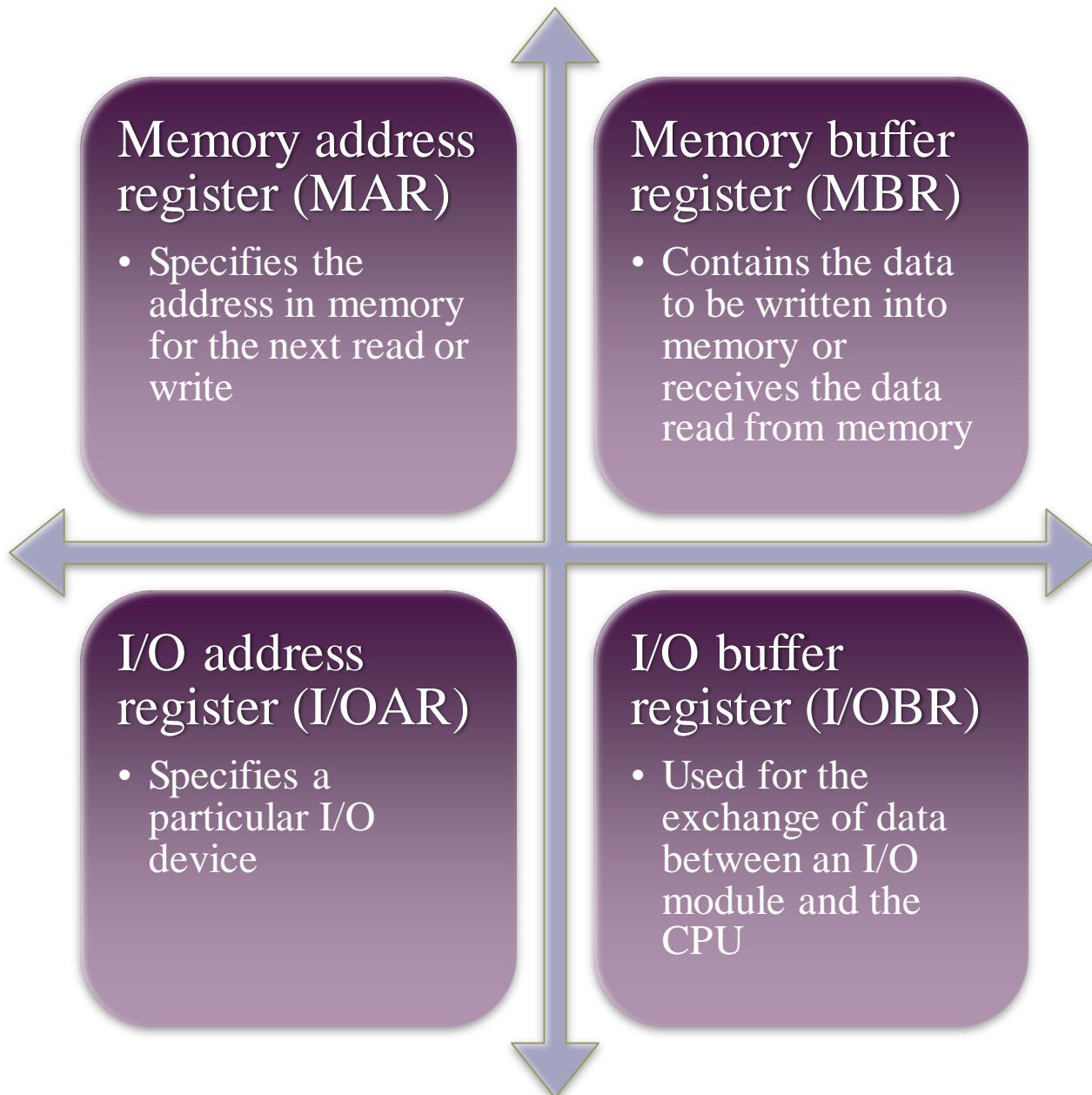
Major components:

- CPU
 - Instruction interpreter
 - Module of general-purpose arithmetic and logic functions
- I/O Components
 - Input module
 - Contains basic components for accepting data and instructions and converting them into an internal form of signals usable by the system
 - Output module
 - Means of reporting results

Software

I/O
Components



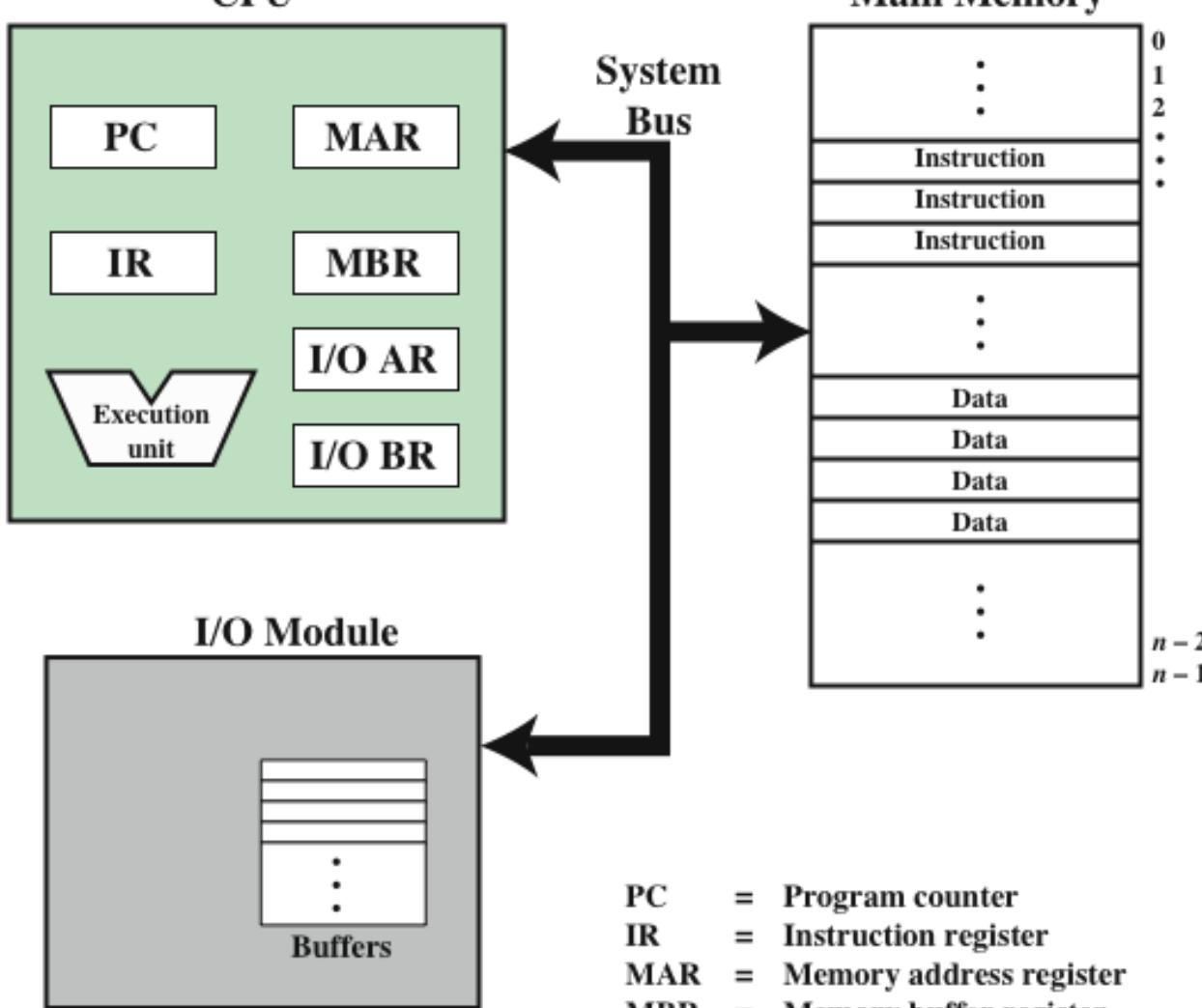


MEMORY

MAR

MBR

Computer Components: Top Level View



PC	= Program counter
IR	= Instruction register
MAR	= Memory address register
MBR	= Memory buffer register
I/O AR	= Input/output address register
I/O BR	= Input/output buffer register

Figure 3.2 Computer Components: Top-Level View

+

Computer Function

- The basic function performed by a computer is execution of a program, which consists of a set of instructions stored in memory.
- Instruction processing consists of two steps:
 - The processor reads (fetches) instructions from memory one at a time and
 - executes each instruction

Basic Instruction Cycle

The processing required for a single instruction is called an **instruction cycle**.

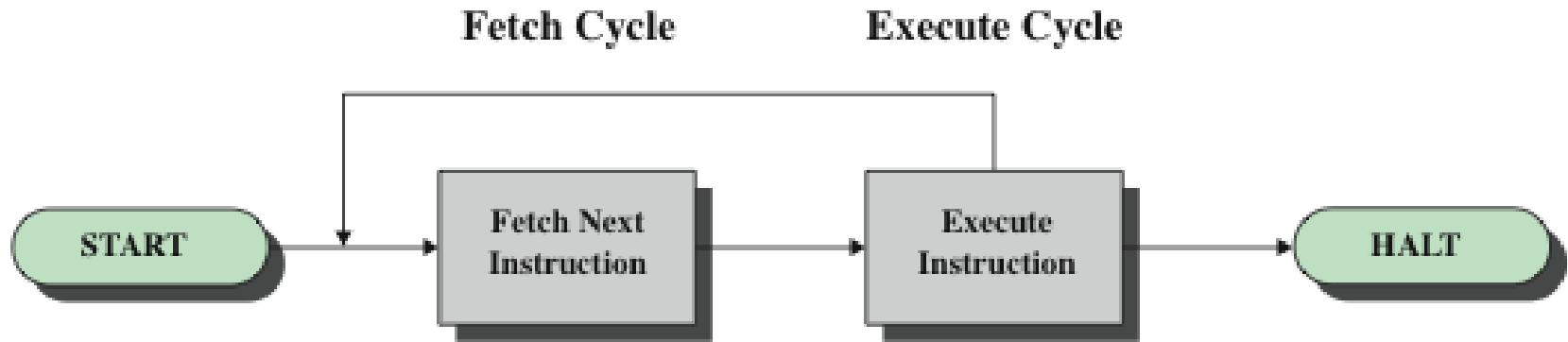


Figure 3.3 Basic Instruction Cycle

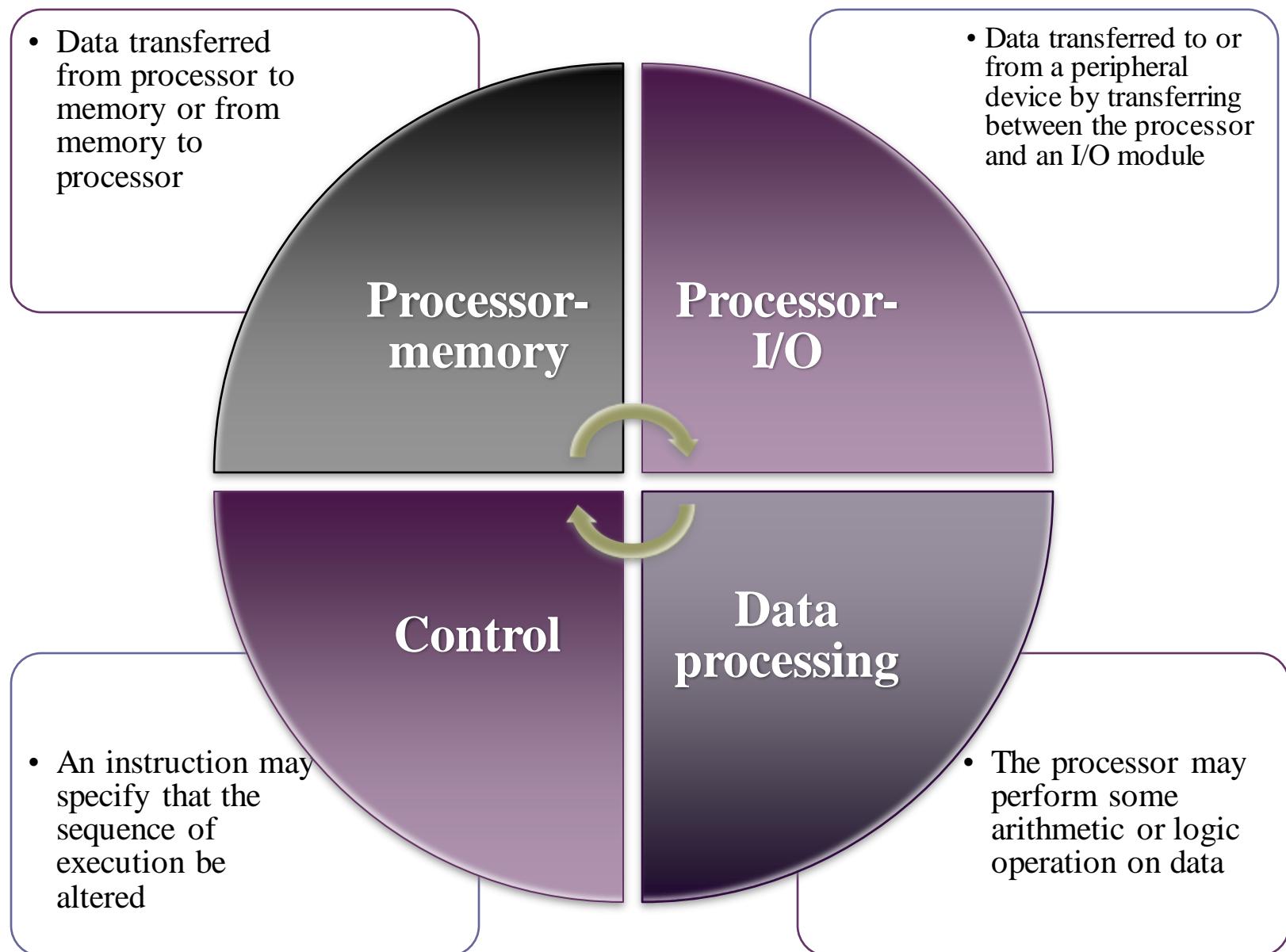


Fetch Cycle

- At the beginning of each instruction cycle the processor fetches an instruction from memory
- The program counter (PC) holds the address of the instruction to be fetched next
- The processor increments the PC after each instruction fetch so that it will fetch the next instruction in sequence
- The fetched instruction is loaded into the instruction register (IR)
- The processor interprets the instruction and performs the required action



Action Categories





Hypothetical machine: Example

- Consider a simple example using a hypothetical machine that includes the
 - The processor contains a single **data register**, called an accumulator (AC).
 - Both **instructions and data** are 16 bits long. Thus, it is convenient to organize memory using 16-bit words.
 - The instruction format provides **4 bits** for the **opcode**, so that there can be as many as $2^4 = 16$ different opcodes, and up to $2^{12} = 4096$ (4K) words of memory can be directly addressed.

0 3 4

Opcode	Address
--------	---------

(a) Instruction format

0 1

	Magnitude
--	-----------

(b) Integer format

Program counter (PC) = Address of instruction

Instruction register (IR) = Instruction being executed

Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from memory

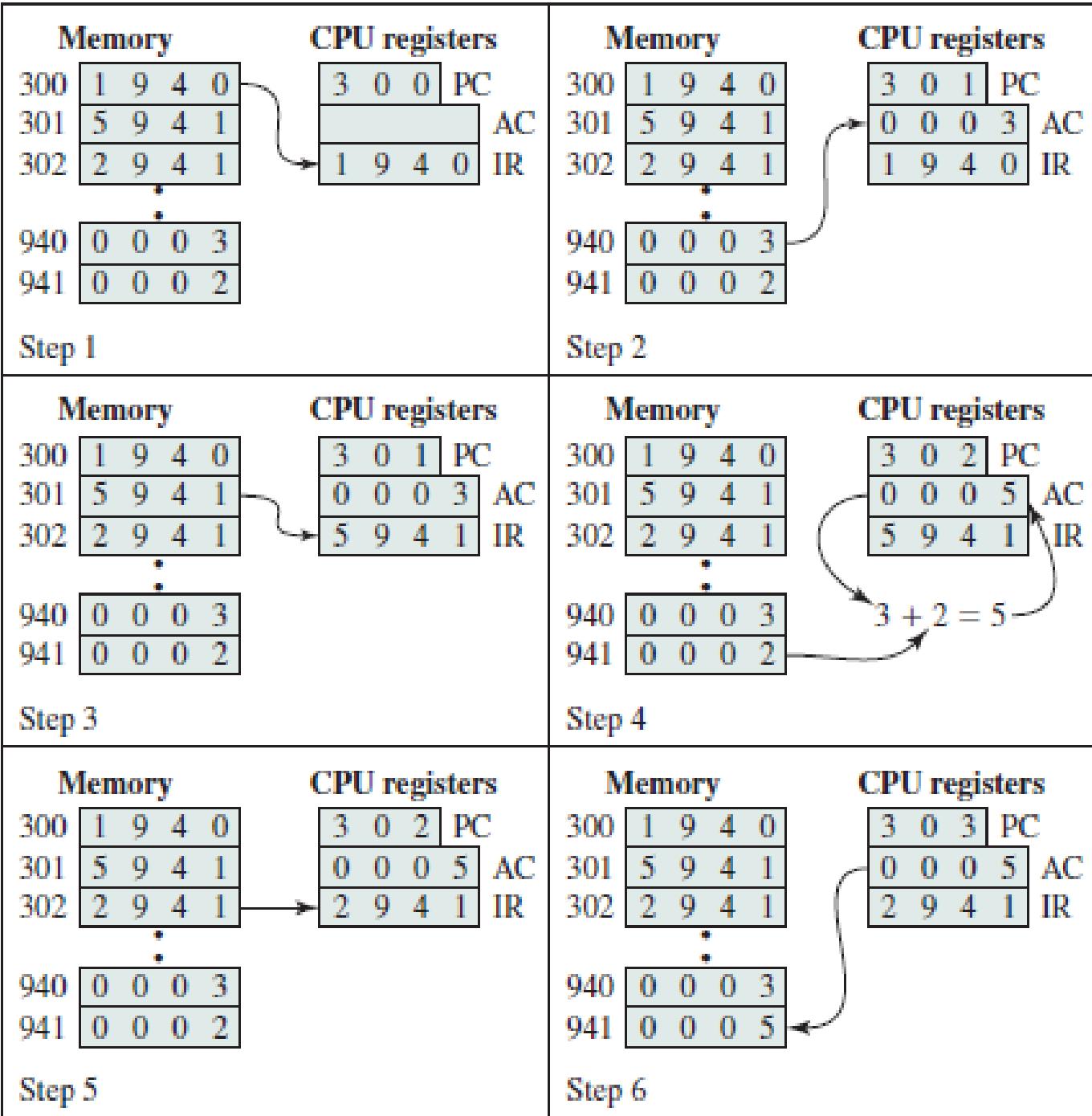
0010 = Store AC to memory

0101 = Add to AC from memory

(d) Partial list of opcodes

Figure 3.4 Characteristics of a Hypothetical Machine

Example of Program Execution



Instruction Cycle State Diagram

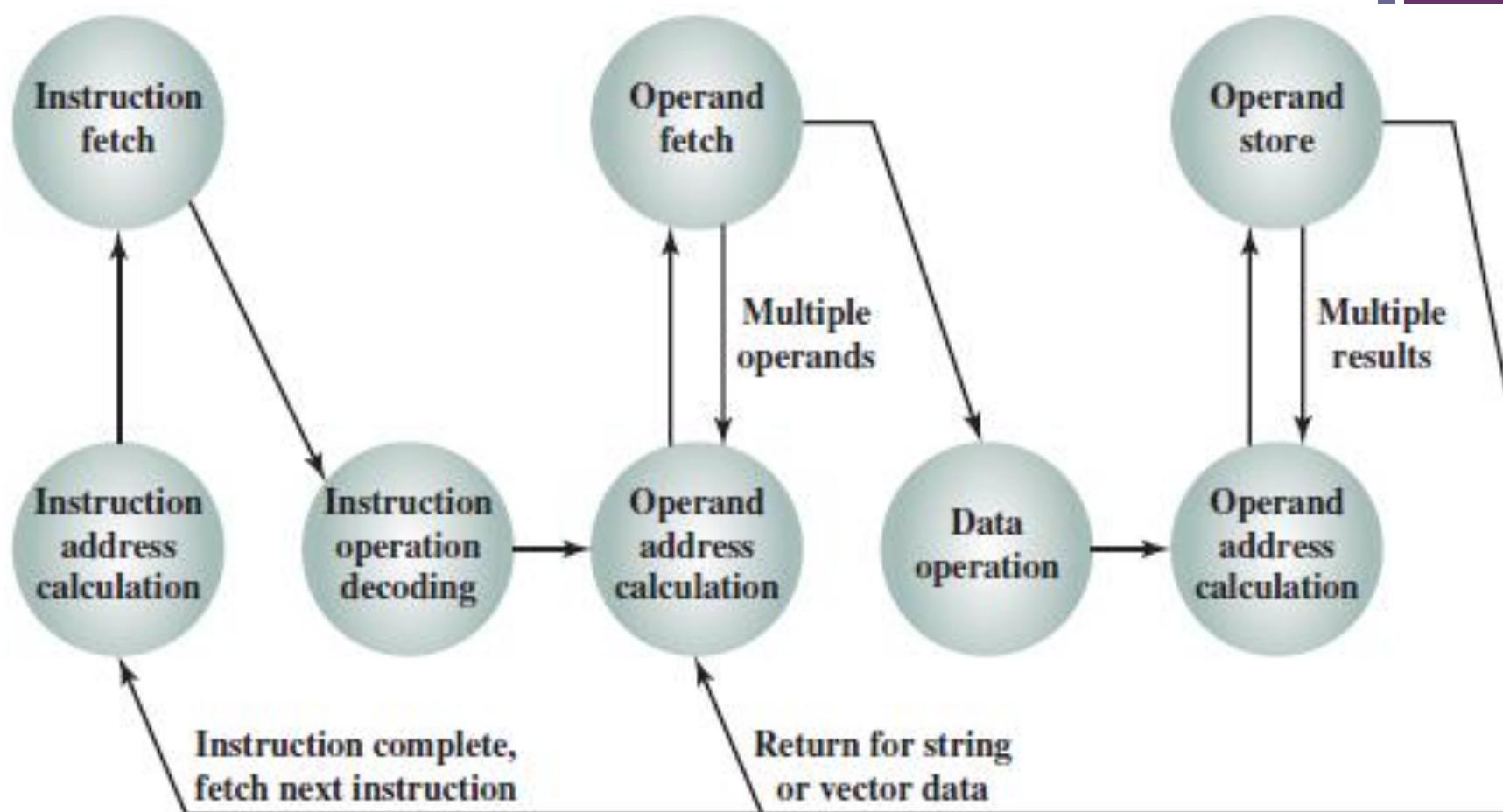


Figure 3.6 Instruction Cycle State Diagram



Instruction Cycle State Diagram

- **Instruction address calculation (iac):** Determine the address of the next instruction to be executed. Usually, this involves adding a fixed number to the address of the previous instruction.
- **Instruction fetch (if):** Read instruction from its memory location into the processor.
- **Instruction operation decoding (iod):** Analyze instruction to determine type of operation to be performed and operand(s) to be used.
- **Operand address calculation (oac):** If the operation involves reference to an operand in memory or available via I/O, then determine the address of the operand.



Instruction Cycle State Diagram

- **Operand fetch (of):** Fetch the operand from memory or read it in from I/O.
- **Data operation (do):** Perform the operation indicated in the instruction.
- **Operand store (os):** Write the result into memory or out to I/O.

Example: PDP-11 instruction **ADD A,B**

Sequence of states: iac, if, iod, oac, of, oac, of, do, oac, os



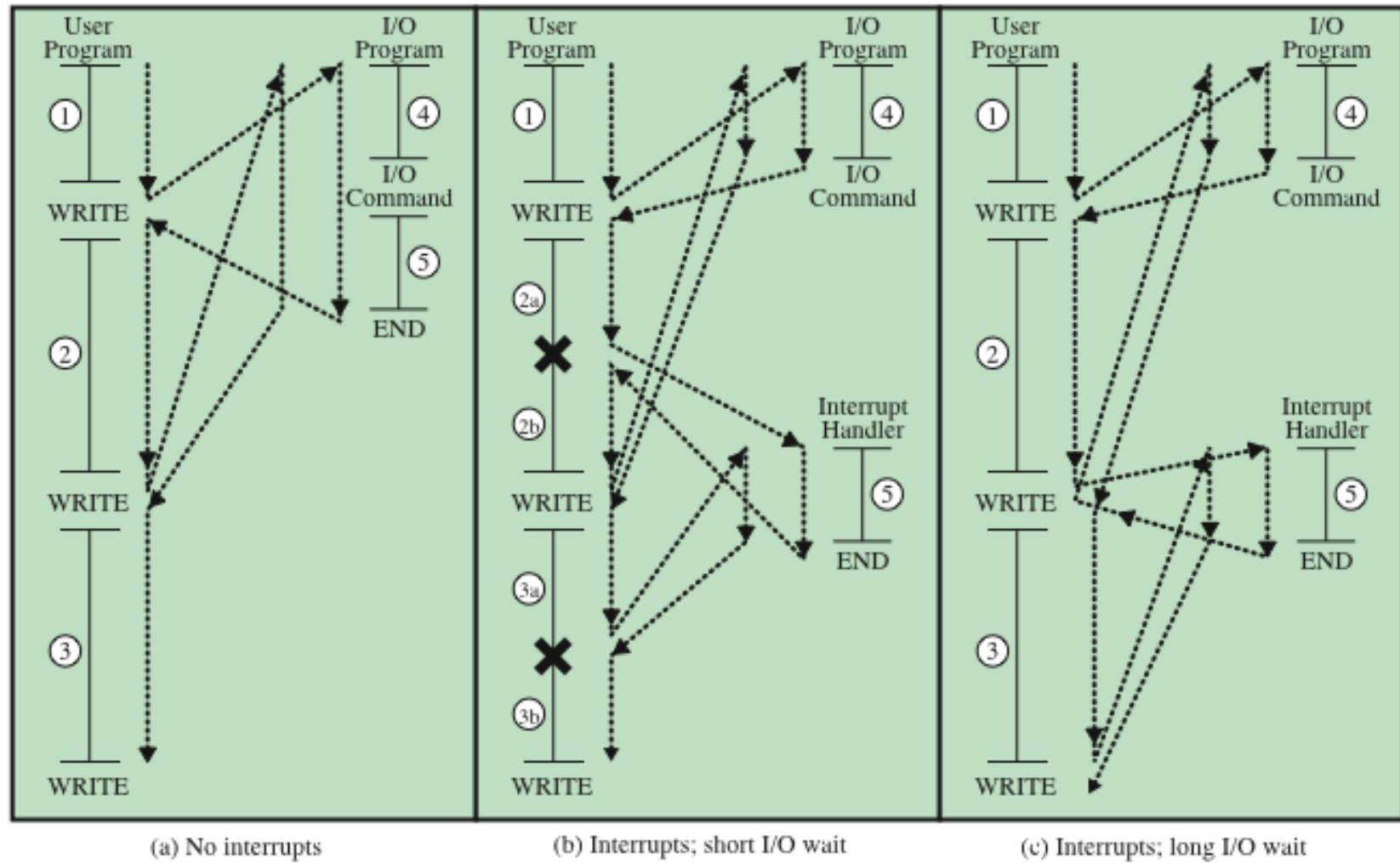
Classes of Interrupts

All computers provide a mechanism by which other modules (I/O, memory) may interrupt the normal processing of the processor.

Table 3.1 Classes of Interrupts

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, or reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation, request service from the processor, or to signal a variety of error conditions.
Hardware Failure	Generated by a failure such as power failure or memory parity error.

Program Flow Control



✗ = interrupt occurs during course of execution of user program

Figure 3.7 Program Flow of Control Without and With Interrupts

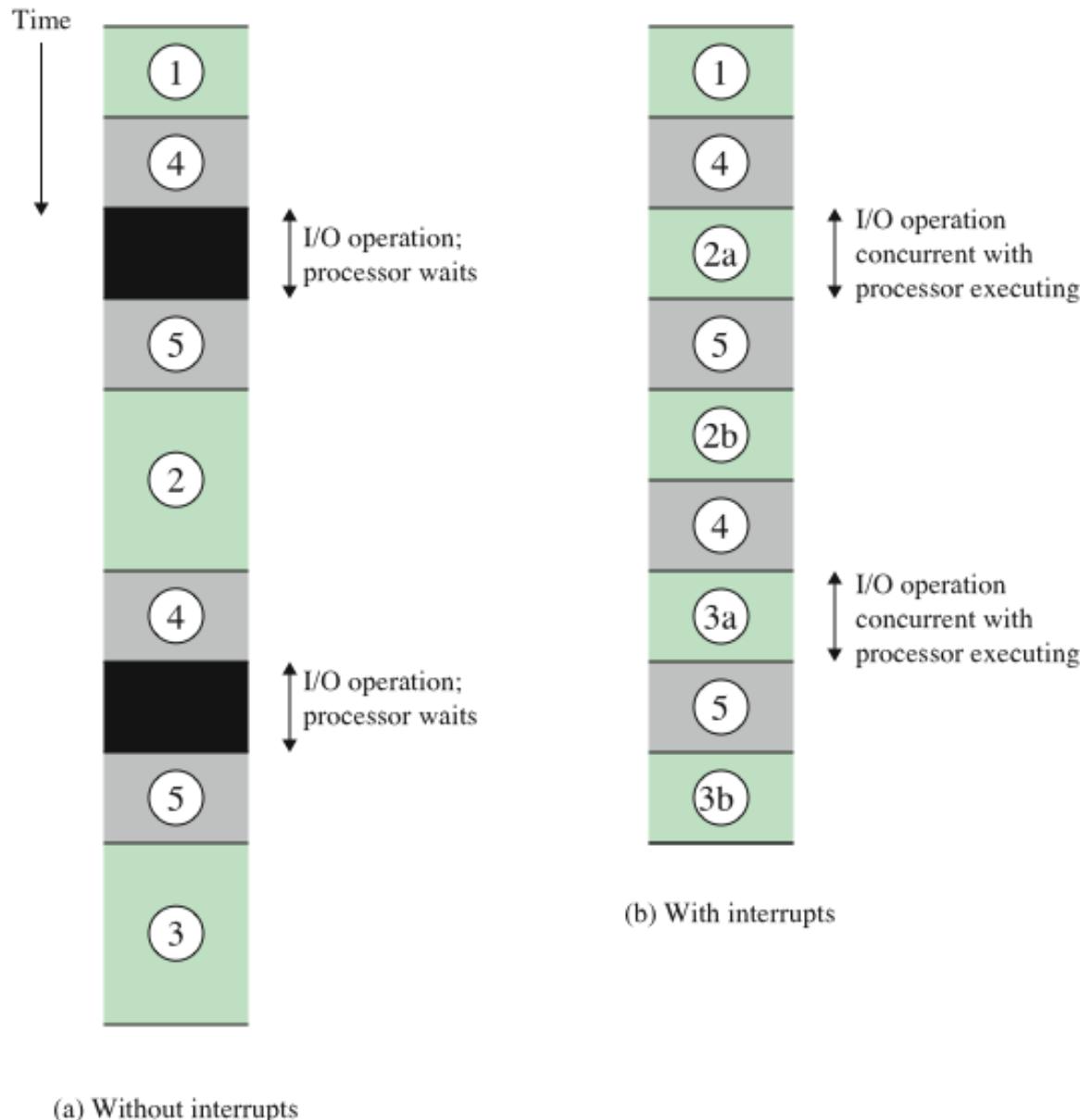
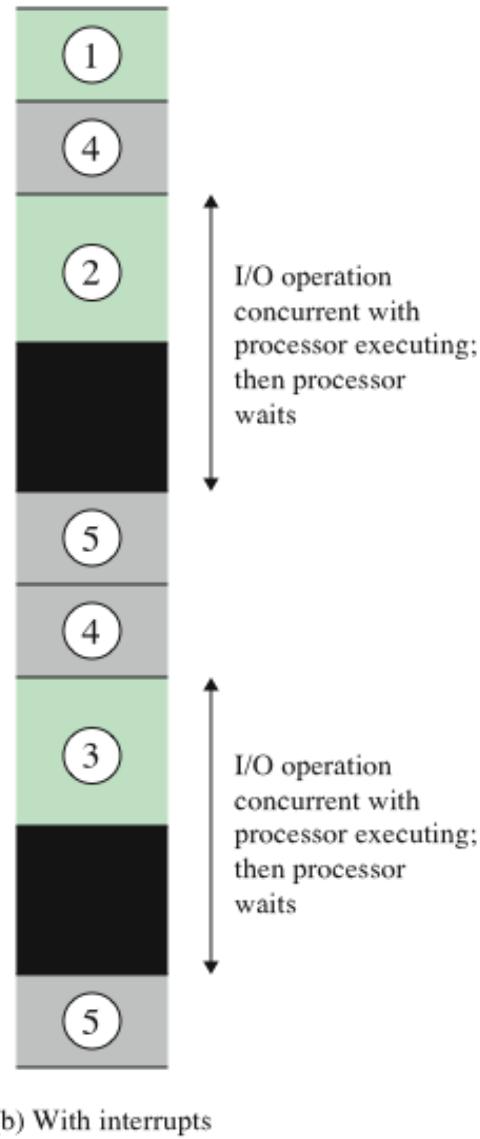
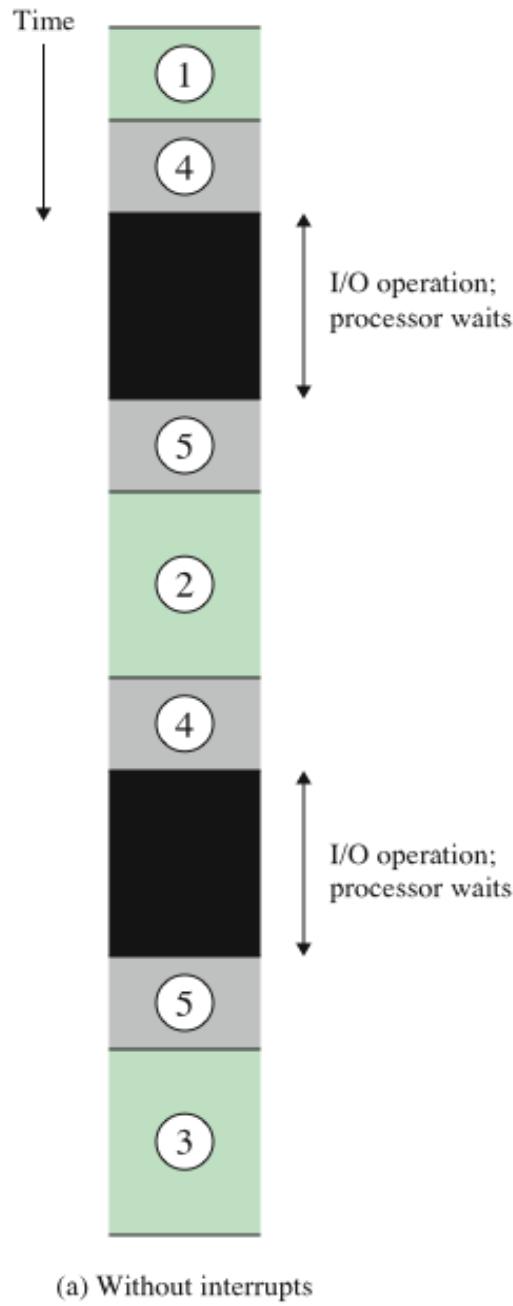


Figure 3.10 Program Timing: Short I/O Wait

Program Timing: Short I/O Wait



Program Timing: Long I/O Wait

Figure 3.11 Program Timing: Long I/O Wait

Instruction Cycle With Interrupts

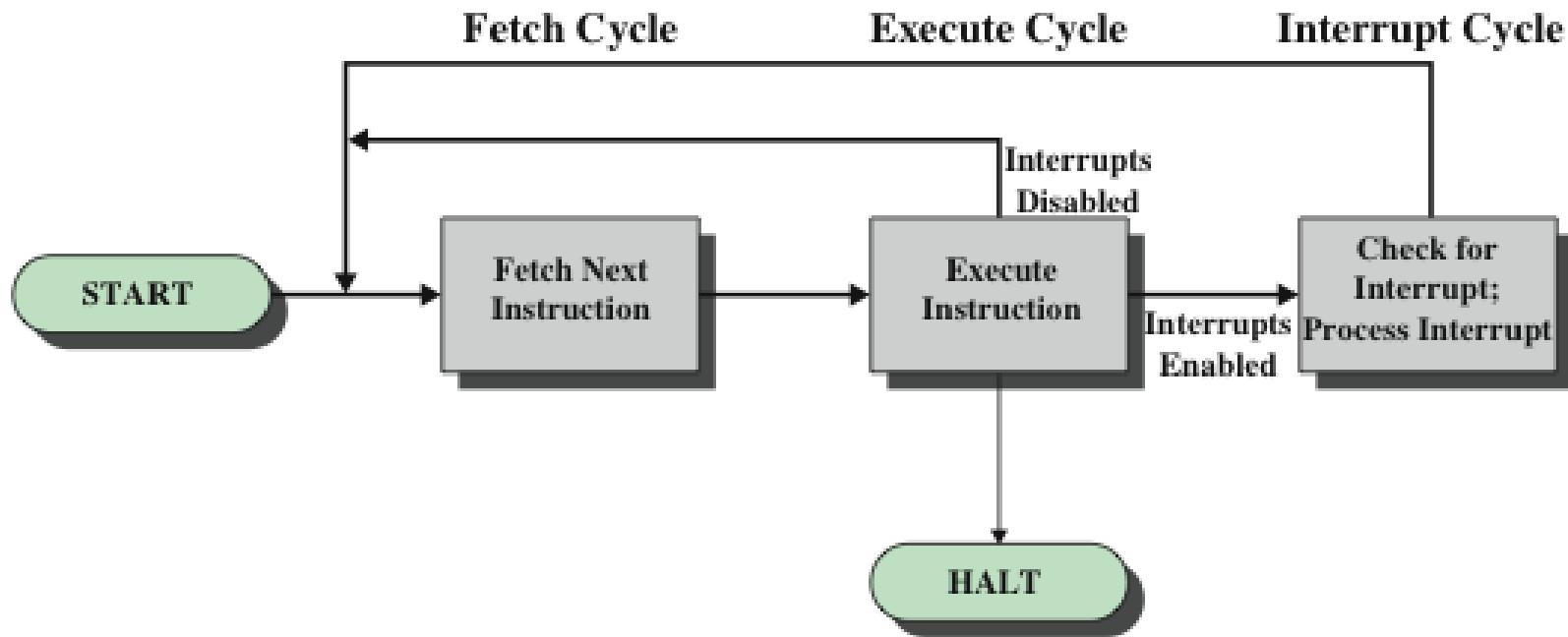


Figure 3.9 Instruction Cycle with Interrupts

Instruction Cycle State Diagram With Interrupts

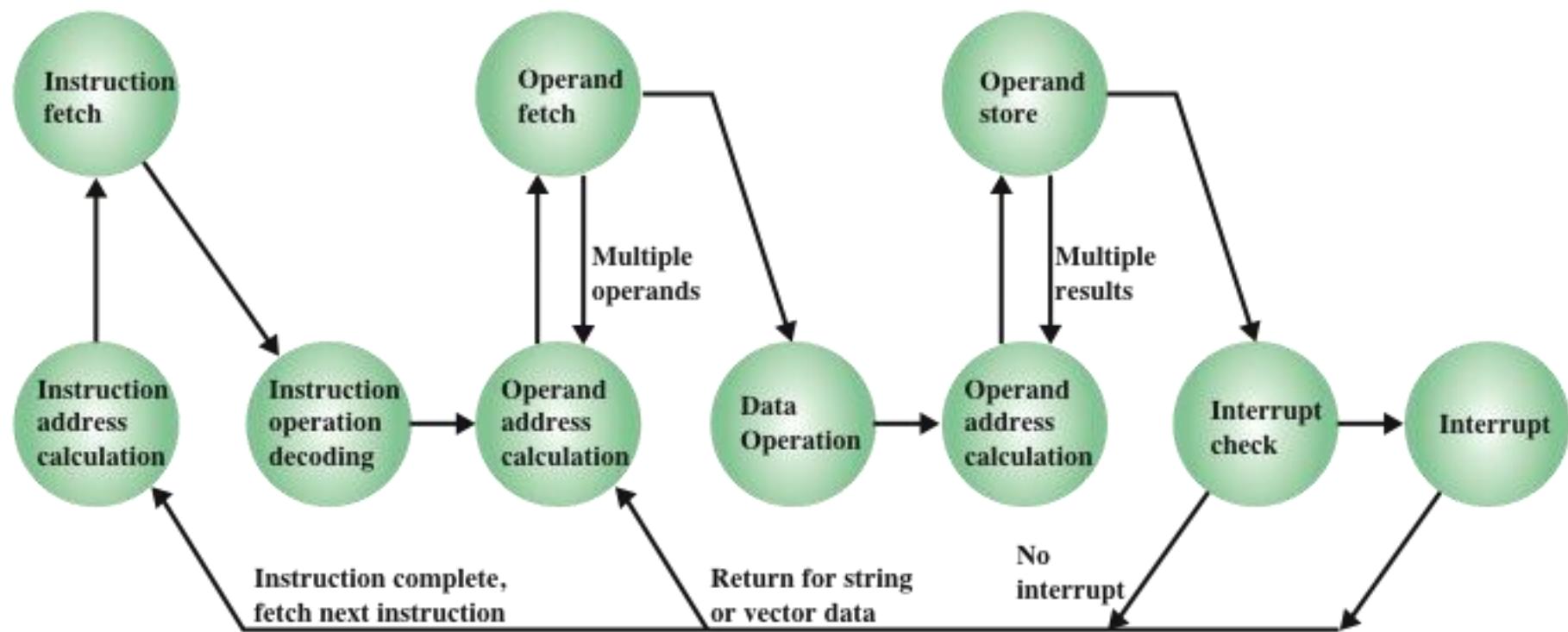


Figure 3.12 Instruction Cycle State Diagram, With Interrupts



Transfer of Control via Interrupts

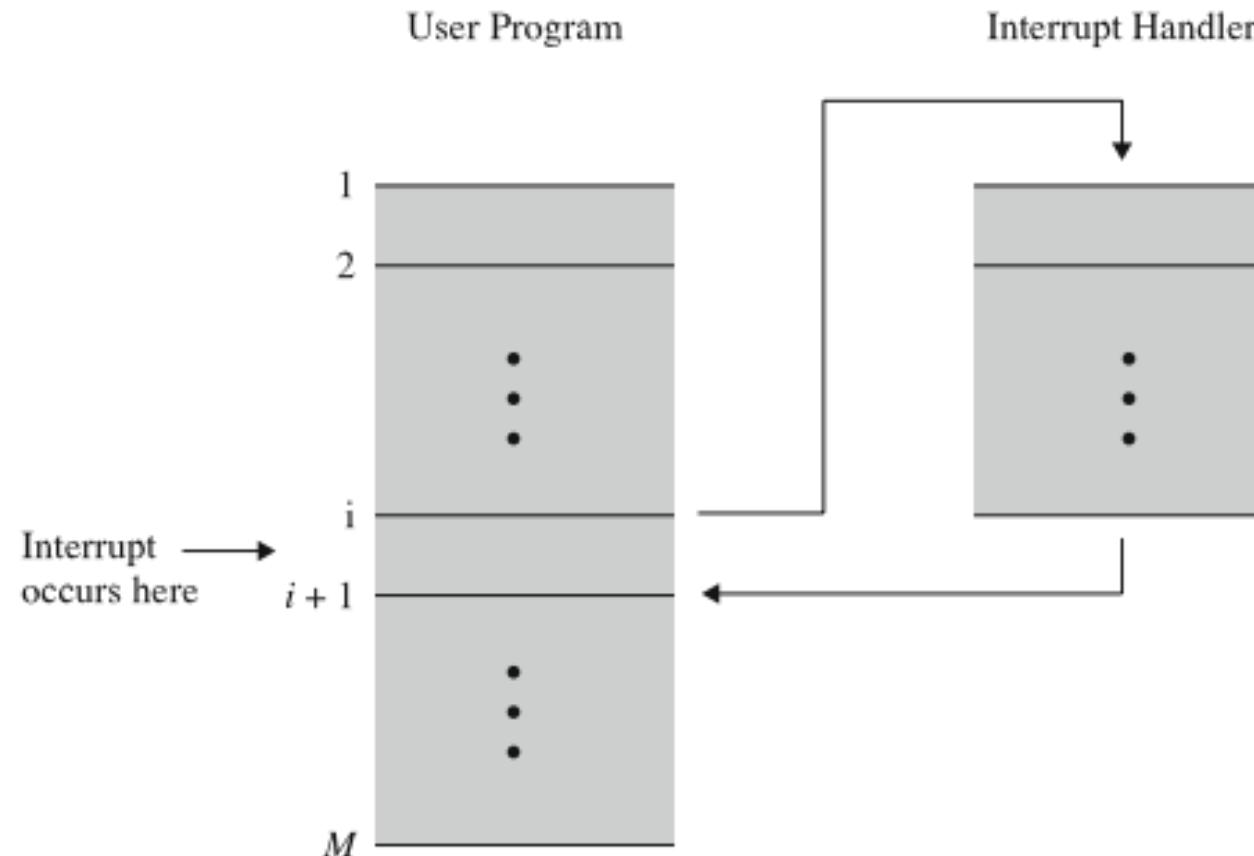
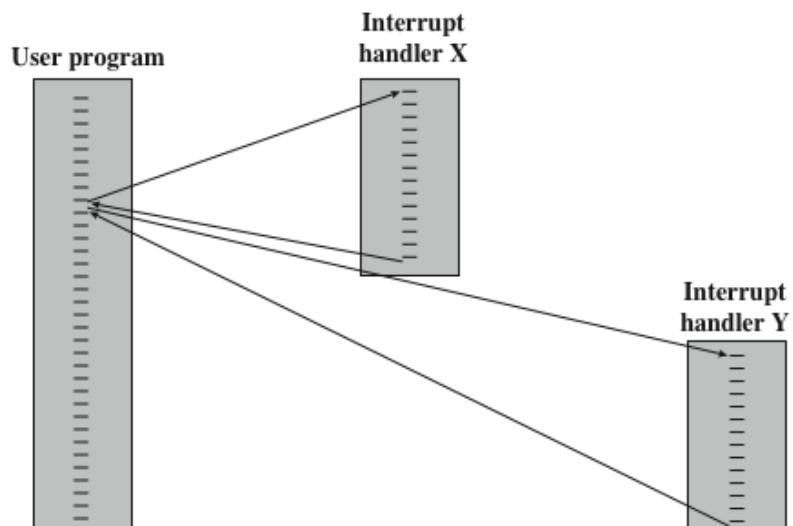


