



**William Stallings
Computer Organization
and Architecture
10th Edition**



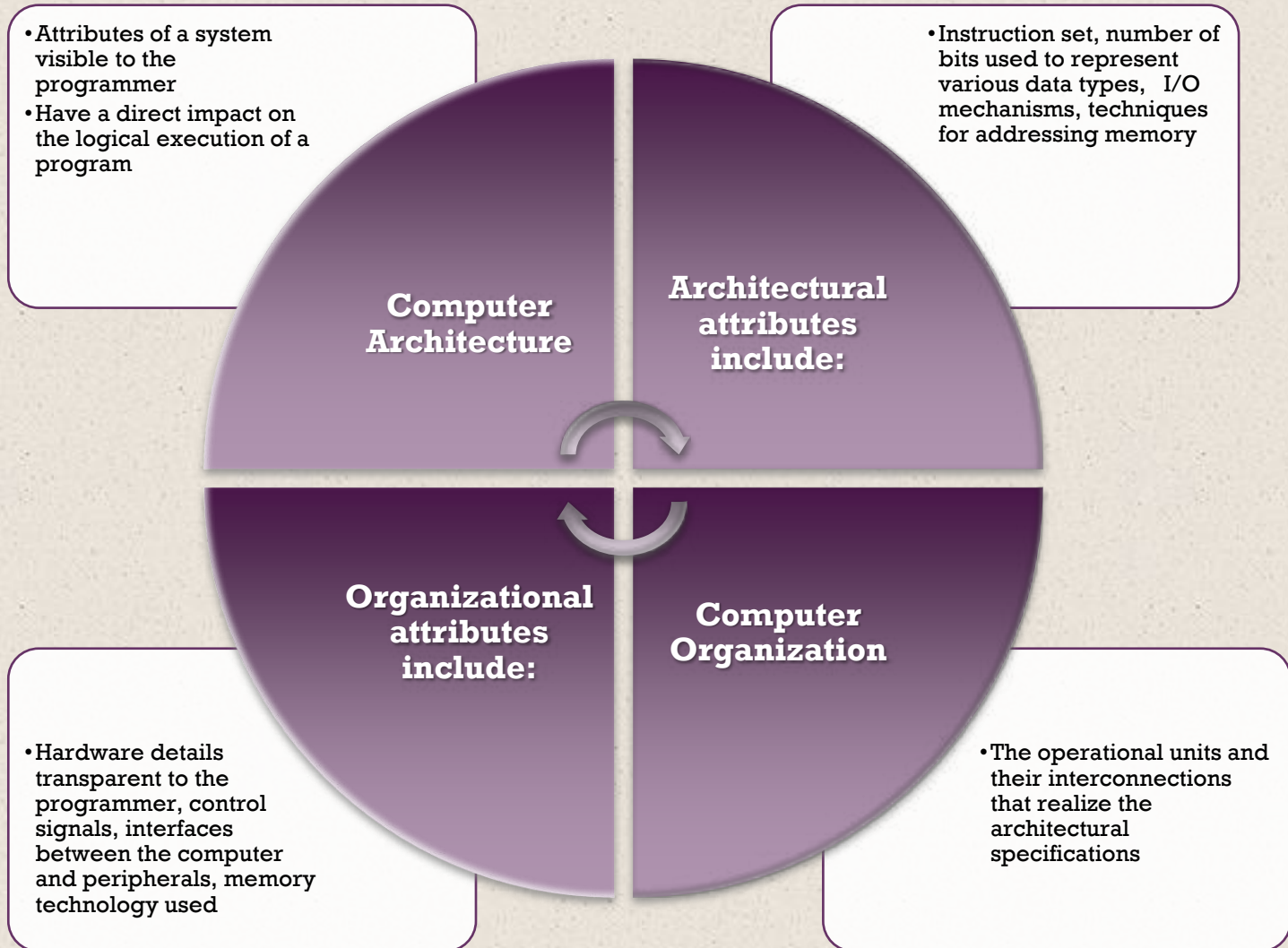
Chapter 1

+

Basic Concepts and Computer Evolution

Computer Architecture

Computer Organization



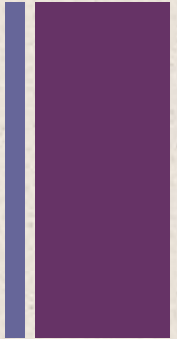


IBM System 370 Architecture

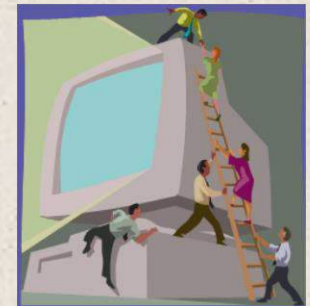
- IBM System/370 architecture
 - Was introduced in 1970
 - Included a number of models
 - New models are introduced with improved technology, but retain the same architecture so that the customer's software investment is protected
 - Architecture has survived to this day as the architecture of IBM's mainframe product line



+ 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



+ Function

- There are four basic functions that a computer can perform:
 - Data processing
 - Processor
 - Data storage
 - Short-term
 - Long-term
 - Data movement
 - Input-output (I/O) - when data are received from or delivered to a device (peripheral) that is directly connected to the computer
 - Data communications – when data are moved over longer distances, to or from a remote device (e-mail)
 - Control
 - A control unit manages the computer's resources and manages the performance of its functional parts in response to instructions

Structure

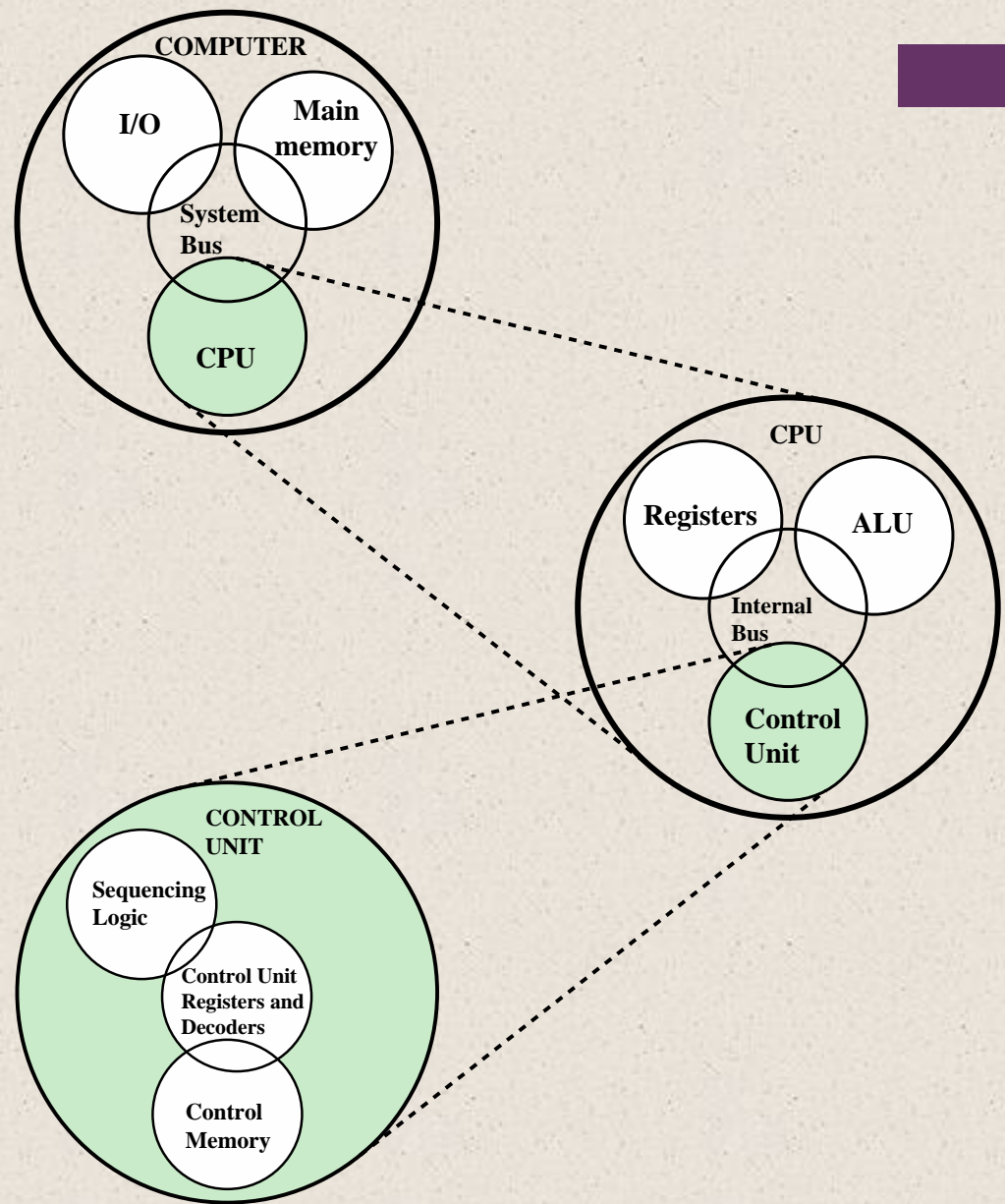
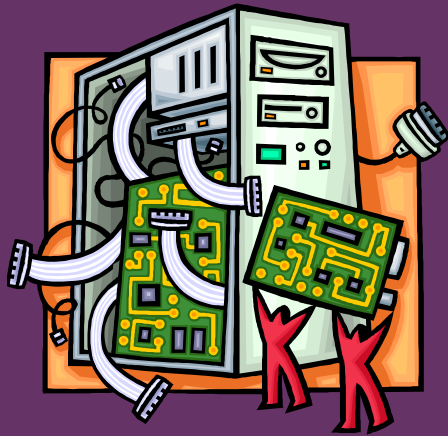


Figure 1.1 A Top-Down View of a Computer



There are four
main structural
components
of the computer:

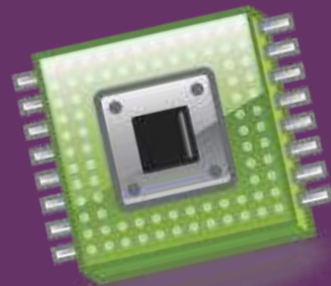
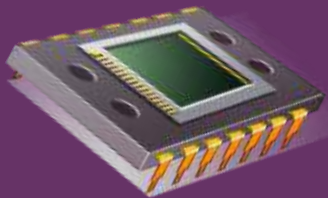


- ✦ CPU – controls the operation of the computer and performs its data processing functions
- ✦ Main Memory – stores data
- ✦ I/O – moves data between the computer and its external environment
- ✦ System Interconnection – some mechanism that provides for communication among CPU, main memory, and I/O



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

+ Multicore Computer Structure

- Central processing unit (CPU)
 - Portion of the computer that fetches and executes instructions
 - Consists of an ALU, a control unit, and registers
 - Referred to as a processor in a system with a **single** processing unit
- Core
 - An individual processing unit on a processor chip
 - May be equivalent in functionality to a CPU on a single-CPU system
 - Specialized processing units are also referred to as cores
- Processor
 - A physical piece of silicon containing one or more cores
 - Is the computer component that interprets and executes instructions
 - Referred to as a *multicore processor* if it contains multiple cores

+ Cache Memory



- Multiple layers of memory between the processor and main memory
- Is smaller and faster than main memory
- Used to speed up memory access by placing in the cache data from main memory that is likely to be used in the near future
- A greater performance improvement may be obtained by using multiple levels of cache, with level 1 (L1) closest to the core and additional levels (L2, L3, etc.) progressively farther from the core