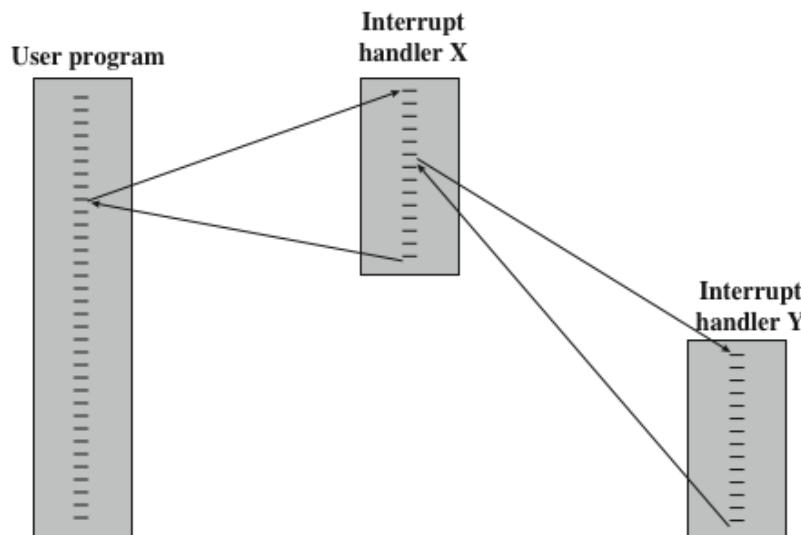
Figure 3.8 Transfer of Control via Interrupts

Transfer of Control

Multiple Interrupts



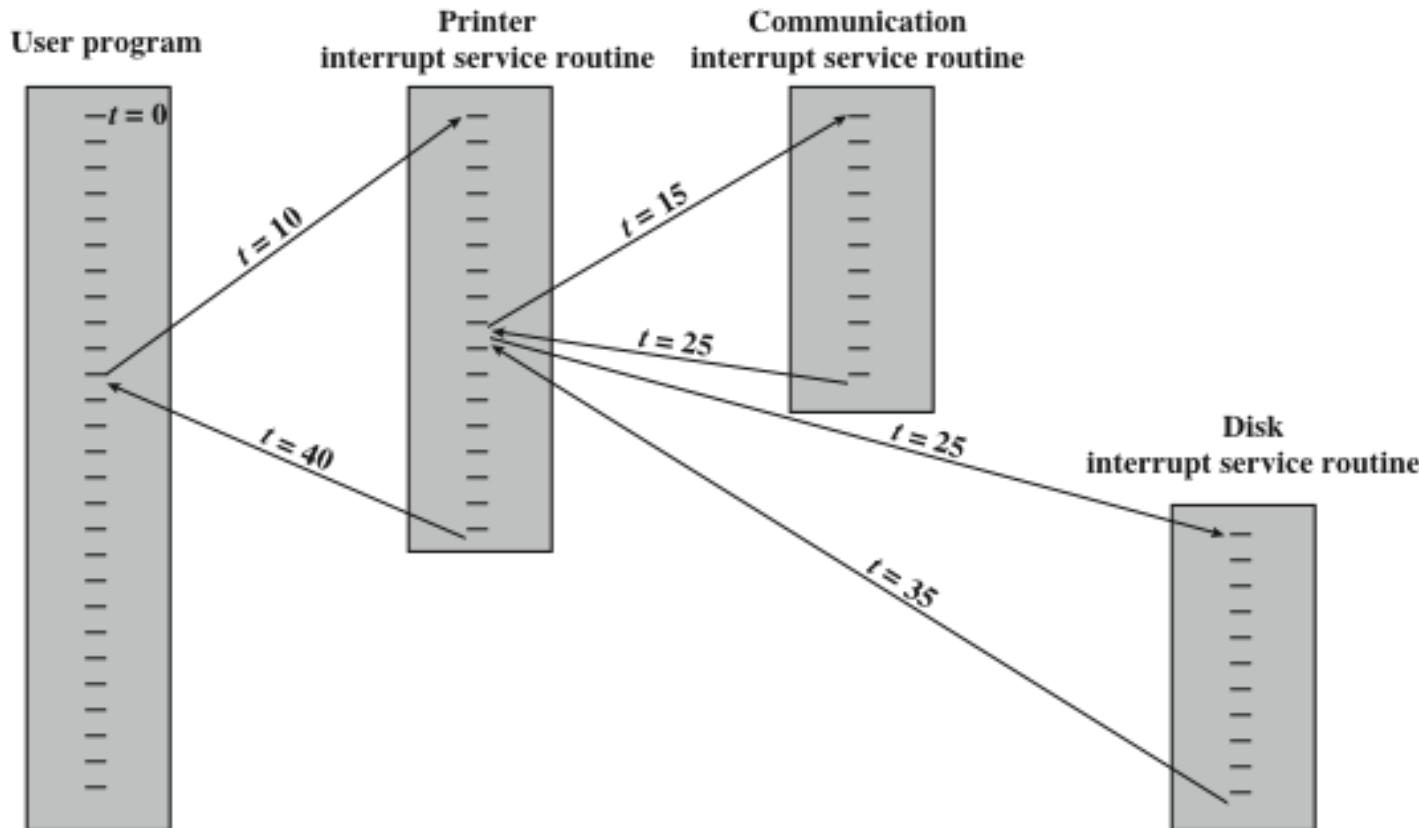
(a) Sequential interrupt processing



(b) Nested interrupt processing

Figure 3.13 Transfer of Control with Multiple Interrupts

+ Time Sequence of Multiple Interrupts



E
x
a
m
p
l
e

Figure 3.14 Example Time Sequence of Multiple Interrupts



I/O Function

- I/O module can exchange data directly with the processor
- Processor can read data from or write data to an I/O module
 - Processor identifies a specific device that is controlled by a particular I/O module
 - I/O instructions rather than memory referencing instructions
- In some cases it is desirable to allow I/O exchanges to occur directly with memory
 - The processor grants to an I/O module the authority to read from or write to memory so that the I/O memory transfer can occur without tying up the processor
 - The I/O module issues read or write commands to memory relieving the processor of responsibility for the exchange
 - This operation is known as direct memory access (DMA)

+ Computer Modules

The collection of paths connecting the various modules is called the **interconnection structure**.

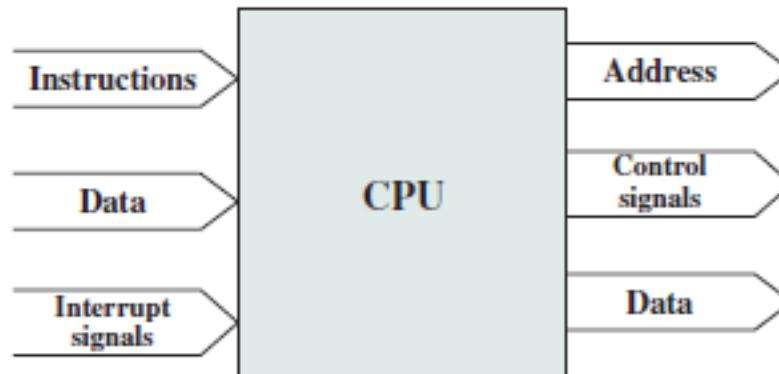
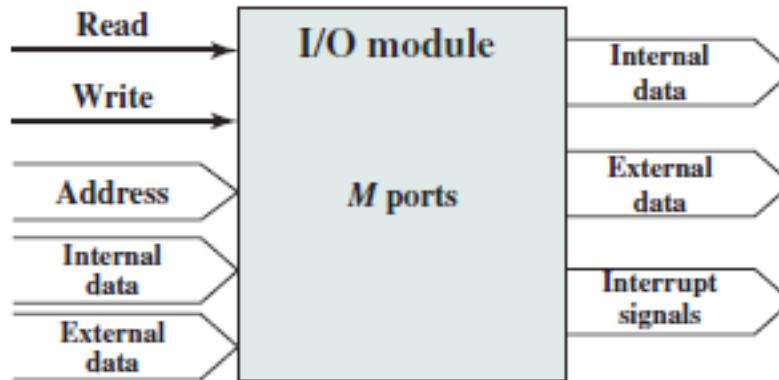
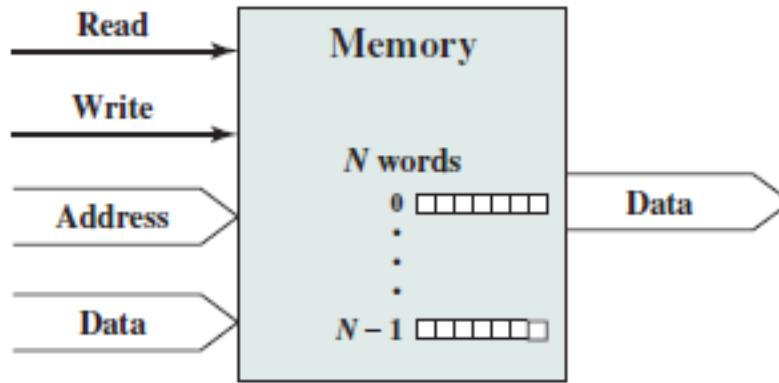


Figure 3.15 Computer Modules

The interconnection structure must support the following types of transfers:

Memory to processor

Processor reads an instruction or a unit of data from memory

Processor to memory

Processor writes a unit of data to memory

I/O to processor

Processor reads data from an I/O device via an I/O module

Processor to I/O

Processor sends data to the I/O device

I/O to or from memory

An I/O module is allowed to exchange data directly with memory without going through the processor using direct memory access

A communication pathway connecting two or more devices

- Key characteristic is that it is a shared transmission medium

Signals transmitted by any one device are available for reception by all other devices attached to the bus

- If two devices transmit during the same time period their signals will overlap and become garbled



Typically consists of multiple communication lines

- Each line is capable of transmitting signals representing binary 1 and binary 0

Computer systems contain a number of different buses that provide pathways between components at various levels of the computer system hierarchy



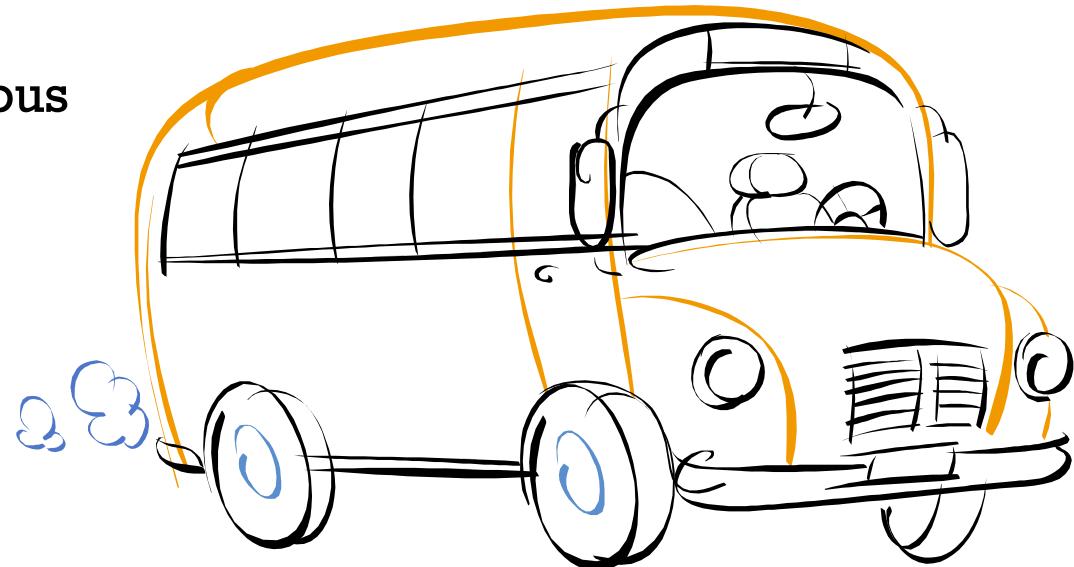
System bus

- A bus that connects major computer components (processor, memory, I/O)

The most common computer interconnection structures are based on the use of one or more system buses

Data Bus

- Data lines that provide a path for moving data among system modules
- May consist of 32, 64, 128, or more separate lines
- The number of lines is referred to as the *width* of the data bus
- The number of lines determines how many bits can be transferred at a time
- The width of the data bus is a key factor in determining overall system performance



Address Bus

- Used to designate the source or destination of the data on the data bus
 - If the processor wishes to read a word of data from memory it puts the address of the desired word on the address lines
- Width determines the maximum possible memory capacity of the system
- Also used to address I/O ports
 - The higher order bits are used to select a particular module on the bus and the lower order bits select a memory location or I/O port within the module

Control Bus

- Used to control the access and the use of the data and address lines
- Because the data and address lines are shared by all components there must be a means of controlling their use
- Control signals transmit both command and timing information among system modules
- Timing signals indicate the validity of data and address information
- Command signals specify operations to be performed

Bus Interconnection Scheme

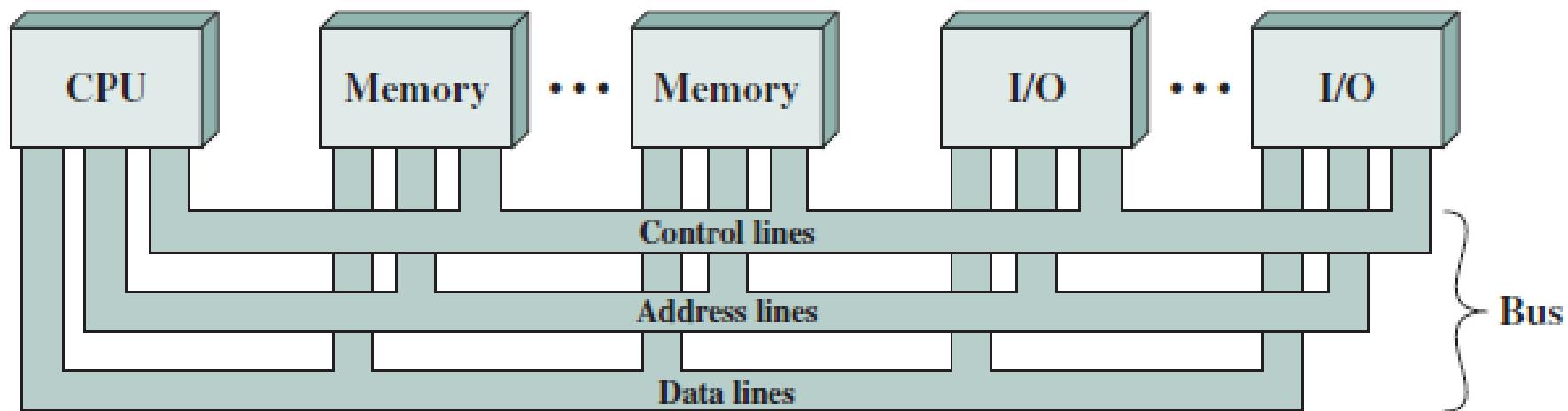
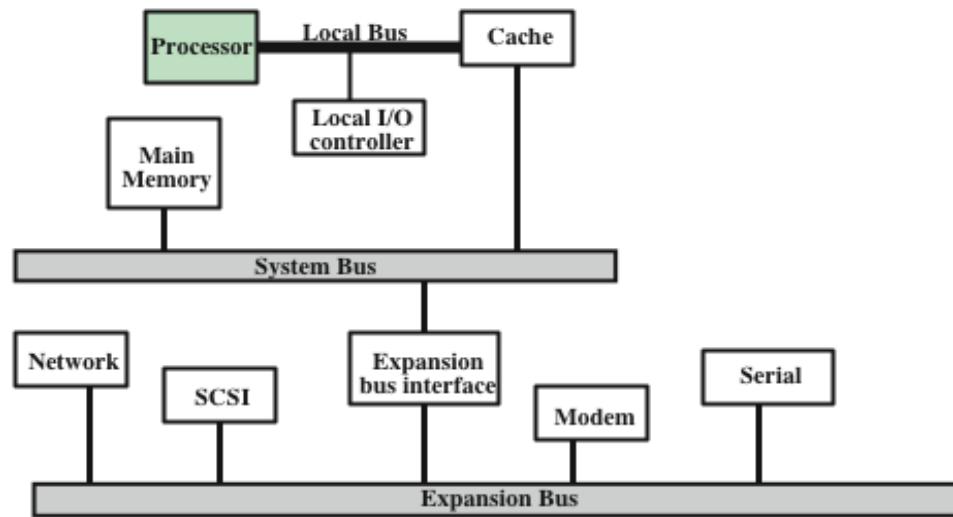
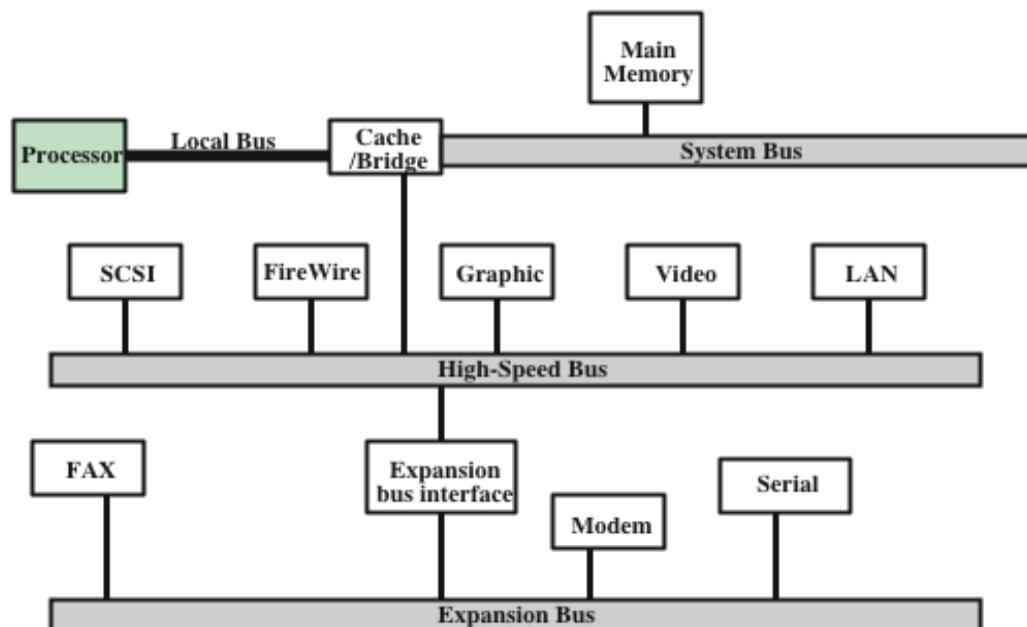


Figure 3.16 Bus Interconnection Scheme

C
o
n
f
i
g
u
r
a
t
i
o
n
s



(a) Traditional Bus Architecture



(b) High-Performance Architecture

Figure 3.17 Example Bus Configurations



Elements of Bus Design

Type	Bus Width
Dedicated	Address
Multiplexed	Data
Method of Arbitration	Data Transfer Type
Centralized	Read
Distributed	Write
Timing	Read-modify-write
Synchronous	Read-after-write
Asynchronous	Block

Timing of Synchronous Bus Operations

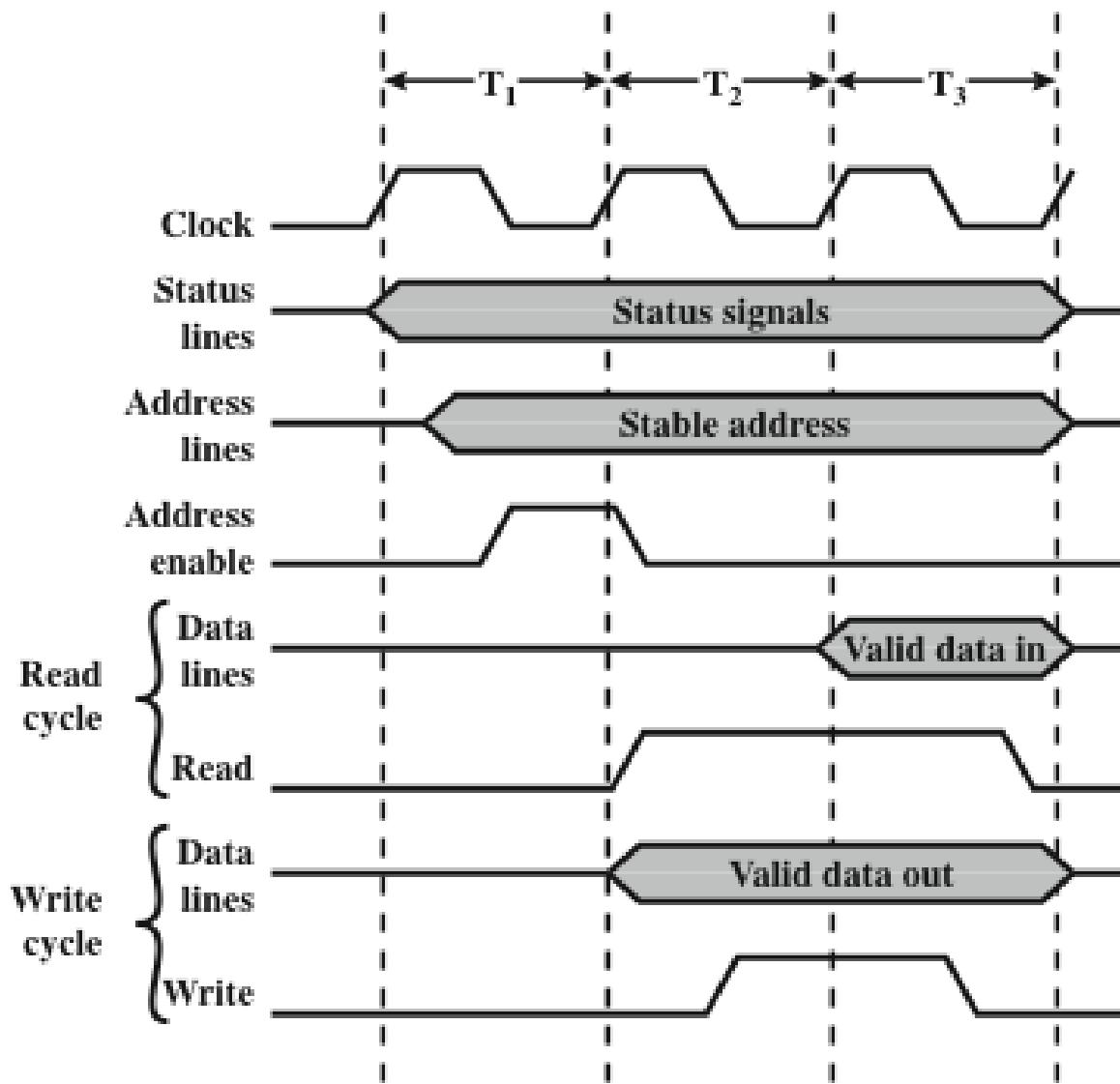


Figure 3.18 Timing of Synchronous Bus Operations



(a) System bus read cycle



(b) System bus write cycle

Timing of Asynchronous Bus Operations

Figure 3.19 Timing of Asynchronous Bus Operations



Point-to-Point Interconnect

Principal reason for driving the change from bus to point-to-point interconnect change was the electrical constraints encountered with increasing the frequency of wide synchronous buses

At higher and higher data rates it becomes increasingly difficult to perform the synchronization and arbitration functions in a timely fashion

A conventional shared bus on the same chip magnified the difficulties of increasing bus data rate and reducing bus latency to keep up with the processors

Has lower latency, higher data rate, and better scalability

+ Quick Path Interconnect

- Introduced in 2008

Significant characteristics of QPI:

- Multiple direct connections

- Direct pairwise connections to other components eliminating the need for arbitration found in shared transmission systems

- Layered protocol architecture

- These processor level interconnects use a layered protocol architecture rather than the simple use of control signals found in shared bus arrangements

- Packetized data transfer

- Data are sent as a sequence of packets each of which includes control headers and error control codes

QPI



Multicore Configuration Using QPI

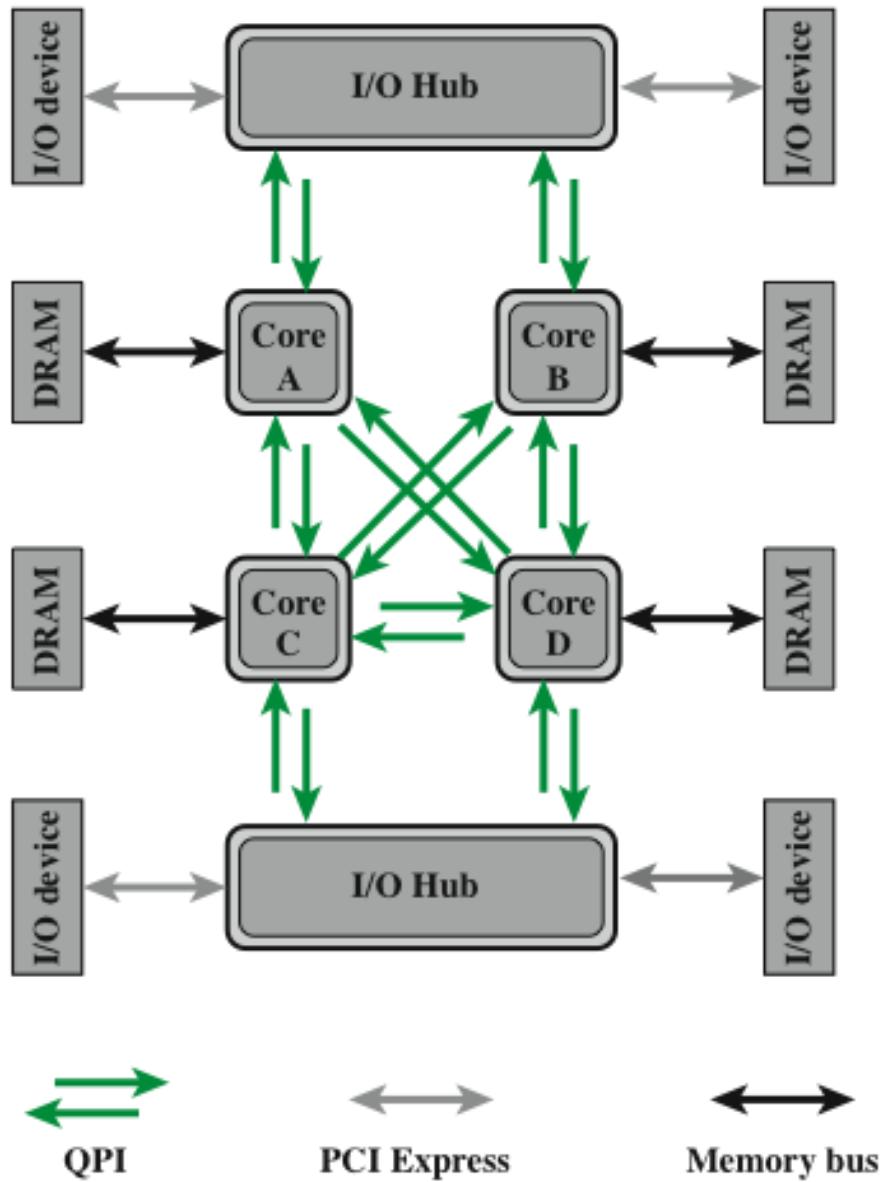


Figure 3.20 Multicore Configuration Using QPI

QPI Layers

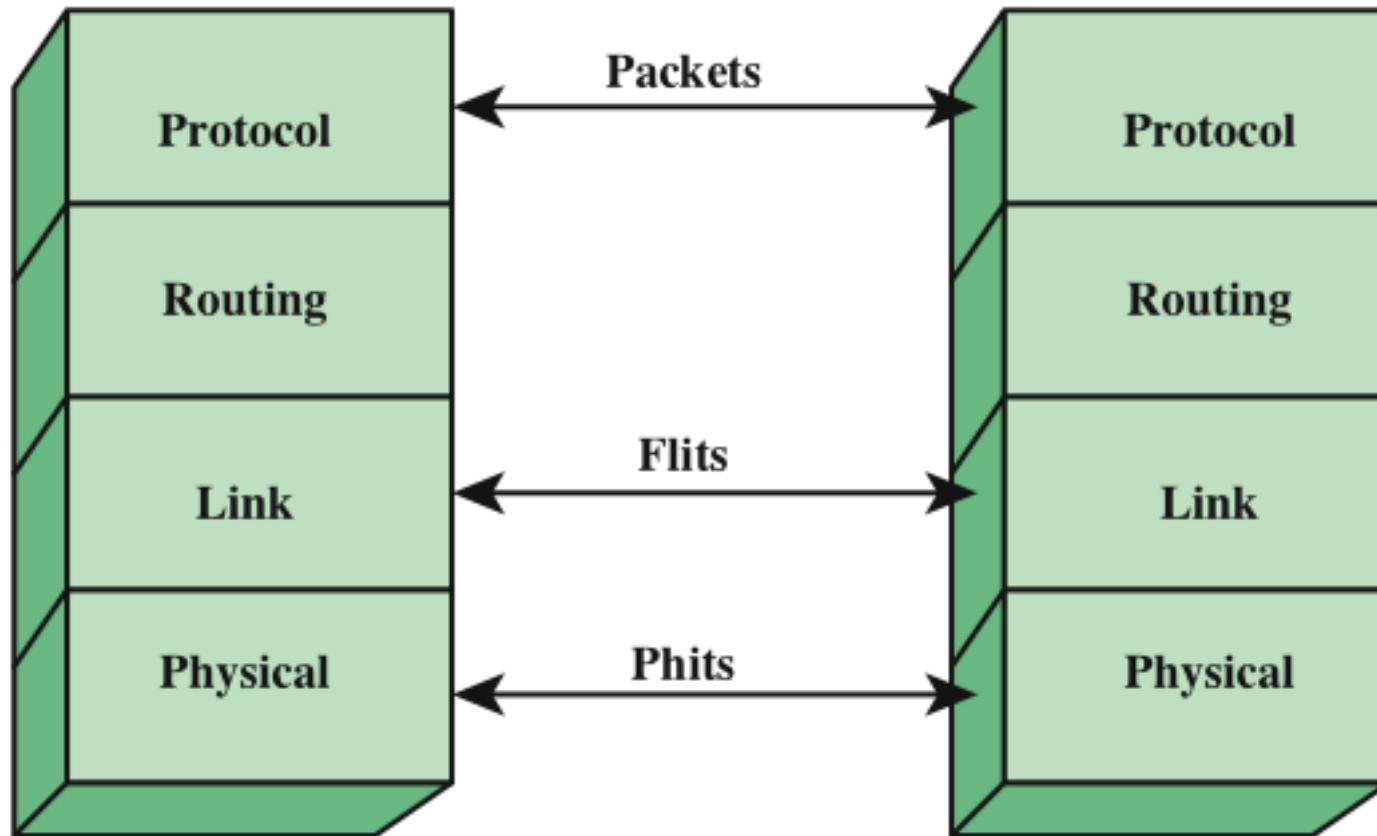


Figure 3.21 QPI Layers



Physical Interface of the Intel QPI Interconnect

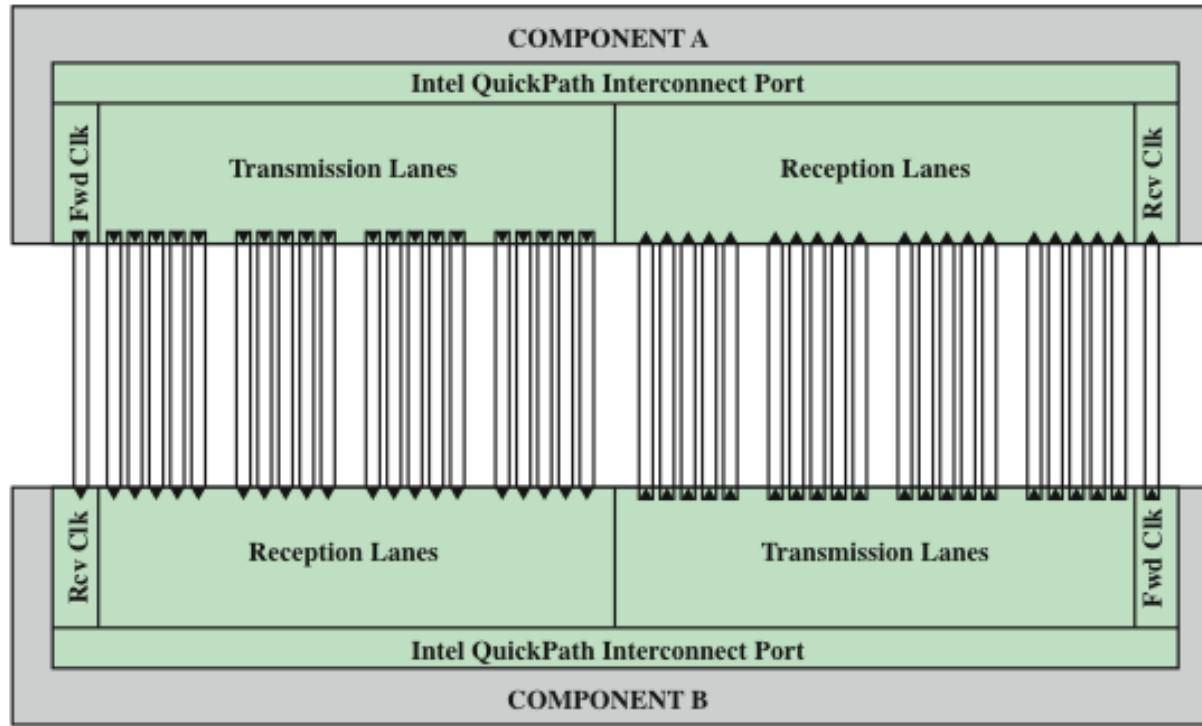


Figure 3.22 Physical Interface of the Intel QPI Interconnect

QPI Multilane Distribution

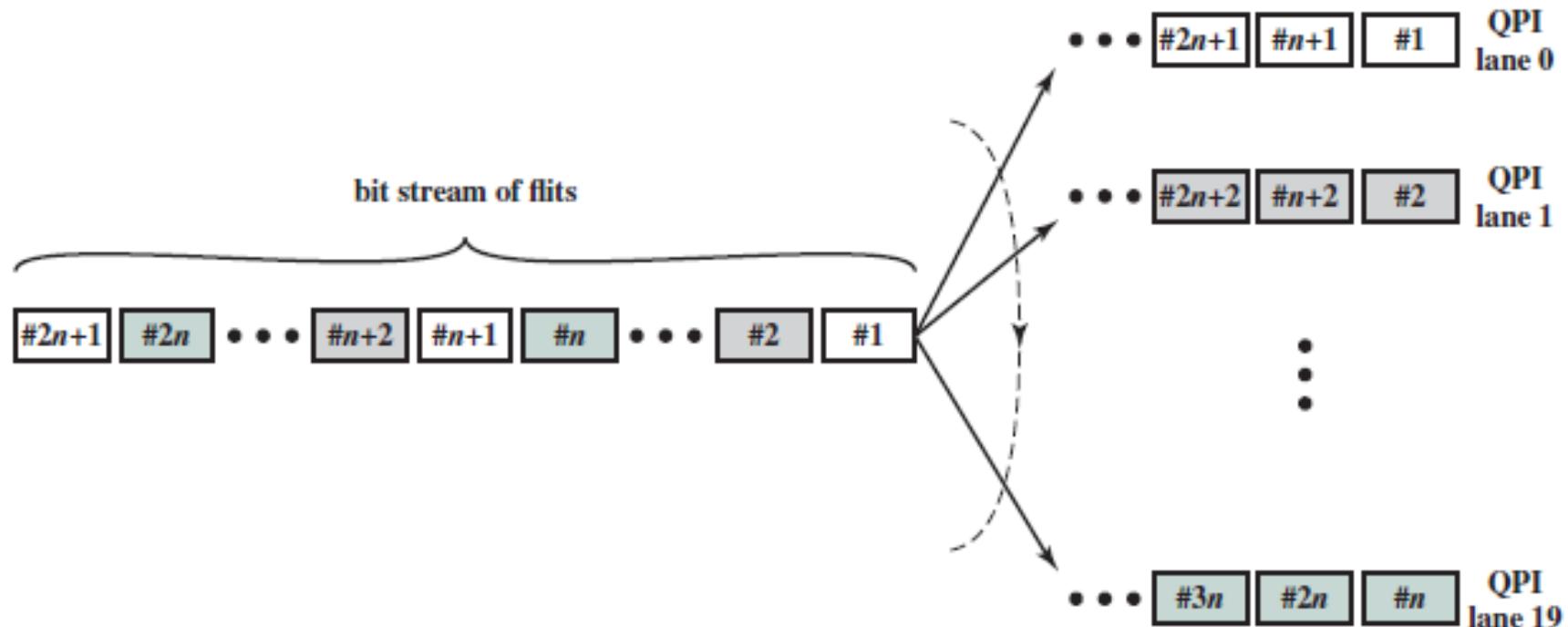


Figure 3.20 QPI Multilane Distribution



QPI Link Layer

- Performs two key functions: ***flow control*** and ***error control***
 - Operate on the level of the flit (flow control unit)
 - Each flit consists of a 72-bit message payload and an 8-bit error control code called a *cyclic redundancy check* (CRC)
- Flow control function
 - Needed to ensure that a sending QPI entity does not overwhelm a receiving QPI entity by sending data faster than the receiver can process the data and clear buffers for more incoming data
- Error control function
 - Detects and recovers from bit errors, and so isolates higher layers from experiencing bit errors



QPI Routing and Protocol Layers

Routing Layer

- Used to determine the course that a packet will traverse across the available system interconnects
- Defined by firmware and describe the possible paths that a packet can follow

Protocol Layer

- Packet is defined as the unit of transfer
- One key function performed at this level is a cache coherency protocol which deals with making sure that main memory values held in multiple caches are consistent
- A typical data packet payload is a block of data being sent to or from a cache



Peripheral Component Interconnect (PCI)

- A popular high bandwidth, processor independent bus that can function as a mezzanine or peripheral bus
- Delivers better system performance for high speed I/O subsystems
- PCI Special Interest Group (SIG)
 - Created to develop further and maintain the compatibility of the PCI specifications
- PCI Express (PCIe)
 - Point-to-point interconnect scheme intended to replace bus-based schemes such as PCI
 - Key requirement is high capacity to support the needs of higher data rate I/O devices, such as Gigabit Ethernet
 - Another requirement deals with the need to support time dependent data streams



PCIe Configuration

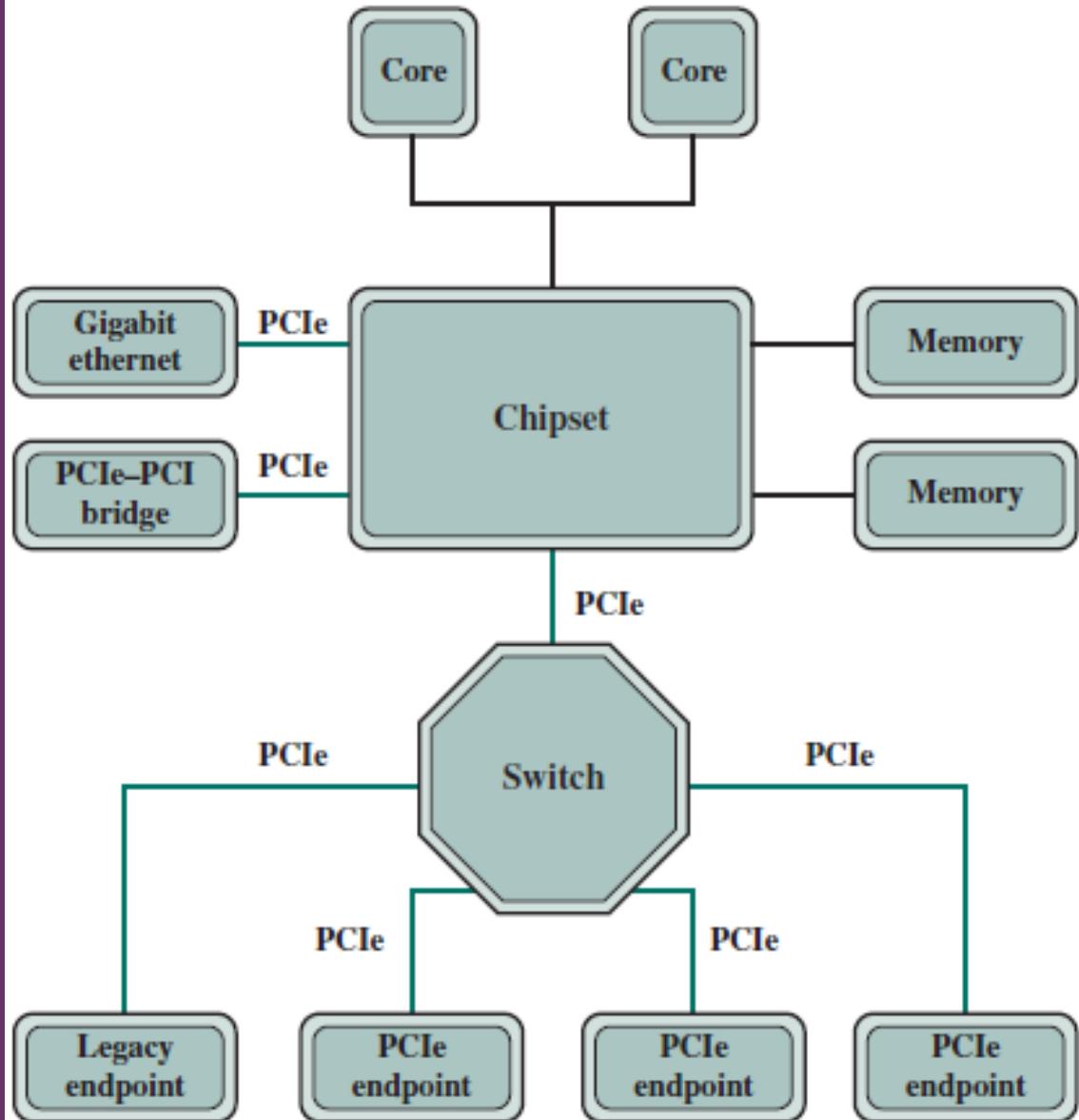


Figure 3.21 Typical Configuration Using PCIe

PCIe Protocol Layers

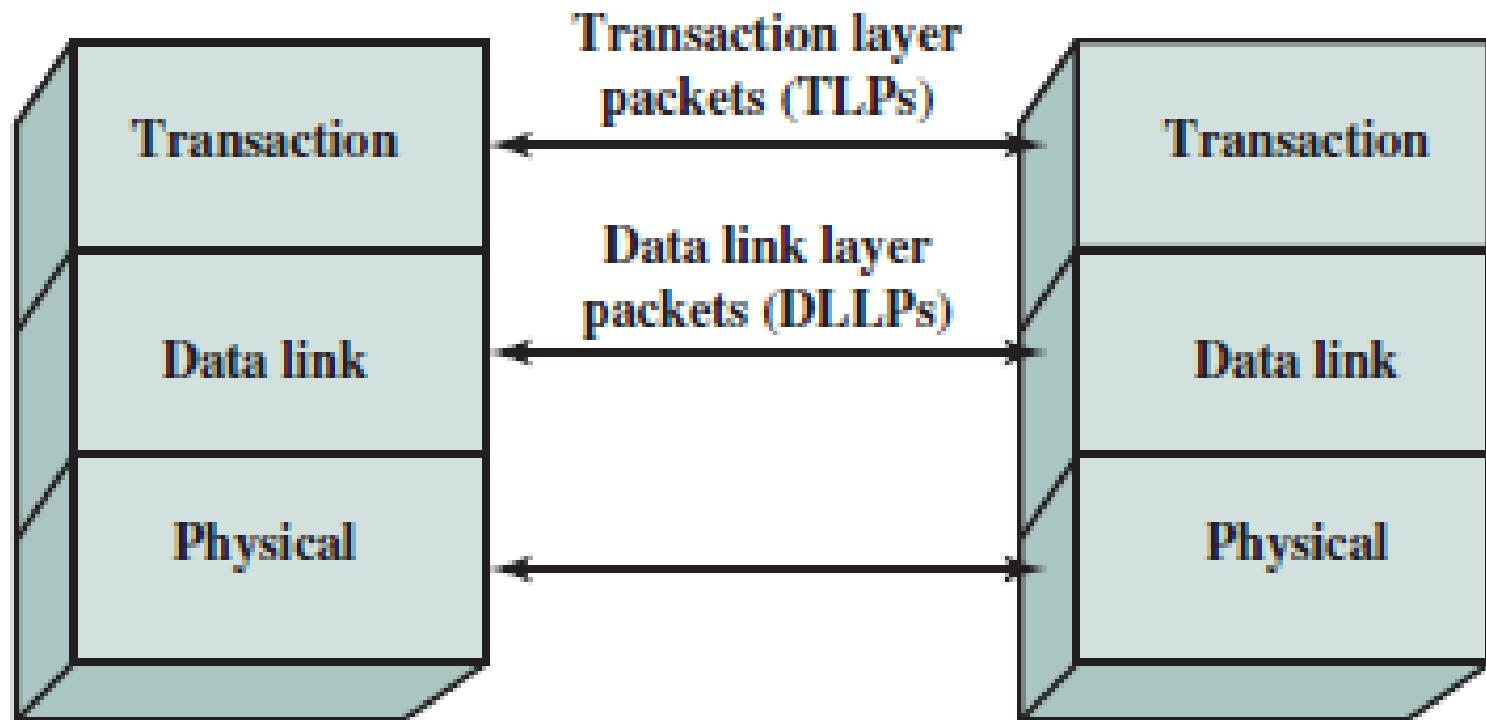
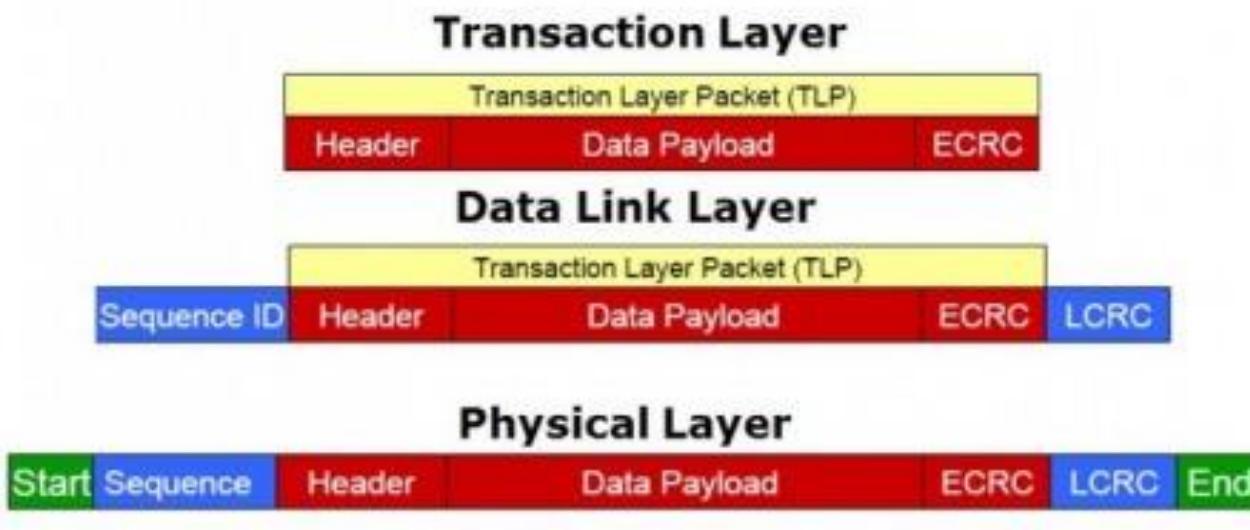


Figure 3.22 PCIe Protocol Layers

PCIe Protocol Layers



Transaction Layer : Responsible for converting requests or completion data from device core to a valid PCIe transaction

Data Link Layer : Integrity of transactions across the link.

Physical Layer : Actually transmit and receive transactions across a PCIe link. On power-on the physical layer initializes the number of layers to be used , link speed etc. The TL and DLL are oblivious to how the data is transmitted, it is taken care by the PL.



PCIe

Transaction Layer (TL)

- Receives read and write requests from the software above the TL and creates request packets for transmission to a destination via the link layer
- Most transactions use a *split transaction* technique
 - A request packet is sent out by a source PCIe device which then waits for a response called a *completion packet*
 - TL messages and some write transactions are posted transactions (meaning that no response is expected)
 - TL packet format supports 32-bit memory addressing and extended 64-bit memory addressing

Transaction Layer –Overview

There are major types of transaction layer packets :

1. **Memory** - Transaction to and from a memory mapped location. They can use 32 bit or 64 bit addressing.
2. **I/O** - Transactions targeting to and from a I/O location. PCIe mainly supports this for backward compatibility of address space. Uses only 32 bit addressing.
3. **Configuration** - Transactions targeting the configuration space targeted for device config and setup during enumeration
4. **Message (new type specific to PCIe)** - This is a new class of transaction in PCIe. Since there are no side band signals as in PCI. The interrupts, error and power management signals are mapped and transmitted as message transactions.

PCIe TLP Transaction Types

Address Space	TLP Type	Purpose
Memory	Memory Read Request	Transfer data to or from a location in the system memory map.
	Memory Read Lock Request	
	Memory Write Request	
I/O	I/O Read Request	Transfer data to or from a location in the system memory map for legacy devices.
	I/O Write Request	
Configuration	Config Type 0 Read Request	Transfer data to or from a location in the configuration space of a PCIe device.
	Config Type 0 Write Request	
	Config Type 1 Read Request	
	Config Type 1 Write Request	
Message	Message Request	Provides in-band messaging and event reporting.
	Message Request with Data	
Memory, I/O, Configuration	Completion	Returned for certain requests.
	Completion with Data	
	Completion Locked	
	Completion Locked with Data	

Data Link Layer - overview

The data link layer assigns a 12 bit sequence number to each TLP as it is passed from the transmit to receive side. Sequence number checks is on a per link basis i.e., Sequence number can be different across links

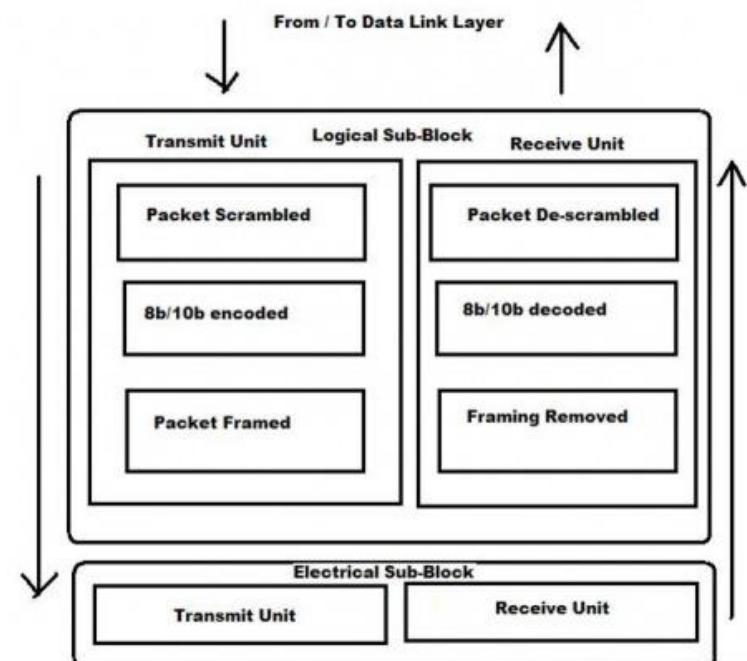
1. Data Link Layer is the gatekeeper for each individual link in the PCI Express system
2. Ensures data sent across the link is correct and received in the same order it was sent
3. Link Management functions are accomplished with the DLLPs (Data Link Layer Packets) which are used by the DLL for error notification, power management, flow control etc.

Data Link Layer - overview

Services of a DLL can be broadly classified into :

- ✓ Data Exchange
- ✓ Error Detection and Retry
- ✓ TLP Sequence number and LCRC generation
- ✓ Initialization and Power Management

Physical Layer- overview



The two key sub-blocks of the physical layer are : Logical sub-block and electrical sub-block.

Logical Sub-block – Decision maker for the physical layer.

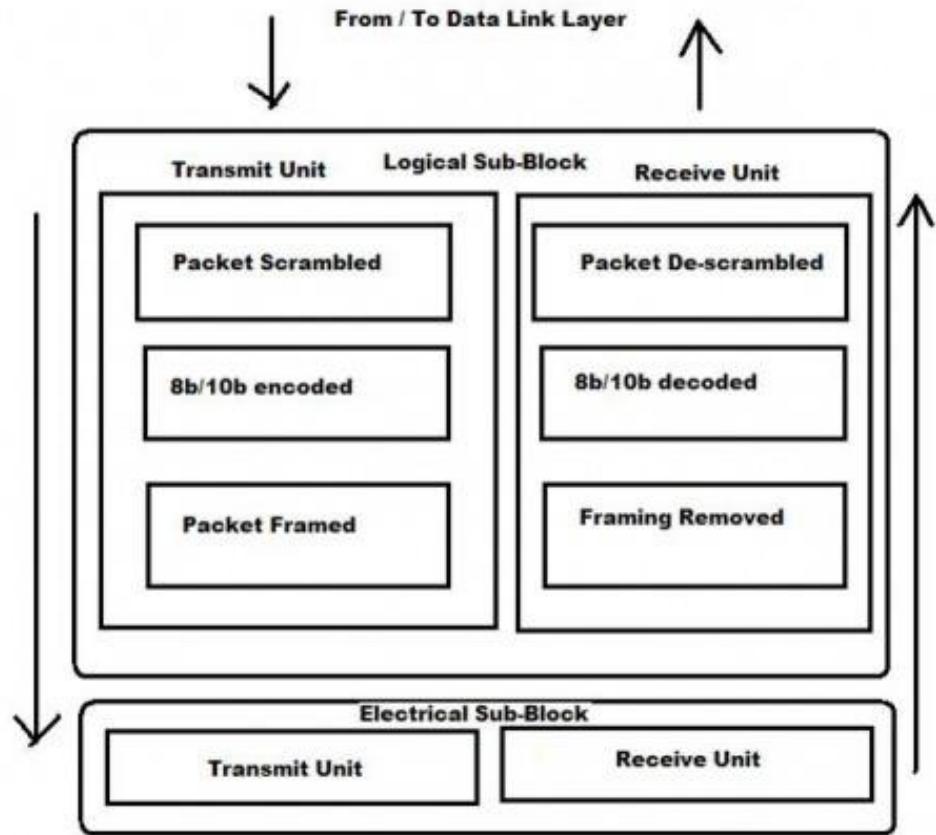
The logical sub-block has a transmit and receive unit.

Transmit Unit has three primary stages of operation :

Data scrambling

128b/130b encoding and
packet framing

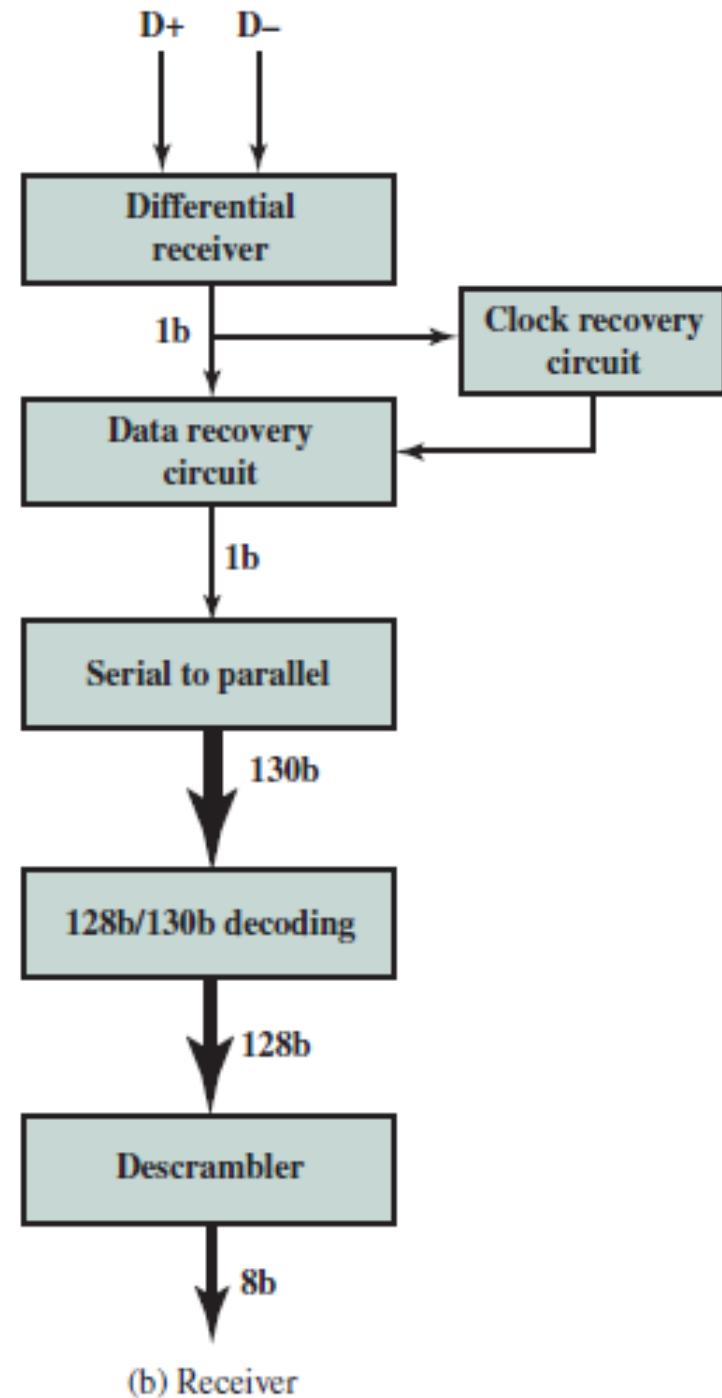
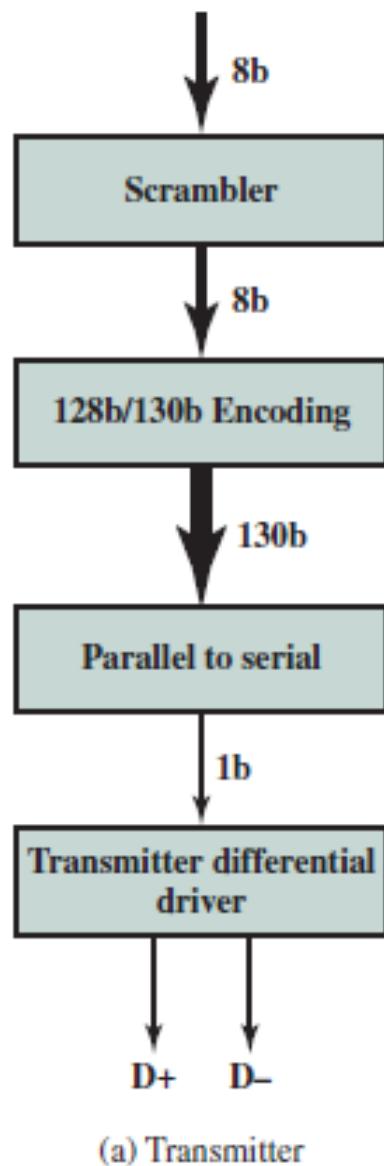
Physical Layer- overview



Receive unit takes the de-serialized physical packet from link, removes framing, decodes and descrambles the packet

The electrical sub-block functions as the delivery mechanism for the physical link. Its primary functions include: Serial- parallel conversion, Clock extraction and Lane-to-Lane de-skew.

PCIe Transmit and Receive Block Diagrams



PCIe Multilane Distribution

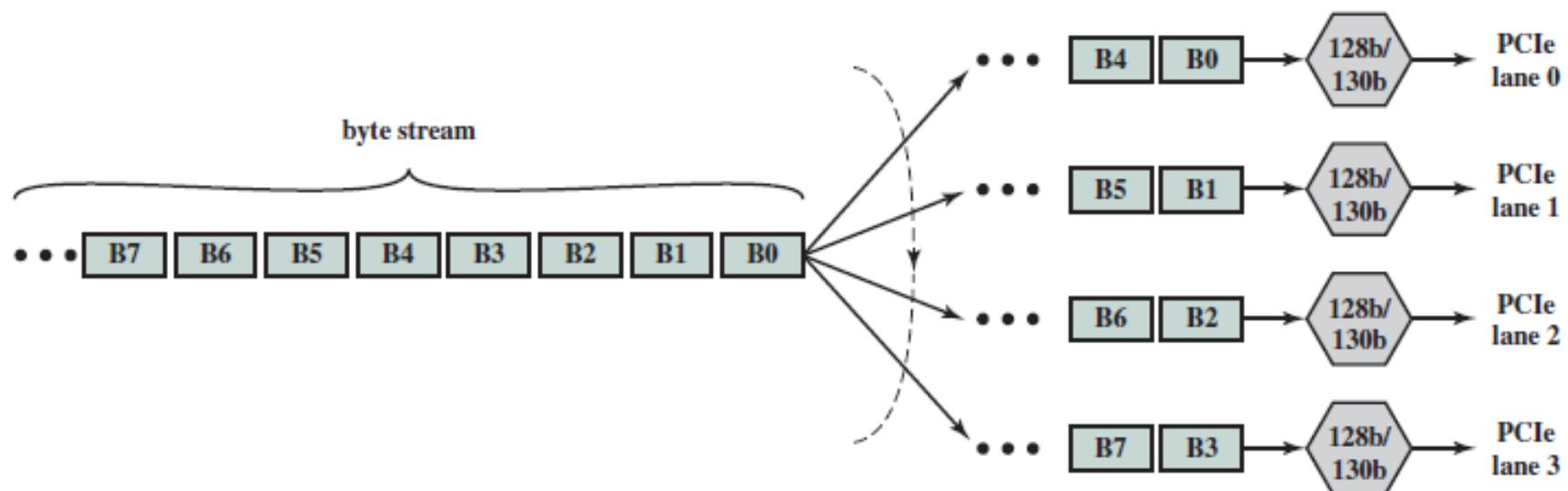


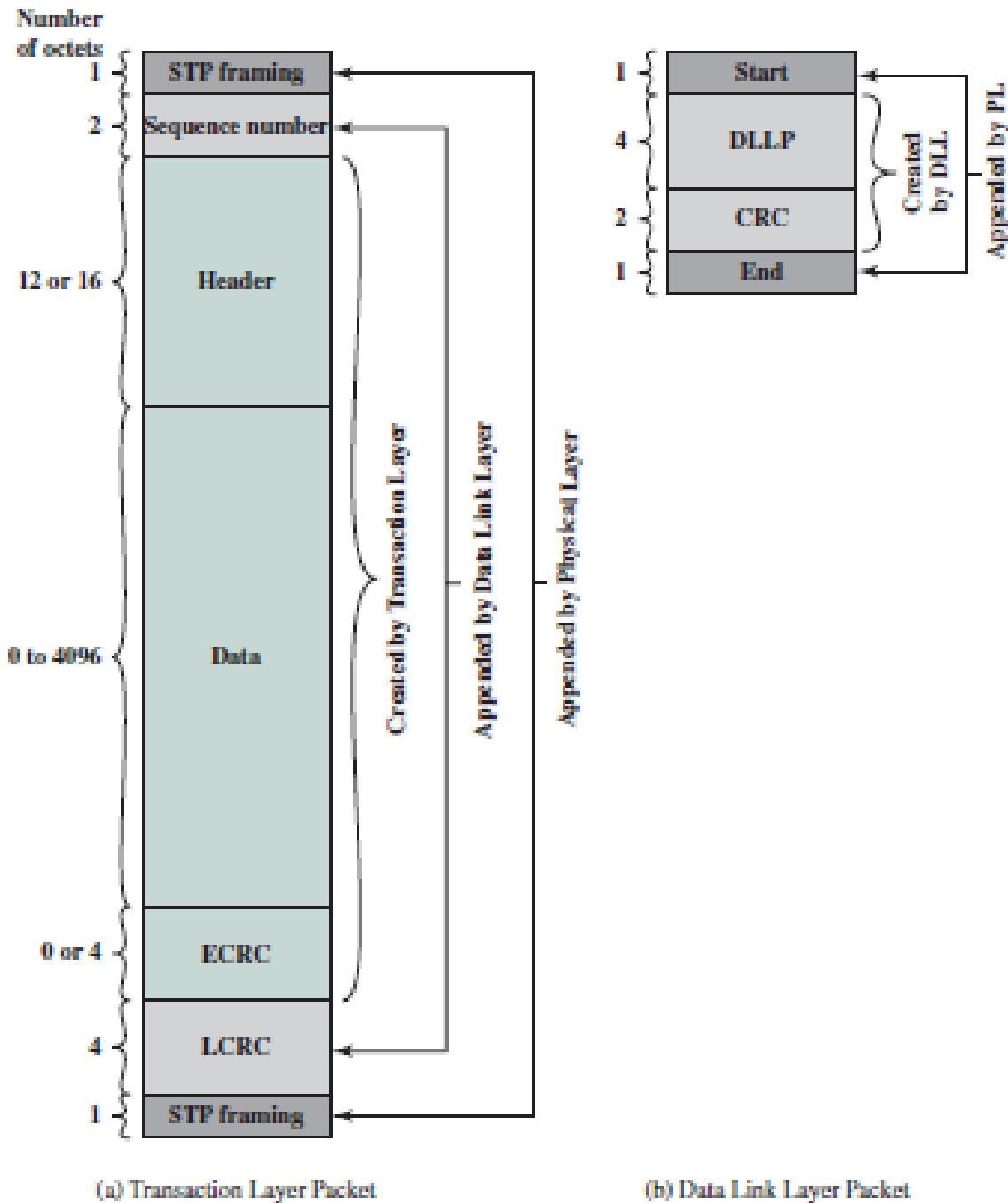
Figure 3.23 PCIe Multilane Distribution



The TL supports four address spaces:

- Memory
 - The memory space includes system main memory and PCIe I/O devices
 - Certain ranges of memory addresses map into I/O devices
- Configuration
 - This address space enables the TL to read/write configuration registers associated with I/O devices
- I/O
 - This address space is used for legacy PCI devices, with reserved address ranges used to address legacy I/O devices
- Message
 - This address space is for control signals related to interrupts, error handling, and power management

PCIe Protocol Data Unit Format



Thank You :)

CSE 213

Computer Architecture

Lecture 4: Cache Memory

Military Institute of Science and Technology

Outline

- Characteristics of Memory Systems
- The Memory Hierarchy
- Cache Memory Principles
- Cache Memory Design
- Pentium 4 Cache Organization

Key Characteristics of Computer Memory Systems

Location	Internal (e.g., processor registers, cache, main memory) External (e.g., optical disks, magnetic disks, tapes)	Performance	Access time Cycle time Transfer rate
Capacity	Number of words Number of bytes	Physical Type	Semiconductor Magnetic Optical Magneto-optical
Unit of Transfer	Word Block	Physical Characteristics	Volatile/nonvolatile Erasable/nonerasable
Access Method	Sequential Direct Random Associative	Organization	Memory modules

Table 4.1 Key Characteristics of Computer Memory Systems

Characteristics of Memory Systems

■ Location

- Refers to whether memory is internal and external to the computer
- Internal memory is often equated with main memory
- Processor requires its own local memory, in the form of registers
- Cache is another form of internal memory
- External memory consists of peripheral storage devices that are accessible to the processor via I/O controllers

■ Capacity

- Memory is typically expressed in terms of bytes

■ Unit of transfer

- For internal memory the unit of transfer is equal to the number of electrical lines into and out of the memory module

Method of Accessing Units of Data

Sequential access

Memory is organized into units of data called records

Access must be made in a specific linear sequence

Access time is variable

Example: Tape units

Direct access

Involves a shared read-write mechanism

Individual blocks or records have a unique address based on physical location

Access time is variable

Example: Disk units

Random access

Each addressable location in memory has a unique, physically wired-in addressing mechanism

The time to access a given location is independent of the sequence of prior accesses and is constant

Any location can be selected at random and directly addressed and accessed

Example: Main memory and some cache systems

Associative

A word is retrieved based on a portion of its contents rather than its address

Each location has its own addressing mechanism and retrieval time is constant independent of location or prior access patterns

Cache memories may employ associative access

The two most important characteristics of memory – Capacity & Performance

Three performance parameters are used:

Access time (latency)

- For random-access memory it is the time it takes to perform a read or write operation
- For non-random-access memory it is the time it takes to position the read-write mechanism at the desired location

Memory cycle time

- Access time plus any additional time required before second access can commence
- Additional time may be required for transients to die out on signal lines or to regenerate data if they are read destructively
- Concerned with the system bus, not the processor

Transfer rate

- The rate at which data can be transferred into or out of a memory unit
- For random-access memory it is equal to $1/(cycle\ time)$



Physical Type



- The most common forms are:
 - Semiconductor memory
 - Magnetic surface memory
 - Optical
 - Magneto-optical

Physical characteristics

- Volatile memory
 - Information decays naturally or is lost when electrical power is switched off
- Nonvolatile memory
 - Once recorded, information remains without deterioration until deliberately changed
 - No electrical power is needed to retain information
 - **Magnetic-surface memories - are nonvolatile**
 - Semiconductor memory - may be either volatile or nonvolatile
- Nonerasable memory
 - Cannot be altered, except by destroying the storage unit
 - Semiconductor memory of this type is known as read-only memory (ROM)

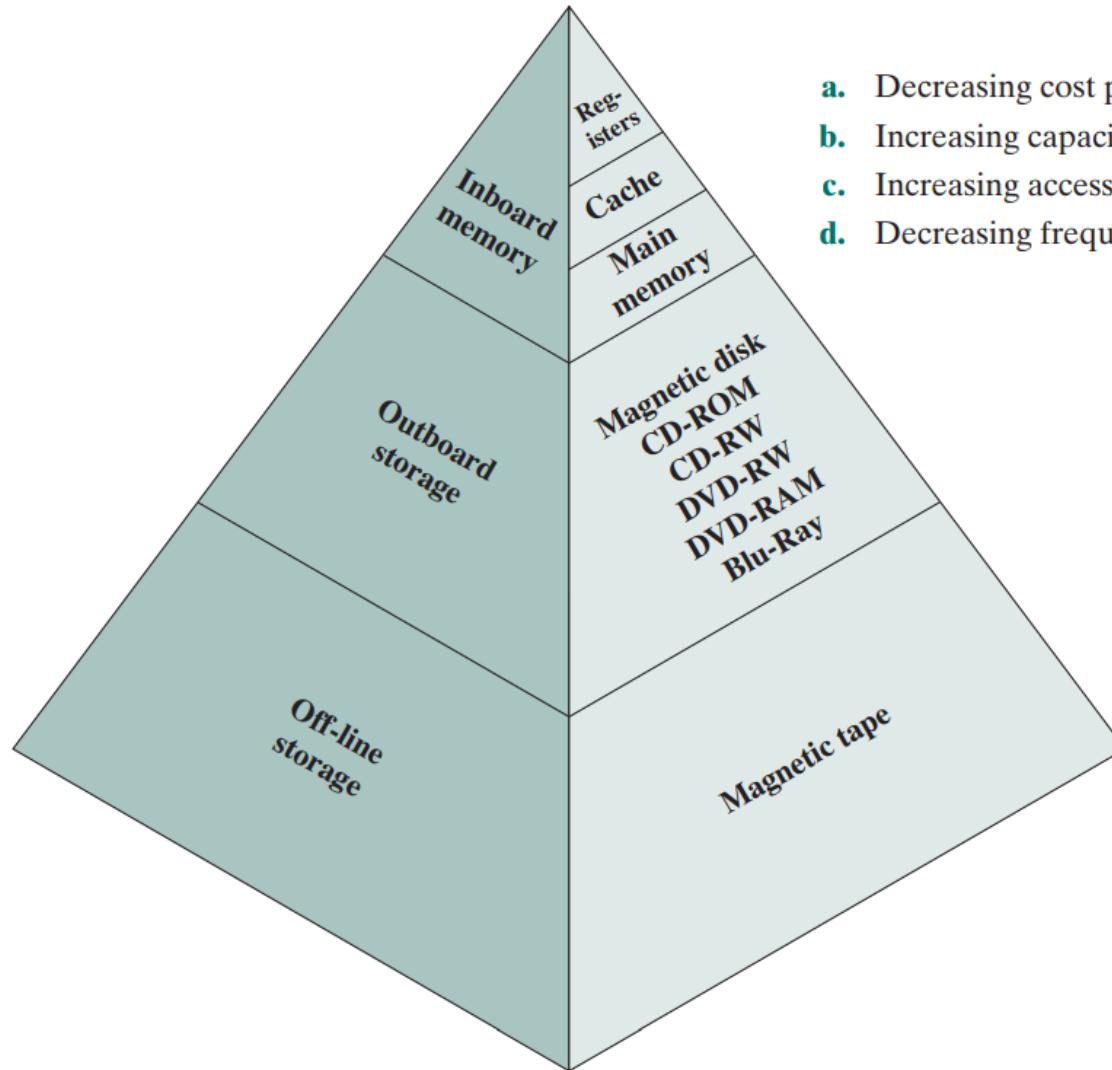
Organization: For random-access memory the organization is a key design issue. Organization refers to the physical arrangement of bits to form words



Memory Hierarchy

- Design constraints on a computer's memory can be summed up by three questions:
 - How much, how fast, how expensive
- There is a trade-off among capacity, access time, and cost
 - Faster access time, greater cost per bit
 - Greater capacity, smaller cost per bit
 - Greater capacity, slower access time
- The way out of the memory dilemma is not to rely on a single memory component or technology, but to employ a memory hierarchy

+ Memory Hierarchy - Diagram



- a. Decreasing cost per bit;
- b. Increasing capacity;
- c. Increasing access time;
- d. Decreasing frequency of access of the memory by the processor.

Figure 4.1 The Memory Hierarchy



Locality of Reference –

(d) decreasing the frequency of access

- During the course of the execution of a program, memory references tend to cluster e.g. loops
- **Locality of reference**, also known as the **principle of locality**, is a phenomenon describing the same value, or related storage locations, being frequently accessed. There are two basic types of reference locality – temporal and spatial locality.
- Temporal locality. If at one point in time a particular memory location is referenced, then it is likely that the **same location** will be referenced again in the near future.
- Spatial locality. If a particular memory location is referenced at a particular time, then it is likely that **nearby memory locations** will be referenced in the near future.



Operation of Two-Level Memory

- ❑ The locality property can be exploited in the formation of a two-level memory. The upper-level memory (M1) is smaller, faster, and more expensive (per bit) than the lower-level memory (M2).
- ❑ M1 is used as a temporary store for part of the contents of the larger M2.
- ❑ When a memory reference is made, an attempt is made to access the item in M1. If this succeeds, then a quick access is made. If not, then a block of memory locations is copied from M2 to M1 and the access then takes place via M1.
- ❑ Because of locality, once a block is brought into M1, there should be a number of accesses to locations in that block, resulting in fast overall service.



Operation of Two-Level Memory

To express the average time to access an item, we must consider not only the speeds of the two levels of memory, but also the probability that a given reference can be found in M1. We have

$$Ts = H \times T1 + (1-H) \times (T1+T2)$$

where

Ts = average (system) access time

$T1$ = access time of M1 (e.g., cache, disk cache)

$T2$ = access time of M2 (e.g., main memory, disk)

H = hit ratio (fraction of time reference is found in M1)



Example-4.1

EXAMPLE 4.1 Suppose that the processor has access to two levels of memory. Level 1 contains 1000 words and has an access time of $0.01 \mu s$; level 2 contains 100,000 words and has an access time of $0.1 \mu s$. Assume that if a word to be accessed is in level 1, then the processor accesses it directly. If it is in level 2, then the word is first transferred to level 1 and then accessed by the processor. For simplicity, we ignore the time required for the processor to determine whether the word is in level 1 or level 2. Figure 4.2 shows the general shape of the curve that covers this situation. The figure shows the average access time to a two-level memory as a function of the hit ratio H , where H is defined as the fraction of all memory accesses that are found in the faster memory (e.g., the cache), T_1 is the access time to level 1, and T_2 is the access time to level 2.¹ As can be seen, for high percentages of level 1 access, the average total access time is much closer to that of level 1 than that of level 2.

In our example, suppose 95% of the memory accesses are found in level 1. Then the average time to access a word can be expressed as

$$(0.95)(0.01 \mu s) + (0.05)(0.01 \mu s + 0.1 \mu s) = 0.0095 + 0.0055 = 0.015 \mu s$$

The average access time is much closer to $0.01 \mu s$ than to $0.1 \mu s$, as desired.



CACHE MEMORY PRINCIPLES



Cache

A technique, sometimes referred to as a **disk cache**, improves performance in two ways:

- Disk writes are clustered. Instead of many small transfers of data, we have a few large transfers of data. This improves disk performance and minimizes processor involvement.
- Some data destined for write-out may be referenced by a program before the next dump to disk. In that case, the data are retrieved rapidly from the Cache rather than slowly from the disk.