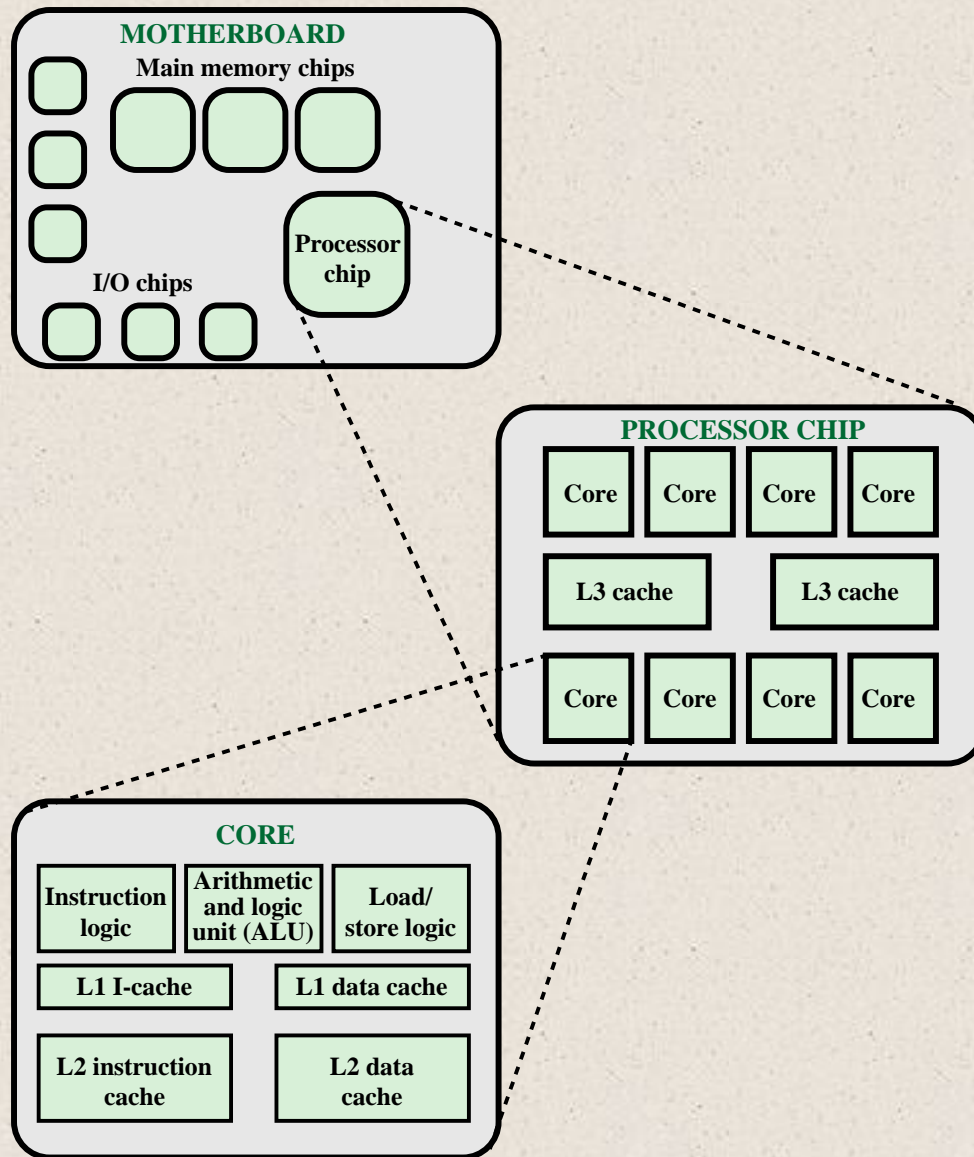
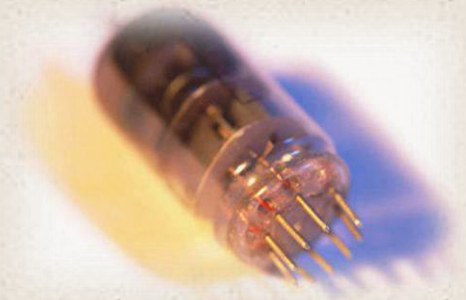


Figure 1.2 Simplified View of Major Elements of a Multicore Computer

+ History of Computers

First Generation: Vacuum Tubes

- Vacuum tubes were used for digital logic elements and memory
- IAS computer
 - Fundamental design approach was the stored program concept
 - Attributed to the mathematician John von Neumann
 - First publication of the idea was in 1945 for the EDVAC
 - Design began at the Princeton Institute for Advanced Studies
 - Completed in 1952
 - Prototype of all subsequent general-purpose computers



+ First Generation: Vacuum Tubes



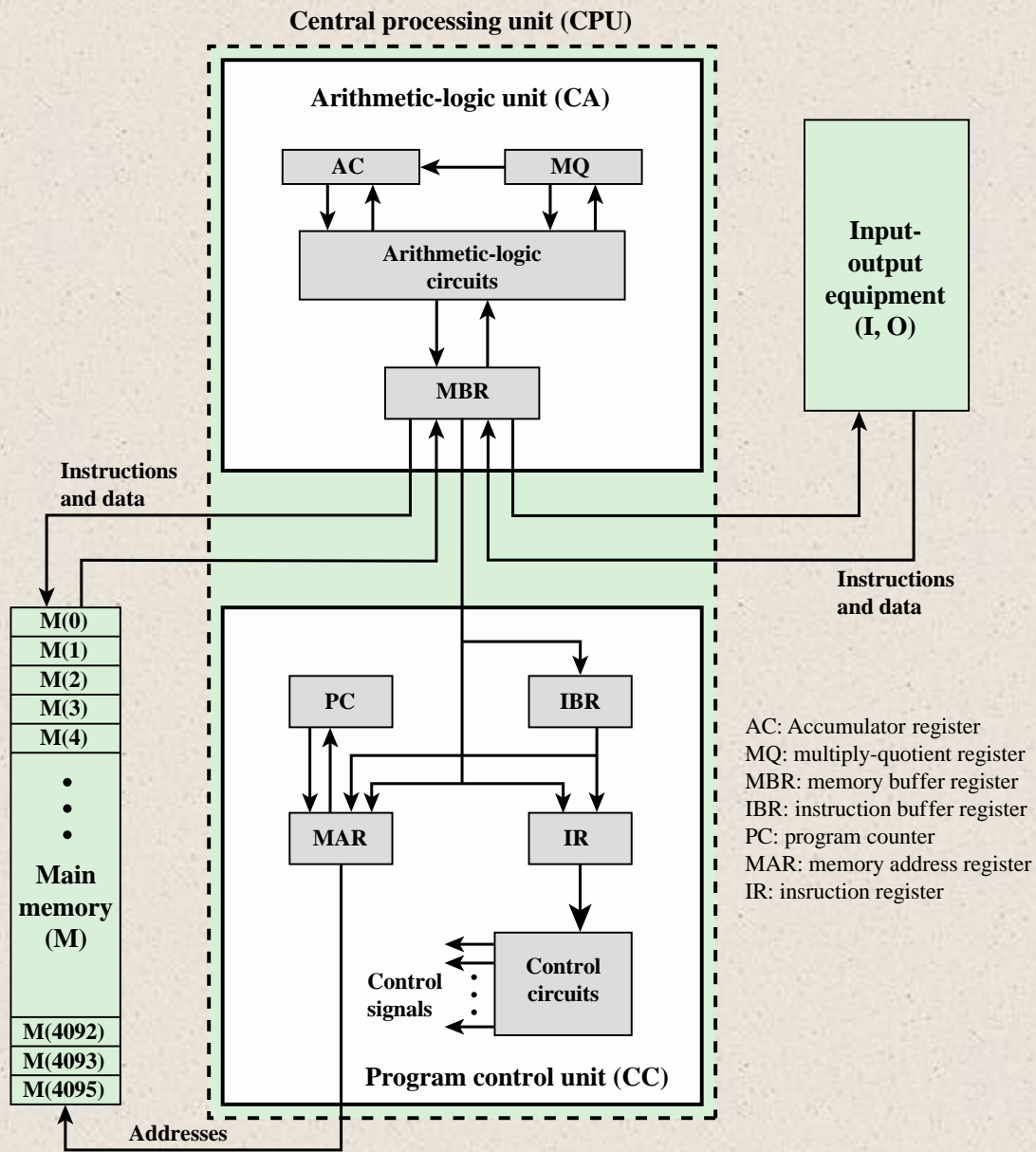


Figure 1.6 IAS Structure

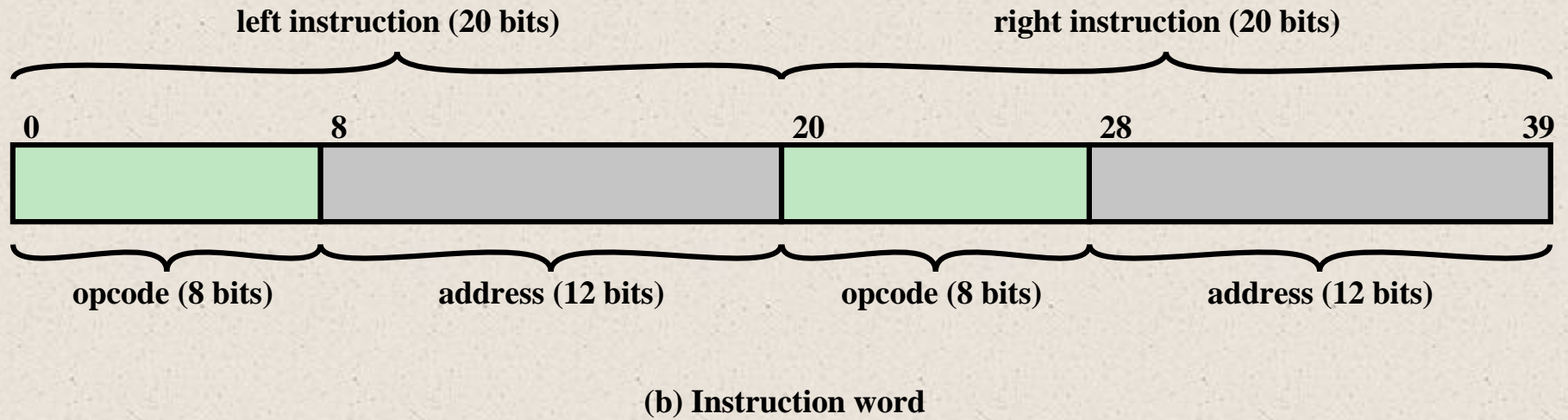
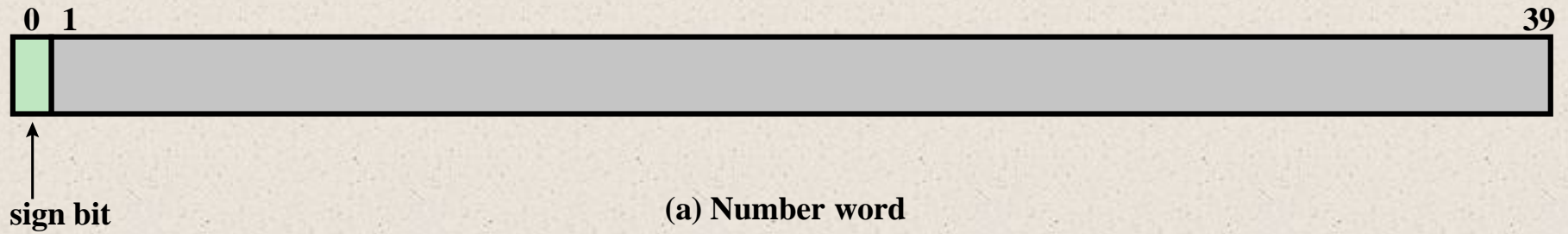


Figure 1.7 IAS Memory Formats



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)

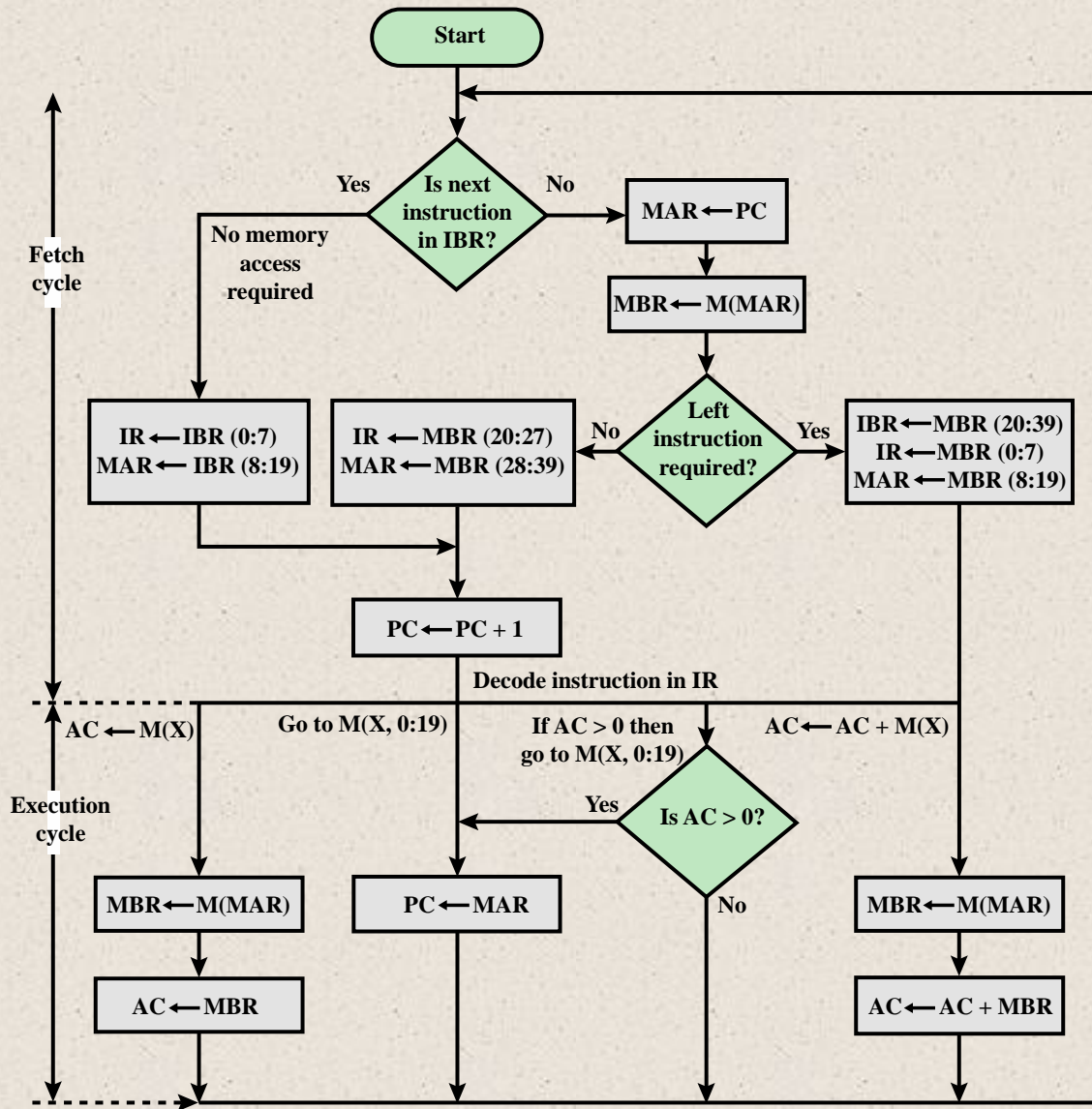
- Employed to temporarily hold the right-hand instruction from a word in memory