+

Cache

- Small amount of fast memory
- Sits between normal main memory and CPU
- May be located on CPU chip or module

Cache and Main Memory

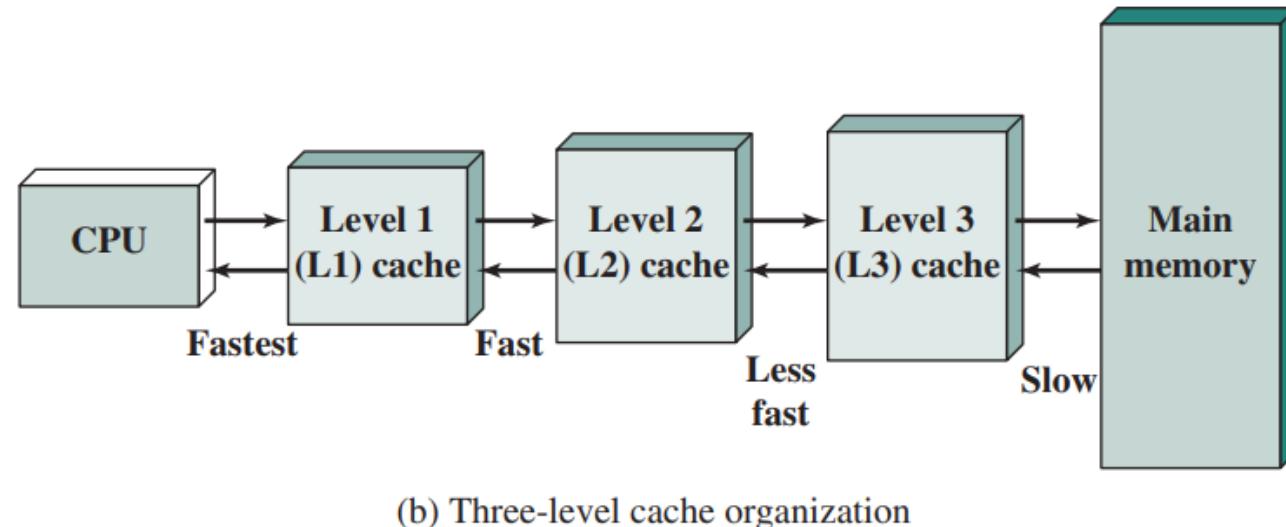
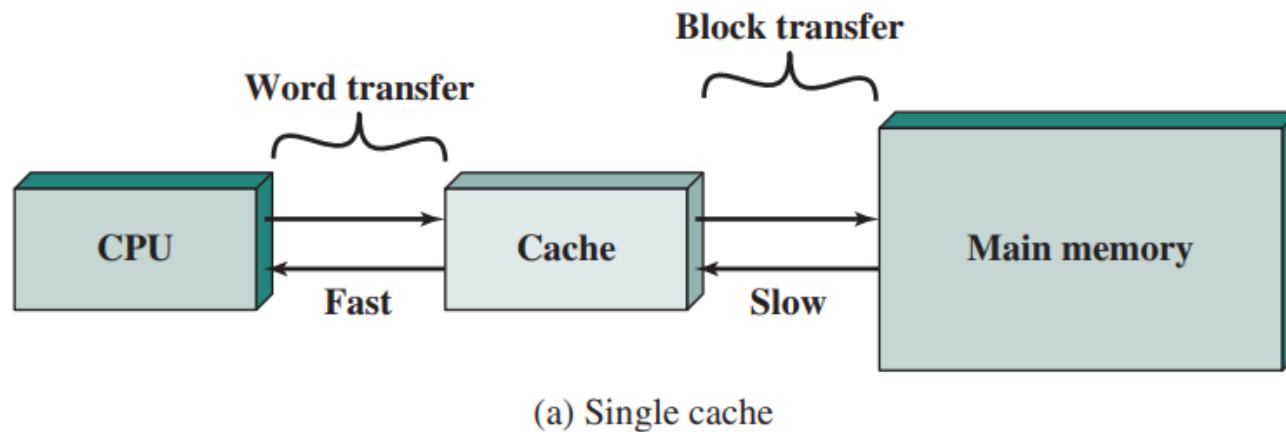


Figure 4.3 Cache and Main Memory

Cache/Main Memory Structure

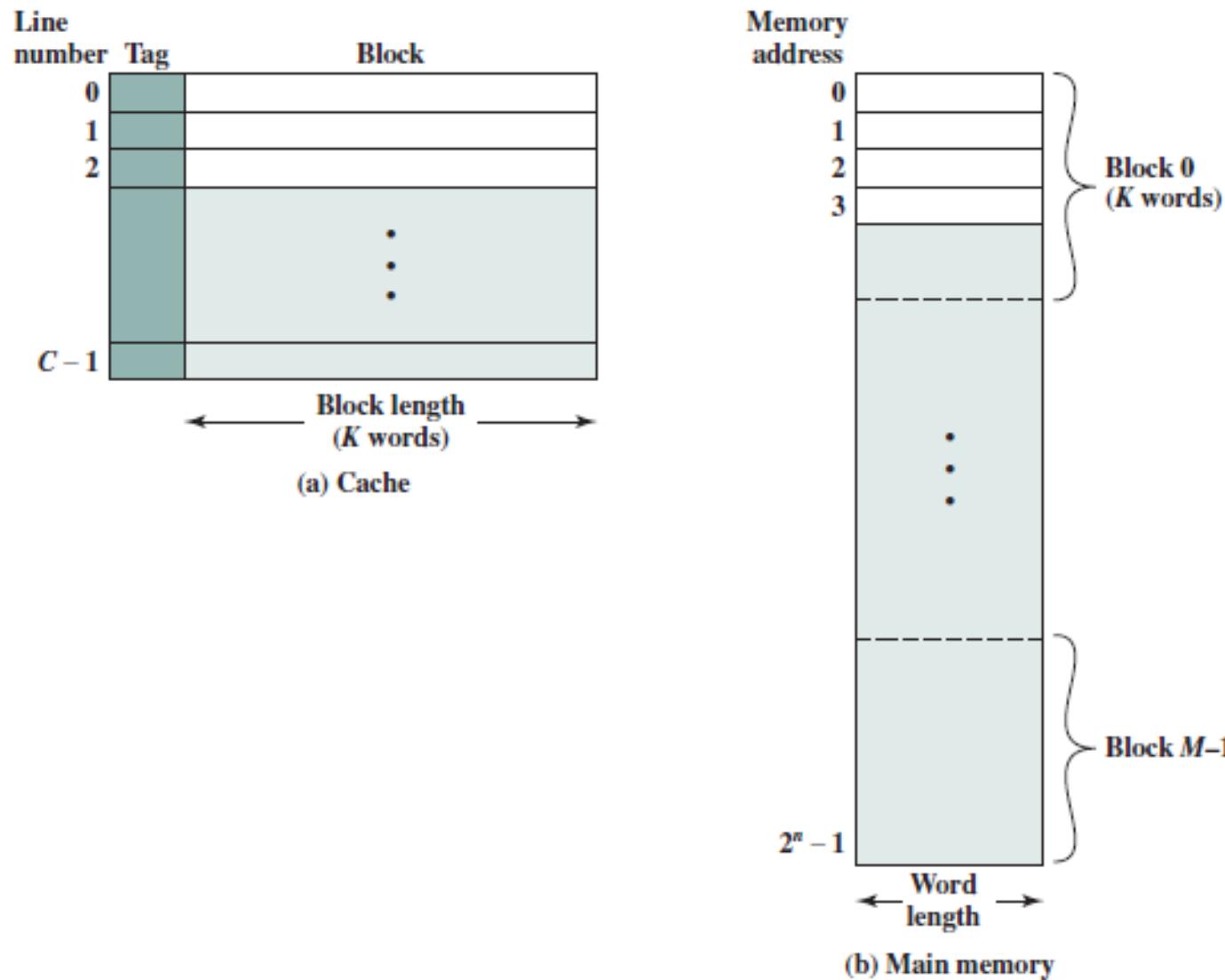


Figure 4.4 Cache/Main Memory Structure

Cache Read Operation

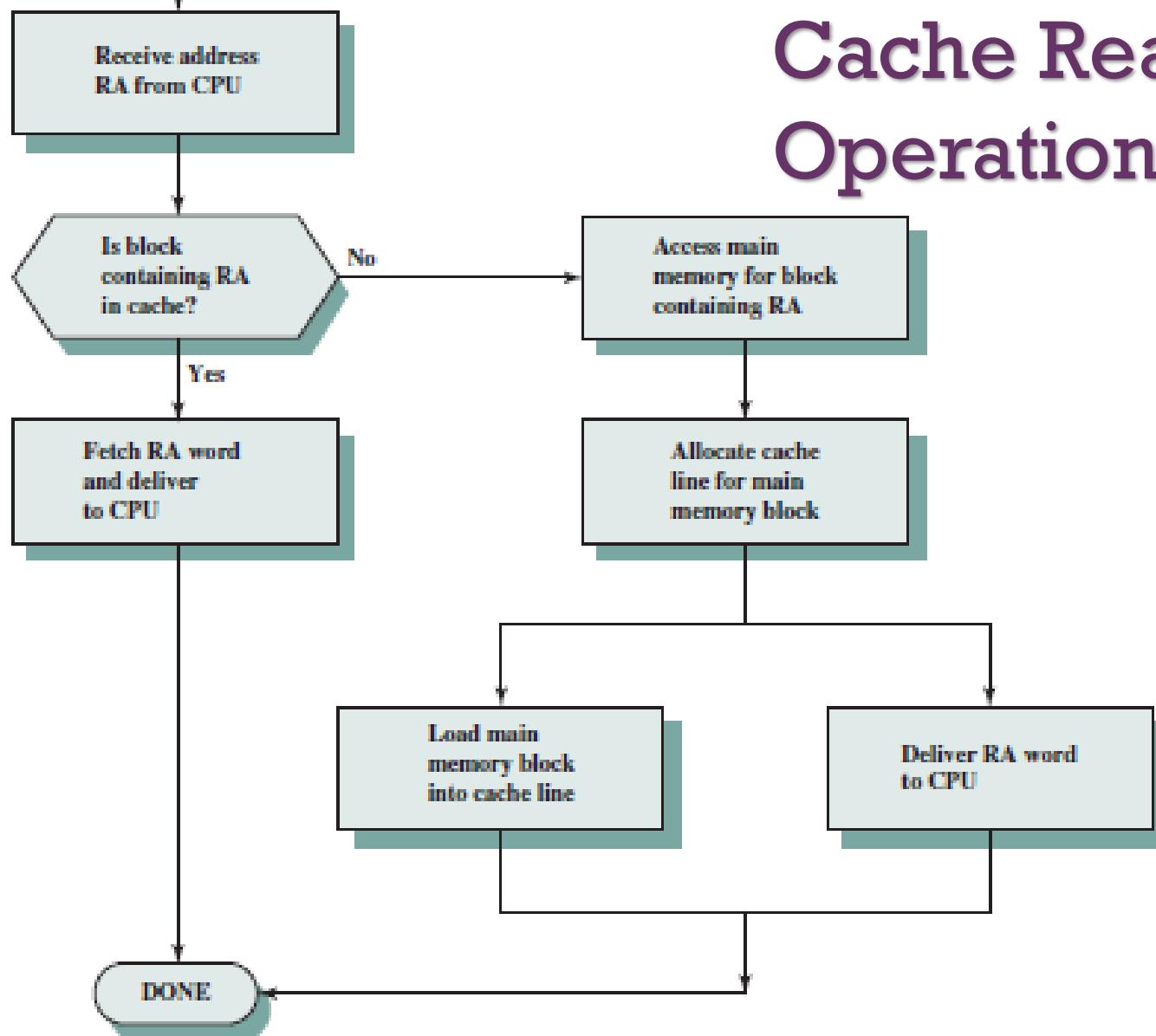


Figure 4.5 Cache Read Operation



Typical Cache Organization

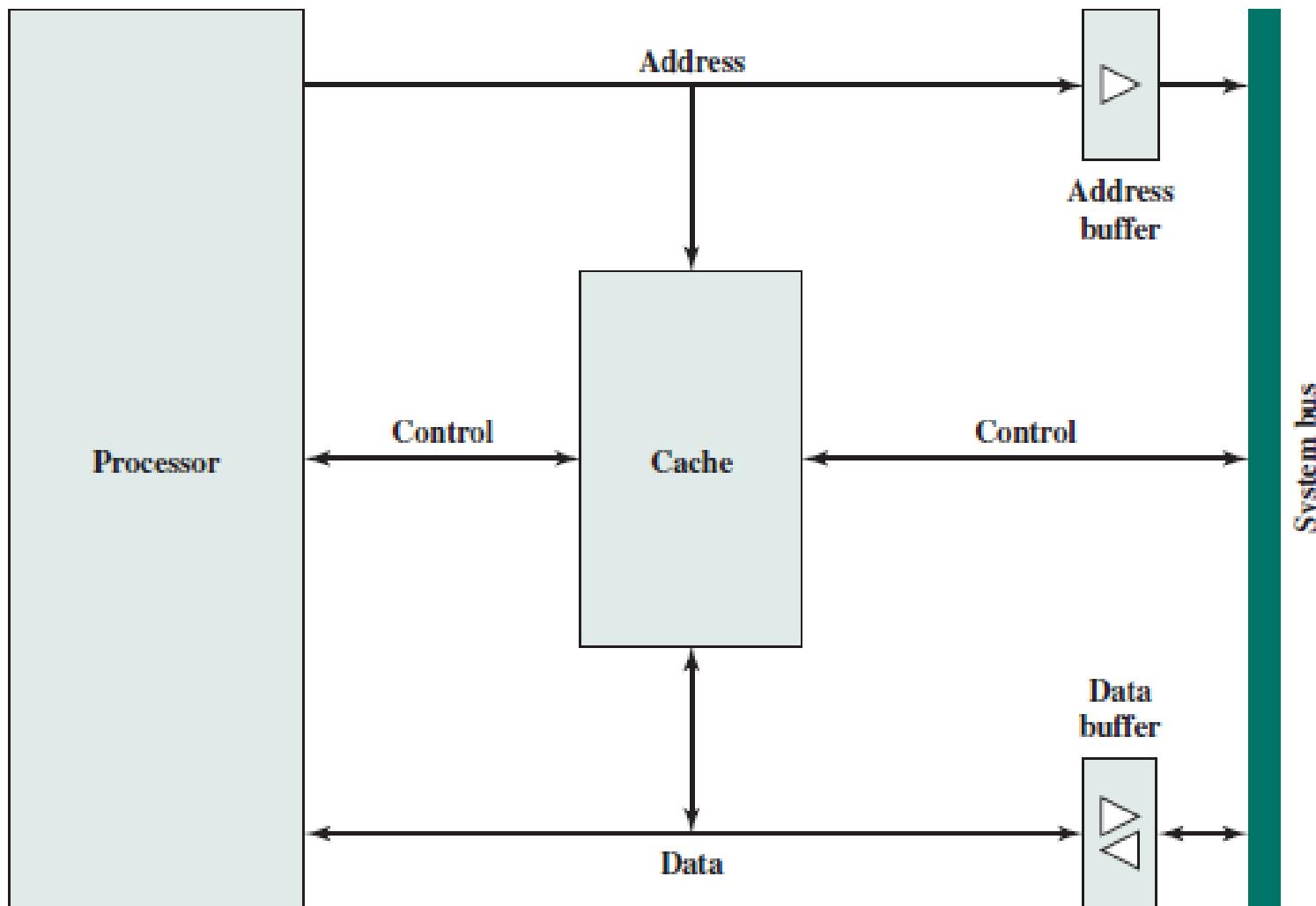


Figure 4.6 Typical Cache Organization

ELEMENTS OF CACHE DESIGN

Table 4.2 Elements of Cache Design

Cache Addresses	Write Policy
Logical	Write through
Physical	Write back
Cache Size	Line Size
Mapping Function	Number of Caches
Direct	Single or two level
Associative	Unified or split
Set associative	
Replacement Algorithm	
Least recently used (LRU)	
First in first out (FIFO)	
Least frequently used (LFU)	
Random	

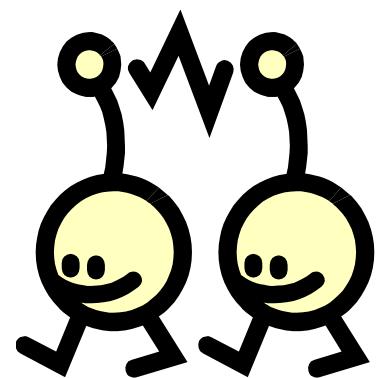


Cache Addresses

Virtual Memory

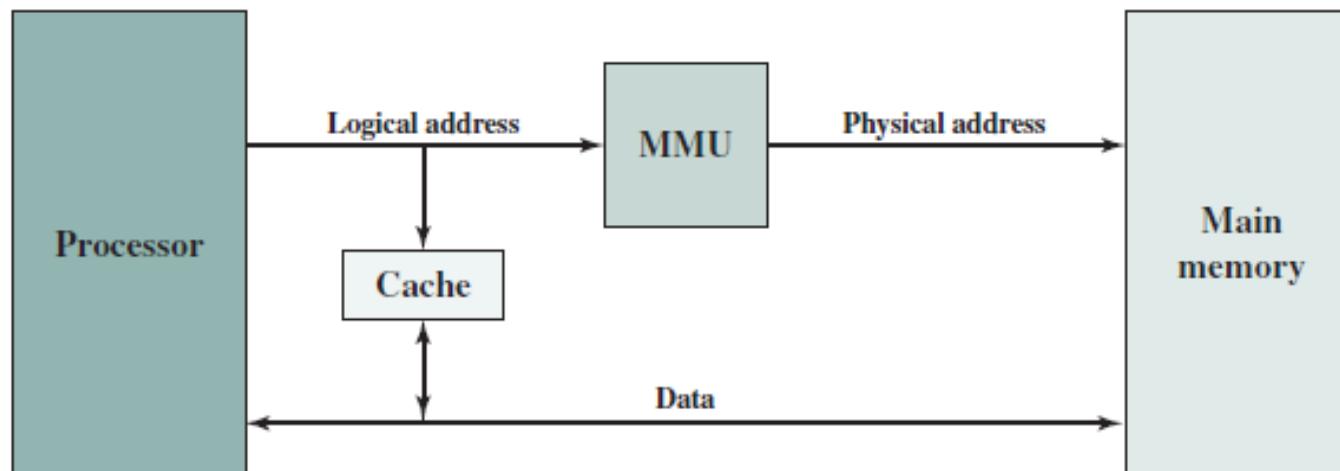
■ Virtual memory

- Facility that allows programs to address memory from a logical point of view, without regard to the amount of main memory physically available
- When used, the address fields of machine instructions contain virtual addresses
- For reads to and writes from main memory, a hardware memory management unit (MMU) translates each virtual address into a physical address in main memory

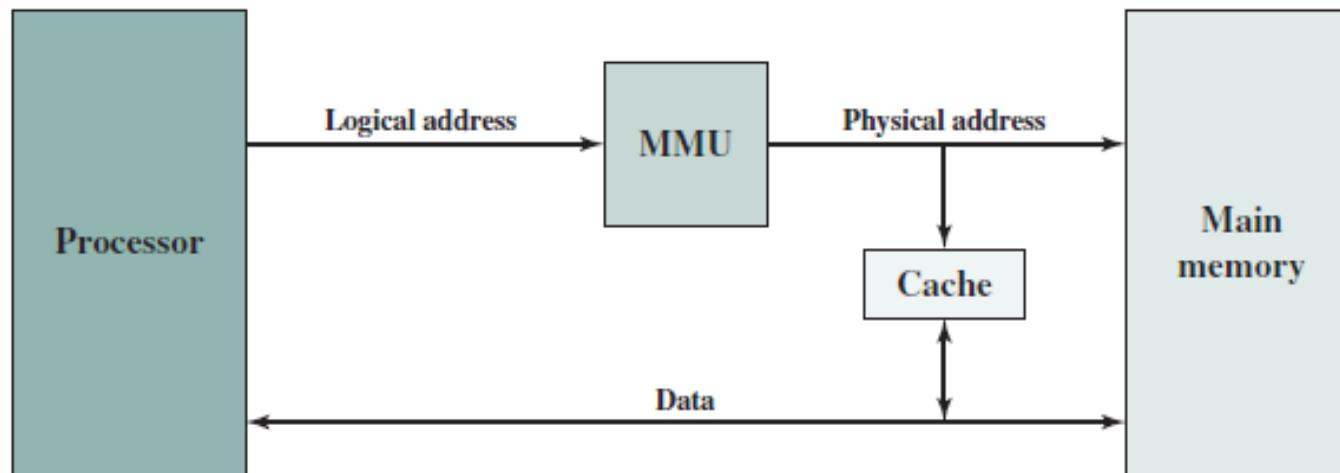




Logical and Physical Caches



(a) Logical cache



(b) Physical cache

Table 4.3
Cache Sizes of Some Processors

Processor	Type	Year of Introduction	L1 Cache ^a	L2 cache	L3 Cache
IBM 360/85	Mainframe	1968	16 to 32 kB	—	—
PDP-11/70	Minicomputer	1975	1 kB	—	—
VAX 11/780	Minicomputer	1978	16 kB	—	—
IBM 3033	Mainframe	1978	64 kB	—	—
IBM 3090	Mainframe	1985	128 to 256 kB	—	—
Intel 80486	PC	1989	8 kB	—	—
Pentium	PC	1993	8 kB/8 kB	256 to 512 KB	—
PowerPC 601	PC	1993	32 kB	—	—
PowerPC 620	PC	1996	32 kB/32 kB	—	—
PowerPC G4	PC/server	1999	32 kB/32 kB	256 KB to 1 MB	2 MB
IBM S/390 G6	Mainframe	1999	256 kB	8 MB	—
Pentium 4	PC/server	2000	8 kB/8 kB	256 KB	—
IBM SP	High-end server/ supercomputer	2000	64 kB/32 kB	8 MB	—
CRAY MTA _b	Supercomputer	2000	8 kB	2 MB	—
Itanium	PC/server	2001	16 kB/16 kB	96 KB	4 MB
Itanium 2	PC/server	2002	32 kB	256 KB	6 MB
IBM POWER5	High-end server	2003	64 kB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 kB/64 kB	1MB	—
IBM POWER6	PC/server	2007	64 kB/64 kB	4 MB	32 MB
IBM z10	Mainframe	2008	64 kB/128 kB	3 MB	24-48 MB
Intel Core i7 EE 990	Workstation/ server	2011	6 × 32 kB/32 kB	1.5 MB	12 MB
IBM zEnterprise 196	Mainframe/ Server	2011	24 × 64 kB/ 128 kB	24 × 1.5 MB	24 MB L3 192 MB L4

^a Two values separated by a slash refer to instruction and data caches.

^b Both caches are instruction only; no data caches.

Mapping Function

- Because there are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines
- Three techniques can be used:

Direct

- The simplest technique
- Maps each block of main memory into only one possible cache line

Associative

- Permits each main memory block to be loaded into any line of the cache
- The cache control logic interprets a memory address simply as a Tag and a Word field
- To determine whether a block is in the cache, the cache control logic must simultaneously examine every line's Tag for a match

Set Associative

- A compromise that exhibits the strengths of both the direct and associative approaches while reducing their disadvantages

Direct Mapping

DIRECT MAPPING The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line. The mapping is expressed as

$$i = j \text{ modulo } m$$

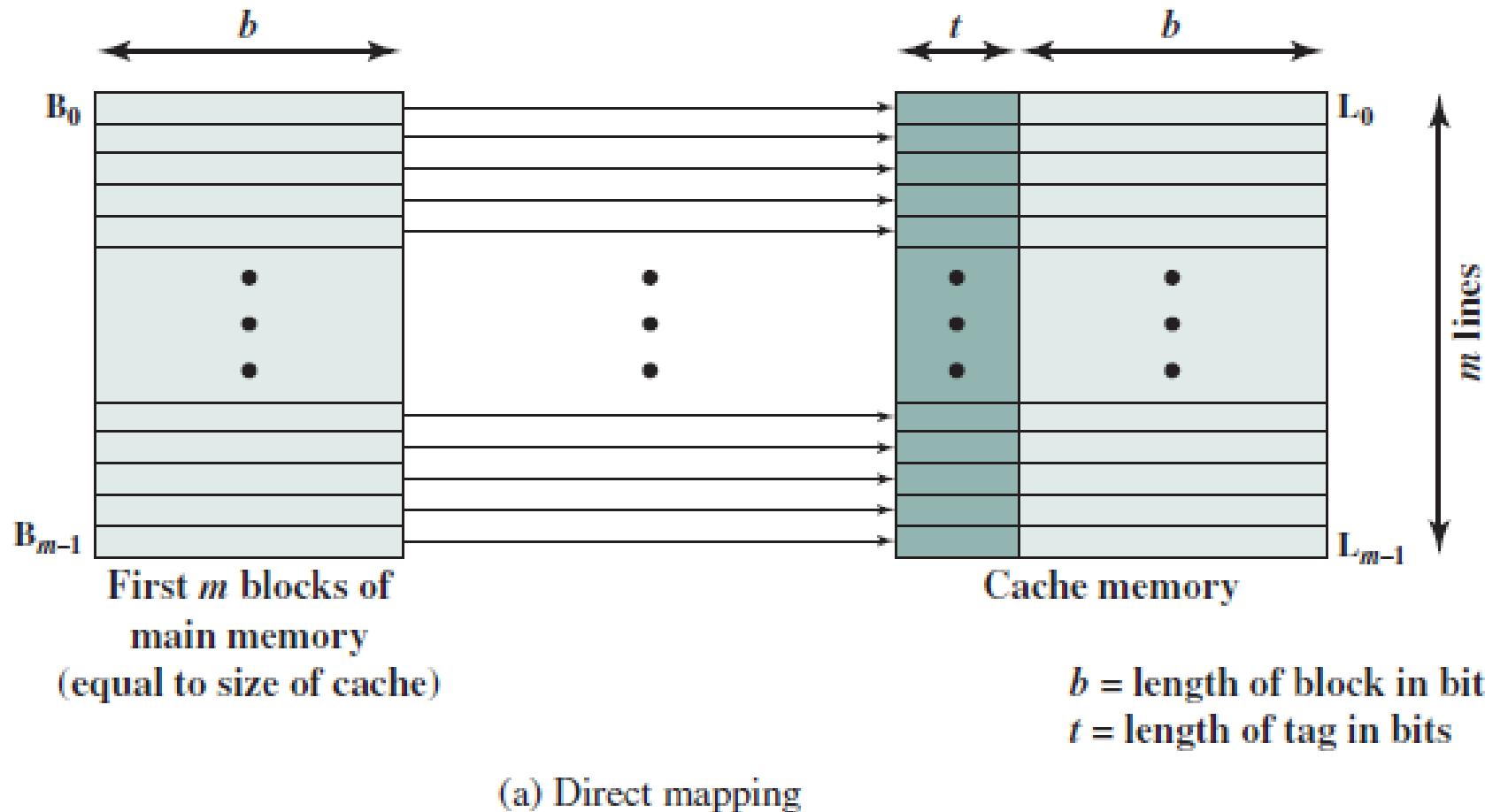
where

i = cache line number

j = main memory block number

m = number of lines in the cache

Direct Mapping



Direct Mapping Cache Organization

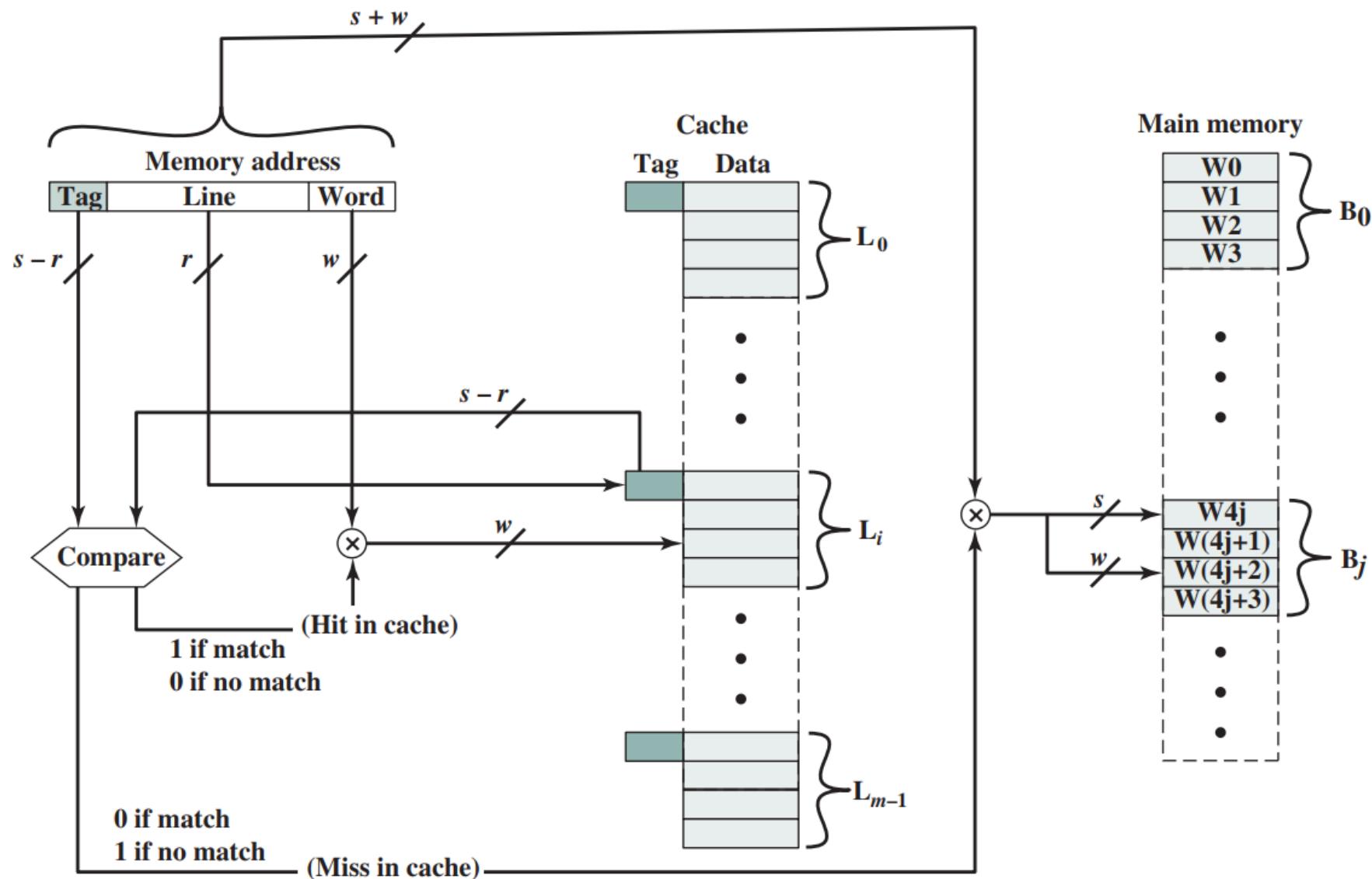


Figure 4.9 Direct-Mapping Cache Organization

Direct Mapping Cache Example

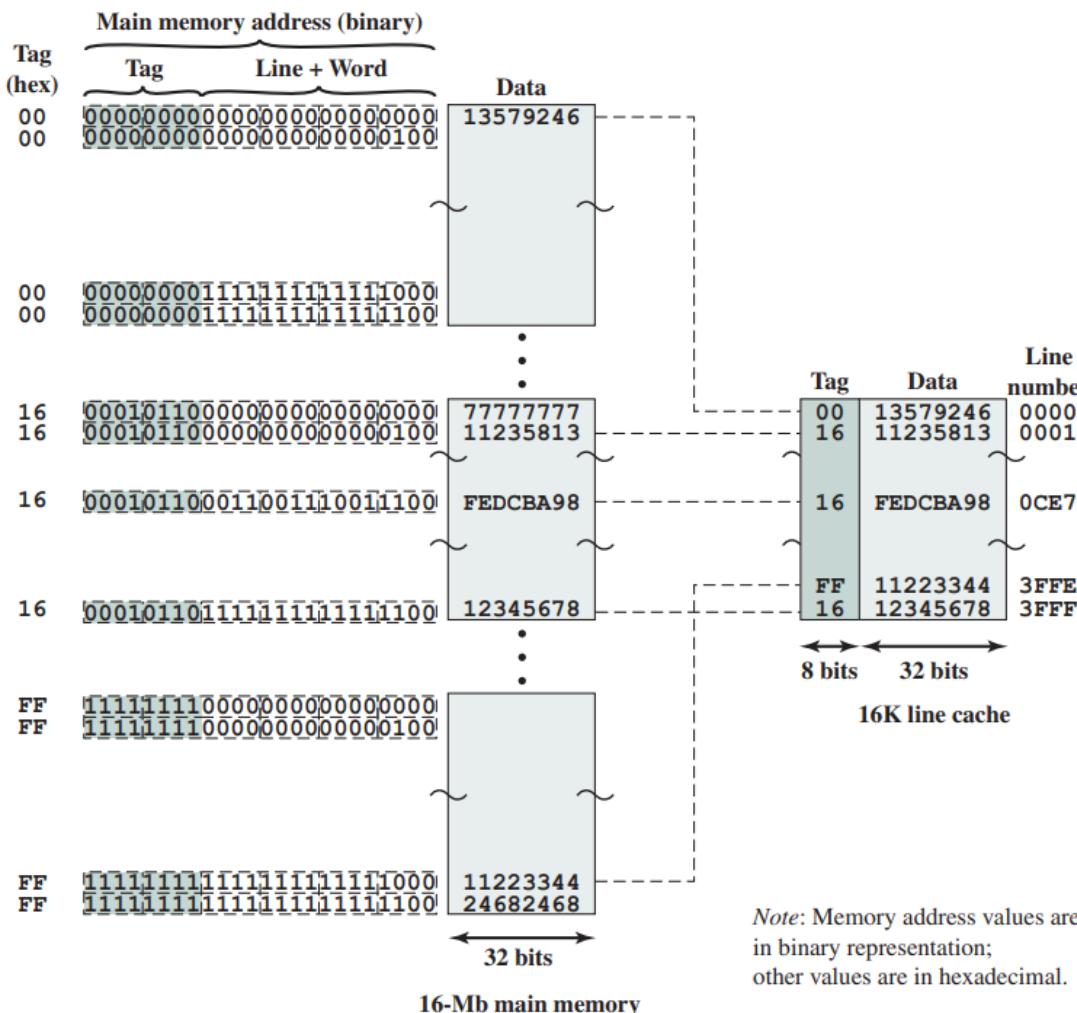
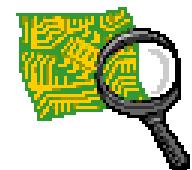


Figure 4.10 Direct Mapping Example

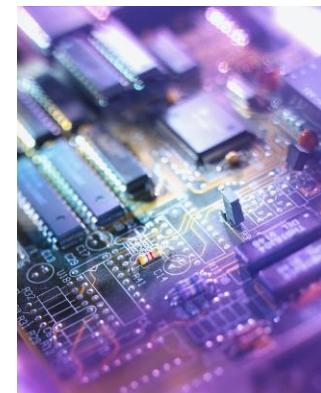


Direct Mapping Summary

- Address length = $(s + w)$ bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^w words or bytes
- Number of blocks in main memory = $2^{s+w}/2^w = 2^s$
- Number of lines in cache = $m = 2^r$
- Size of tag = $(s - r)$ bits

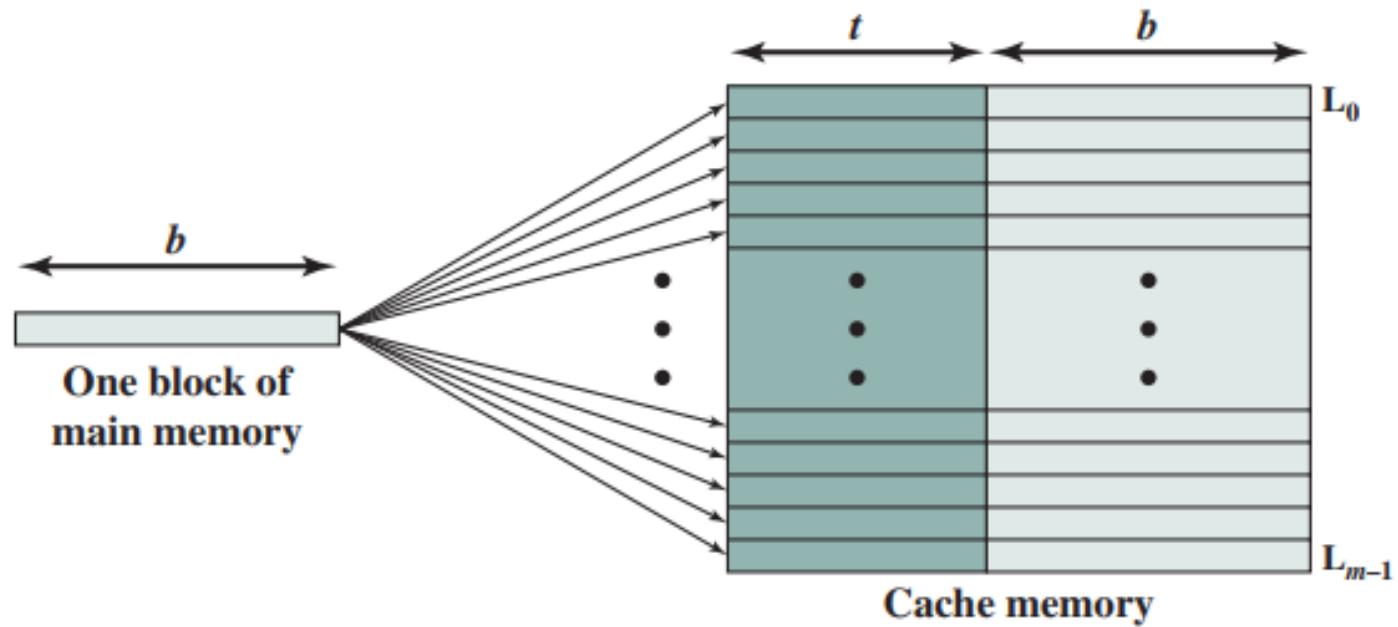


Victim Cache



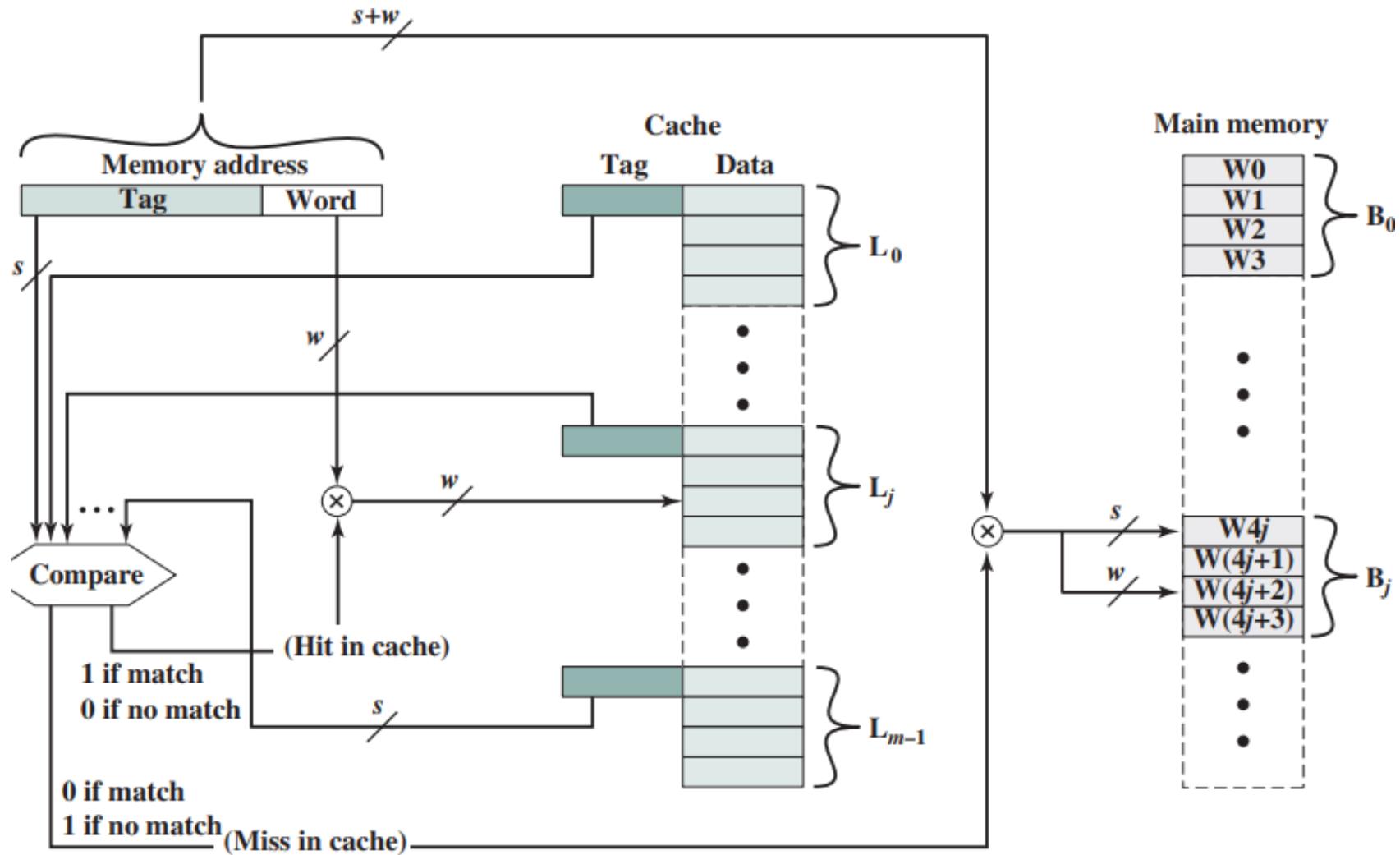
- Originally proposed as an approach to reduce the conflict misses of direct mapped caches without affecting its fast access time
- Fully associative cache
- Typical size is 4 to 16 cache lines
- Residing between direct mapped L1 cache and the next level of memory

Associative Cache Mapping



(b) Associative mapping

Fully Associative Cache Organization



Associative Mapping Example

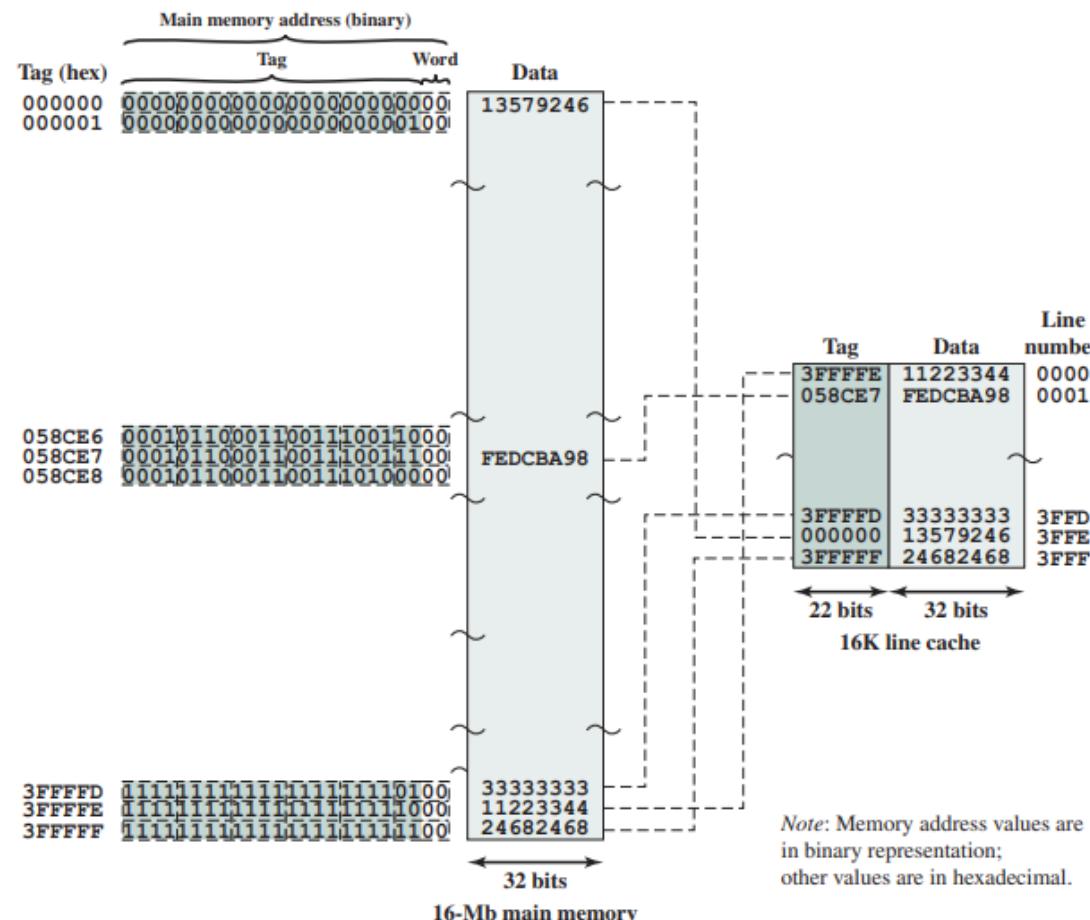
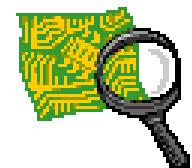


Figure 4.12 Associative Mapping Example



Associative Mapping Summary

- Address length = $(s + w)$ bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^w words or bytes
- Number of blocks in main memory = $2^{s+w}/2^w = 2^s$
- Number of lines in cache = **undetermined**
- Size of tag = s bits





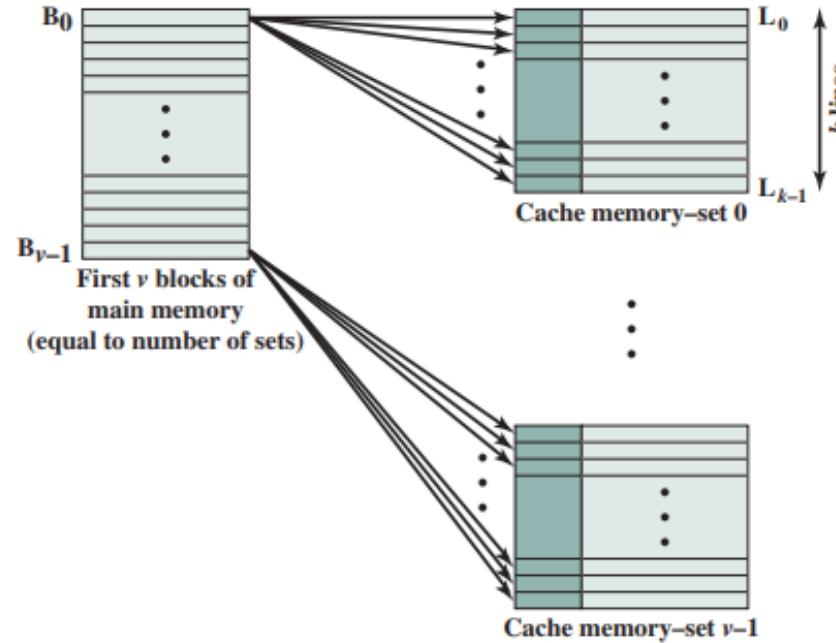
Set Associative Mapping

- Compromise that exhibits the strengths of both the direct and associative approaches while reducing their disadvantages
- Cache consists of a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
- e.g. 2 lines per set
 - 2 way associative mapping
 - A given block can be in one of 2 lines in only one set