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



M(X) = contents of memory location whose address is X
(i:j) = bits i through j

Figure 1.8 Partial Flowchart of IAS Operation

Instruction Type	Opcode	Symbolic Representation	Description
Data transfer	00001010	LOAD MQ	Transfer contents of register MQ to the accumulator AC
	00001001	LOAD MQ,M(X)	Transfer contents of memory location X to MQ
	00100001	STOR M(X)	Transfer contents of accumulator to memory location X
	00000001	LOAD M(X)	Transfer M(X) to the accumulator
	00000010	LOAD -M(X)	Transfer -M(X) to the accumulator
	00000011	LOAD M(X)	Transfer absolute value of M(X) to the accumulator
	00000100	LOAD - M(X)	Transfer - M(X) to the accumulator
Unconditional branch	00001101	JUMP M(X,0:19)	Take next instruction from left half of M(X)
	00001110	JUMP M(X,20:39)	Take next instruction from right half of M(X)
Conditional branch	00001111	JUMP+ M(X,0:19)	If number in the accumulator is nonnegative, take next instruction from left half of M(X)
		<i>JU MP + M(X ,20: 39)</i>	<i>If number in the accumulator is nonnegative, take next instruction from right half of M(X)</i>
Arithmetic	00000101	ADD M(X)	Add M(X) to AC; put the result in AC
	00000111	ADD M(X)	Add M(X) to AC; put the result in AC
	00000110	SUB M(X)	Subtract M(X) from AC; put the result in AC
	00001000	SUB M(X)	Subtract M(X) from AC; put the remainder in AC
	00001011	MUL M(X)	Multiply M(X) by MQ; put most significant bits of result in AC, put least significant bits in MQ
	00001100	DIV M(X)	Divide AC by M(X); put the quotient in MQ and the remainder in AC
	00010100	LSH	Multiply accumulator by 2; i.e., shift left one bit position
	00010101	RSH	Divide accumulator by 2; i.e., shift right one position
Address modify	00010010	STOR M(X,8:19)	Replace left address field at M(X) by 12 rightmost bits of AC
	00010011	STOR M(X,28:39)	Replace right address field at M(X) by 12 rightmost bits of AC

Table 1.1
The IAS
Instruction Set

(Table can be found on page 17 in the textbook.)

+ 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

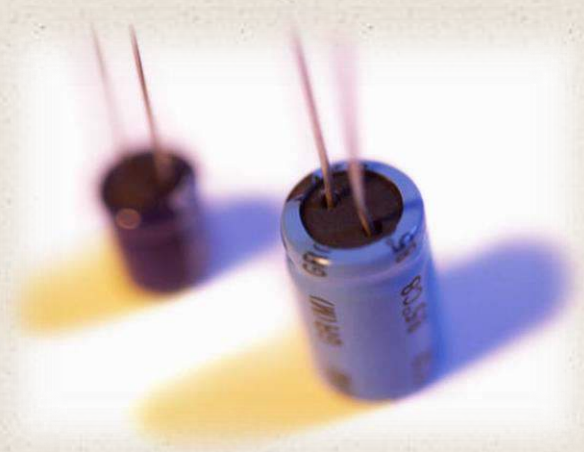




Table 1.2

Computer Generations

Generation	Approximate Dates	Technology	Typical Speed (operations per second)
1	1946–1957	Vacuum tube	40,000
2	1957–1964	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

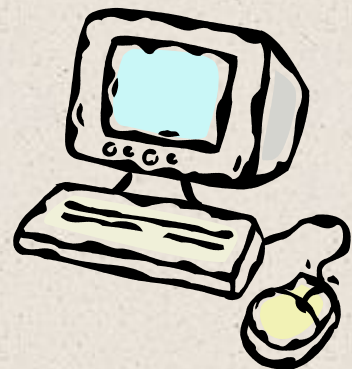


Second Generation Computers



■ 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
 - Libraries perform common computations



2. In a computer system with such devices, the CPU does **not execute** detailed I/O instructions. Such instructions are stored in a main memory to be executed by a **special-purpose processor** in the data channel itself.

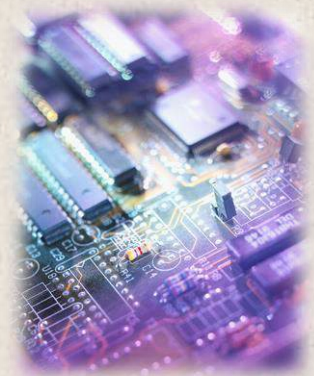
1. Several differences from the IAS computer are worth noting. The most important of these is the use of data channels. A data channel is an independent I/O module with its own processor and instruction set

3. multiplexor, which is the central termination point for data channels, the CPU, and memory. The multiplexor schedules access to the memory from the CPU and data channels, allowing these devices to act independently.

Figure 1.9 An IBM 7094

History of Computers

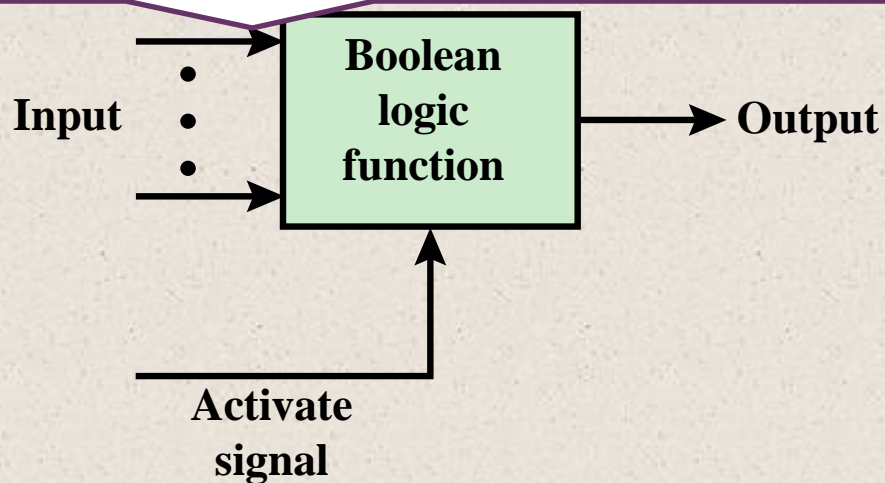
Third Generation: Integrated Circuits



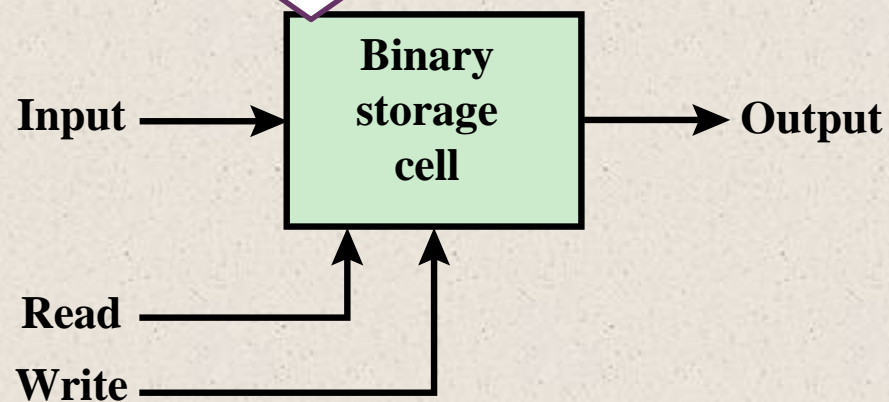
- 1958 – the invention of the integrated circuit
- *Discrete component*
 - Single, self-contained transistor
 - Manufactured separately, packaged in their own containers, and soldered or wired together onto masonite-like circuit boards
 - Manufacturing process was expensive and cumbersome
- The two most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

A gate is a device that implements a simple Boolean or logical function, such as *IF A AND B ARE TRUE THEN C IS TRUE (AND gate)*. Such devices are called gates because they control data flow in much the same way that canal gates control the flow of water

The memory cell is a device that can store one bit of data; that is, the device can be in one of two stable states at any time



(a) Gate



(b) Memory cell

Figure 1.10 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

• **Control:** The paths among components can carry control signals. For example, a gate will have one or two data inputs plus a control signal input that activates the gate. When the control signal is ON, the gate performs its function on the data inputs and produces a data output. Similarly, the memory cell will store the bit that is on its input lead when the WRITE control signal is ON and will place the bit that is in the cell on its output lead when the READ control signal is ON.



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



Figure 1.11 depicts the key concepts in an integrated circuit. A thin **wafer** of silicon is divided into a matrix of small areas, each a few millimeters square. The identical circuit pattern is fabricated in each area, and the wafer is broken up into **chips**. Each chip consists of many gates and/or memory cells plus a number of input and output attachment points. This chip is then packaged in housing that protects it and provides pins for attachment to devices beyond the chip. A number of these packages can then be interconnected on a printed circuit board to produce larger and more complex circuits.

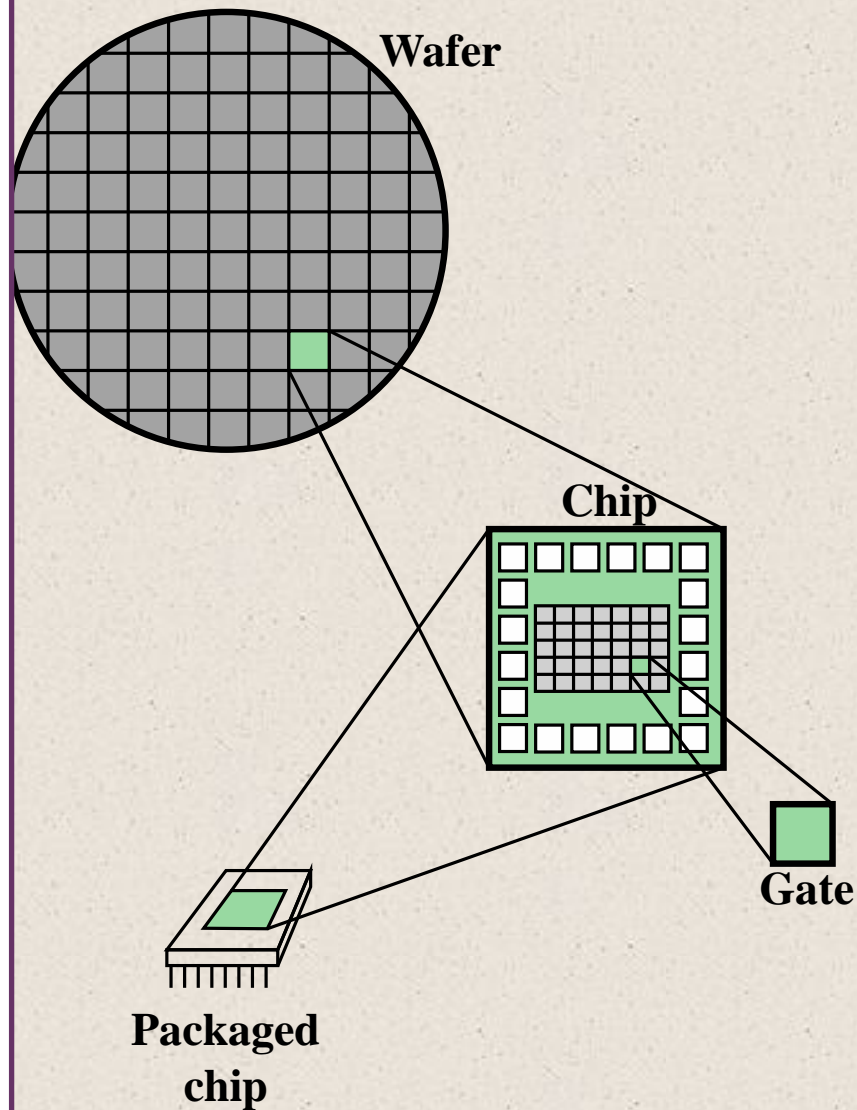


Figure 1.11 Relationship Among Wafer, Chip, and Gate

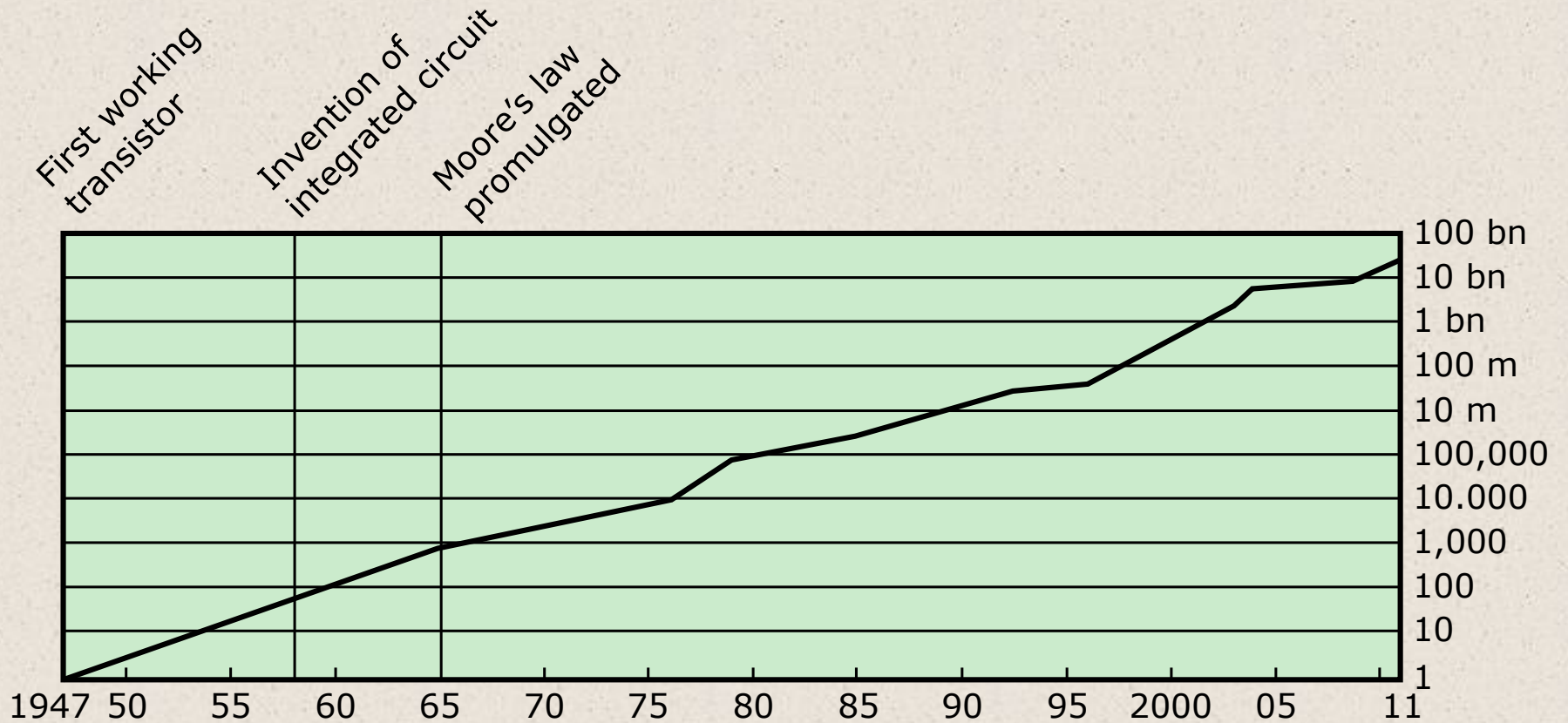


Figure 1.12 Growth in Transistor Count on Integrated Circuits (DRAM memory)

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

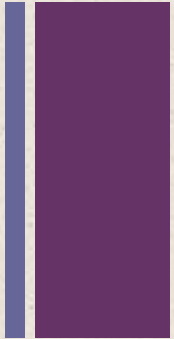
The electrical path length is shortened, increasing operating speed

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

Reduction in power and cooling requirements

Fewer interchip connections

+ IBM System/360



- By 1964, IBM had a firm grip on the computer market with its 7000 series of machines. In that year, IBM announced the System/360, a new family of computer products. Although the announcement itself was no surprise, it contained some unpleasant news for current IBM customers: the 360 product line was incompatible with older IBM machines. Thus, the transition to the 360 would be difficult for the current customer base
- This was a bold step by IBM, but one IBM felt was necessary to break out of some of the constraints of the 7000 architecture and to produce a system capable of evolving with the new integrated circuit technology
- The 360 was the success of the decade and cemented IBM as the overwhelmingly dominant computer vendor
- The architecture remains to this day the architecture of IBM's mainframe computers
- The models were compatible in the sense that a program written for one model should be capable of being executed by another model in the series, with only a difference in the time it takes to execute

+ Family Characteristics

Similar or identical instruction set

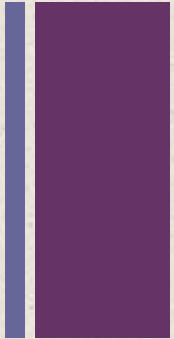
Similar or identical operating system

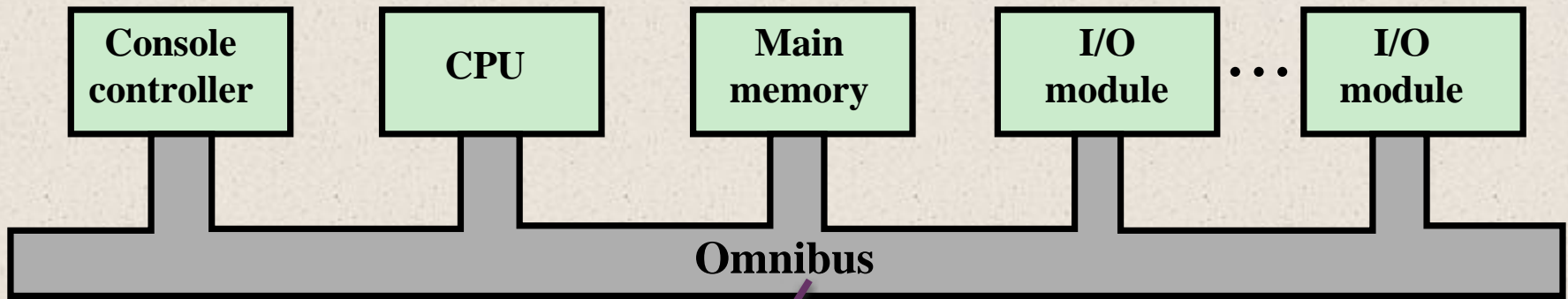
Increasing speed

Increasing number of I/O ports

Increasing memory size

Increasing cost





consists of 96 separate signal paths, used to carry control, address, and data signals. Because all system components share a common set of signal paths, their use can be controlled by the CPU. This architecture is highly flexible, allowing modules to be plugged into the bus to create various configurations. It is only in recent years that the bus structure has given way to a structure known as Point-to-point interconnect

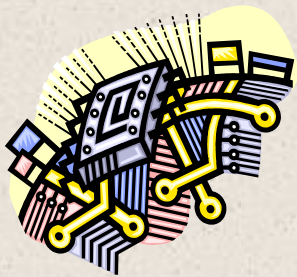
Figure 1.13 PDP-8 Bus Structure



Later Generations

LSI
Large
Scale
Integration

VLSI
Very Large
Scale
Integration



**Semiconductor
Memory
Microprocessors**

ULSI
Ultra Large
Scale
Integration

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
- This evolution can be seen most easily in the number of bits that the processor deals with at a time. There is no clear-cut measure of this, but perhaps the best measure is the data bus width: the number of bits of data that can be brought into or sent out of the processor at a time. Another measure is the number of bits in the accumulator or in the set of general-purpose registers.
- 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	8	6	3	6
Addressable memory	640 Bytes	16 KB	64 KB	1 MB	1 MB

(a) 1970s Processors

Evolution of Intel Microprocessors



	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

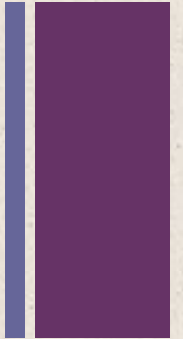
Evolution of Intel Microprocessors

	Pentium III	Pentium 4	Core 2 Duo	Core i7 EE 4960X
Introduced	1999	2000	2006	2013
Clock speeds	450 - 660 MHz	1.3 - 1.8 GHz	1.06 - 1.2 GHz	4 GHz
Bus width	64 bits	64 bits	64 bits	64 bits
Number of transistors	9.5 million	42 million	167 million	1.86 billion
Feature size (nm)	250	180	65	22
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 L2/15 MB L3
Number of cores	1	1	2	6

(d) Recent Processors



The Evolution of the Intel x86 Architecture



- Two processor families are the Intel x86 and the ARM architectures
- Current x86 offerings represent the results of decades of design effort on complex instruction set computers (CISCs)
- An alternative approach to processor design is the reduced instruction set computer (RISC)
- 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

Highlights of the Evolution of the Intel Product Line:



8080

- World's first general-purpose microprocessor
- 8-bit machine, 8-bit data path to memory
- Was used in the first personal computer

8086

- A more powerful 16-bit machine
- Wider data path and larger registers
- Has an instruction cache, or queue, that prefetches a few instructions before they are executed
- The first appearance of the x86 architecture
- The 8088 was a variant of this processor and used in IBM's first personal computer (securing the success of Intel)

80286

- Extension of the 8086 enabling addressing a 16-MB memory instead of just 1MB

80386

- Intel's first 32-bit machine
- First Intel processor to support multitasking meaning it could run multiple programs at the same time.

80486

- Introduced the use of much more sophisticated and powerful cache technology and sophisticated instruction pipelining
- Also offered a built-in math coprocessor which offloading complex math operations from the main CPU.

Highlights of the Evolution of the Intel Product Line:

Pentium

- Intel introduced the use of superscalar techniques, which allow multiple instructions to execute in parallel

Pentium Pro

- Continued the move into superscalar organization with aggressive use of register renaming, branch prediction, data flow analysis, and speculative execution

Pentium II

- Incorporated Intel MMX technology, which is designed specifically to process video, audio, and graphics data efficiently

Pentium III

- Incorporated additional floating-point instructions
- Streaming SIMD Extensions (SSE) instruction set extension added 70 new instructions designed to increase performance when exactly the same operations are to be performed on multiple data objects

Pentium 4

- Includes additional floating-point and other enhancements for multimedia

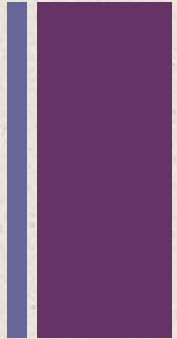
Core

- First Intel x86 micro-core referring to the implementation of two cores on a single chip.

Core 2

- Extends the Core architecture to 64 bits
- Core 2 Quad provides four cores on a single chip
- More recent Core offerings have up to 10 cores per chip
- An important addition to the architecture was the Advanced Vector Extensions instruction set