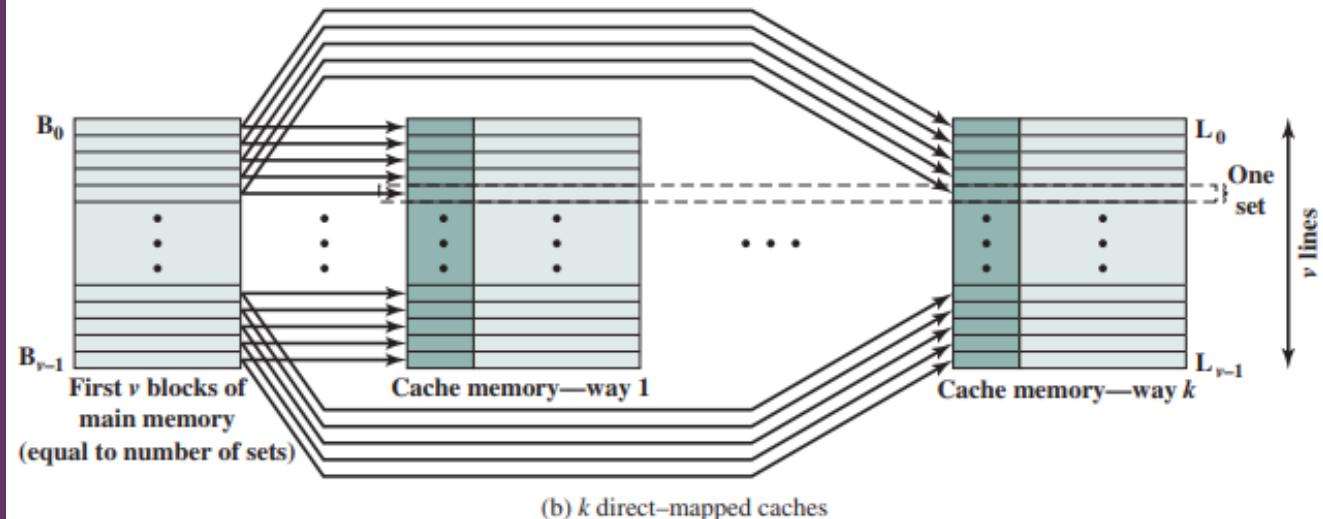


Mapping From Main Memory to Cache:

k -Way Set Associative



(a) v associative-mapped caches



(b) k direct-mapped caches

Figure 4.13 Mapping from Main Memory to Cache: k -Way Set Associative

k -Way Set Associative Cache Organization

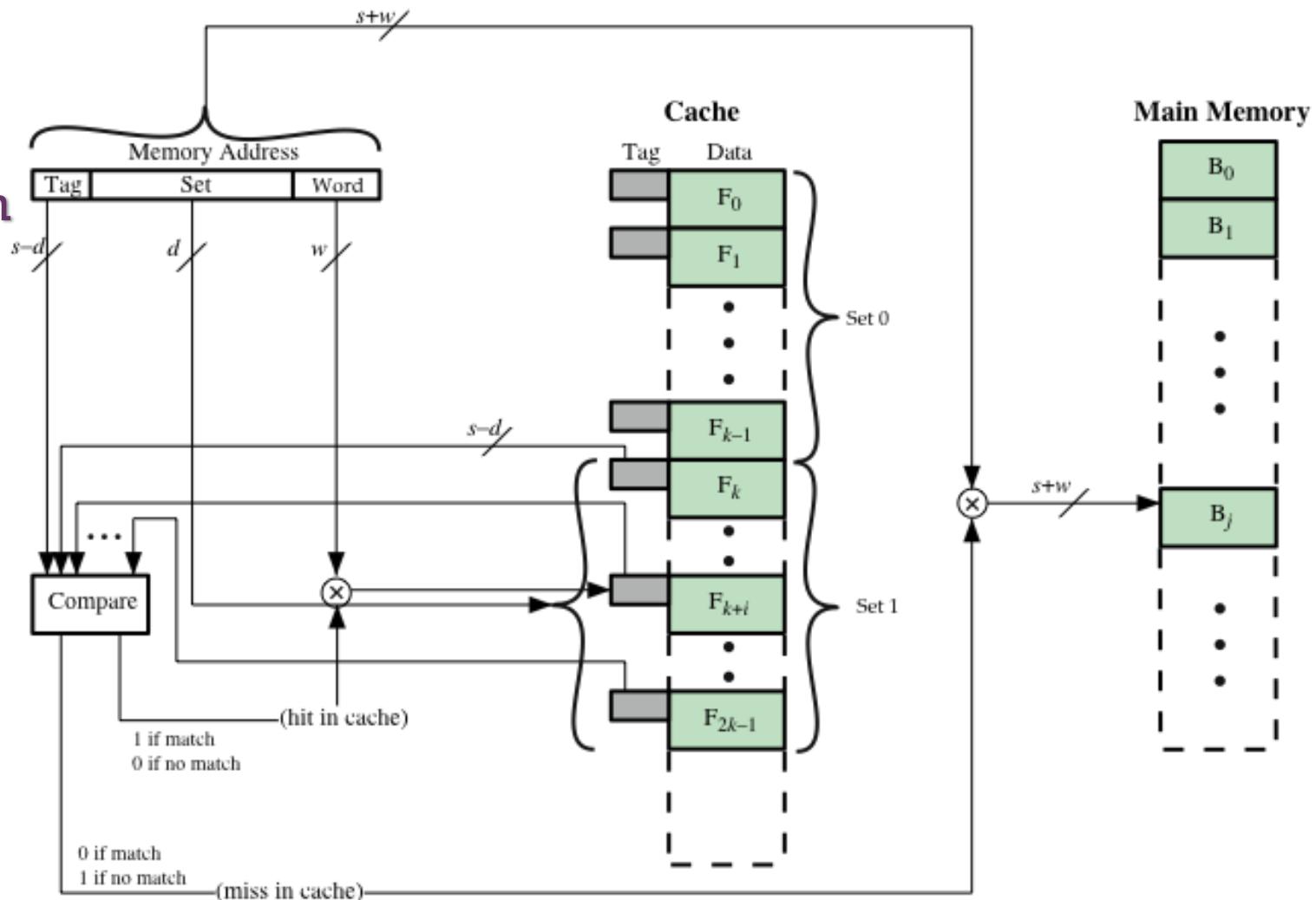
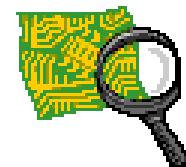


Figure 4.14 k -Way Set Associative Cache Organization



Set Associative Mapping Summary

- Address length = $(s + w)$ bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^w words or bytes
- Number of blocks in main memory = $2^{s+w}/2^w=2^s$
- Number of lines in set = k
- Number of sets = $v = 2^d$
- Number of lines in cache = $m=kv = k * 2^d$
- Size of cache = $k * 2^{d+w}$ words or bytes
- Size of tag = $(s - d)$ bits



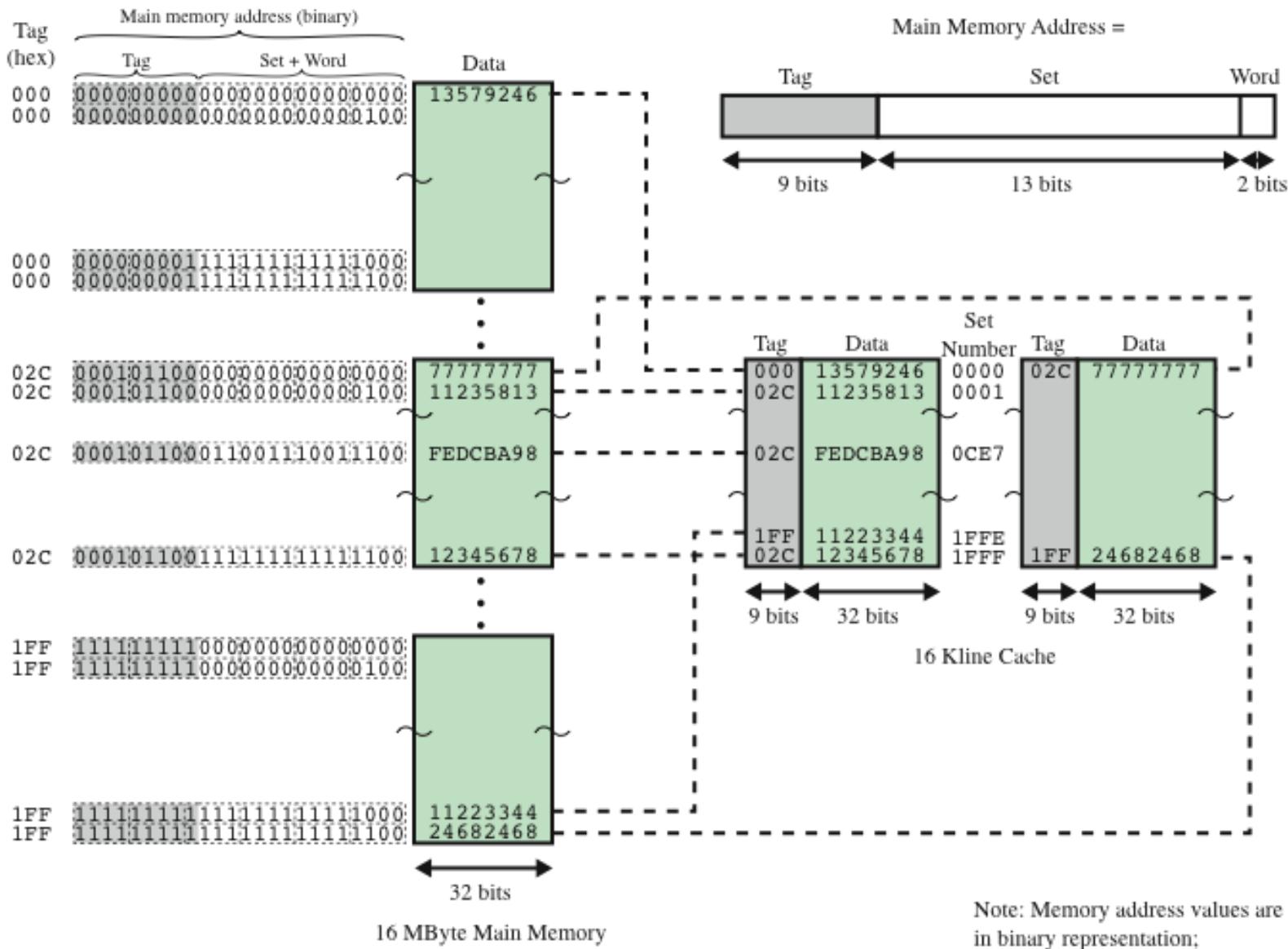


Figure 4.15 Two-Way Set Associative Mapping Example



Varying Associativity Over Cache Size

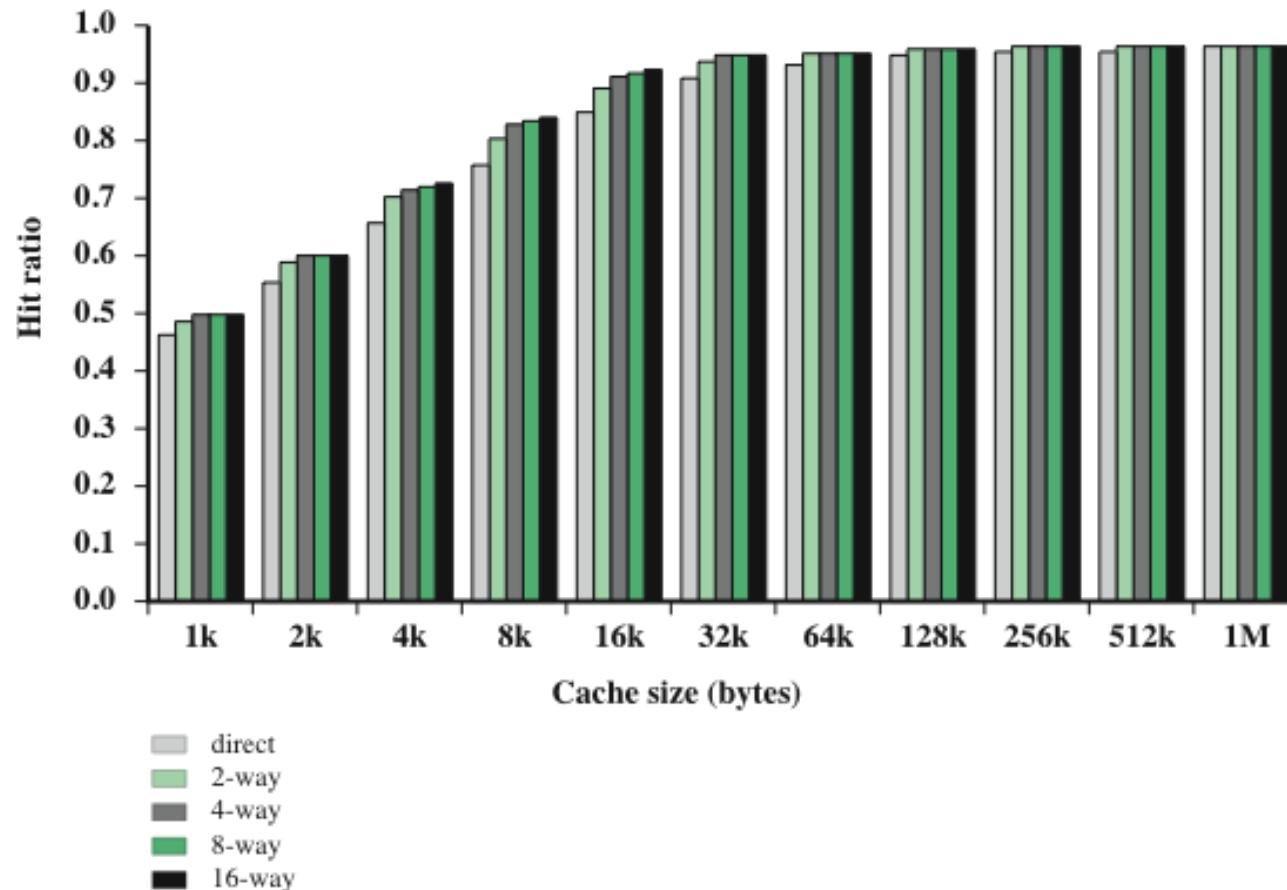


Figure 4.16 Varying Associativity over Cache Size



Problem

- Assume the size of a main memory in a computer is **1Mbytes**. The block size of the main memory is **16 Bytes**. The size of each word is 1Byte. The size of the cache memory is **64KBytes**.
- Draw the Main memory format for direct mapping, associative mapping and two way set associative mapping.
- direct mapping

4 bit tag	12 bits	4 bit word
-----------	---------	------------
- associative mapping

16 bit Tag	4 bit word
------------	------------
- 2 way set associative mapping

5 bit tag	11 bit set	4 bit word
-----------	------------	------------

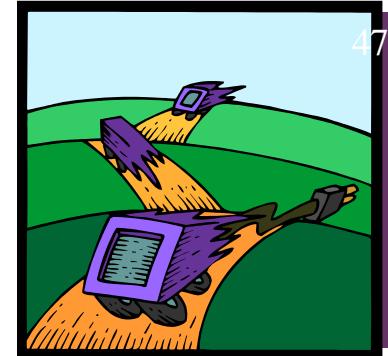


Problem

- A set-associative cache consists of 64 lines, or slots, divided into four-line sets. Main memory contains 4K blocks of 128 words each. Show the format of main memory addresses.
- A two-way set-associative cache has lines of 16 bytes and a total size of 8 Kbytes. The 64-Mbyte main memory is byte addressable. Show the format of main memory addresses.



Replacement Algorithms



- Once the cache has been filled, when a new block is brought into the cache, one of the existing blocks must be replaced
- For direct mapping there is only one possible line for any particular block and no choice is possible
- For the associative and set-associative techniques a replacement algorithm is needed
- To achieve high speed, an algorithm must be implemented in hardware



The four most common replacement algorithms are:

- Least recently used (LRU)
 - Most effective
 - Replace that block in the set that has been in the cache longest with no reference to it
 - Because of its simplicity of implementation, LRU is the most popular replacement algorithm
- First-in-first-out (FIFO)
 - Replace that block in the set that has been in the cache longest
 - Easily implemented as a round-robin or circular buffer technique
- Least frequently used (LFU)
 - Replace that block in the set that has experienced the fewest references
 - Could be implemented by associating a counter with each line

Write Policy

When a block that is resident in the cache is to be replaced there are two cases to consider:

If the old block in the cache has not been altered then it may be overwritten with a new block without first writing out the old block

If at least one write operation has been performed on a word in that line of the cache then main memory must be updated by writing the line of cache out to the block of memory before bringing in the new block

There are two problems to contend with:

More than one device may have access to main memory

A more complex problem occurs when multiple processors are attached to the same bus and each processor has its own local cache - if a word is altered in one cache it could conceivably invalidate a word in other caches



Write Through and Write Back

- Write through
 - Simplest technique
 - All write operations are made to main memory as well as to the cache
 - The main disadvantage of this technique is that it generates substantial memory traffic and may create a bottleneck
- Write back
 - Minimizes memory writes
 - Updates are made only in the cache
 - Portions of main memory are invalid and hence accesses by I/O modules can be allowed only through the cache
 - This makes for complex circuitry and a potential bottleneck



Cache coherency

Even if a write-through policy is used, the other caches may contain invalid data. A system that prevents this problem is said to maintain cache coherency. Possible approaches to cache coherency include the following:

1. **Hardware transparency:** Additional hardware is used to ensure all updates
2. **Non-cacheable memory:** A portion of main memory is shared by more than one processor, and non-cacheable.
3. **Bus watching with write through:** Each cache controller monitors the address lines to detect write operations to memory by other bus masters.

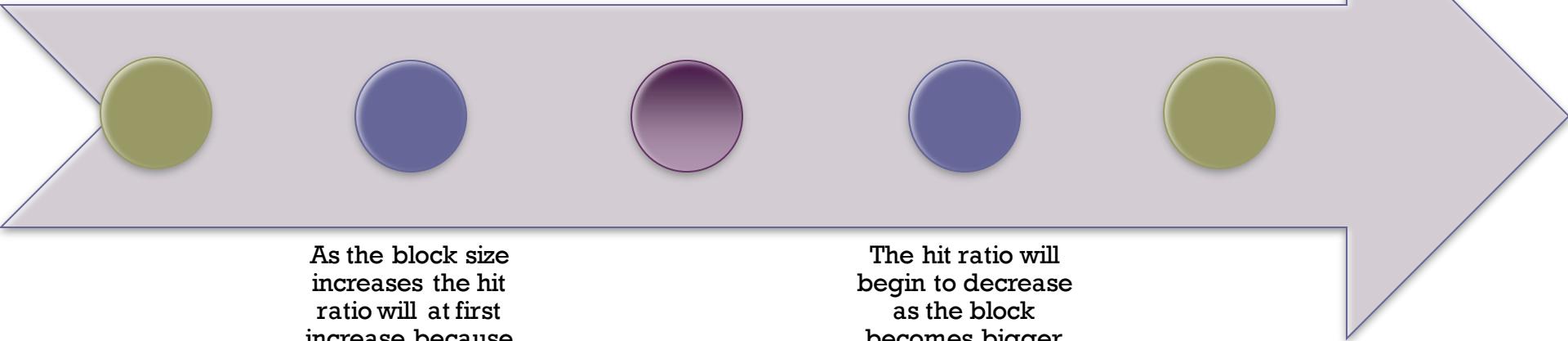
Line Size

When a block of data is retrieved and placed in the cache not only the desired word but also some number of adjacent words are retrieved

As the block size increases more useful data are brought into the cache

Two specific effects come into play:

- Larger blocks reduce the number of blocks that fit into a cache
- As a block becomes larger each additional word is farther from the requested word



As the block size increases the hit ratio will at first increase because of the principle of locality

The hit ratio will begin to decrease as the block becomes bigger and the probability of using the newly fetched information becomes less than the probability of reusing the information that has to be replaced



Multilevel Caches

- As logic density has increased it has become possible to have a cache on the same chip as the processor
- The on-chip cache reduces the processor's external bus activity and speeds up execution time and increases overall system performance
 - When the requested instruction or data is found in the on-chip cache, the bus access is eliminated
 - On-chip cache accesses will complete appreciably faster than would even zero-wait state bus cycles
 - During this period the bus is free to support other transfers
- Two-level cache:
 - Internal cache designated as level 1 (L1)
 - External cache designated as level 2 (L2)
- Potential savings due to the use of an L2 cache depends on the hit rates in both the L1 and L2 caches
- The use of multilevel caches complicates all of the design issues related to caches, including size, replacement algorithm, and write policy

Hit Ratio (L1 & L2) For 8 Kbyte and 16 Kbyte L1

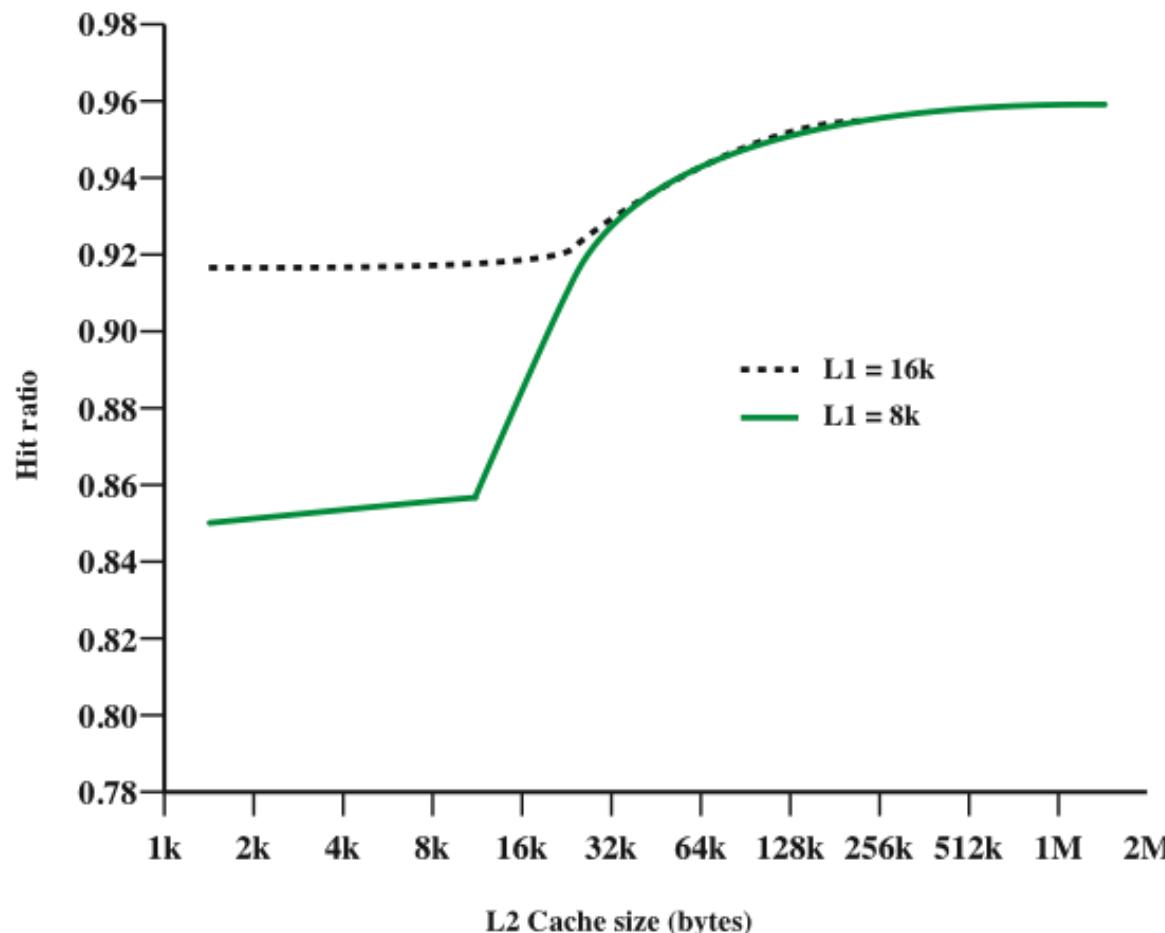


Figure 4.17 Total Hit Ratio (L1 and L2) for 8 Kbyte and 16 Kbyte L1



Unified Versus Split Caches

- Has become common to split cache:
 - One dedicated to instructions
 - One dedicated to data
 - Both exist at the same level, typically as two L1 caches
- Advantages of unified cache:
 - Higher hit rate
 - Balances load of instruction and data fetches automatically
 - Only one cache needs to be designed and implemented
- Trend is toward split caches at the L1 and unified caches for higher levels
- Advantages of split cache:
 - Eliminates cache contention between instruction fetch/decode unit and execution unit
 - Important in pipelining

PENTIUM 4 CACHE ORGANIZATION

Intel Cache Evolution

Problem	Solution	Processor on Which Feature First Appears
External memory slower than the system bus.	Add external cache using faster memory technology.	386
Increased processor speed results in external bus becoming a bottleneck for cache access.	Move external cache on-chip, operating at the same speed as the processor.	486
Internal cache is rather small, due to limited space on chip.	Add external L2 cache using faster technology than main memory.	486
Contention occurs when both the Instruction Prefetcher and the Execution Unit simultaneously require access to the cache. In that case, the Prefetcher is stalled while the Execution Unit's data access takes place.	Create separate data and instruction caches.	Pentium
Increased processor speed results in external bus becoming a bottleneck for L2 cache access.	Create separate back-side bus that runs at higher speed than the main (front-side) external bus. The BSB is dedicated to the L2 cache.	Pentium Pro
	Move L2 cache on to the processor chip.	Pentium II
Some applications deal with massive databases and must have rapid access to large amounts of data. The on-chip caches are too small.	Add external L3 cache.	Pentium III
	Move L3 cache on-chip.	Pentium 4

Pentium 4 Block Diagram

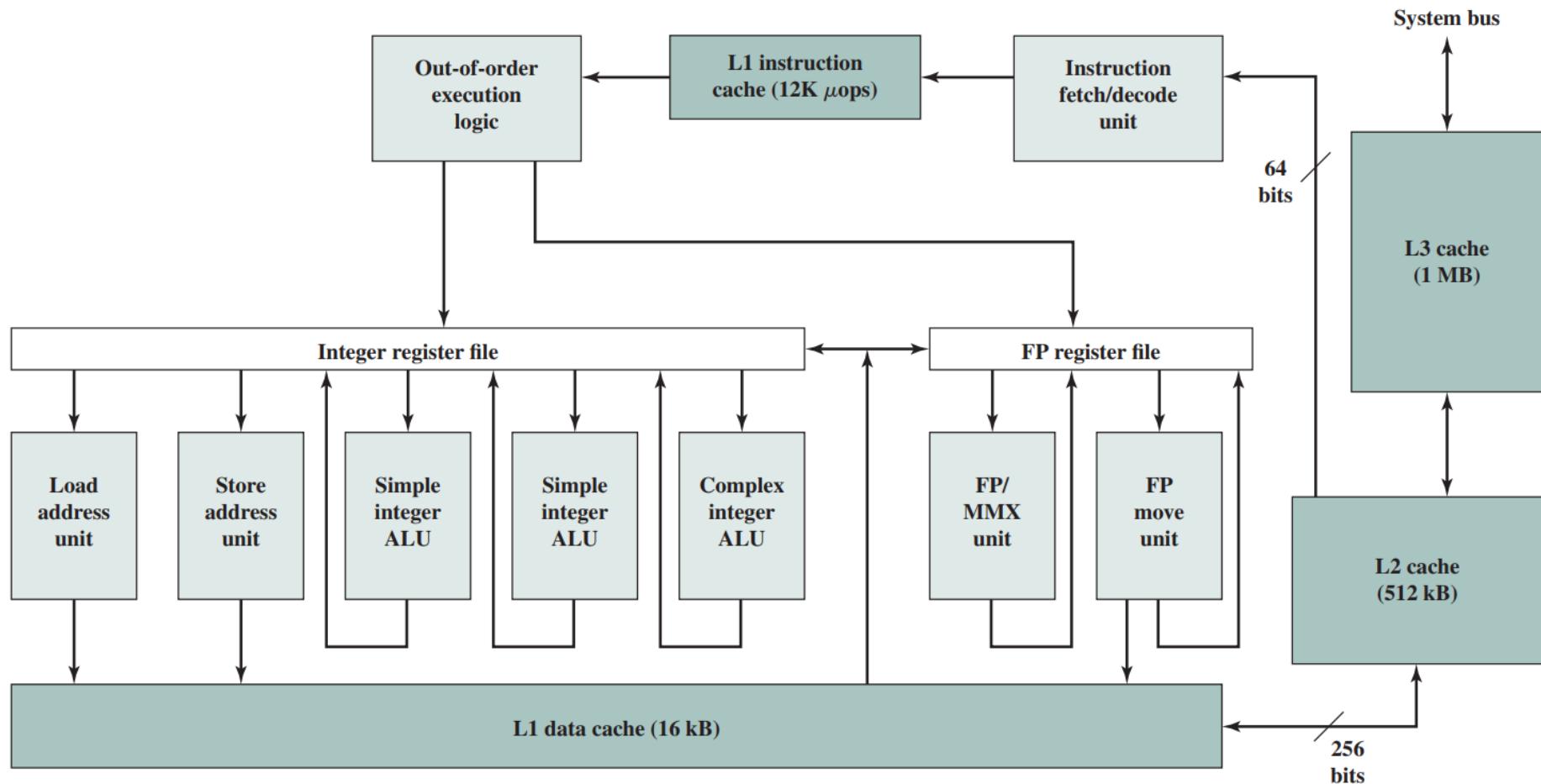


Figure 4.18 Pentium 4 Block Diagram

Pentium 4 Cache Operating Modes

Control Bits		Operating Mode		
CD	NW	Cache Fills	Write Throughs	Invalidates
0	0	Enabled	Enabled	Enabled
1	0	Disabled	Enabled	Enabled
1	1	Disabled	Disabled	Disabled

Note: CD = 0; NW = 1 is an invalid combination.

CD (cache disable)

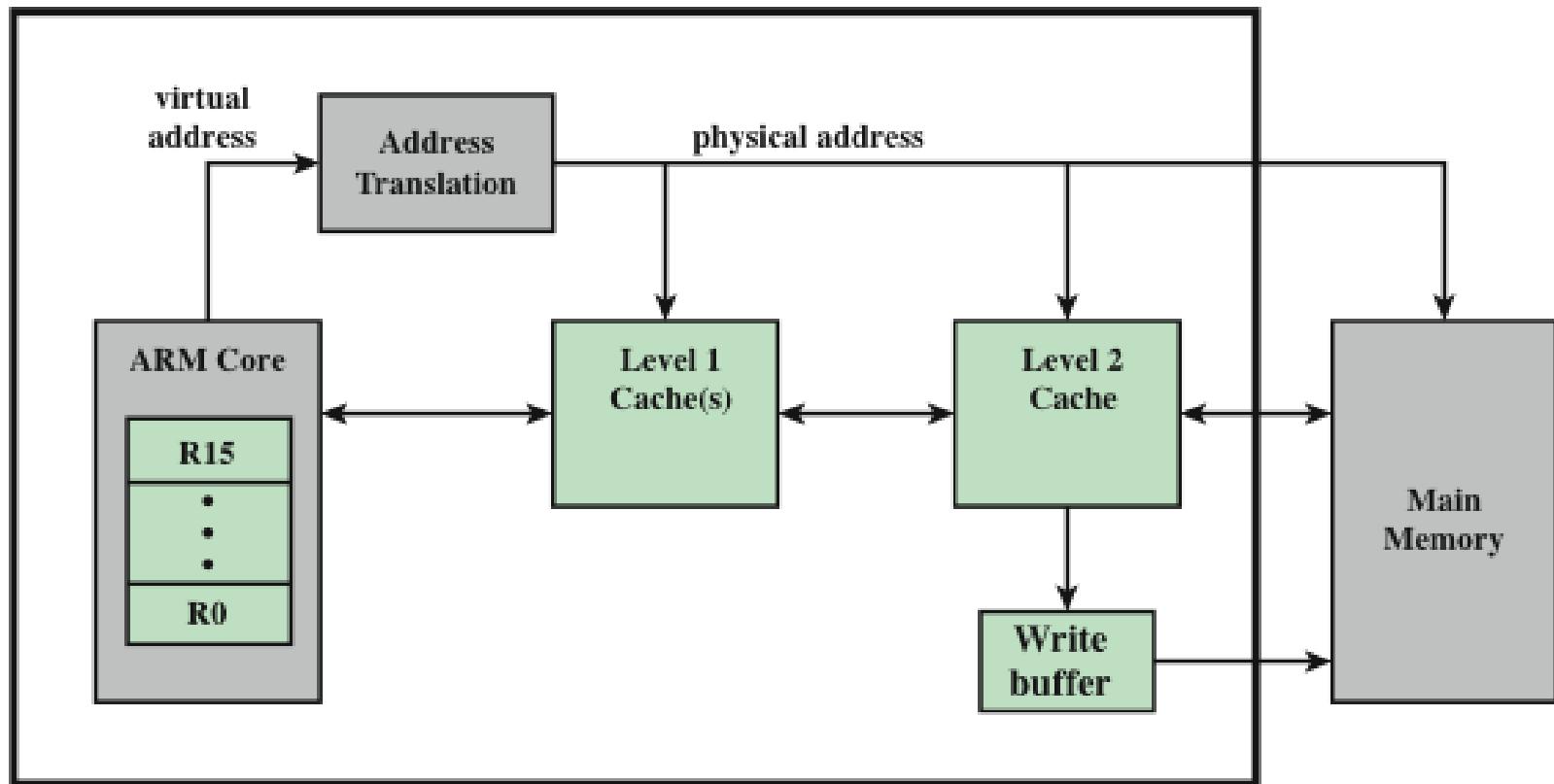
NW (not write-through)

Table 4.5 Pentium 4 Cache Operating Modes

ARM Cache Features

Core	Cache Type	Cache Size (kB)	Cache Line Size (words)	Associativity	Location	Write Buffer Size (words)
ARM720T	Unified	8	4	4-way	Logical	8
ARM920T	Split	16/16 D/I	8	64-way	Logical	16
ARM926EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	16
ARM1022E	Split	16/16 D/I	8	64-way	Logical	16
ARM1026EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	8
Intel StrongARM	Split	16/16 D/I	4	32-way	Logical	32
Intel Xscale	Split	32/32 D/I	8	32-way	Logical	32
ARM1136-JF-S	Split	4-64/4-64 D/I	8	4-way	Physical	32

ARM Cache and Write Buffer Organization



+ Summary

Chapter 4

Cache Memory

- Characteristics of Memory Systems
 - Location
 - Capacity
 - Unit of transfer
- Memory Hierarchy
 - How much?
 - How fast?
 - How expensive?
- Cache memory principles
- Elements of cache design
 - Cache addresses
 - Cache size
 - Mapping function
 - Replacement algorithms
 - Write policy
 - Line size
 - Number of caches
- Pentium 4 cache organization
- ARM cache organization

Thank you for the patience :)

CSE 213

Computer Architecture

Lecture 5: Internal Memory

Military Institute of Science and Technology

Outline

- Semiconductor Main Memory

- Organization

- DRAM and SRAM

- Types of ROM

- Chip Logic

- Chip Packaging

- Module Organization

- Interleaved Memory

- Error Correction

- DDR DRAM

- Flash Memory



Memory Cell Operation

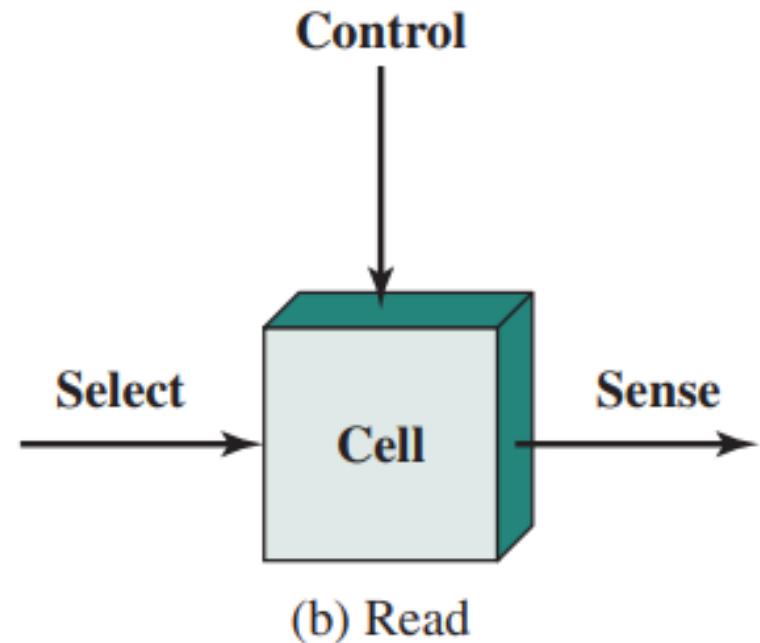
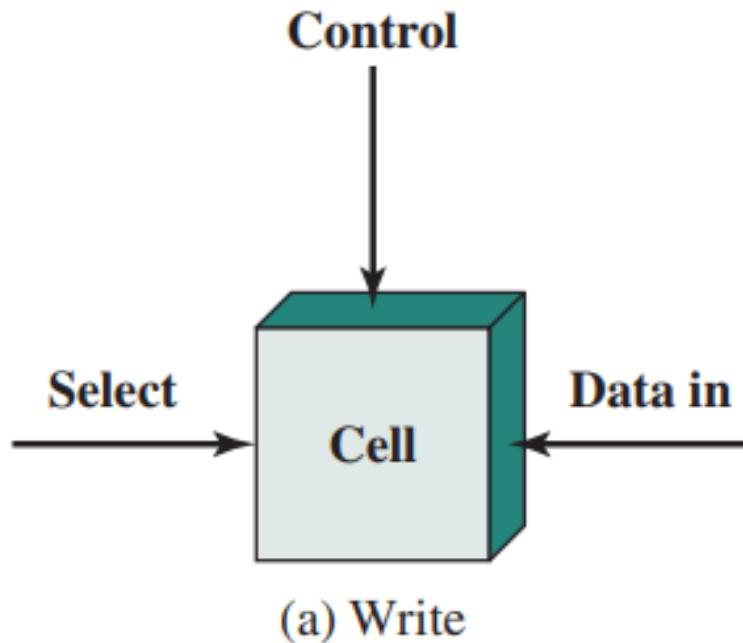


Figure 5.1 Memory Cell Operation

Semiconductor Memory Types

Memory Type	Category	Erasure	Write Mechanism	Volatility	
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile	
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile	
Programmable ROM (PROM)			Electrically		
Erasable PROM (EPROM)	Read-mostly memory	UV light, chip-level			
Electrically Erasable PROM (EEPROM)		Electrically, byte-level			
Flash memory		Electrically, block-level			

Table 5.1 Semiconductor Memory Types



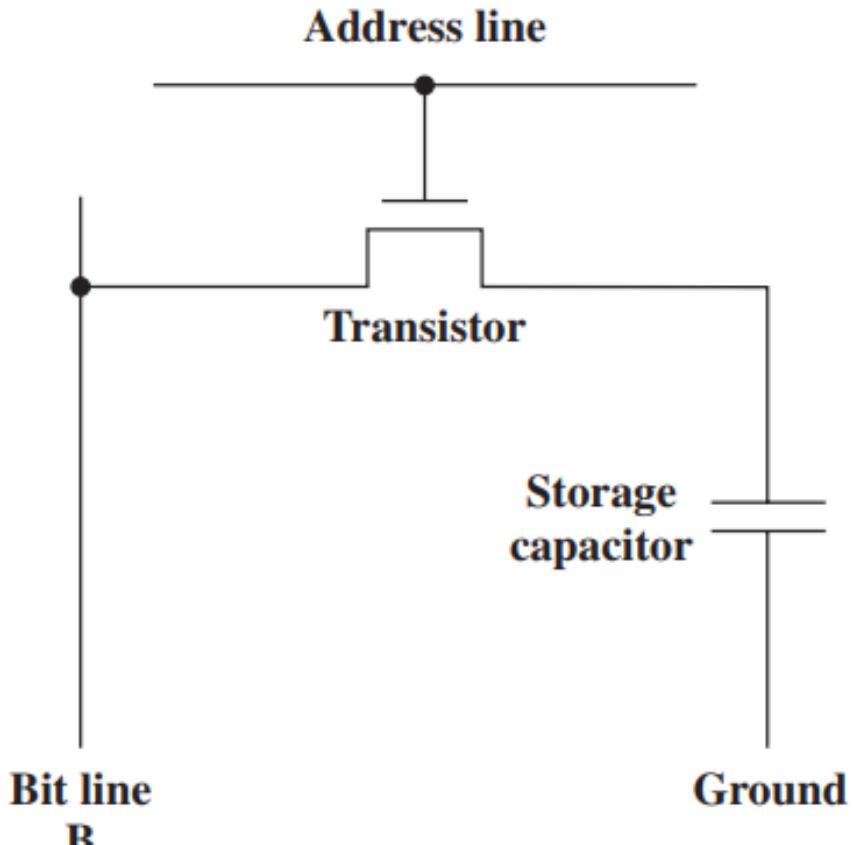
Dynamic RAM (DRAM)

- RAM technology is divided into two technologies:
 - Dynamic RAM (DRAM)
 - Static RAM (SRAM)
- **DRAM**
 - Made with cells that store data as charge on capacitors
 - Presence or absence of charge in a capacitor is interpreted as a binary 1 or 0
 - Requires periodic charge refreshing to maintain data storage
 - The term *dynamic* refers to tendency of the stored charge to leak away, even with power continuously applied

Dynamic RAM Structure

Figure 5.2a

Typical Memory Cell Structures



(a) Dynamic RAM (DRAM) cell



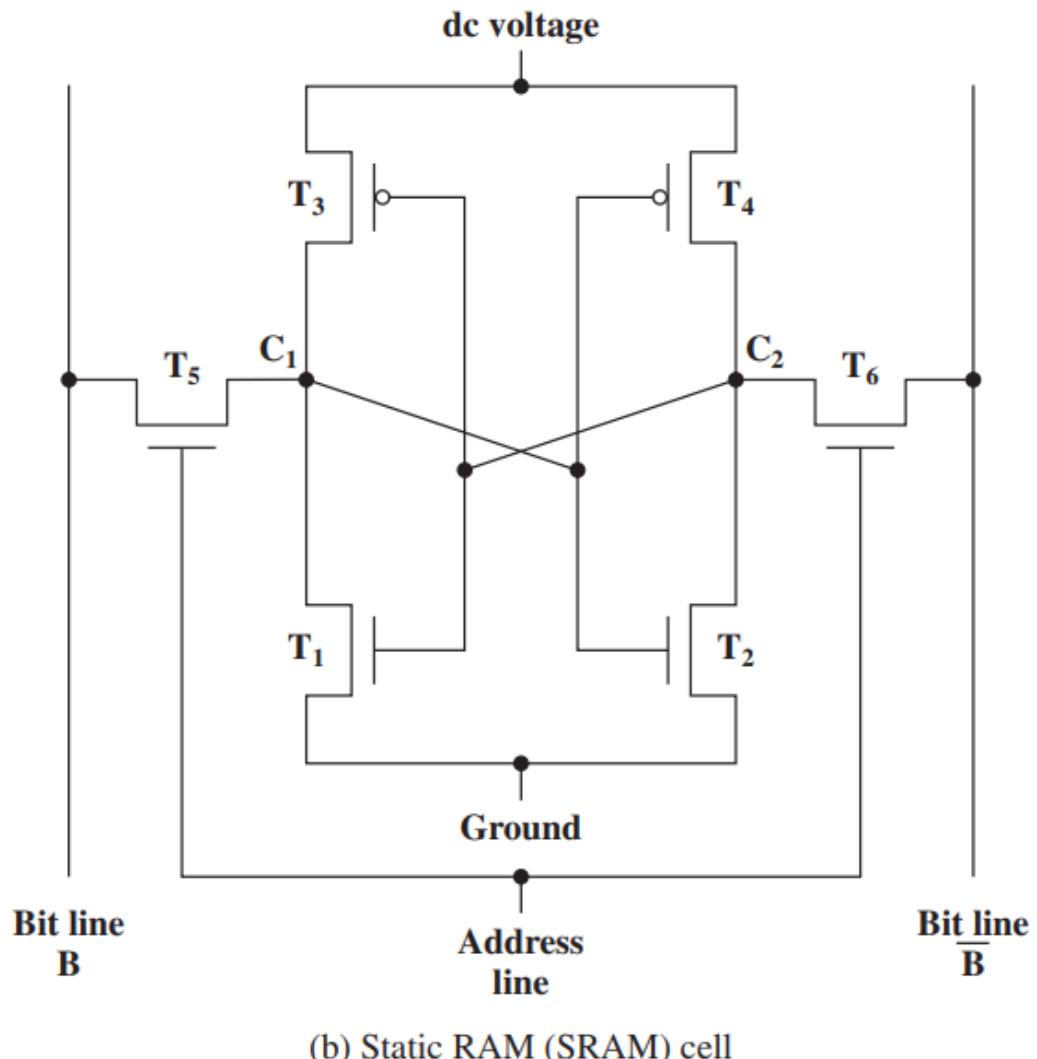
Static RAM (SRAM)

- RAM technology is divided into two technologies:
 - Dynamic RAM (DRAM)
 - Static RAM (SRAM)
- **SRAM**
 - Digital device that uses the same logic elements used in the processor
 - Binary values are stored using traditional flip-flop logic gate configurations
 - Will hold its data as long as power is supplied to it

Static RAM Structure

Figure 5.2b

Typical Memory Cell Structures



SRAM versus DRAM

- Both volatile
 - Power must be continuously supplied to the memory to preserve the bit values

- Dynamic cell
 - Simpler to build, smaller
 - More dense (smaller cells = more cells per unit area)
 - Less expensive
 - Requires the supporting refresh circuitry
 - Tend to be favored for large memory requirements
 - Used for main memory

- Static
 - Faster
 - Used for cache memory (both on and off chip)

SRAM

DRAM



Read Only Memory (ROM)

- Contains a permanent pattern of data that cannot be changed or added to
- No power source is required to maintain the bit values in memory
- Data or program is permanently in main memory and never needs to be loaded from a secondary storage device
- Data is actually wired into the chip as part of the fabrication process
 - Disadvantages of this:
 - No room for error, if one bit is wrong the whole batch of ROMs must be thrown out
 - Data insertion step includes a relatively large fixed cost



Programmable ROM (PROM)

- Less expensive **alternative**
- Nonvolatile and may be written into only **once**
- Writing process is performed **electrically** and may be performed by supplier or customer at a time later than the original chip fabrication
- **Special equipment** is required for the writing process
- Provides **flexibility** and convenience
- Attractive for **high volume** production runs

Read-Mostly Memory

EPROM

Erasable programmable
read-only memory

Erasure process can be
performed repeatedly

More expensive than
PROM but it has the
advantage of the multiple
update capability

EEPROM

Electrically erasable
programmable read-only
memory

Can be written into at any
time without erasing prior
contents

Combines the advantage of
non-volatility with the
flexibility of being
updatable in place

More expensive than
EPROM

Flash Memory

Intermediate between
EPROM and EEPROM in
both cost and functionality

Uses an electrical erasing
technology, does not
provide byte-level erasure

Microchip is organized so
that a section of memory
cells are erased in a single
action or “flash”

Typical 16 Mb DRAM (4M x 4)

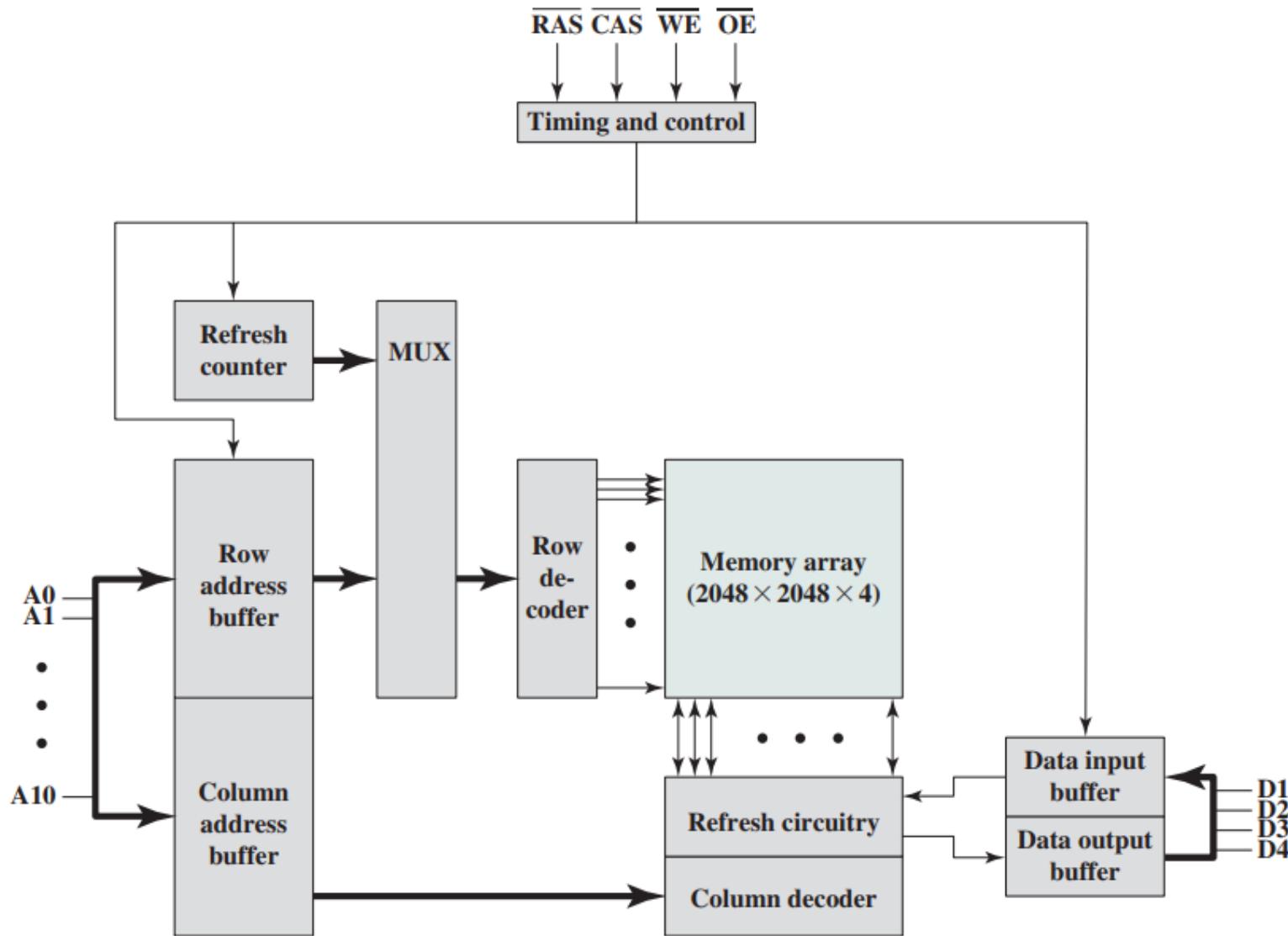
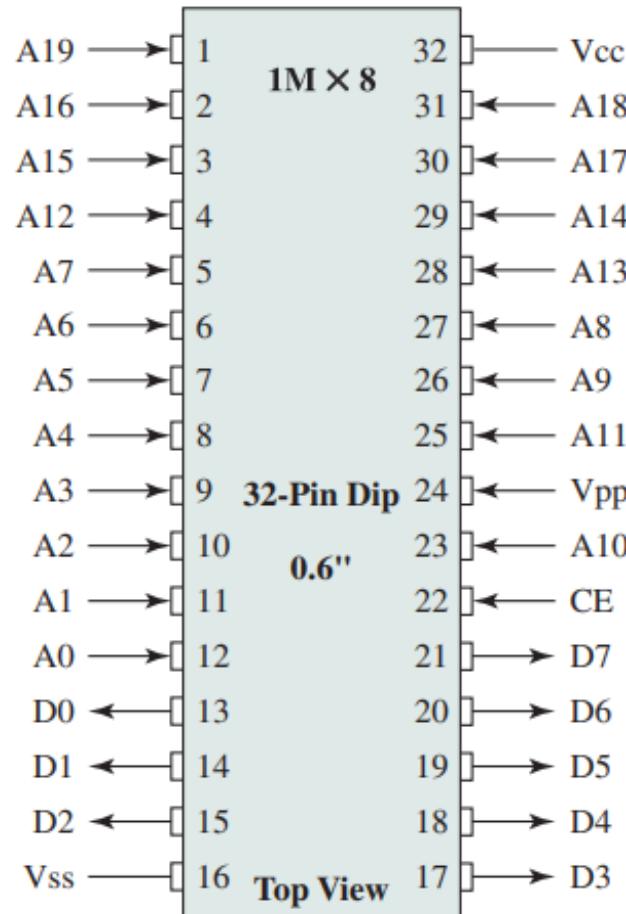
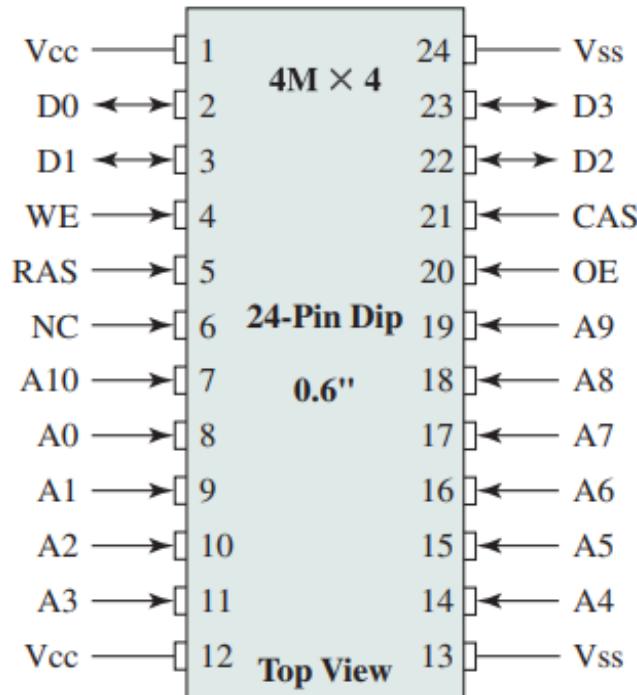


Figure 5.3 Typical 16-Mbit DRAM (4M × 4)

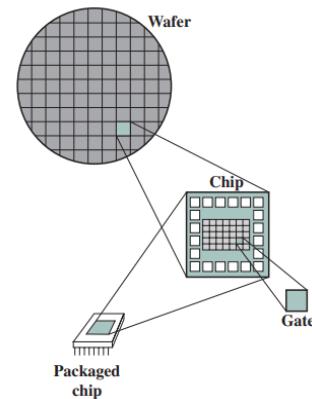
Chip Packaging



(a) 8-Mbit EPROM



(b) 16-Mbit DRAM

**Figure 5.4** Typical Memory Package Pins and Signals

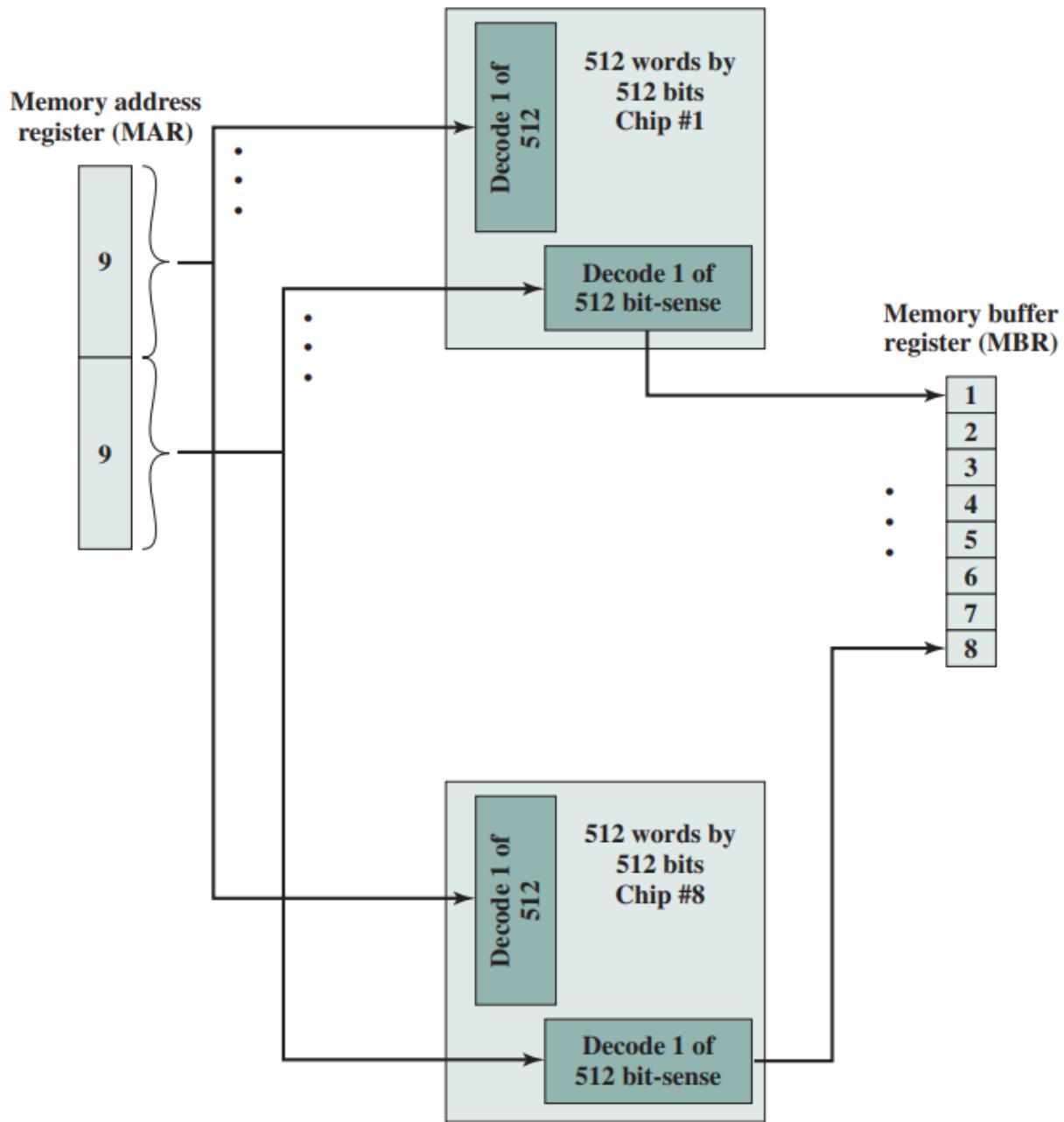


Figure 5.5 256-KByte Memory Organization

Figure 5.5

256-KByte
Memory
Organization

+ Memory Organization

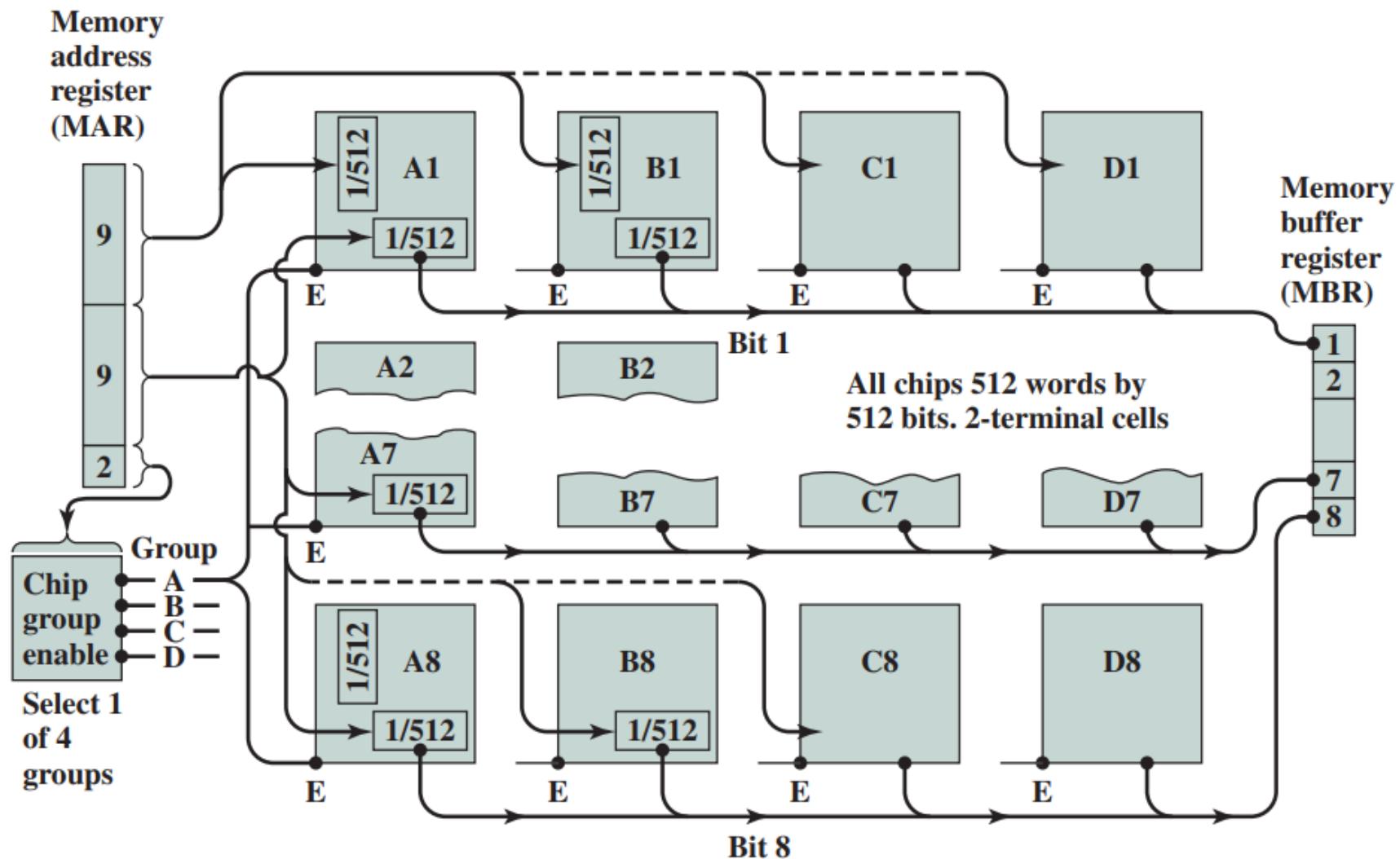
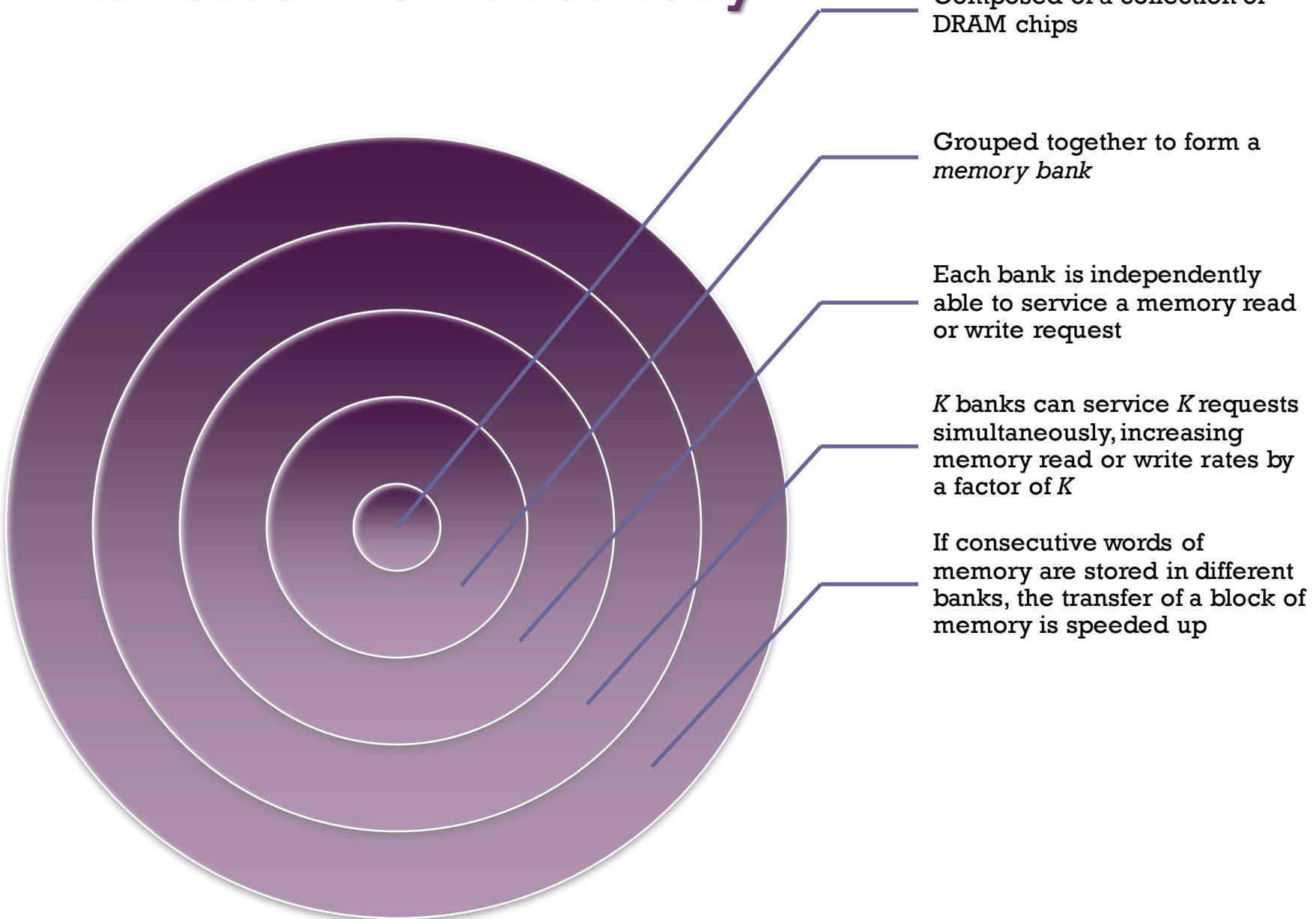


Figure 5.6 1-MB Memory Organization

Interleaved Memory



Error Correction



Error Correction

- Hard Failure
 - Permanent physical defect
 - Memory cell or cells affected cannot reliably store data but become stuck at 0 or 1 or switch erratically between 0 and 1
 - Can be caused by:
 - Harsh environmental abuse
 - Manufacturing defects
 - Wear
- Soft Error
 - Random, non-destructive event that alters the contents of one or more memory cells
 - No permanent damage to memory
 - Can be caused by:
 - Power supply problems
 - Alpha particles from radioactive decay *

*(Radioactive decay is the process by which an unstable atomic nucleus loses energy by radiation.)

Error Correcting Code Function

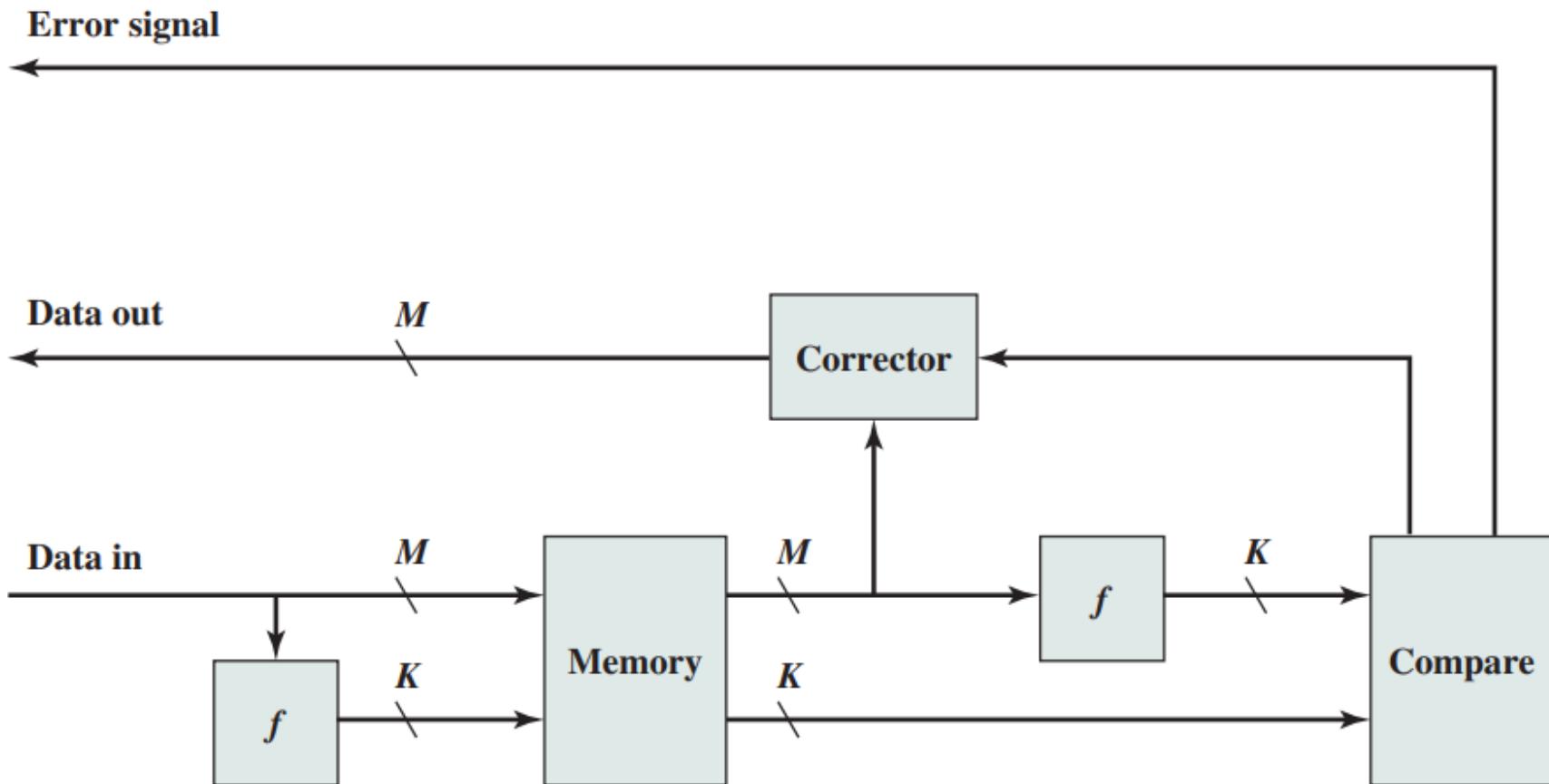


Figure 5.7 Error-Correcting Code Function

Hamming Error Correcting Code

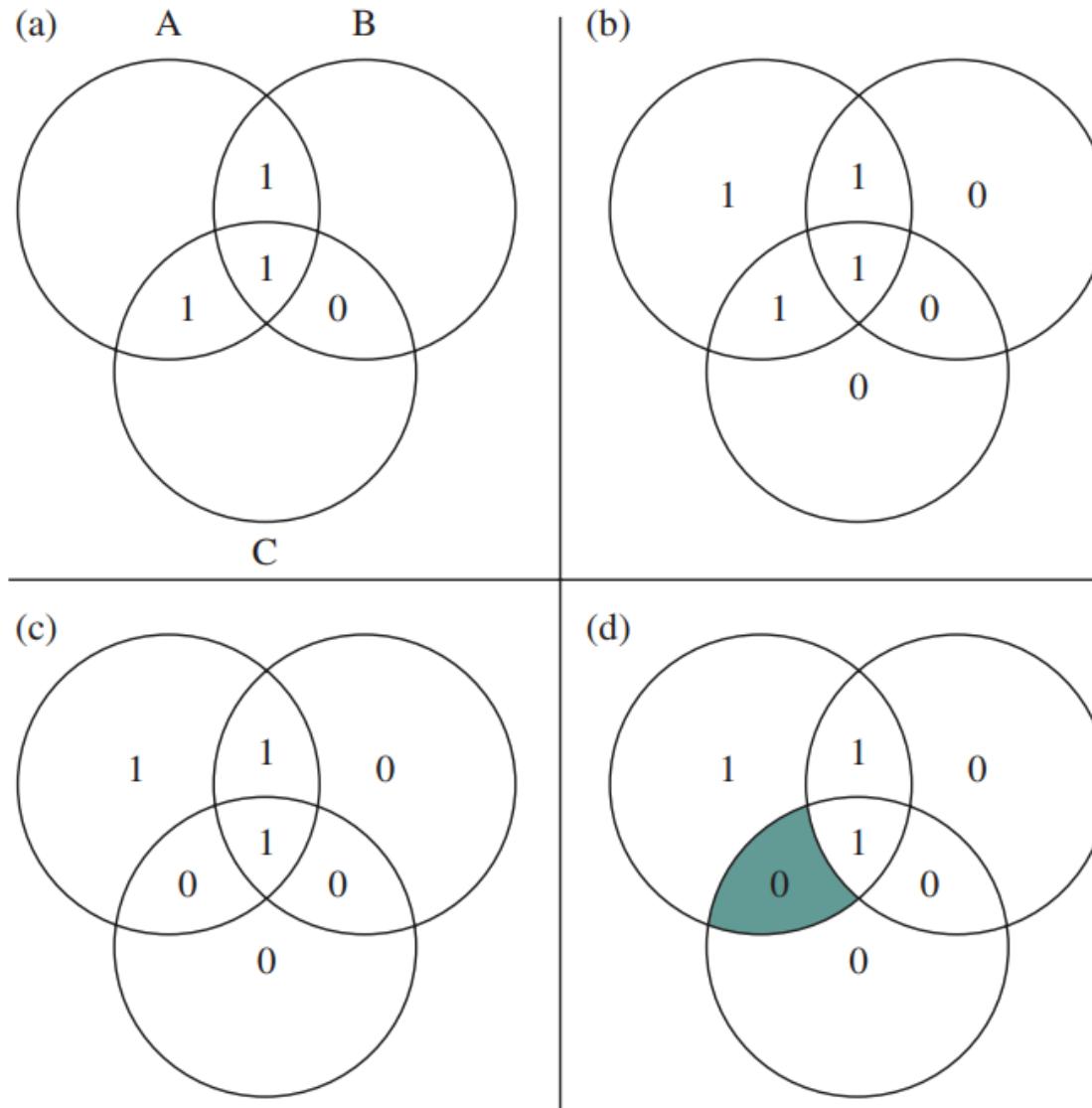


Figure 5.8 Hamming Error-Correcting Code

Hamming SEC-DED Code

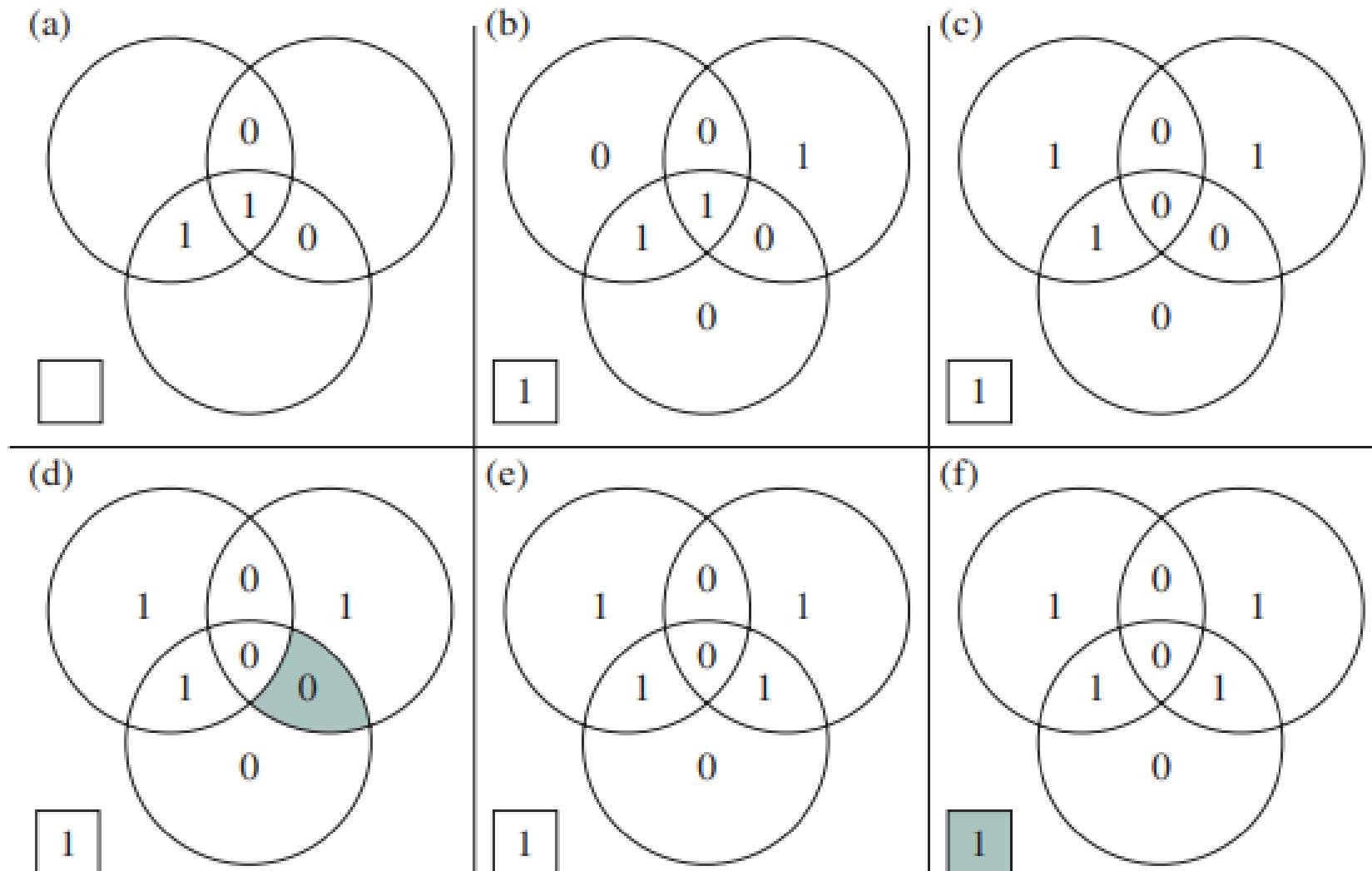


Figure 5.11 Hamming SEC-DEC Code



Hamming Code Syndrome

If we compare the read K bits with the write K bits, using an exclusive or function, the result is called the “syndrome”.

- If the syndrome is all zeros, there were no errors, i.e. the bits were exactly alike
- If there is a 1 bit somewhere, we know that 1 represents an error.



Hamming Code Design – determining K

- To store an M bit word with detection/correction takes $M+K$ bit words.
- If $K = 1$, we can detect single bit errors but not correct them
- If $2^K - 1 \geq M + K$, we can detect, identify, and correct all single bit errors, i.e. the syndrome contains the information to correct any single bit error

■ Example: For $M = 8$:

and $K = 3$: $2^3 - 1 = 7 < 8 + 3$ (doesn't work)

and $K = 4$: $2^4 - 1 = 15 > 8 + 4$ (works!)

Therefore, we must choose $K = 4$, i.e., the minimum size of the syndrome is 4

Check bits required for various data word lengths

Table 5.2 Increase in Word Length with Error Correction

Data Bits	Single-Error Correction		Single-Error Correction/ Double-Error Detection	
	Check Bits	% Increase	Check Bits	% Increase
8	4	50.0	5	62.5
16	5	31.25	6	37.5
32	6	18.75	7	21.875
64	7	10.94	8	12.5
128	8	6.25	9	7.03
256	9	3.52	10	3.91

Hamming Code Syndrome Design Criteria

- For convenience, we would like to generate a 4-bit syndrome for an 8-bit data word with the following characteristics:
 - If the syndrome contains all 0s, no error has been detected.
 - If the syndrome contains one and only one bit set to 1, then an error has occurred in one of the 4 check bits. No correction is needed.
 - If the syndrome contains more than one bit set to 1, then the numerical value of the syndrome indicates the position of the data bit in error. This data bit is inverted for correction

A Layout of Data and Check Bits that Achieves Our Design Criteria:

Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check bit					C8				C4		C2	C1

C1 is a parity check on every data bit whose position is xxx1

$$C1 = D1 \text{ exor } D2 \text{ exor } D4 \text{ exor } D5 \text{ exor } D7$$

C2 is a parity check on every data bit whose position is xx1x

$$C2 = D1 \text{ exor } D3 \text{ exor } D4 \text{ exor } D6 \text{ exor } D7$$

C4 is a parity check on every data bit whose position is x1xx

$$C4 = D2 \text{ exor } D3 \text{ exor } D4 \text{ exor } D8$$

C8 is a parity check on every data bit whose position is 1xxx

$$C8 = D5 \text{ exor } D6 \text{ exor } D7 \text{ exor } D8$$

Why this ordering? Because we want the *syndrome*, the Hamming test word, to yield the address of the error.

Example:

Data need to be stored = 00111001

Check bits: $C_1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$

$$C_2 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$C_4 = 0 \oplus 0 \oplus 1 \oplus 0 = 1$$

$$C_8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

Putting it together:

Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check bit					C8				C4		C2	C1
Word stored as	0	0	1	1	0	1	0	0	1	1	1	1

Example:

Word fetched:

Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check bit					C8				C4		C2	C1
Word fetched as	0	0	1	1	0	1	1	0	1	1	1	1

$$C1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$

$$C2 = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C4 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$$

$$C8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

Comparing:

C8	C4	C2	C1
0	1	1	1
\oplus	0	0	1
	0	1	0

Original Check Bits

New Check Bits

Syndrome

0110 = 6 \rightarrow bit position 6 is wrong, i.e. bit D3 is wrong

Example:

Putting it all together:

Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check bit					C8				C4		C2	C1
Word stored as	0	0	1	1	0	1	0	0	1	1	1	1
Word fetched as	0	0	1	1	0	1	1	0	1	1	1	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Check bit					0				0		0	1

DDR DRAM

Advanced DRAM Organization

SDRAM

DDR-DRAM

RDRAM

- One of the most critical system bottlenecks when using high-performance processors is the interface to main internal memory
- The traditional DRAM chip is constrained both by its internal architecture and by its interface to the processor's memory bus
- A number of enhancements to the basic DRAM architecture have been explored:

+

	Clock Frequency (MHz)	Transfer Rate (GB/s)	Access Time (ns)	Pin Count
SDRAM	166	1.3	18	168
DDR	200	3.2	12.5	184
RDRAM	600	4.8	12	162

Table 5.3 Performance Comparison of Some DRAM Alternatives

Synchronous DRAM (SDRAM)

One of the most widely used forms of DRAM

Exchanges data with the processor synchronized to an external clock signal and running at the full speed of the processor/memory bus without imposing wait states

With synchronous access the DRAM moves data in and out under control of the system clock

- The processor or other master issues the instruction and address information which is latched by the DRAM
- The DRAM then responds after a set number of clock cycles
- Meanwhile the master can safely do other tasks while the SDRAM is processing

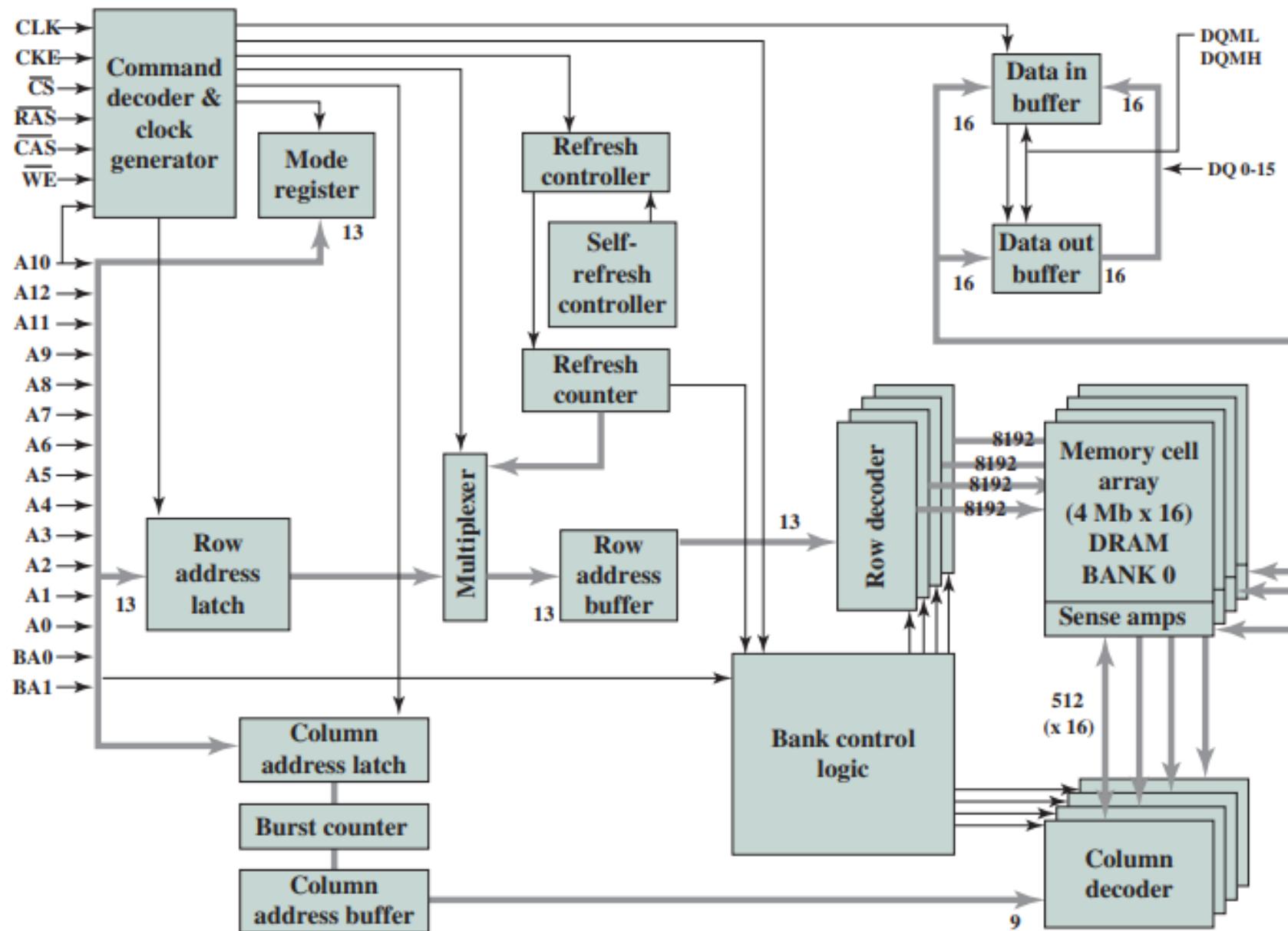


Figure 5.12 256-Mb Synchronous Dynamic RAM (SDRAM)



SDRAM Pin Assignments

A0 to A13	Address inputs
BA0, BA1	Bank address lines
CLK	Clock input
CKE	Clock enable
\overline{CS}	Chip select
\overline{RAS}	Row address strobe
\overline{CAS}	Column address strobe
\overline{WE}	Write enable
DQ0 to DQ7	Data input/output
DQM	Data mask

Table 5.3 SDRAM Pin Assignments

SDRAM Read Timing

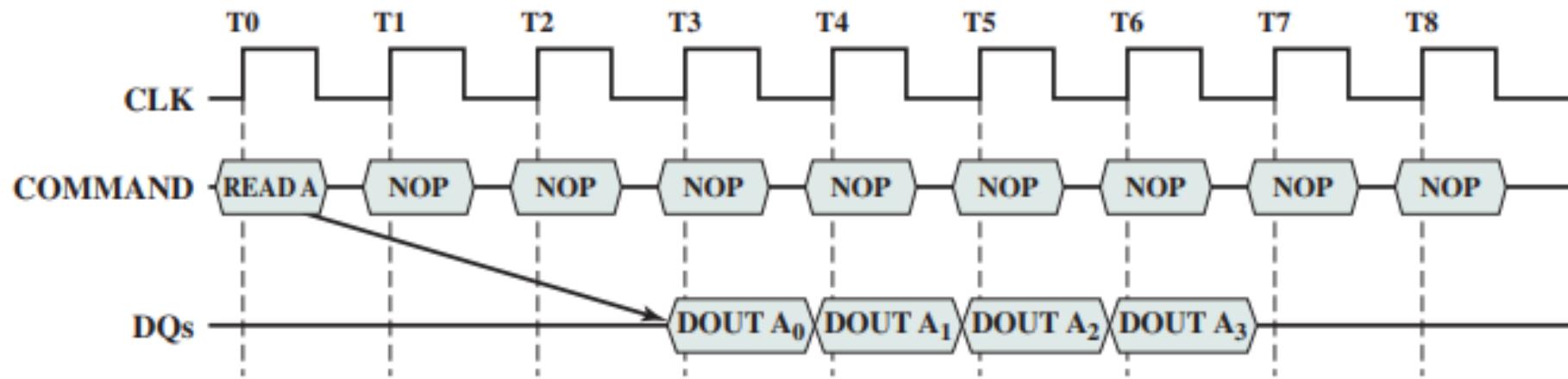


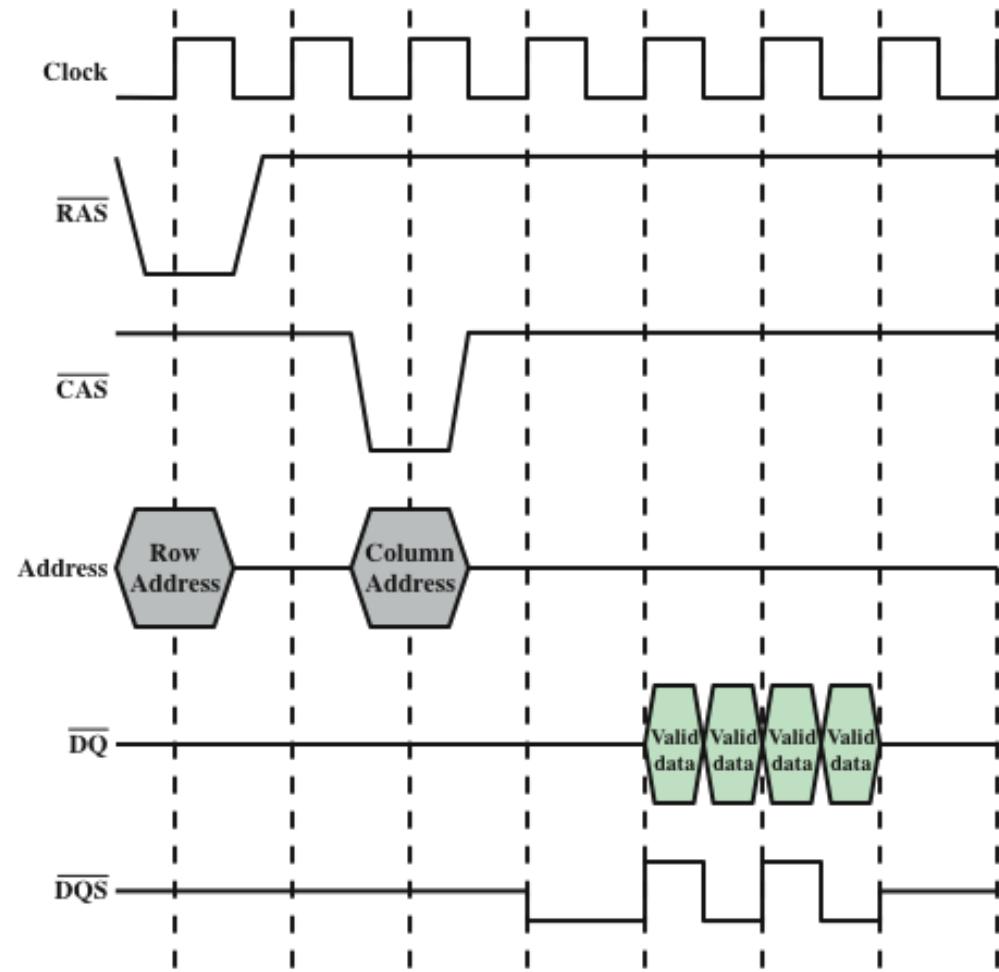
Figure 5.13 SDRAM Read Timing (burst length = 4, $\overline{\text{CAS}}$ latency = 2)



Double Data Rate SDRAM (DDR SDRAM)

- SDRAM can only send data once per bus clock cycle
- Double-data-rate SDRAM can send data twice per clock cycle, once on the rising edge of the clock pulse and once on the falling edge
- Developed by the JEDEC Solid State Technology Association (Electronic Industries Alliance's semiconductor-engineering-standardization body)

DDR SDRAM Read Timing



RAS = row address select
CAS = column address select
DQ = data (in or out)
DQS = DQ select

Figure 5.15 DDR SDRAM Read Timing

+

DDR2 SDRAM

(Double Data Rate Two SDRAM)

Its primary benefit is the ability to operate the external data bus twice as fast as DDR SDRAM. This is achieved by improved bus signal.