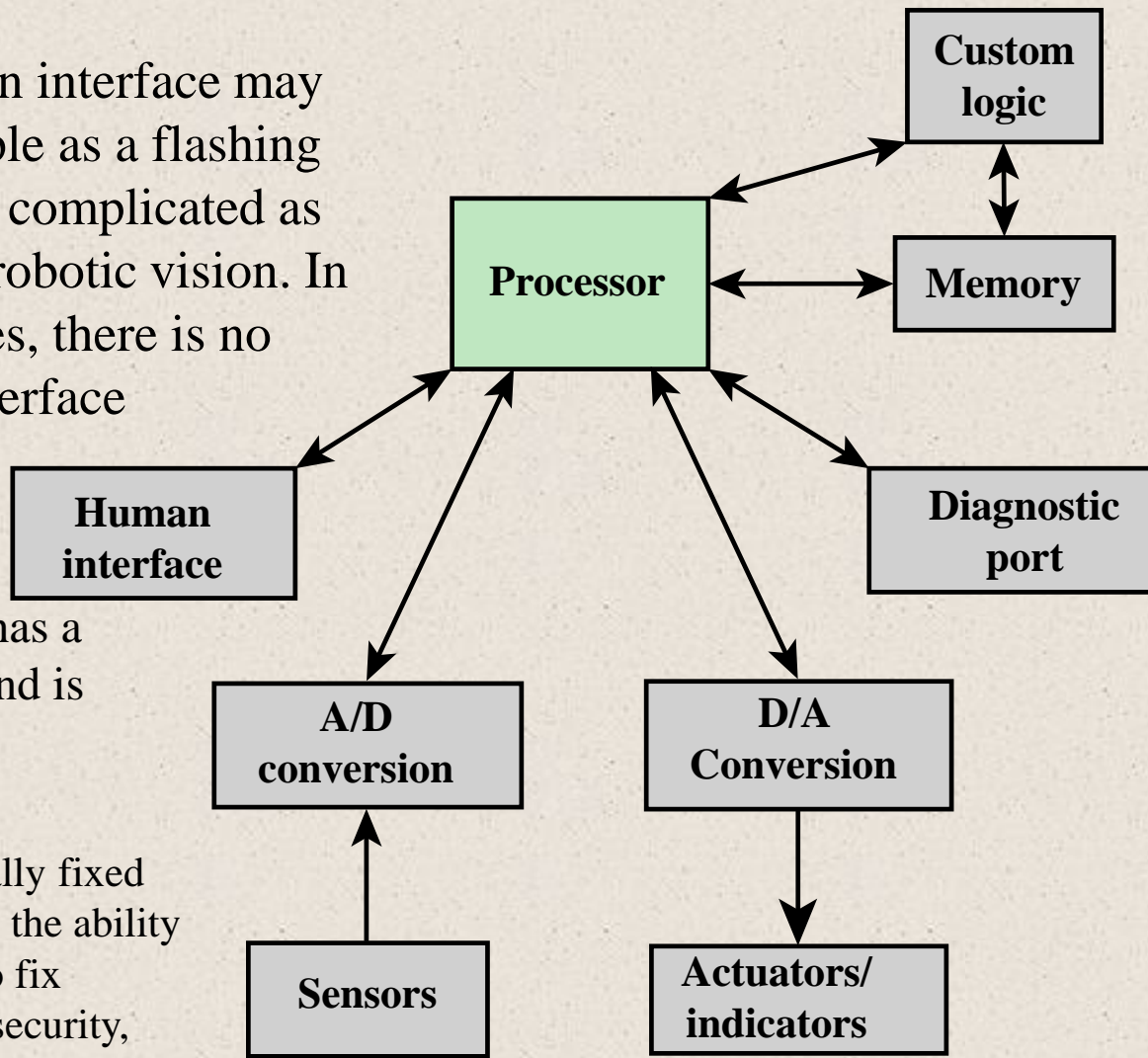
+ Embedded Systems



- Embedded systems are special-purpose computers built into devices not generally considered to be computers
- The use of electronics and software within a product
- Billions of computer systems are produced each year that are embedded within larger devices
- Today many devices that use electric power have an embedded computing system
- Often embedded systems are tightly coupled to their environment
 - This can give rise to real-time constraints imposed by the need to interact with the environment
 - Constraints such as required speeds of motion, required precision of measurement, and required time durations, dictate the timing of software operations
 - If multiple activities must be managed simultaneously this imposes more complex real-time constraints



The human interface may be as simple as a flashing light or as complicated as real-time robotic vision. In many cases, there is no human interface



The diagnostic port may be used for diagnosing the system that is being controlled

Software often has a fixed function and is specific to the application

Even with nominally fixed function software, the ability to field upgrade to fix bugs, to improve security, and to add functionality, has become very important for embedded system

Embedded systems often interact (sense, manipulate, and communicate) with external world through sensors and actuators

Figure 1.14 Possible Organization of an Embedded System



The Internet of Things (IoT)

- Term that refers to the expanding interconnection of smart devices, ranging from appliances to tiny sensors
- Is primarily driven by deeply embedded devices
- Generations of deployment culminating in the IoT:
 - Information technology (IT)
 - PCs, servers, routers, firewalls, and so on, bought as IT devices by enterprise IT people and primarily using wired connectivity
 - Operational technology (OT)
 - Machines/appliances with embedded IT built by non-IT companies, such as medical machinery, SCADA, process control, and kiosks, bought as appliances by enterprise OT people and primarily using wired connectivity
 - Personal technology
 - Smartphones, tablets, and eBook readers bought as IT devices by consumers exclusively using wireless connectivity and often multiple forms of wireless connectivity
 - Sensor/actuator technology
 - Single-purpose devices bought by consumers, IT, and OT people exclusively using wireless connectivity, generally of a single form, as part of larger systems
- It is the fourth generation that is usually thought of as the IoT and it is marked by the use of billions of embedded devices



Embedded Operating Systems

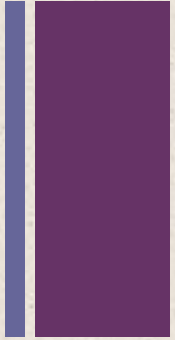
- There are two general approaches to developing an embedded operating system (OS):
 - Take an existing OS and adapt it for the embedded application
 - Design and implement an OS intended solely for embedded use

Application Processors versus Dedicated Processors

- Application processors
 - Defined by the processor's ability to execute complex operating systems
 - General-purpose in nature
 - An example is the smartphone – the embedded system is designed to support numerous apps and perform a wide variety of functions
- Dedicated processor
 - Is dedicated to one or a small number of specific tasks required by the host device
 - Because such an embedded system is dedicated to a specific task or tasks, the processor and associated components can be engineered to reduce size and cost



Microcontroller



- Also called a “computer on a chip,” billions of microcontroller units are embedded each year in myriad products from toys to appliances to automobiles. For example, a single vehicle can use 70 or more microcontrollers. Typically, especially for the smaller, less expensive microcontrollers, they are used as dedicated processors for specific tasks. For example, microcontrollers are heavily utilized in automation processes. By providing simple reactions to input, they can control machinery, turn fans on and off, open and close valves, and so forth. They are integral parts of modern industrial technology and are among the most inexpensive ways to produce machinery that can handle extremely complex functionalities.

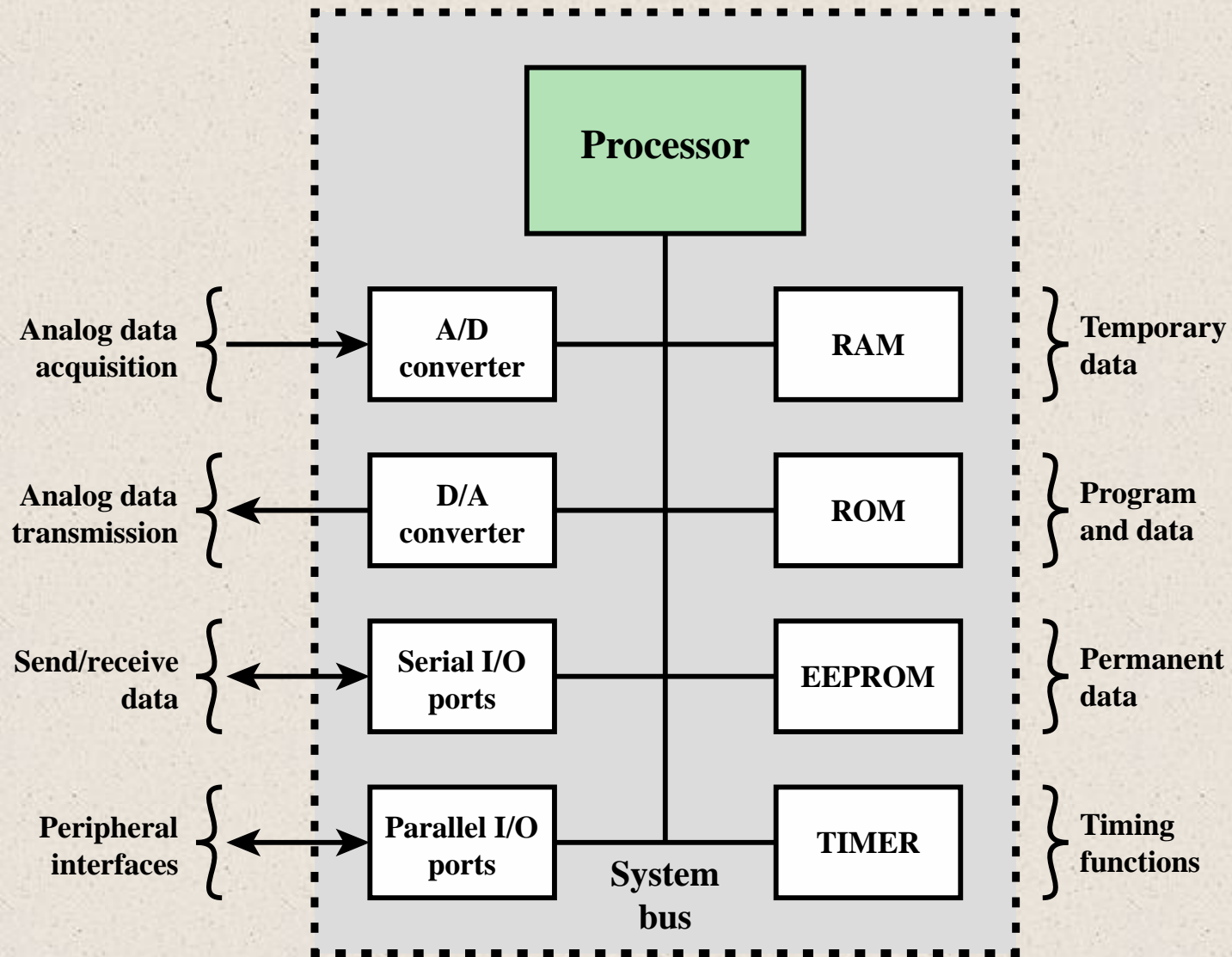


Figure 1.15 Typical Microcontroller Chip Elements

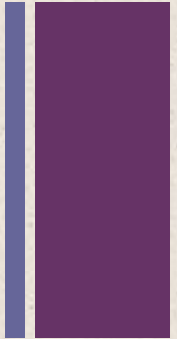


Microprocessor vs Microcontroller

- As we have seen, early microprocessor chips included registers, an ALU, and some sort of control unit or instruction processing logic. As transistor density increased, it became possible to increase the complexity of the instruction set architecture, and ultimately to add memory and more than one processor. Contemporary microprocessor chips, as shown in Figure 1.2, include multiple cores and a substantial amount of cache memory.
- A microcontroller chip makes a substantially different use of the logic space available. Figure 1.15 shows in general terms the elements typically found on a microcontroller chip. As shown, a microcontroller is a single chip that contains the processor, non-volatile memory for the program (ROM), volatile memory for input and output (RAM), a clock, and an I/O control unit. The processor portion of the microcontroller has a much lower silicon area than other microprocessors and much higher energy efficiency
- Microcontrollers come in a range of physical sizes and processing power. Processors range from 4-bit to 32-bit architectures. Microcontrollers tend to be much slower than microprocessors, typically operating in the MHz range rather than the GHz speeds of microprocessors. Another typical feature of a microcontroller is that it does not provide for human interaction. The microcontroller is programmed for a specific task, embedded in its device, and executes as and when required



Deeply Embedded Systems



- Subset of embedded systems
- Has a processor whose behavior is difficult to observe both by the programmer and the user
- Uses a microcontroller rather than a microprocessor
- Is not programmable once the program logic for the device has been burned into ROM
- Has no interaction with a user
- Dedicated, single-purpose devices that detect something in the environment, perform a basic level of processing, and then do something with the results
- Often have wireless capability and appear in networked configurations, such as networks of sensors deployed over a large area
- Typically have extreme resource constraints in terms of memory, processor size, time, and power consumption

ARM



Refers to a processor architecture that has evolved from RISC design principles and is used in embedded systems

Family of RISC-based microprocessors and microcontrollers designed by ARM Holdings, Cambridge, England

Chips are high-speed processors that are known for their small die size and low power requirements

Probably the most widely used embedded processor architecture and indeed the most widely used processor architecture of any kind in the world