The prefetch buffer of DDR2 is 4 bit(double of DDR SDRAM). DDR2 memory is at the same internal clock speed (133~200MHz) as DDR, but the transfer rate of DDR2 can reach 533~800 MT/s with the improved I/O bus signal. DDR2 533 and DDR2 800 memory types are on the market.



DDR3 SDRAM

(Double Data Rate Three SDRAM)

DDR3 memory reduces 40% power consumption compared to current DDR2 modules, allowing for lower operating currents and voltages (1.5 V, compared to DDR2's 1.8 V or DDR's 2.5 V).

The transfer rate of DDR3 is 800~1600 MT/s.

DDR3's prefetch buffer width is 8 bit, whereas DDR2's is 4 bit, and DDR's is 2 bit.

DDR3 also adds two functions, such as ASR (Automatic Self-Refresh) and SRT (Self-Refresh Temperature). They can make the memory control the refresh rate according to the temperature variation.



DDR4 SDRAM

(Double Data Rate Fourth SDRAM)

DDR4 SDRAM provides the lower operating voltage (1.2V) and higher transfer rate. The transfer rate of DDR4 is 2133~3200 MT/s.

DDR4 adds four new Bank Groups technology. Each bank group has the feature of singlehanded operation.

DDR4 can process 4 data within a clock cycle, so DDR4's efficiency is better than DDR3 obviously.

DDR4 also adds some functions, such as DBI (Data Bus Inversion), CRC (Cyclic Redundancy Check) and CA parity. They can enhance DDR4 memory's signal integrity, and improve the stability of data transmission/access.

+

Comparison between SDRAM, DDR1, DDR2, DDR3 and DDR4

DDR SDRAM Standard	Internal rate (MHz)	Bus clock (MHz)	Prefetch	Data rate (MT/s)	Transfer rate (GB/s)	Voltage (V)
SDRAM	100-166	100-166	1n	100-166	0.8-1.3	3.3
DDR	133-200	133-200	2n	266-400	2.1-3.2	2.5/2.6
DDR2	133-200	266-400	4n	533-800	4.2-6.4	1.8
DDR3	133-200	533-800	8n	1066-1600	8.5-14.9	1.35/1.5
DDR4	133-200	1066-1600	8n	2133-3200	17-21.3	1.2

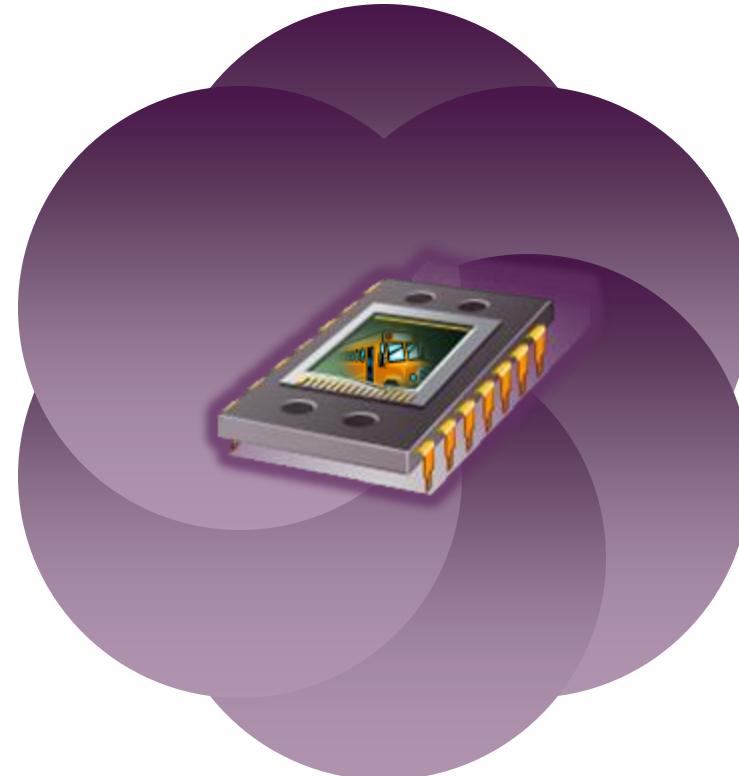
RDRAM

Developed by Rambus

Bus delivers address and control information using an asynchronous block-oriented protocol

- Gets a memory request over the high-speed bus
- Request contains the desired address, the type of operation, and the number of bytes in the operation

Bus can address up to 320 RDRAM chips and is rated at 1.6 GBps



Adopted by Intel for its Pentium and Itanium processors

Has become the main competitor to SDRAM

Chips are vertical packages with all pins on one side

- Exchanges data with the processor over 28 wires no more than 12 centimeters long

+

RDRAM Structure

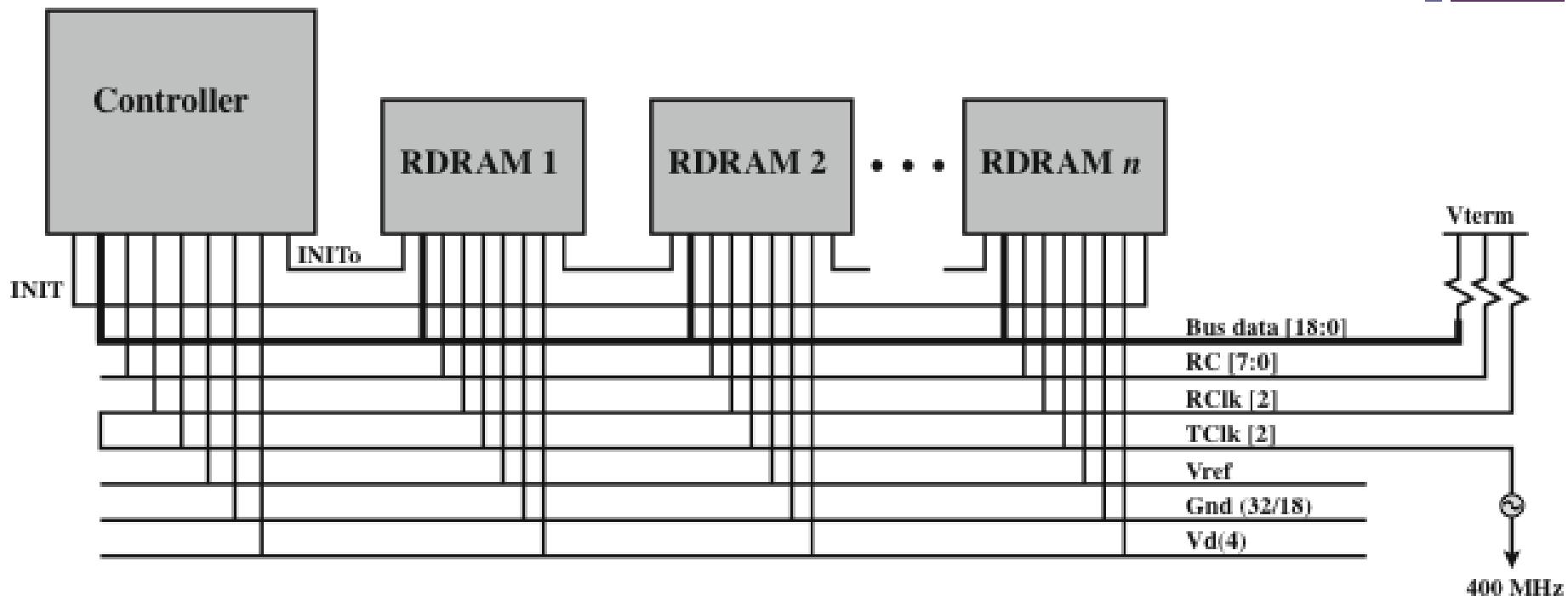


Figure 5.14 RDRAM Structure



RDRAM - Performance

- Compared to other contemporary standards, Rambus showed increase in latency, heat output, manufacturing complexity, and cost. Because of more complex interface circuitry and increased number of memory banks, RDRAM die size was larger than that of contemporary SDRAM chips, and results in a 10–20 percent price premium at 16 Mbit densities



Cache DRAM (CDRAM)

- Developed by Mitsubishi
- Integrates a small SRAM cache onto a generic DRAM chip
- SRAM on the CDRAM can be used in two ways:
 - It can be used as a true cache consisting of a number of 64-bit lines
 - Cache mode of the CDRAM is effective for ordinary random access to memory
 - Can also be used as a buffer to support the serial access of a block of data

Summary

Chapter 5

- Semiconductor main memory
 - Organization
 - DRAM and SRAM
 - Types of ROM
 - Chip logic
 - Chip packaging
 - Module organization
 - Interleaved memory
- Error correction
 - Hard failure
 - Soft error
- Hamming code
- Advanced DRAM organization
 - Synchronous DRAM
 - Rambus DRAM
 - DDR SDRAM
 - Cache DRAM

Internal Memory

CSE 213

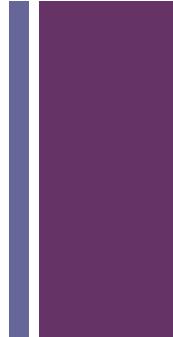
Computer Architecture

Lecture 6: External Memory

Military Institute of Science and Technology



Magnetic Disk



- A disk is a circular *platter* constructed of nonmagnetic material, called the *substrate*, coated with a magnetizable material
 - Traditionally the substrate has been an aluminium or aluminium alloy material
 - Recently glass substrates have been introduced
- Benefits of the glass substrate:
 - Improvement in the uniformity of the magnetic film surface to increase disk reliability
 - A significant reduction in overall surface defects to help reduce read-write errors
 - Ability to support lower fly heights
 - Better stiffness to reduce disk dynamics
 - Greater ability to withstand shock and damage



Magnetic Read and Write Mechanisms

Data are recorded on and later retrieved from the disk via a conducting coil named the *head*

- In many systems there are two heads, a read head and a write head
- During a read or write operation the head is stationary while the platter rotates beneath it

Electric pulses are sent to the write head and the resulting magnetic patterns are recorded on the surface below, with different patterns for positive and negative currents

The write mechanism exploits the fact that electricity flowing through a coil produces a magnetic field

The write head itself is made of easily magnetizable material and is in the shape of a rectangular doughnut with a gap along one side and a few turns of conducting wire along the opposite side

An electric current in the wire induces a magnetic field across the gap, which in turn magnetizes a small area of the recording medium

Reversing the direction of the current reverses the direction of the magnetization on the recording medium

Inductive Write/Magnetoresistive Read Head

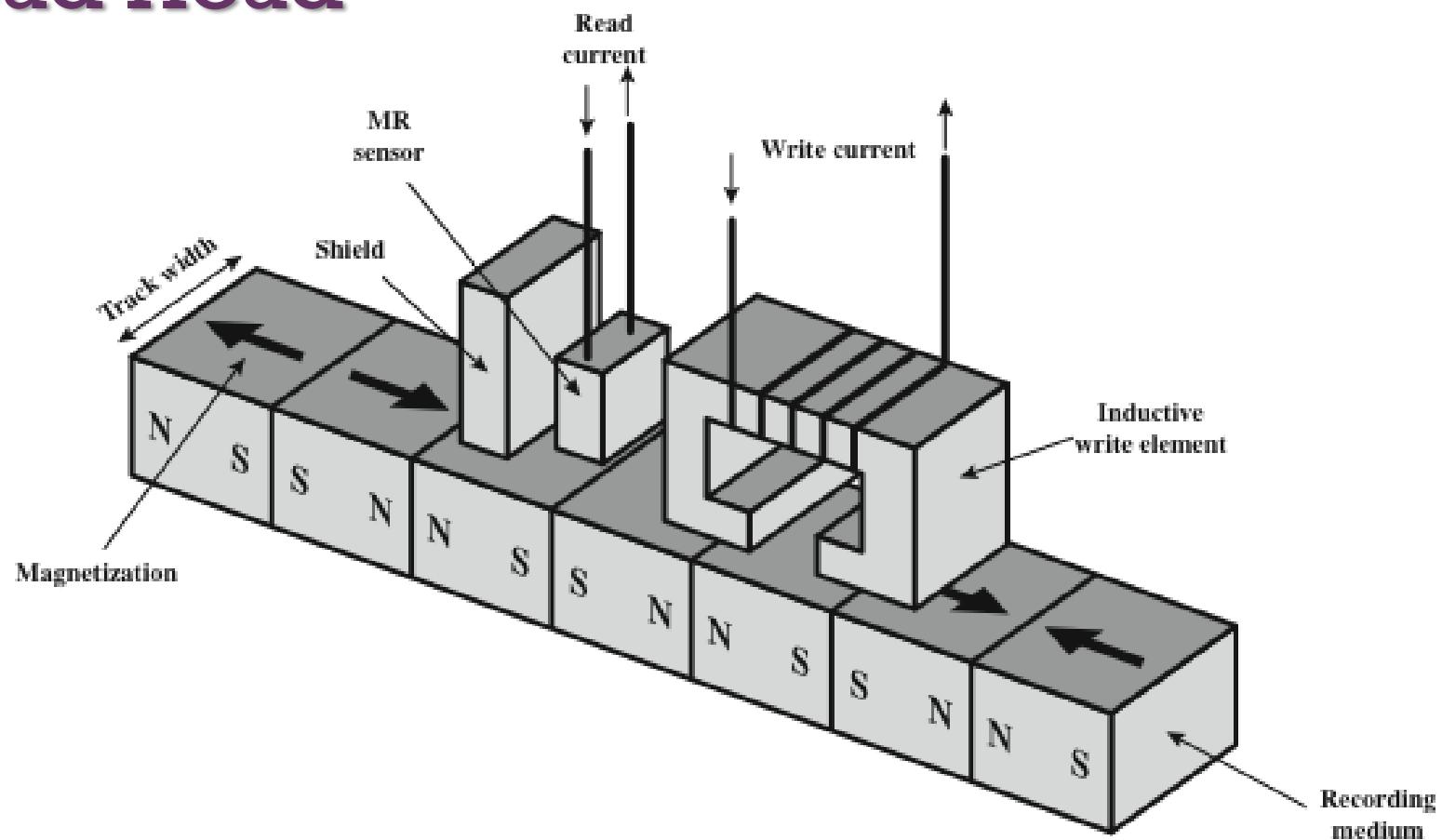
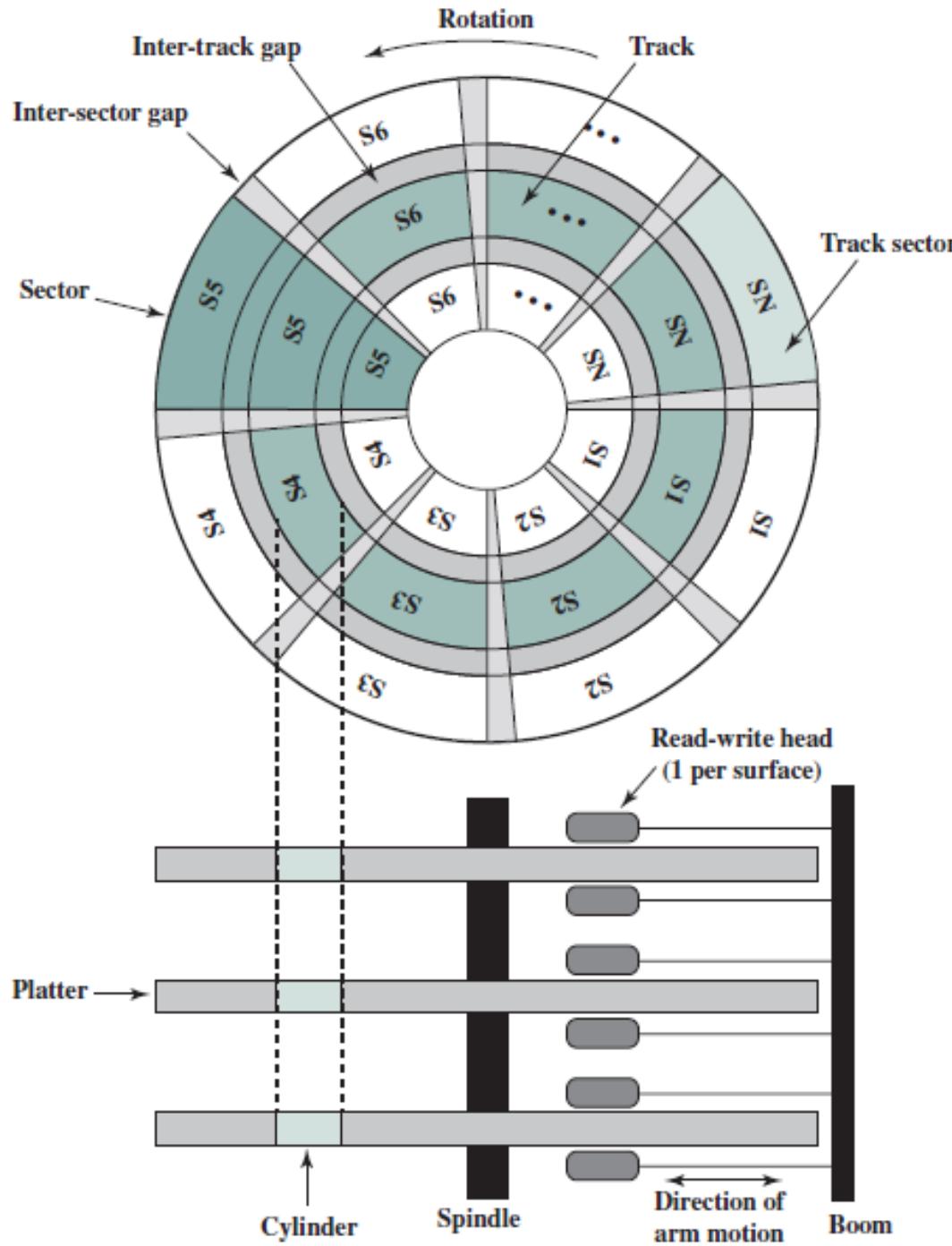
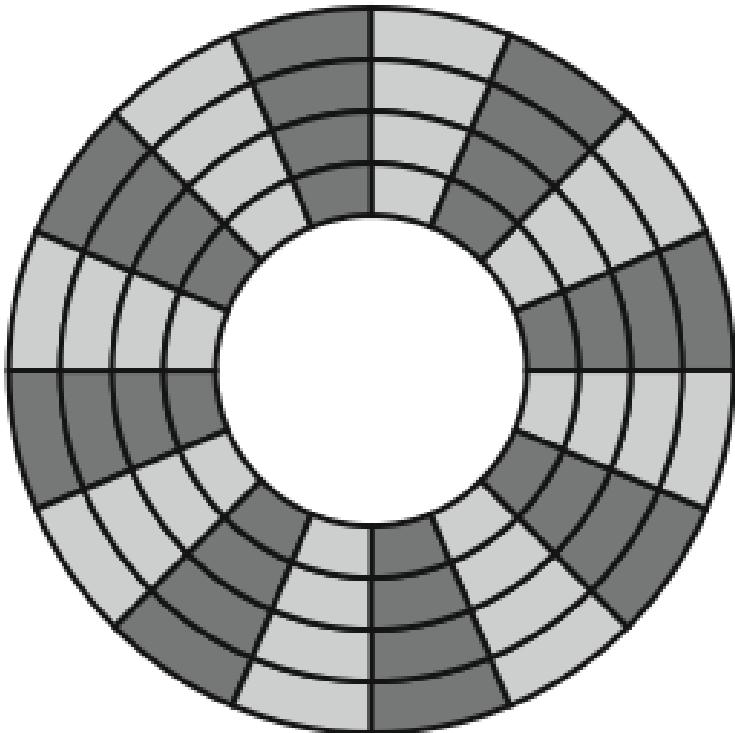


Figure 6.1 Inductive Write/Magnetoresistive Read Head

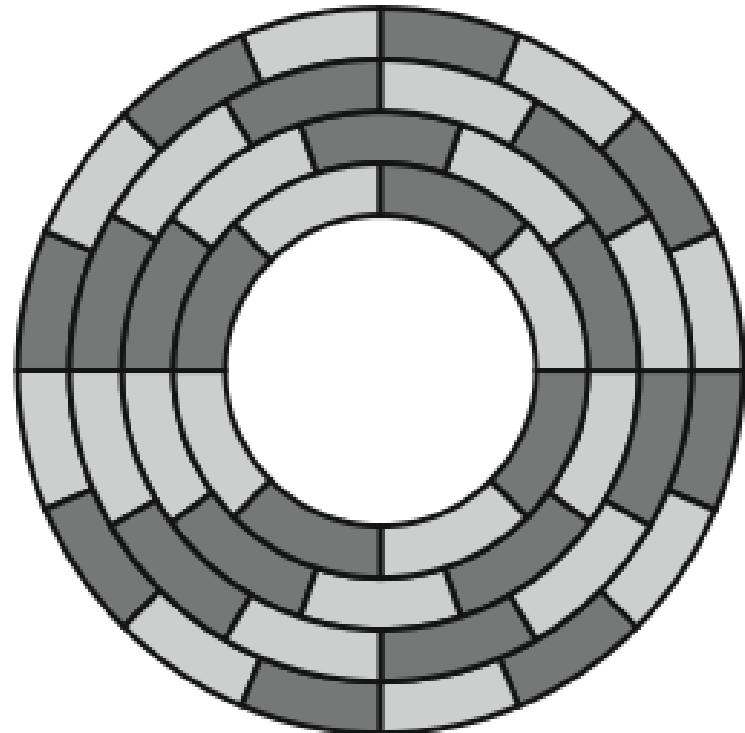
Disk Data Layout



Disk Layout Methods Diagram



(a) Constant angular velocity



(b) Multiple zoned recording

Figure 6.3 Comparison of Disk Layout Methods

Winchester Disk Format

Seagate ST506

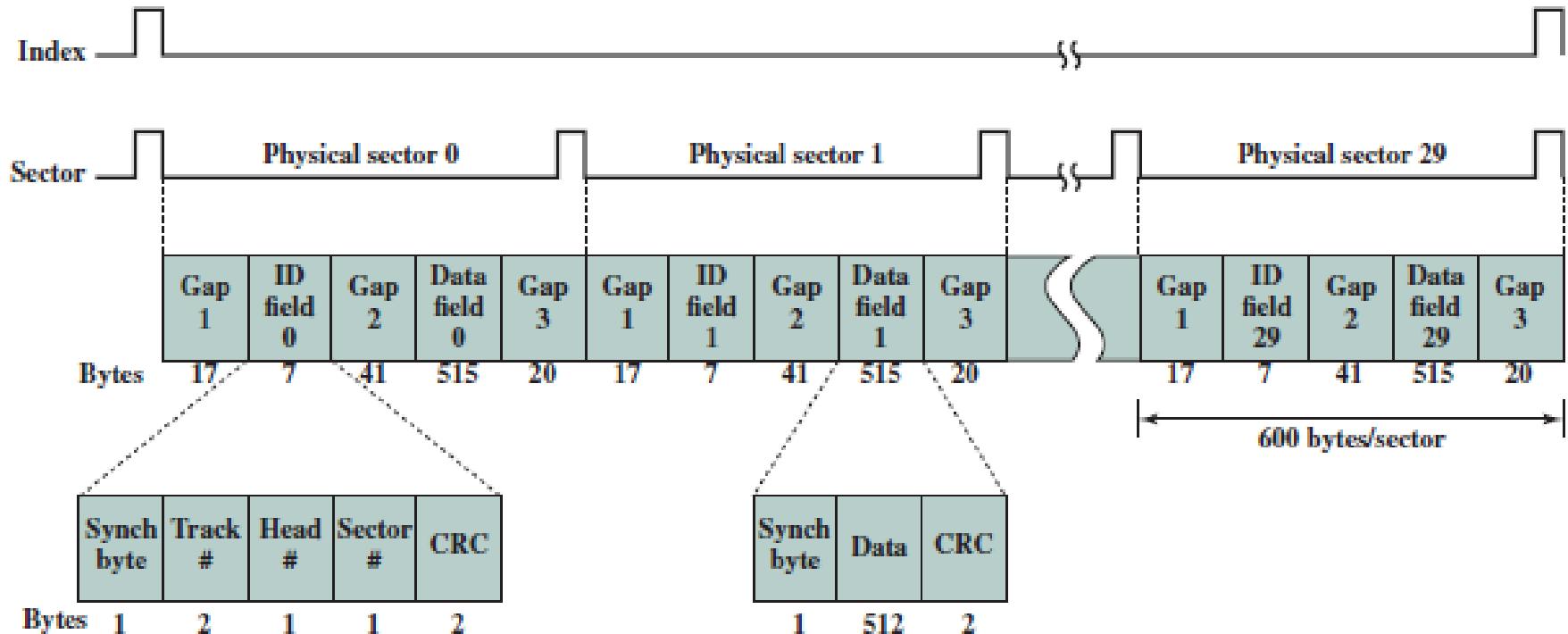


Figure 6.4 Winchester Disk Format (Seagate ST506)



Physical Characteristics of Disk Systems

Table 6.1 Physical Characteristics of Disk Systems

Head Motion	Platters
Fixed head (one per track)	Single platter
Movable head (one per surface)	Multiple platter
Disk Portability	Head Mechanism
Nonremovable disk	Contact (floppy)
Removable disk	Fixed gap
Sides	Aerodynamic gap (Winchester)
Single sided	
Double sided	

Characteristics

■ Fixed-head disk

- One read-write head per track
- Heads are mounted on a fixed ridged arm that extends across all tracks

■ Movable-head disk

- One read-write head
- Head is mounted on an arm
- The arm can be extended or retracted

■ Non-removable disk

- Permanently mounted in the disk drive
- The hard disk in a personal computer is a non-removable disk

■ Removable disk

- Can be removed and replaced with another disk
- Advantages:
 - Unlimited amounts of data are available with a limited number of disk systems
 - A disk may be moved from one computer system to another
- Floppy disks and ZIP cartridge disks are examples of removable disks

■ Double sided disk

- Magnetizable coating is applied to both sides of the platter





Multiple Platters

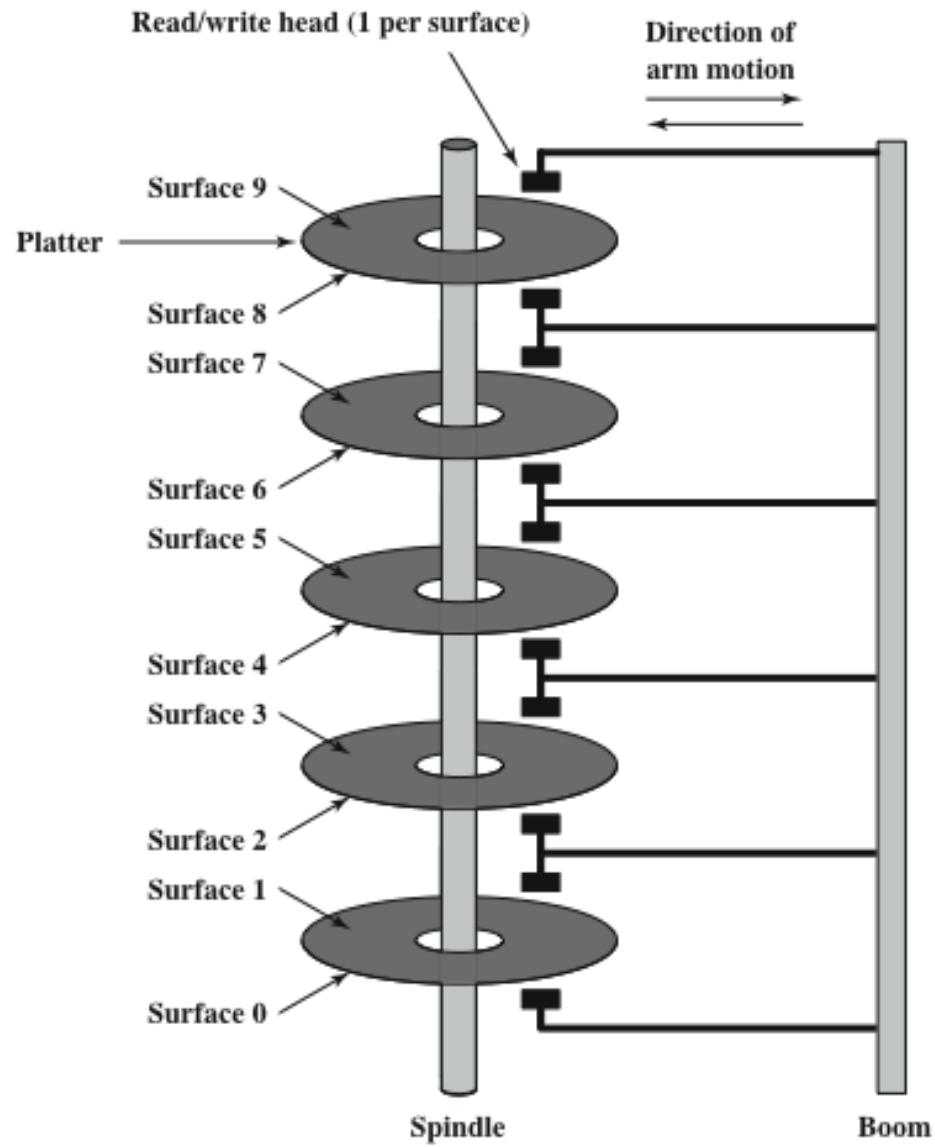
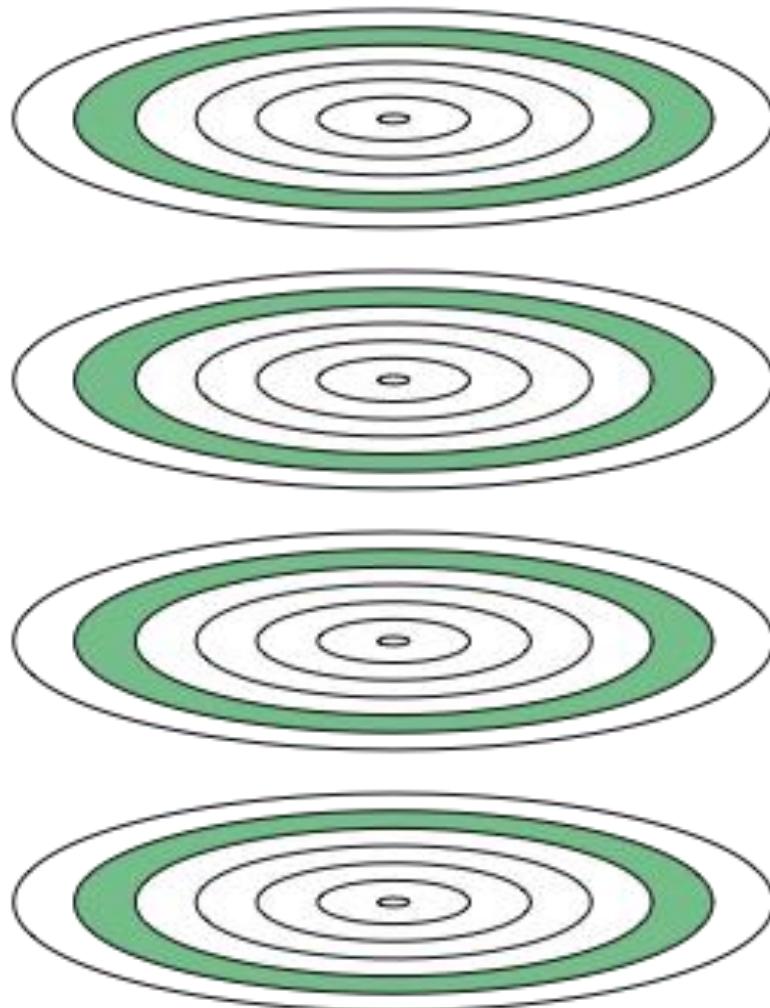


Figure 6.5 Components of a Disk Drive



Tracks

Cylinders

Figure 6.6 Tracks and Cylinders

Disk

Classification

The head mechanism provides a classification of disks into three types

- The head must generate or sense an electromagnetic field of sufficient magnitude to write and read properly
- The narrower the head, the closer it must be to the platter surface to function
 - A narrower head means narrower tracks and therefore greater data density
- The closer the head is to the disk the greater the risk of error from impurities or imperfections

Winchester Heads

- Used in sealed drive assemblies that are almost free of contaminants
- Designed to operate closer to the disk's surface than conventional rigid disk heads, thus allowing greater data density
- Is actually an **aerodynamic foil** that rests lightly on the platter's surface when the disk is motionless
 - The air pressure generated by a spinning disk is enough to make the foil rise above the surface

Typical Hard Disk Parameters

Table 6.2 Typical Hard Disk Drive Parameters

Characteristics	Seagate Enterprise	Seagate Barracuda XT	Seagate Cheetah NS	Seagate Laptop HDD
Application	Enterprise	Desktop	Network-attached storage, application servers	Laptop
Capacity	6 TB	3 TB	600 GB	2 TB
Average seek time	4.16 ms	N/A	3.9 ms read 4.2 ms write	13 ms
Spindle speed	7200 rpm	7200 rpm	10,075 rpm	5400 rpm
Average latency	4.16 ms	4.16 ms	2.98	5.6 ms
Maximum sustained transfer rate	216 MB/sec	149 MB/sec	97 MB/sec	300 MB/sec
Bytes per sector	512/4096	512	512	4096
Tracks per cylinder (number of platter surfaces)	8	10	8	4
Cache	128 MB	64 MB	16 MB	8 MB

Timing of Disk I/O Transfer

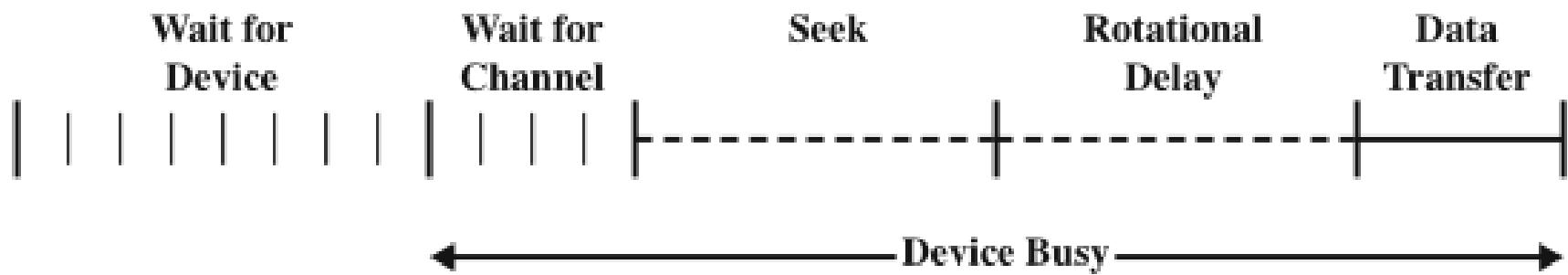


Figure 6.7 Timing of a Disk I/O Transfer

Disk Performance Parameters

- When the disk drive is operating the disk is rotating at constant speed
- To read or write the head must be positioned at the desired track and at the beginning of the desired sector on the track
 - Track selection involves moving the head in a movable-head system or electronically selecting one head on a fixed-head system
 - Once the track is selected, the disk controller waits until the appropriate sector rotates to line up with the head



Disk Performance Parameters

- **Seek time**
 - On a movable-head system, the time it takes to position the head at the track
- **Rotational delay (*rotational latency*)**
 - The time it takes for the beginning of the sector to reach the head
- **Access time**
 - The sum of the seek time and the rotational delay
 - The time it takes to get into position to read or write
- **Transfer time**
 - Once the head is in position, the read or write operation is then performed as the sector moves under the head
 - This is the data transfer portion of the operation



Disk Performance Parameters

$$T = \frac{b}{rN}$$

where

T = transfer time

b = number of bytes to be transferred

N = number of bytes on a track

r = rotation speed, in revolutions per second

Thus the total average read or write time T_{total} can be expressed as

$$T_{total} = T_s + \frac{1}{2r} + \frac{b}{rN}$$

where T_s is the average seek time.

Disk Performance Parameters

Consider a disk with an advertised average seek time of 4 ms, rotation speed of 15,000 rpm, and 512-byte sectors with 500 sectors per track. Suppose that we wish to read a file consisting of 2500 sectors for a total of 1.28 Mbytes. Calculate the total time for the transfer for both Sequential access and Random access

Answer:

0.034 seconds for Sequential access

15.02 seconds for Random access



RAID

Redundant Array of
Independent Disks

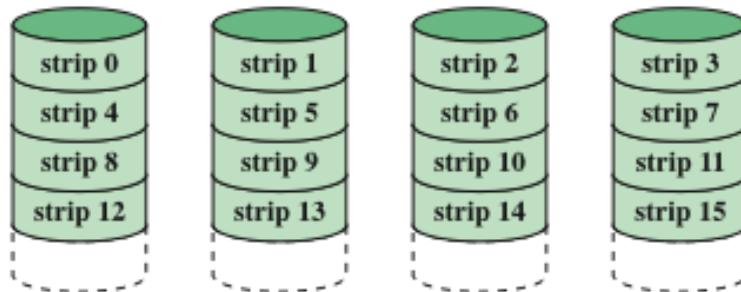
- Consists of 7 levels
- Levels do not imply a hierarchical relationship but designate different design architectures that share three common characteristics:
 - 1) Set of physical disk drives viewed by the operating system as a single logical drive
 - 2) Data are distributed across the physical drives of an array in a scheme known as **striping**
 - 3) Redundant disk capacity is used to store parity information, which guarantees data recoverability in case of a disk failure

Table 6.3 RAID Levels

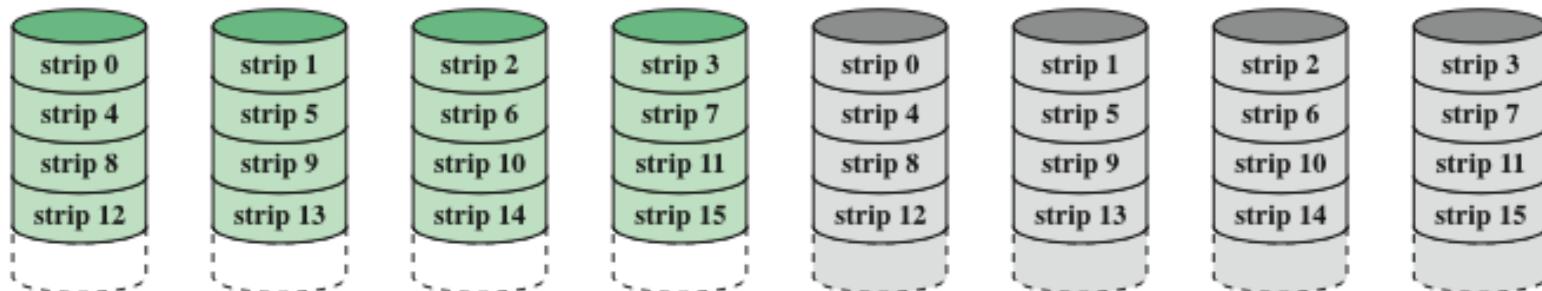
Category	Level	Description	Disks Required	Data Availability	Large I/O Data Transfer Capacity	Small I/O Request Rate
Striping	0	Nonredundant	N	Lower than single disk	Very high	Very high for both read and write
Mirroring	1	Mirrored	$2N$	Higher than RAID 2, 3, 4, or 5; lower than RAID 6	Higher than single disk for read; similar to single disk for write	Up to twice that of a single disk for read; similar to single disk for write
Parallel access	2	Redundant via Hamming code	$N + m$	Much higher than single disk; comparable to RAID 3, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
	3	Bit-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
Independent access	4	Block-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 5	Similar to RAID 0 for read; significantly lower than single disk for write	Similar to RAID 0 for read; significantly lower than single disk for write
	5	Block-interleaved distributed parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 4	Similar to RAID 0 for read; lower than single disk for write	Similar to RAID 0 for read; generally lower than single disk for write
	6	Block-interleaved dual distributed parity	$N + 2$	Highest of all listed alternatives	Similar to RAID 0 for read; lower than RAID 5 for write	Similar to RAID 0 for read; significantly lower than RAID 5 for write

N = number of data disks; m proportional to $\log N$

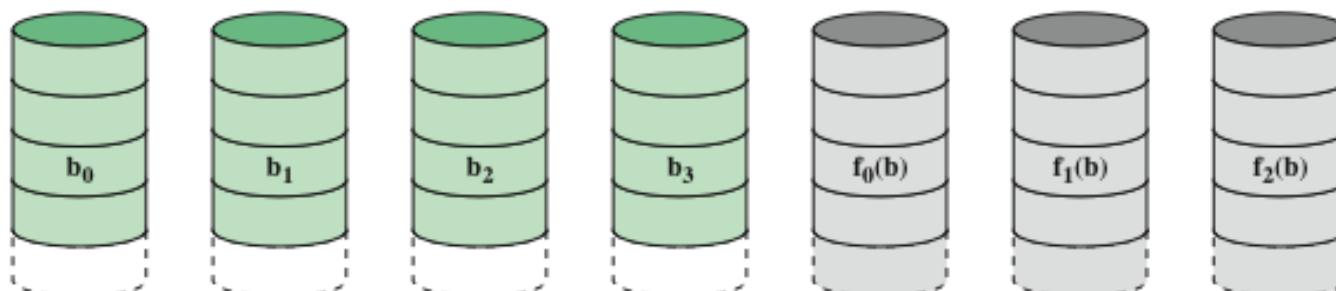
RAID Levels 0, 1, 2



(a) RAID 0 (non-redundant)

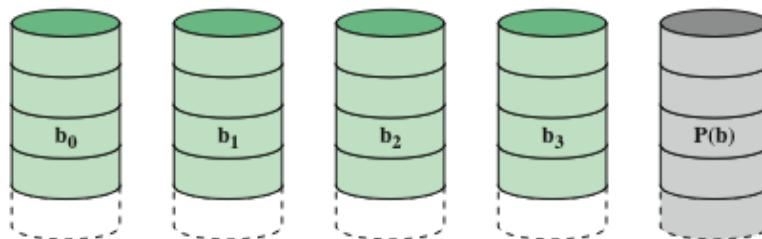


(b) RAID 1 (mirrored)

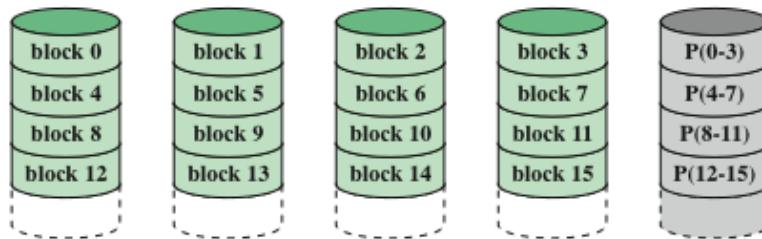


(c) RAID 2 (redundancy through Hamming code)

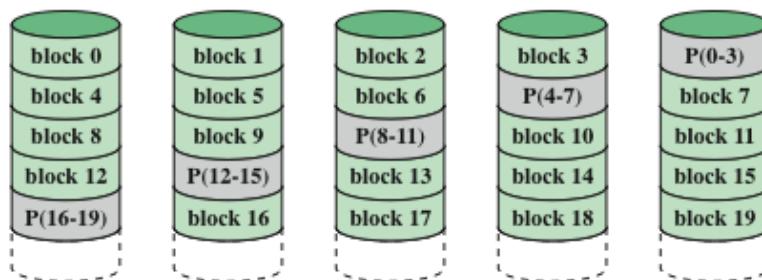
Figure 6.8 RAID Levels (page 1 of 2)



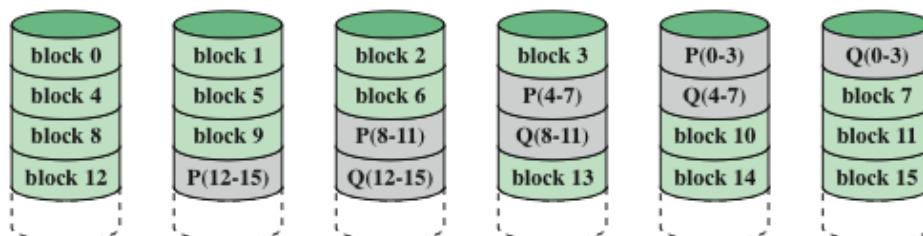
(d) RAID 3 (bit-interleaved parity)



(e) RAID 4 (block-level parity)



(f) RAID 5 (block-level distributed parity)



(g) RAID 6 (dual redundancy)

RAID Levels

3, 4, 5, 6

Figure 6.8 RAID Levels (page 2 of 2)

Data Mapping for a RAID Level 0 Array

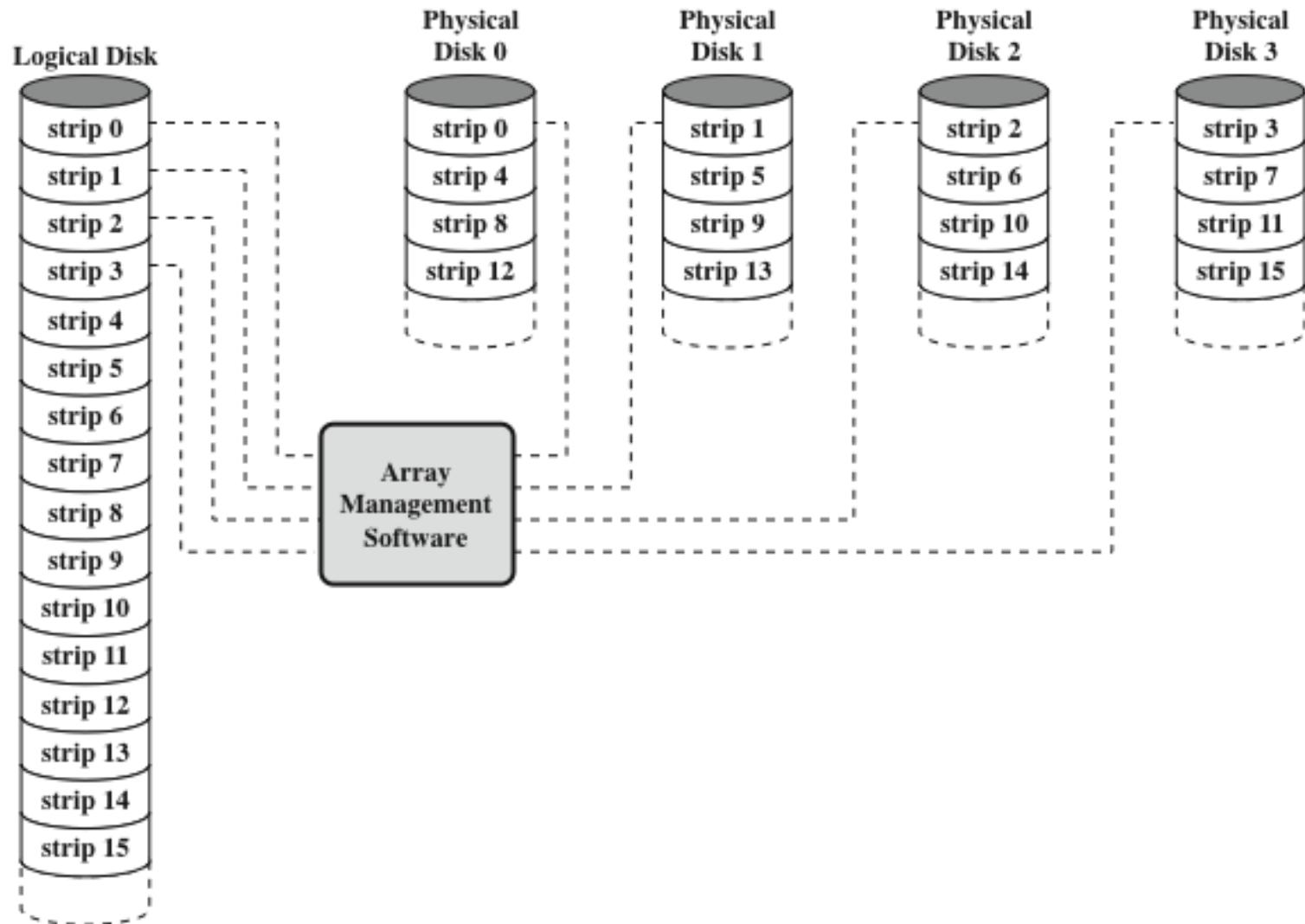


Figure 6.9 Data Mapping for a RAID Level 0 Array

+ RAID Level 0

RAID 0 for High Data Transfer Capacity

- For applications to experience a high transfer rate two requirements must be met:
 1. A high transfer capacity must exist along the entire path between host memory and the individual disk drives
 2. The application must make I/O requests that drive the disk array efficiently

RAID 0 for High I/O Request Rate

- For an individual I/O request for a small amount of data the I/O time is dominated by the seek time and rotational latency
- A disk array can provide high I/O execution rates by balancing the I/O load across multiple disks
- If the strip size is relatively large multiple waiting I/O requests can be handled in parallel, reducing the queuing time for each request



RAID

Level 1

Characteristics

- Differs from RAID levels 2 through 6 in the way in which redundancy is achieved
- Redundancy is achieved by the simple expedient of duplicating all the data
- **Data striping is used** but each logical strip is mapped to two separate physical disks so that every disk in the array has a mirror disk that contains the same data
- **RAID 1 can also be implemented without data striping, although this is less common**

Positive Aspects

- A read request can be serviced by either of the two disks that contains the requested data
- There is no “write penalty”
- Recovery from a failure is simple, when a drive fails the data can be accessed from the second drive
- Provides real-time copy of all data
- Can achieve high I/O request rates if the bulk of the requests are reads
- Principal disadvantage is the cost

RAID Level 2

Characteristics

- Makes use of a parallel access technique
- In a parallel access array all member disks participate in the execution of every I/O request
- Spindles of the individual drives are synchronized so that each disk head is in the same position on each disk at any given time
- **Data striping is used**
 - Strips are very small, often as small as a single byte or word

Performance

- An error-correcting code is calculated across corresponding bits on each data disk and the bits of the code are stored in the corresponding bit positions on multiple parity disks
- Typically a Hamming code is used, which is able to correct single-bit errors and detect double-bit errors
- The number of redundant disks is proportional to the log of the number of data disks
- Would only be an effective choice in an environment in which many disk errors occur



RAID Level 3

Redundancy

- Requires only a single redundant disk, no matter how large the disk array
- Employs parallel access, with **data distributed in small strips**
- Instead of an error correcting code, a simple parity bit is computed for the set of individual bits in the same position on all of the data disks
- Can achieve very high data transfer rates

Performance

- In the event of a drive failure, the parity drive is accessed and data is reconstructed from the remaining devices
- Once the failed drive is replaced, the missing data can be restored on the new drive and operation resumed
- In the event of a disk failure, all of the data are still available in what is referred to as *reduced mode*
- Return to full operation requires that the failed disk be replaced and the entire contents of the failed disk be regenerated on the new disk
- In a transaction-oriented environment performance suffers



RAID Level 4

Characteristics

- Makes use of an independent access technique
 - In an independent access array, each member disk operates independently so that separate I/O requests can be satisfied in parallel
- **Data striping is used**
 - Strips are relatively large
- To calculate the new parity the array management software must read the old user strip and the old parity strip

Performance

- Involves a write penalty when an I/O write request of small size is performed
- Each time a write occurs the array management software must update the user data the corresponding parity bits
- Thus each strip write involves two reads and two writes

RAID Level 5

Characteristics

- Organized in a similar fashion to RAID 4
- Difference is distribution of the parity strips across all disks
- A typical allocation is a round-robin scheme
- The distribution of parity strips across all drives avoids the potential I/O bottleneck found in RAID 4

RAID Level 6

Characteristics

- Two different parity calculations are carried out and stored in separate blocks on different disks
- Advantage is that it provides extremely high data availability
- Three disks would have to fail within the mean time to repair (MTTR) interval to cause data to be lost
- Incurs a substantial write penalty because each write affects two parity blocks

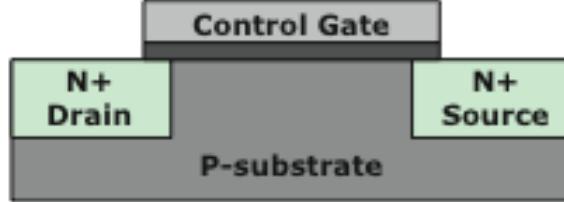
Table 6.4
RAID
Comparison
(page 1 of 2)

Level	Advantages	Disadvantages	Applications
0	I/O performance is greatly improved by spreading the I/O load across many channels and drives No parity calculation overhead is involved Very simple design Easy to implement	The failure of just one drive will result in all data in an array being lost	Video production and Editing Image editing Pre-press applications Any application requiring high bandwidth
1	100% redundancy of data means no rebuild is necessary in case of a disk failure, just a copy to the replacement disk Under certain circumstances, RAID 1 can sustain multiple simultaneous drive failures Simplest RAID storage subsystem design	Highest disk overhead of all RAID types (100%) - inefficient	Accounting Payroll Financial Any application requiring very high availability
2	Extremely high data transfer rates possible The higher the data transfer rate required, the better the ratio of data disks to ECC disks Relatively simple controller design compared to RAID levels 3,4 & 5	Very high ratio of ECC disks to data disks with smaller word sizes - inefficient Entry level cost very high - requires very high transfer rate requirement to justify	No commercial implementations exist / not commercially viable

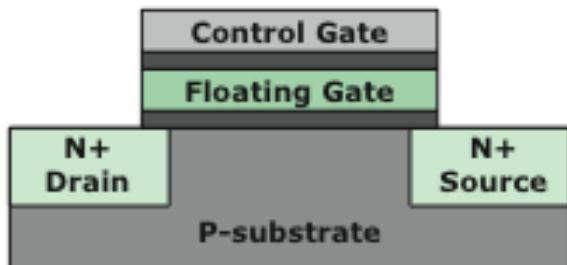


Level	Advantages	Disadvantages	Applications
3	Very high read data transfer rate Very high write data transfer rate Disk failure has an insignificant impact on throughput Low ratio of ECC (parity) disks to data disks means high efficiency	Transaction rate equal to that of a single disk drive at best (if spindles are synchronized) Controller design is fairly complex	Video production and live streaming Image editing Video editing Prepress applications Any application requiring high throughput
4	Very high Read data transaction rate Low ratio of ECC (parity) disks to data disks means high efficiency	Quite complex controller design Worst write transaction rate and Write aggregate transfer rate Difficult and inefficient data rebuild in the event of disk failure	No commercial implementations exist / not commercially viable
5	Highest Read data transaction rate Low ratio of ECC (parity) disks to data disks means high efficiency Good aggregate transfer rate	Most complex controller design Difficult to rebuild in the event of a disk failure (as compared to RAID level 1)	File and application servers Database servers Web, e-mail, and news servers Intranet servers Most versatile RAID level
6	Provides for an extremely high data fault tolerance and can sustain multiple simultaneous drive failures	More complex controller design Controller overhead to compute parity addresses is extremely high	Perfect solution for mission critical applications

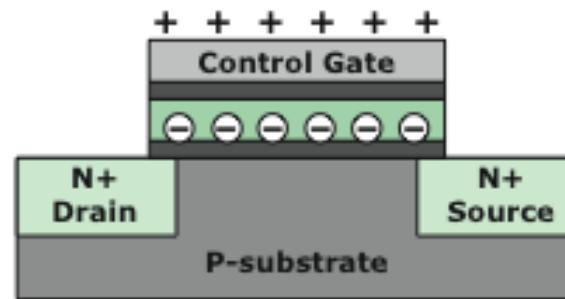
Table 6.4
RAID Comparison
(page 2 of 2)



(a) Transistor structure



(b) Flash memory cell in one state



(c) Flash memory cell in zero state

Flash

Memory

+

Figure 6.10
Flash Memory Operation

Solid State Drive (SSD)

A memory device made with solid state components that can be used as a replacement to a hard disk drive (HDD)

The term *solid state* refers to electronic circuitry built with semiconductors

Flash memory

A type of semiconductor memory used in many consumer electronic products including smart phones, GPS devices, MP3 players, digital cameras, and USB devices

Cost and performance has evolved to the point where it is feasible to use to replace HDDs

Two distinctive types of flash memory:

NOR

- The basic unit of access is a bit
- Provides high-speed random access
- Used to store **cell phone operating system code** and on Windows computers for the **BIOS program** that runs at start-up

NAND

- The basic unit is 16 or 32 bits
- Reads and writes in small blocks
- Used in **USB flash drives, memory cards, and in SSDs**
- Does not provide a random-access external address bus so the data must be read on a block-wise basis



SSD Organization

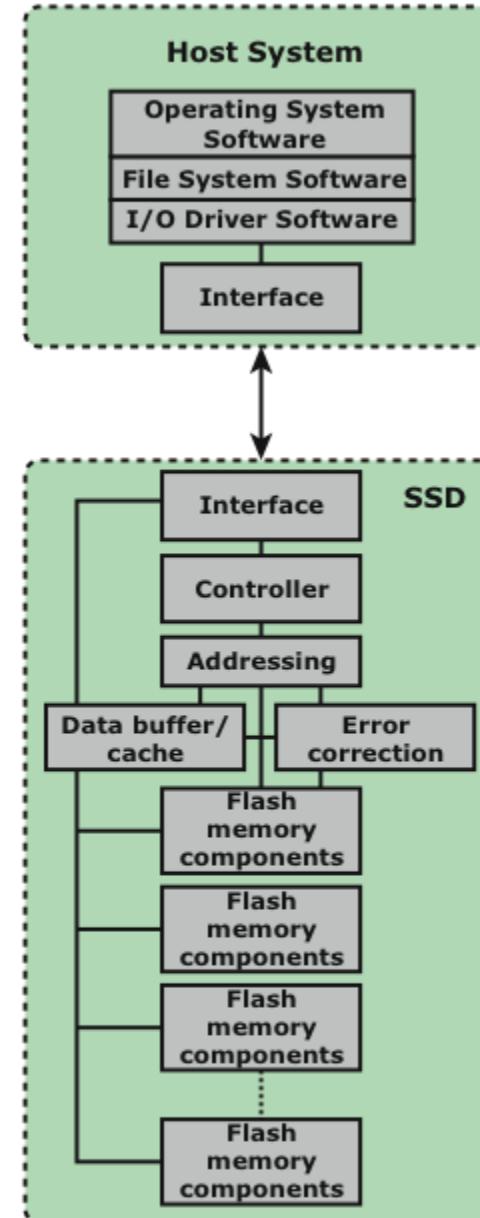


Figure 6.11 Solid State Drive Architecture

SSD Compared to HDD

SSDs have the following advantages over HDDs:

- High-performance input/output operations per second (IOPS)
- Durability
- Longer lifespan
- Lower power consumption
- Quieter and cooler running capabilities
- Lower access times and latency rates

Table
6.5

Comparisons

	NAND Flash Drives	Disk Drives
I/O per second (sustained)	Read: 45,000 Write: 15,000	300
Throughput (MB/s)	Read: 200+ Write: 100+	up to 80
Random access time (ms)	0.1	4–10
Storage capacity	up to 256 GB	up to 4 TB

+ Practical Issues

There are two practical issues peculiar to SSDs that are not faced by HDDs:

- SSD performance has a tendency to slow down as the device is used
- The entire block must be read from the flash memory and placed in a RAM buffer
- Before the block can be written back to flash memory, the entire block of flash memory must be erased
- The entire block from the buffer is now written back to the flash memory
- Flash memory becomes unusable after a certain number of writes
- Techniques for prolonging life:
 - Front-ending the flash with a cache to delay and group write operations
 - Using wear-leveling algorithms that evenly distribute writes across block of cells
 - Bad-block management techniques
- Most flash devices estimate their own remaining lifetimes so systems can anticipate failure and take preemptive action



CD

Compact Disk. A nonerasable disk that stores digitized audio information. The standard system uses 12-cm disks and can record more than 60 minutes of uninterrupted playing time.

CD-ROM

Compact Disk Read-Only Memory. A nonerasable disk used for storing computer data. The standard system uses 12-cm disks and can hold more than 650 Mbytes.

CD-R

CD Recordable. Similar to a CD-ROM. The user can write to the disk only once.

CD-RW

CD Rewritable. Similar to a CD-ROM. The user can erase and rewrite to the disk multiple times.

DVD

Digital Versatile Disk. A technology for producing digitized, compressed representation of video information, as well as large volumes of other digital data. Both 8 and 12 cm diameters are used, with a double-sided capacity of up to 17 Gbytes. The basic DVD is read-only (DVD-ROM).

DVD-R

DVD Recordable. Similar to a DVD-ROM. The user can write to the disk only once. Only one-sided disks can be used.

DVD-RW

DVD Rewritable. Similar to a DVD-ROM. The user can erase and rewrite to the disk multiple times. Only one-sided disks can be used.

Blu-Ray DVD

High definition video disk. Provides considerably greater data storage density than DVD, using a 405-nm (blue-violet) laser. A single layer on a single side can store 25 Gbytes.

Table 6.6 Optical Disk Products



Compact Disk Read-Only Memory (CD-ROM)



- Audio CD and the CD-ROM share a similar technology
 - The main difference is that CD-ROM players are more rugged and have error correction devices to ensure that data are properly transferred
- Production:
 - The disk is formed from a resin such as polycarbonate
 - Digitally recorded information is imprinted as a series of microscopic pits on the surface of the polycarbonate
 - This is done with a finely focused, high intensity laser to create a master disk
 - The master is used, in turn, to make a die to stamp out copies onto polycarbonate
 - The pitted surface is then coated with a highly reflective surface, usually aluminum or gold
 - This shiny surface is protected against dust and scratches by a top coat of clear acrylic
 - Finally a label can be silkscreened onto the acrylic



CD Operation

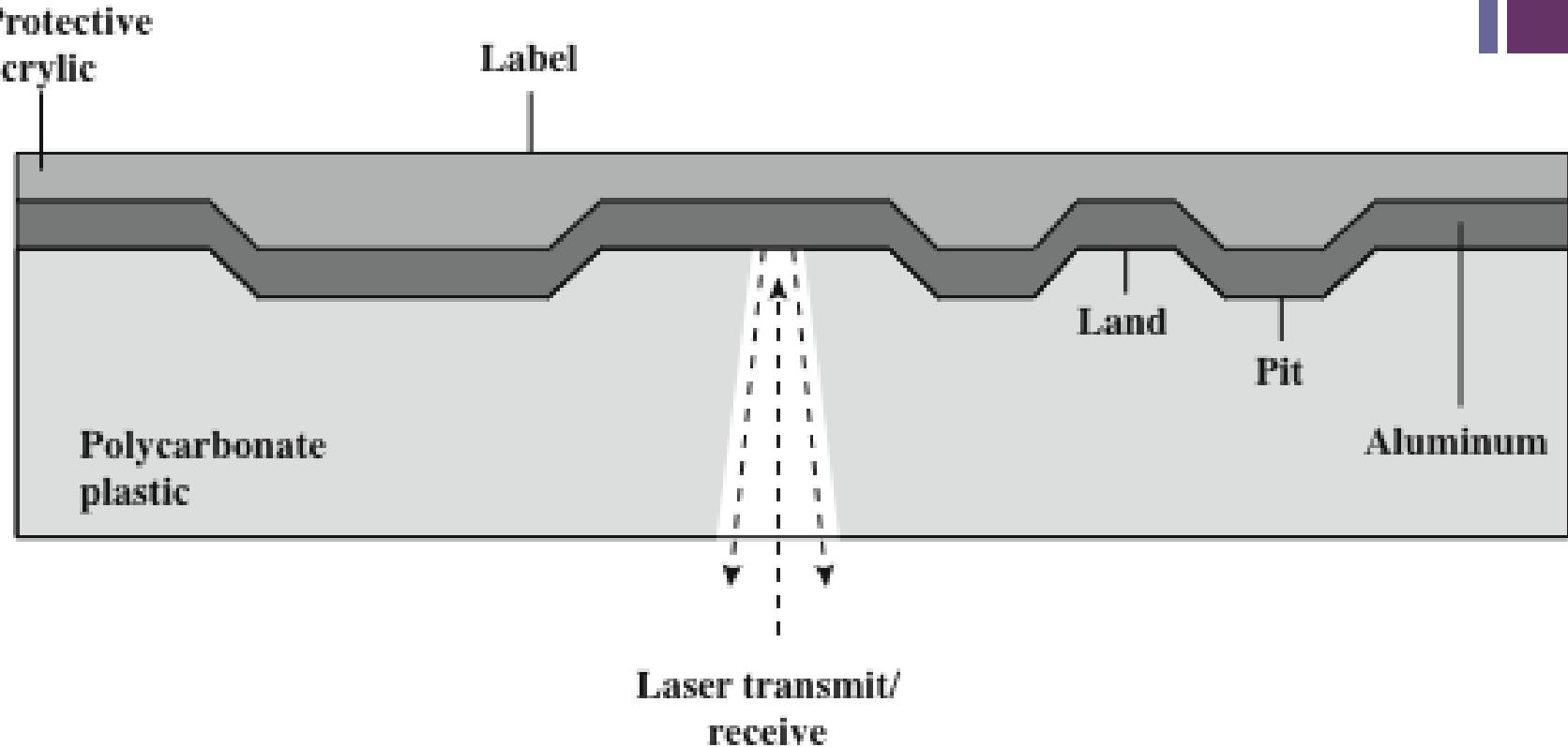
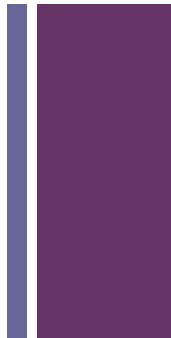


Figure 6.12 CD Operation



CD-ROM Block Format

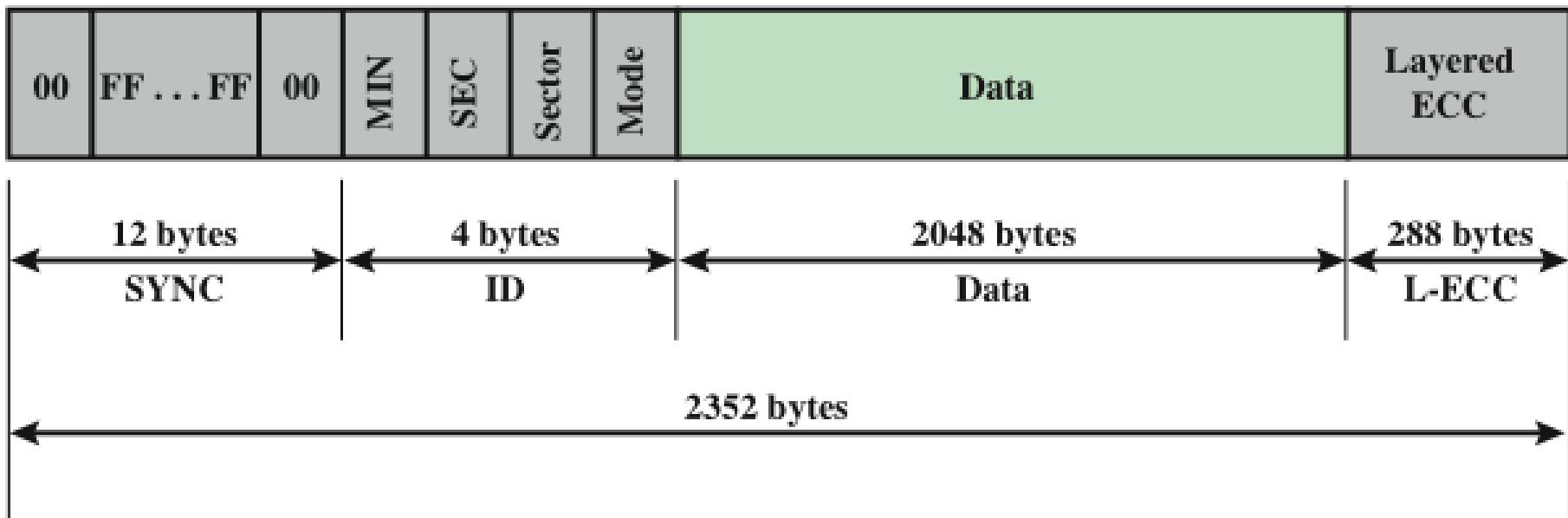
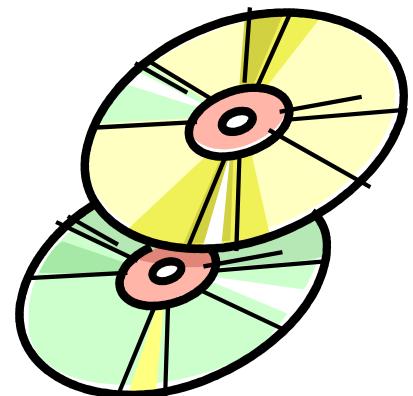


Figure 6.13 CD-ROM Block Format

+

- CD-ROM is appropriate for the distribution of large amounts of data to a large number of users
- Because the expense of the initial writing process it is not appropriate for individualized applications
- The CD-ROM has two advantages:
 - The optical disk together with the information stored on it can be mass replicated inexpensively
 - The optical disk is removable, allowing the disk itself to be used for archival storage
- The CD-ROM disadvantages:
 - It is read-only and cannot be updated
 - It has an access time much longer than that of a magnetic disk drive

CD-ROM





CD Recordable (CD-R)

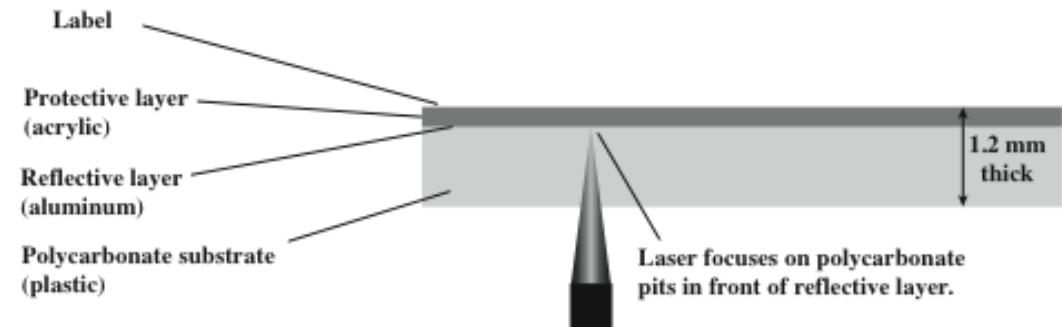
- Write-once read-many
- Accommodates applications in which only one or a small number of copies of a set of data is needed
- Disk is prepared in such a way that it can be subsequently written once with a laser beam of modest-intensity
- Medium includes a dye layer which is used to change reflectivity and is activated by a high-intensity laser
- Provides a permanent record of large volumes of user data

CD Rewritable (CD-RW)

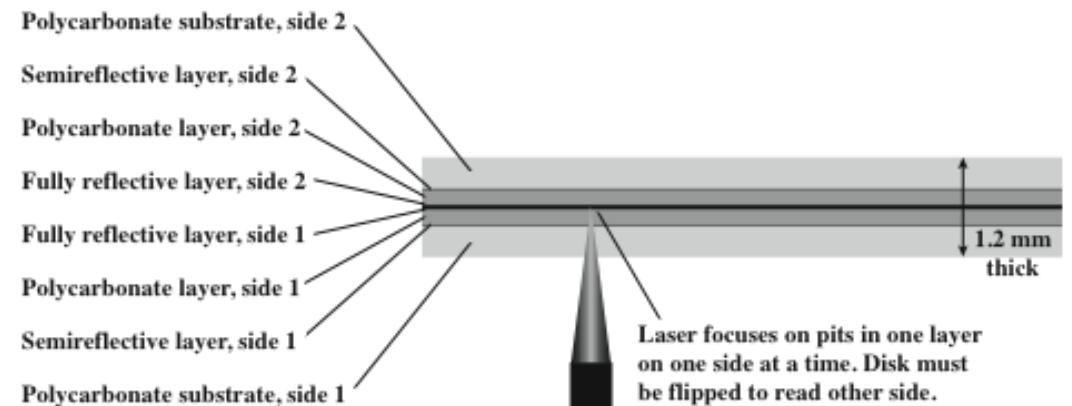
- Can be repeatedly written and overwritten
- Phase change disk uses a material that has two significantly different reflectivities in two different phase states
 - Amorphous state
 - Molecules exhibit a random orientation that reflects light poorly
 - Crystalline state
 - Has a smooth surface that reflects light well
- A beam of laser light can change the material from one phase to the other
- Disadvantage is that the material eventually and permanently loses its desirable properties
- Advantage is that it can be rewritten



Digital Versatile Disk (DVD)



(a) CD-ROM - Capacity 682 MB



(b) DVD-ROM, double-sided, dual-layer - Capacity 17 GB

Figure 6.14 CD-ROM and DVD-ROM

High-Definition Optical Disks

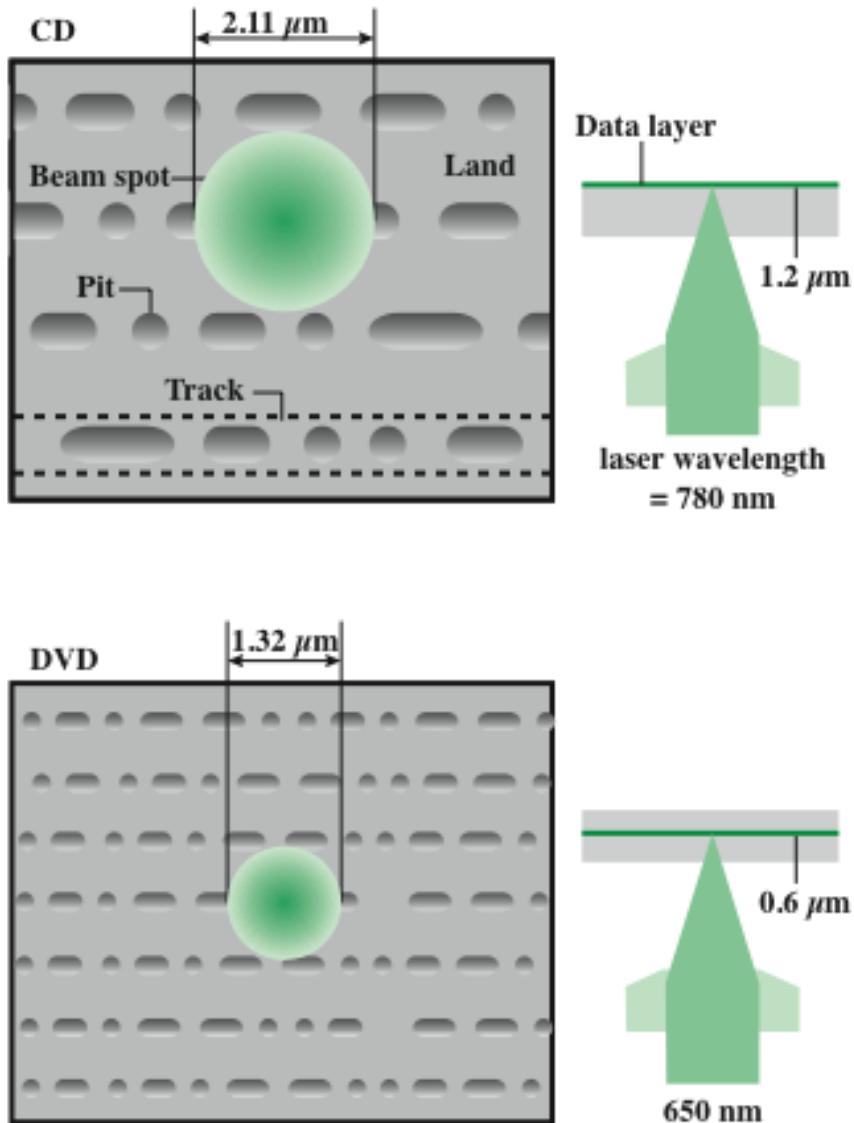
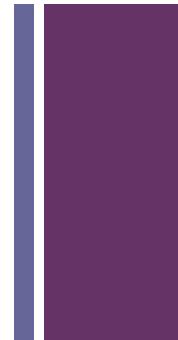


Figure 6.15 Optical Memory Characteristics

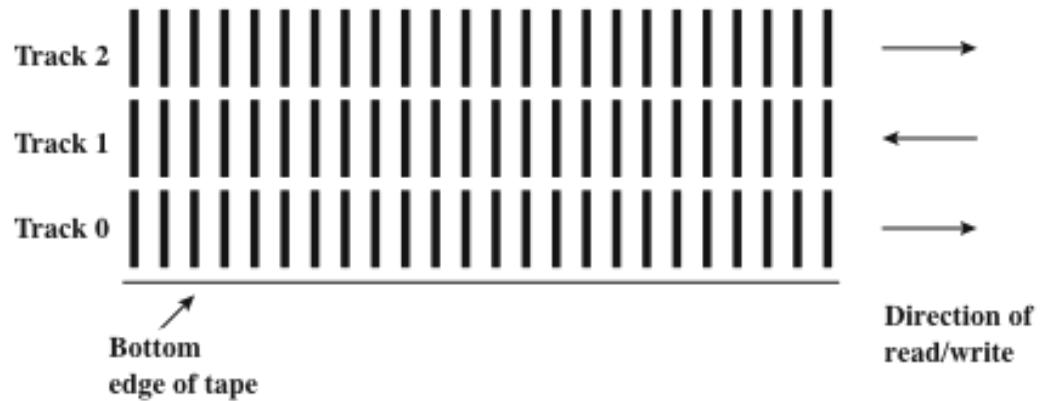
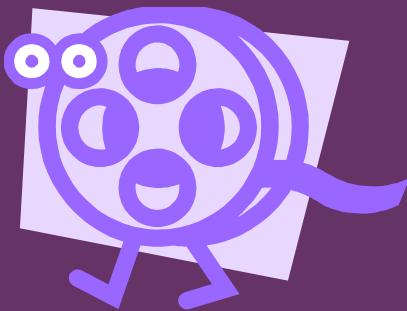


Magnetic Tape

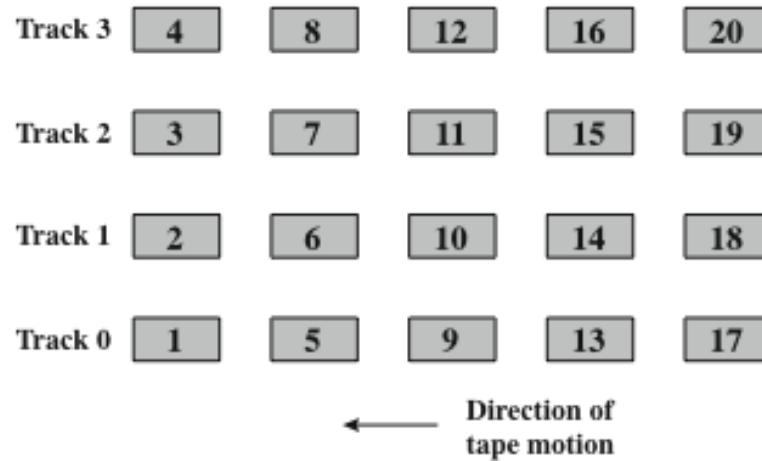


- Tape systems use the same reading and recording techniques as disk systems
- Medium is flexible polyester tape coated with magnetizable material
- Coating may consist of particles of pure metal in special binders or vapor-plated metal films
- Data on the tape are structured as a number of parallel tracks running lengthwise
- Serial recording
 - Data are laid out as a sequence of bits along each track
- Data are read and written in contiguous blocks called *physical records*
- Blocks on the tape are separated by gaps referred to as *inter-record gaps*

Magnetic Tape Features



(a) Serpentine reading and writing



(b) Block layout for system that reads/writes four tracks simultaneously

Figure 6.16 Typical Magnetic Tape Features

+

Table 6.7

LTO Tape Drives

	LTO-1	LTO-2	LTO-3	LTO-4	LTO-5	LTO-6	LTO-7	LTO-8
Release date	2000	2003	2005	2007	2010	TBA	TBA	TBA
Compressed capacity	200 GB	400 GB	800 GB	1600 GB	3.2 TB	8 TB	16 TB	32 TB
Compressed transfer rate (MB/s)	40 MB/s	80 MB/s	160 MB/s	240 MB/s	280 MB/s	525 MB/s	788 MB/s	1.18 GB/s
Linear density (bits/mm)	4880	7398	9638	13250	15142			
Tape tracks	384	512	704	896	1280			
Tape length	609 m	609 m	680 m	820 m	846 m			
Tape width (cm)	1.27	1.27	1.27	1.27	1.27			
Write elements	8	8	16	16	16			
WORM?	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Encryption Capable?	No	No	No	Yes	Yes	Yes	Yes	Yes
Partitioning?	No	No	No	No	No	Yes	Yes	Yes

+

Summary

Chapter 6

- Magnetic disk
 - Magnetic read and write mechanisms
 - Data organization and formatting
 - Physical characteristics
 - Disk performance parameters
- Solid state drives
 - Flash memory
 - SSD compared to HDD
 - SSD organization
 - Practical issues
- Magnetic tape

External Memory

- RAID
 - RAID level 0
 - RAID level 1
 - RAID level 2
 - RAID level 3
 - RAID level 4
 - RAID level 5
 - RAID level 6
- Optical memory
 - Compact disk
 - Digital versatile disk
 - High-definition optical disks