Acorn RISC Machine/Advanced RISC Machine

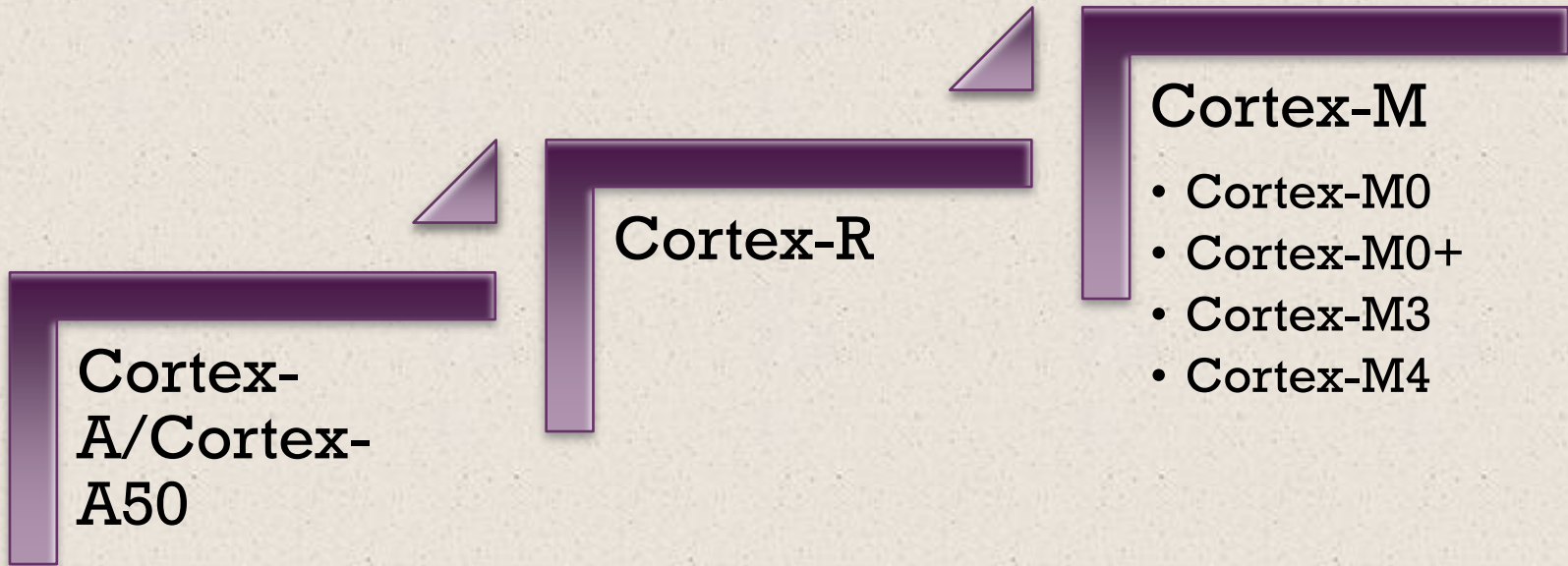


RISC VS CISC

Parameter	RISC	CISC
Instruction types	Simple	Complex
Number of instructions	Reduced (30-40)	Extended (100-200)
Duration of an instruction	One cycle	More cycles (4-120)
Instruction format	Fixed	Variable
Instruction execution	In parallel (pipeline)	Sequential
Addressing modes	Simple	Complex
Instructions accessing the memory	Two: Load and Store	Almost all from the set
Register set	multiple	unique
Complexity	In compiler	In CPU (micro-program)



ARM Products



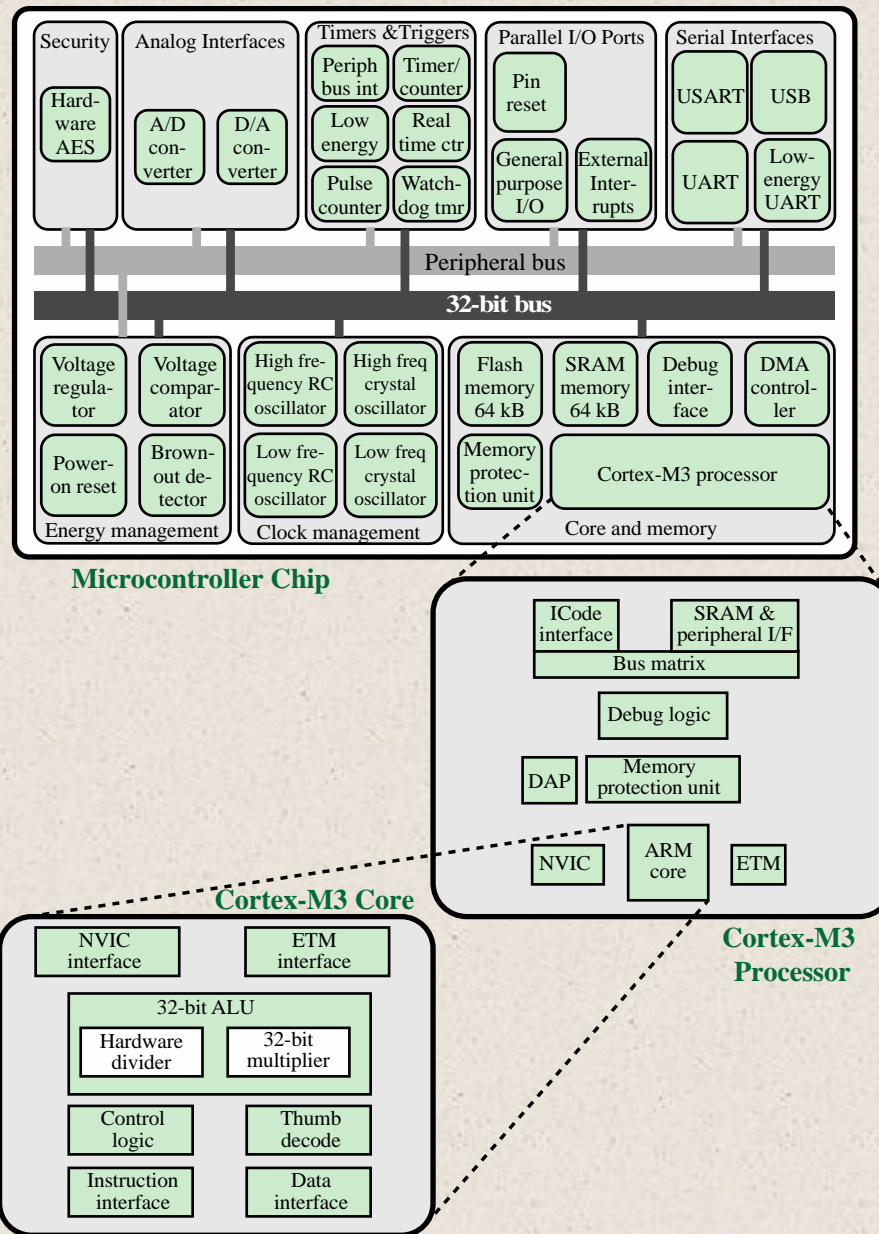


Figure 1.16 Typical Microcontroller Chip Based on Cortex-M3

+ Cloud Computing



- NIST defines cloud computing as:

“A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction.”

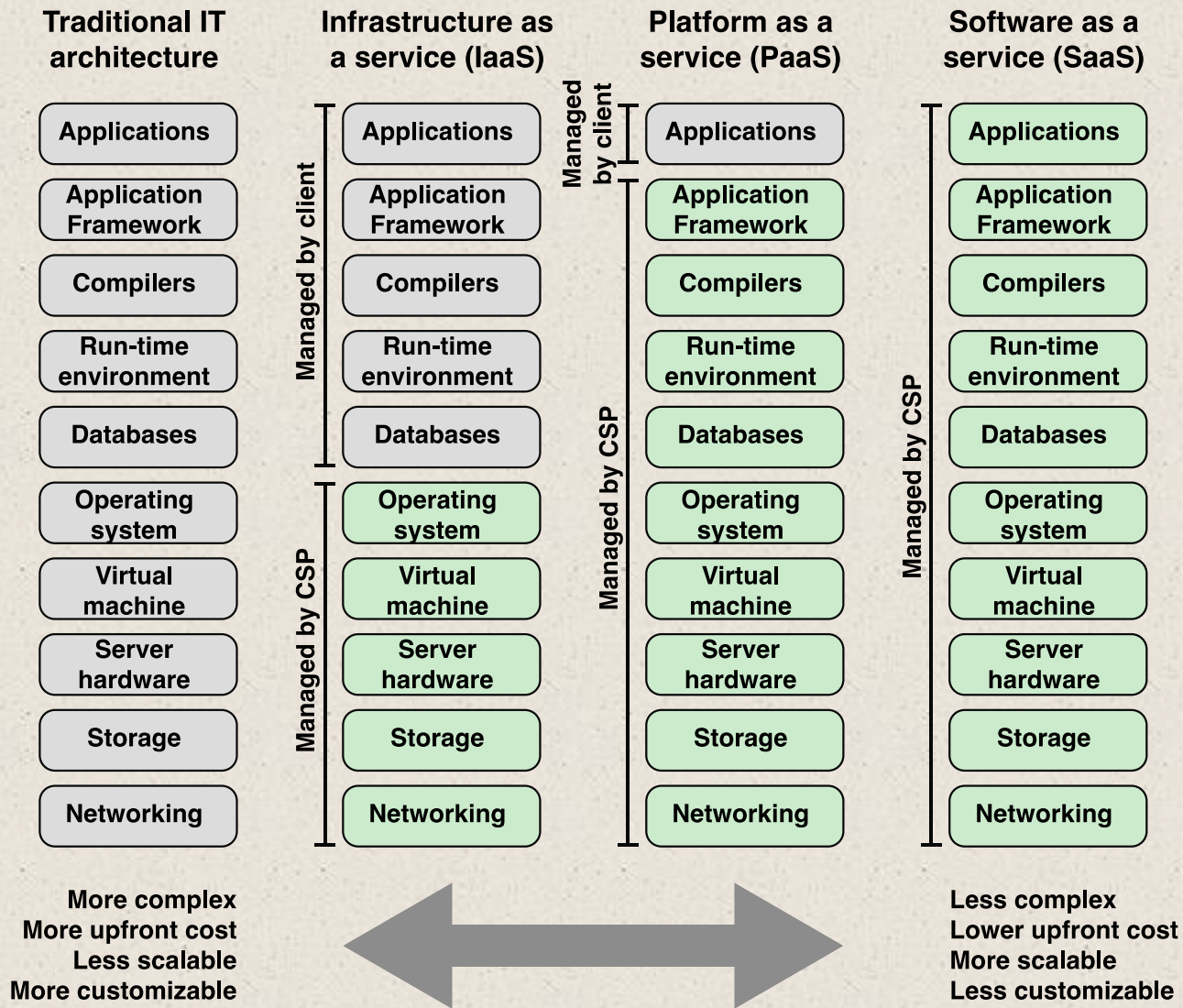
- You get economies of scale, professional network management, and professional security management
- The individual or company only needs to pay for the storage capacity and services they need
- Cloud provider takes care of security

Cloud Networking

- Refers to the networks and network management functionality that must be in place to enable cloud computing
- One example is the provisioning of high-performance and/or high-reliability networking between the provider and subscriber
- The collection of network capabilities required to access a cloud, including making use of specialized services over the Internet, linking enterprise data center to a cloud, and using firewalls and other network security devices at critical points to enforce access security policies

Cloud Storage

- Subset of cloud computing
- Consists of database storage and database applications hosted remotely on cloud servers
- Enables small businesses and individual users to take advantage of data storage that scales with their needs and to take advantage of a variety of database applications without having to buy, maintain, and manage the storage assets



IT = information technology
 CSP = cloud service provider

Figure 1.17 Alternative Information Technology Architectures



SaaS

- Software as a service (SaaS) As the name implies, a SaaS cloud provides service to customers in the form of software, specifically application software, running on and accessible in the cloud. SaaS follows the familiar model of Web services, in this case applied to cloud resources. SaaS enables the customer to use the cloud provider's applications running on the provider's cloud infrastructure. The applications are accessible from various client devices through a simple interface such as a Web browser. Instead of obtaining desktop and server licenses for software products it uses, an enterprise obtains the same functions from the cloud service. SaaS saves the complexity of software installation, maintenance, upgrades, and patches. Examples of services at this level are Gmail, Google's e-mail service, and Salesforce.com, which help firms keep track of their customers.
- Common subscribers to SaaS are organizations that want to provide their employees with access to typical office productivity software, such as document management and email. Individuals also commonly use the SaaS model to acquire cloud resources. Typically, subscribers use specific applications on demand. The cloud provider also usually offers data-related features such as automatic backup and data sharing between subscribers.



PaaS

- Platform as a service (PaaS) A PaaS cloud provides service to customers in the form of a platform on which the customer's applications can run. PaaS enables the customer to deploy onto the cloud infrastructure containing customer- created or acquired applications. A PaaS cloud provides useful software building blocks, plus a number of development tools, such as programming languages, run- time environments, and other tools that assist in deploying new applications. In effect, PaaS is an operating system in the cloud. PaaS is useful for an organization that wants to develop new or tailored applications while paying for the needed computing resources only as needed and only for as long as needed. Google App Engine and the Salesforce1 Platform from Salesforce.com are examples of PaaS.



IaaS



- Infrastructure as a service (IaaS) With IaaS, the customer has access to the underlying cloud infrastructure. IaaS provides virtual machines and other abstracted hardware and operating systems, which may be controlled through a service application programming interface (API). IaaS offers the customer processing, storage, networks, and other fundamental computing resources so that the customer is able to deploy and run arbitrary software, which can include operating systems and applications. IaaS enables customers to combine basic computing services, such as number crunching and data storage, to build highly adaptable computer systems. Examples of IaaS are Amazon Elastic Compute Cloud (Amazon EC2) and Windows Azure.

+ Summary

Chapter 1

- Organization and architecture
- Structure and function
- Brief history of computers
 - The First Generation: Vacuum tubes
 - The Second Generation: Transistors
 - The Third Generation: Integrated Circuits
 - Later generations
- The evolution of the Intel x86 architecture
- Cloud computing
 - Basic concepts
 - Cloud services

Basic Concepts and Computer Evolution

- Embedded systems
 - The Internet of things
 - Embedded operating systems
 - Application processors versus dedicated processors
 - Microprocessors versus microcontrollers
 - Embedded versus deeply embedded systems
- ARM architecture
 - ARM evolution
 - Instruction set architecture
 - ARM products



**William Stallings
Computer Organization
and Architecture
10th Edition**



+ Chapter 3

A Top-Level View of Computer Function and Interconnection

+ Top-Level of Computer



- At a top level, a computer consists of CPU (central processing unit), memory, and I/O components, with one or more modules of each type. These components are interconnected in some fashion to achieve the basic function of the computer, which is to execute programs. Thus, at a top level, we can characterize a computer system by describing (1) the external behavior of each component, that is, the data and control signals that it exchanges with other components and (2) the interconnection structure and the controls required to manage the use of the interconnection structure.



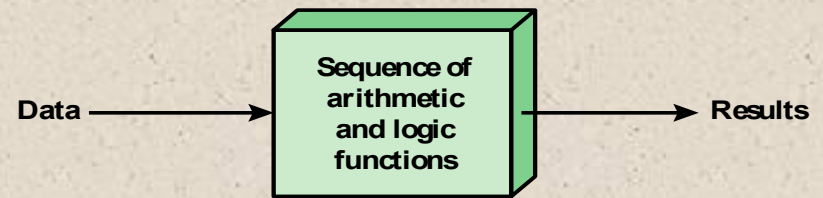
Computer Components



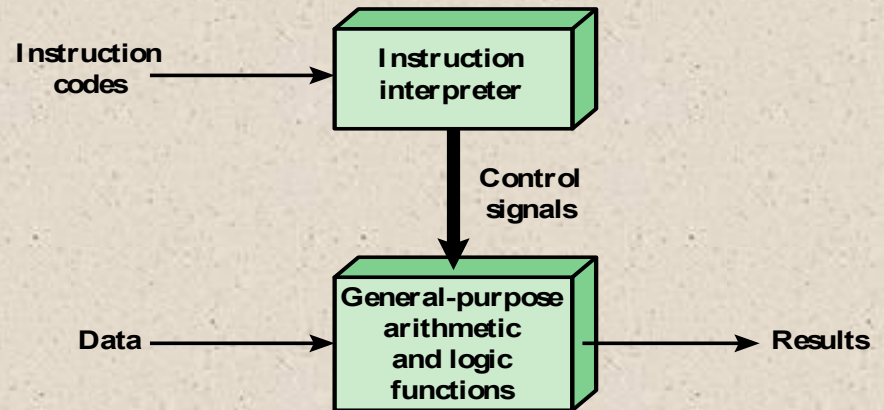
- Contemporary computer designs are based on concepts developed by John von Neumann at the Institute for Advanced Studies, Princeton
- Referred to as the *von Neumann architecture* and is based on three key concepts:
 - Data and instructions are stored in a single read-write memory
 - The contents of 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*
 - The result of the process of **connecting the various components** in the desired configuration



Hardware and Software Approaches



(a) Programming in hardware



(b) Programming in software

Suppose we construct a general-purpose configuration of arithmetic and logic functions. This set of hardware will perform various functions on data depending on control signals applied to the hardware. In the original case of customized hardware, the system accepts data and produces results (Figure 3.1a). With general-purpose hardware, the system accepts data and control signals and produces results. Thus, instead of rewiring the hardware for each new program, the programmer merely needs to supply a new set of control signals.

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

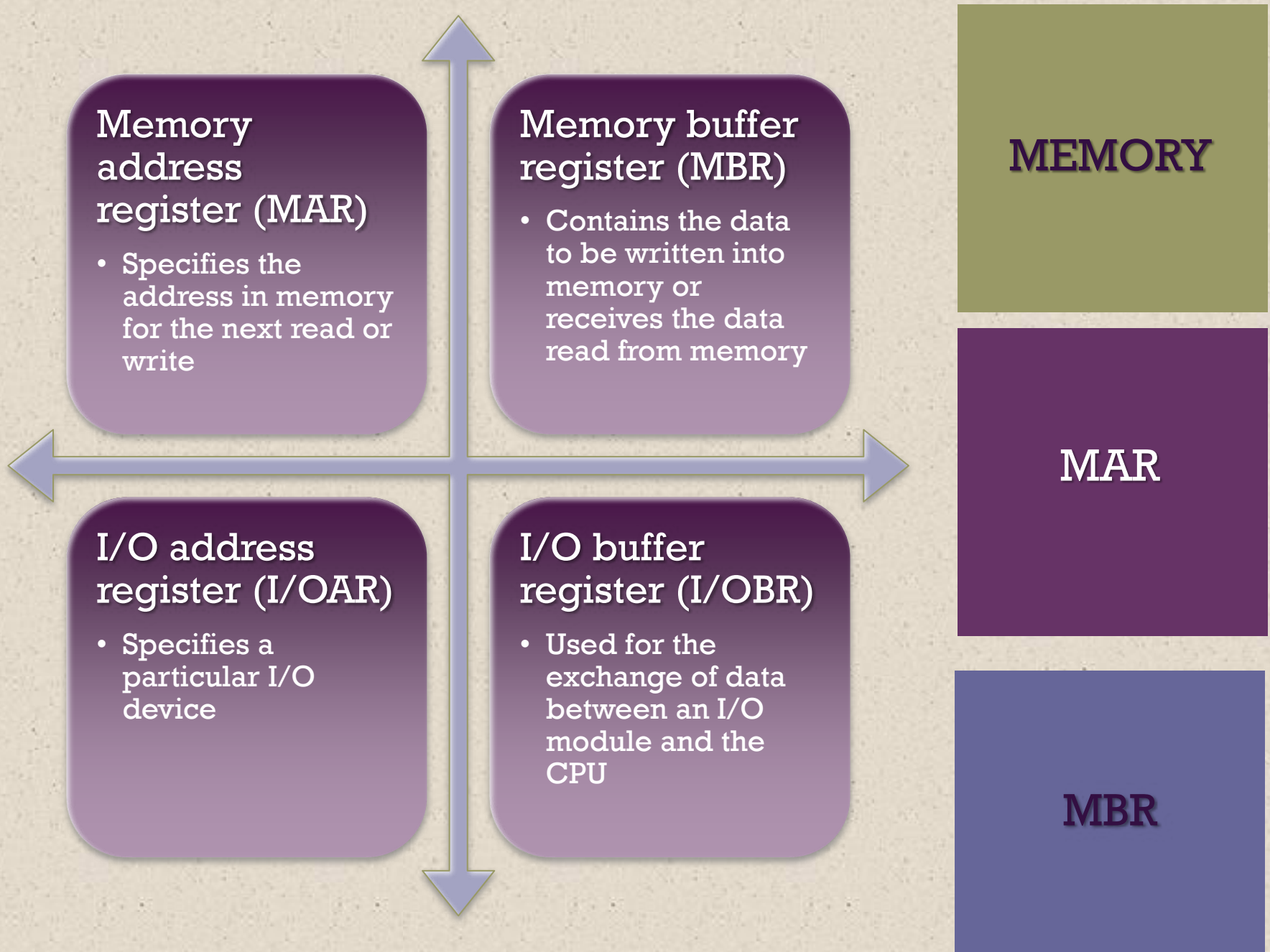
Major components:

- 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





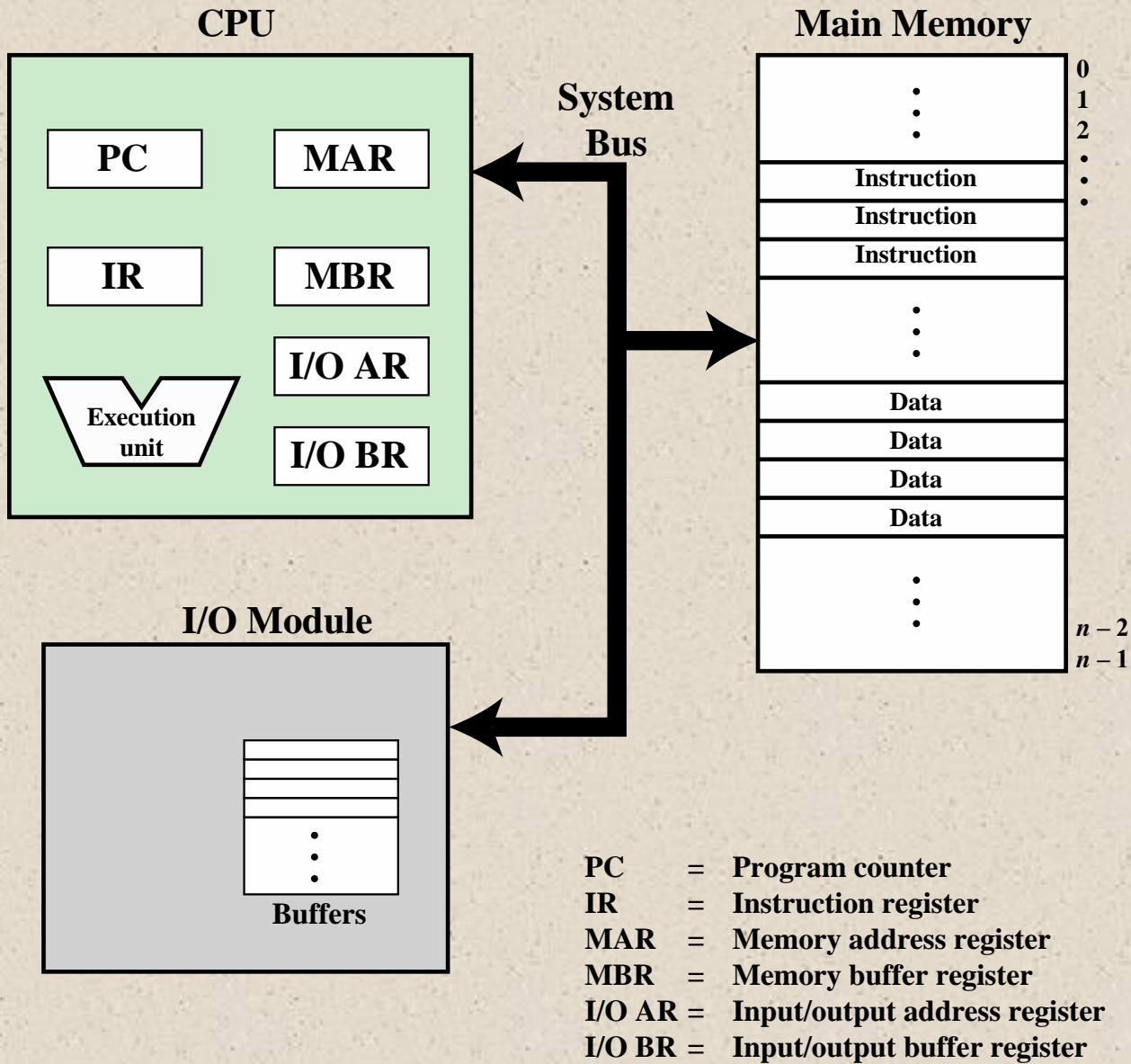


Figure 3.2 Computer Components: Top-Level View

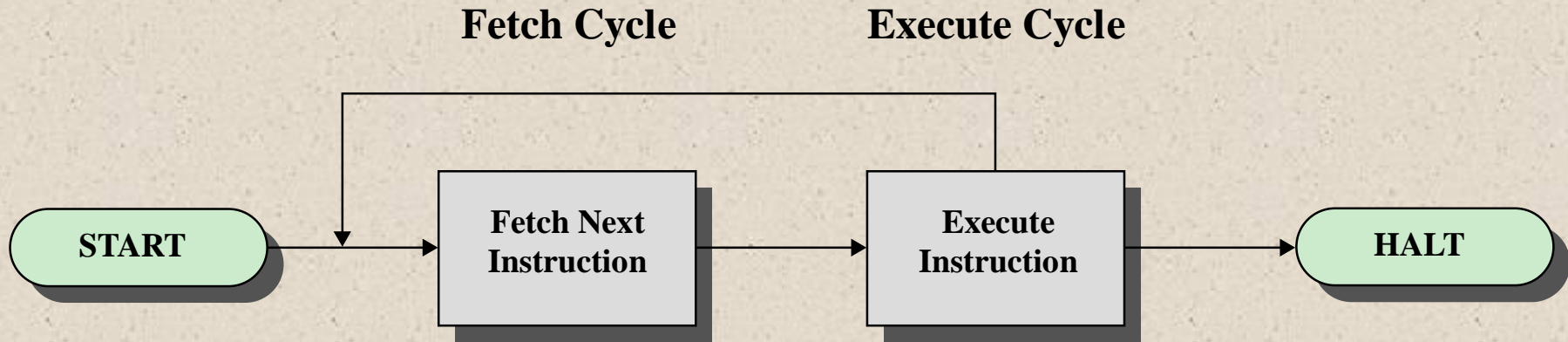


Figure 3.3 Basic Instruction Cycle



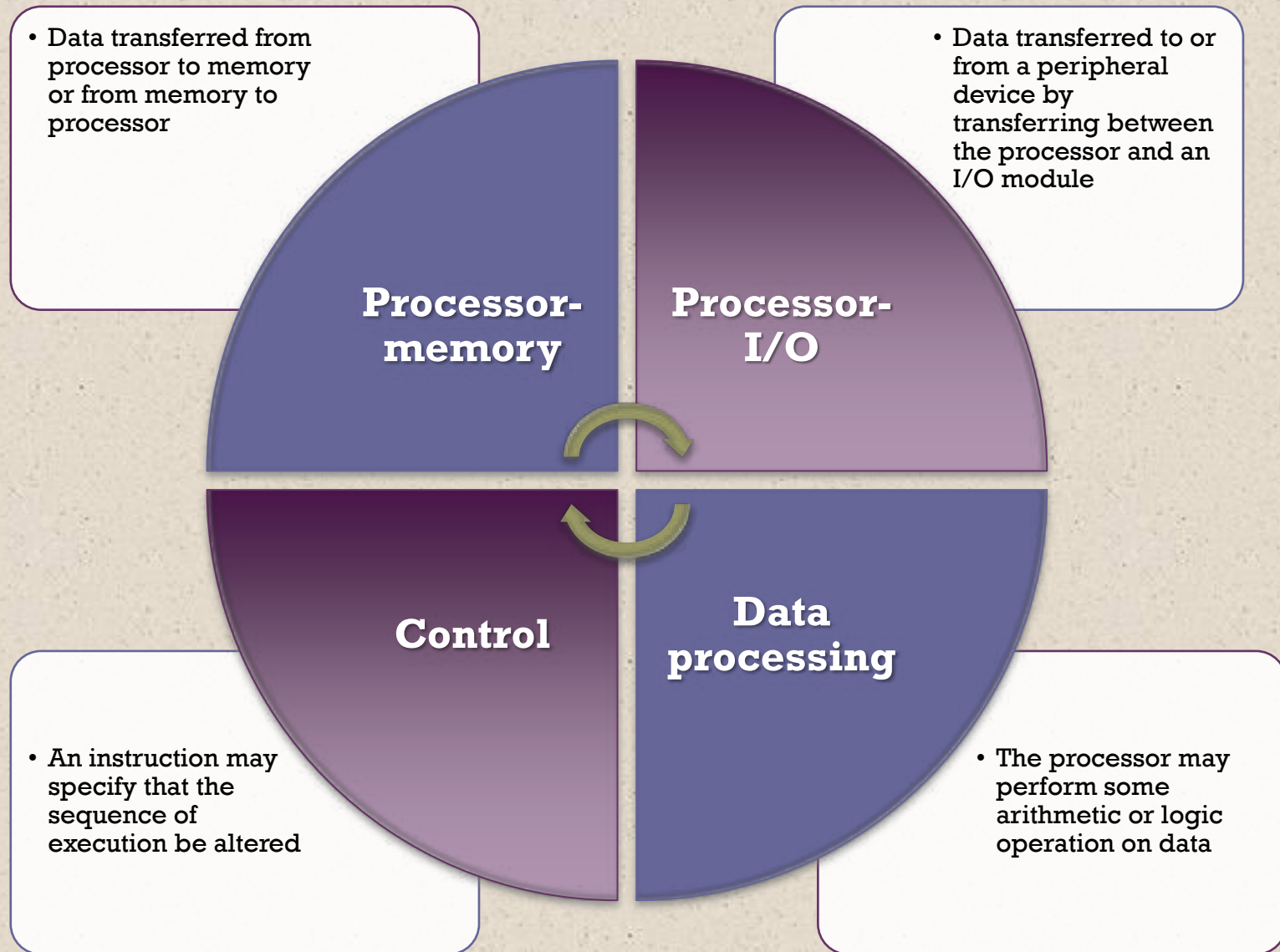
Fetch Cycle



- 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





(a) Instruction format



(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

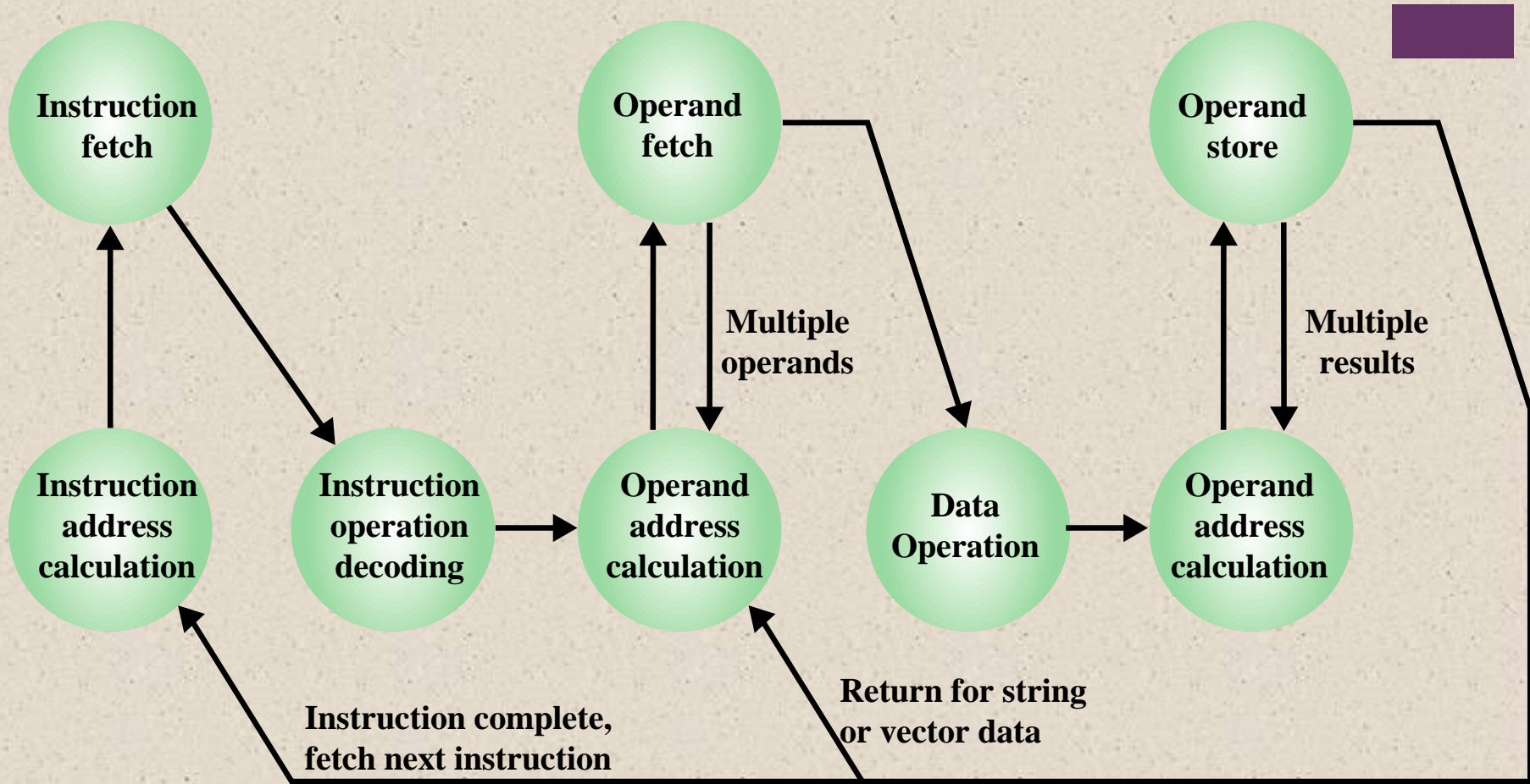


Figure 3.6 Instruction Cycle State Diagram



Interrupt

- The interrupt is a signal emitted by hardware or software when a process or an event needs immediate attention. It alerts the processor to a high-priority process requiring interruption of the current working process. In I/O devices one of the bus control lines is dedicated for this purpose and is called the *Interrupt Service Routine (ISR)*.
- When a device raises an interrupt at let's say process i , the processor first completes the execution of instruction i . Then it loads the Program Counter (PC) with the address of the first instruction of the ISR. Before loading the Program Counter with the address, the address of the interrupted instruction is moved to a temporary location. Therefore, after handling the interrupt the processor can continue with process $i+1$.
- While the processor is handling the interrupts, it must inform the device that its request has been recognized so that it stops sending the interrupt request signal. Also, saving the registers so that the interrupted process can be restored in the future, increases the delay between the time an interrupt is received and the start of the execution of the ISR. This is called Interrupt Latency.

+ Type of interrupt

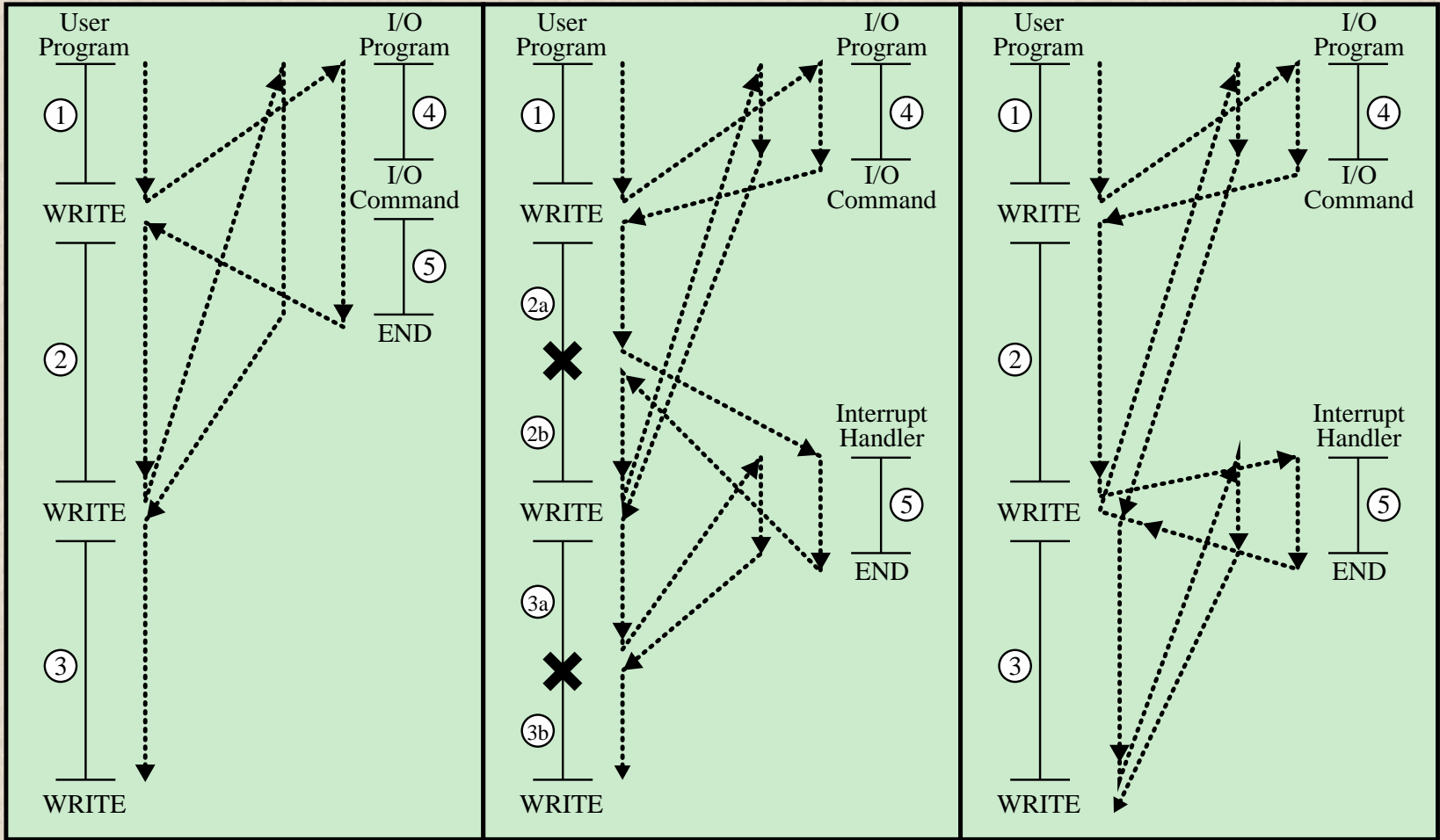
- **Software Interrupts:** A sort of interrupt called a software interrupt is one that is produced by software or a system as opposed to hardware. Traps and exceptions are other names for software interruptions. They serve as a signal for the operating system or a system service to carry out a certain function or respond to an error condition. A particular instruction known as a “interrupt instruction” is used to create software interrupts. When the interrupt instruction is used, the processor stops what it is doing and switches over to a particular interrupt handler code. The interrupt handler routine completes the required work or handles any errors before handing back control to the interrupted application.
- **Hardware Interrupts:** In a hardware interrupt, all the devices are connected to the Interrupt Request Line. A single request line is used for all the n devices. To request an interrupt, a device closes its associated switch. When a device requests an interrupt, the value of INTR is the logical OR of the requests from individual devices.



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.

Table 3.1

Classes of Interrupts



(a) No interrupts

(b) Interrupts; short I/O wait

(c) Interrupts; long I/O wait

X = interrupt occurs during course of execution of user program

Figure 3.7 Program Flow of Control Without and With Interrupts

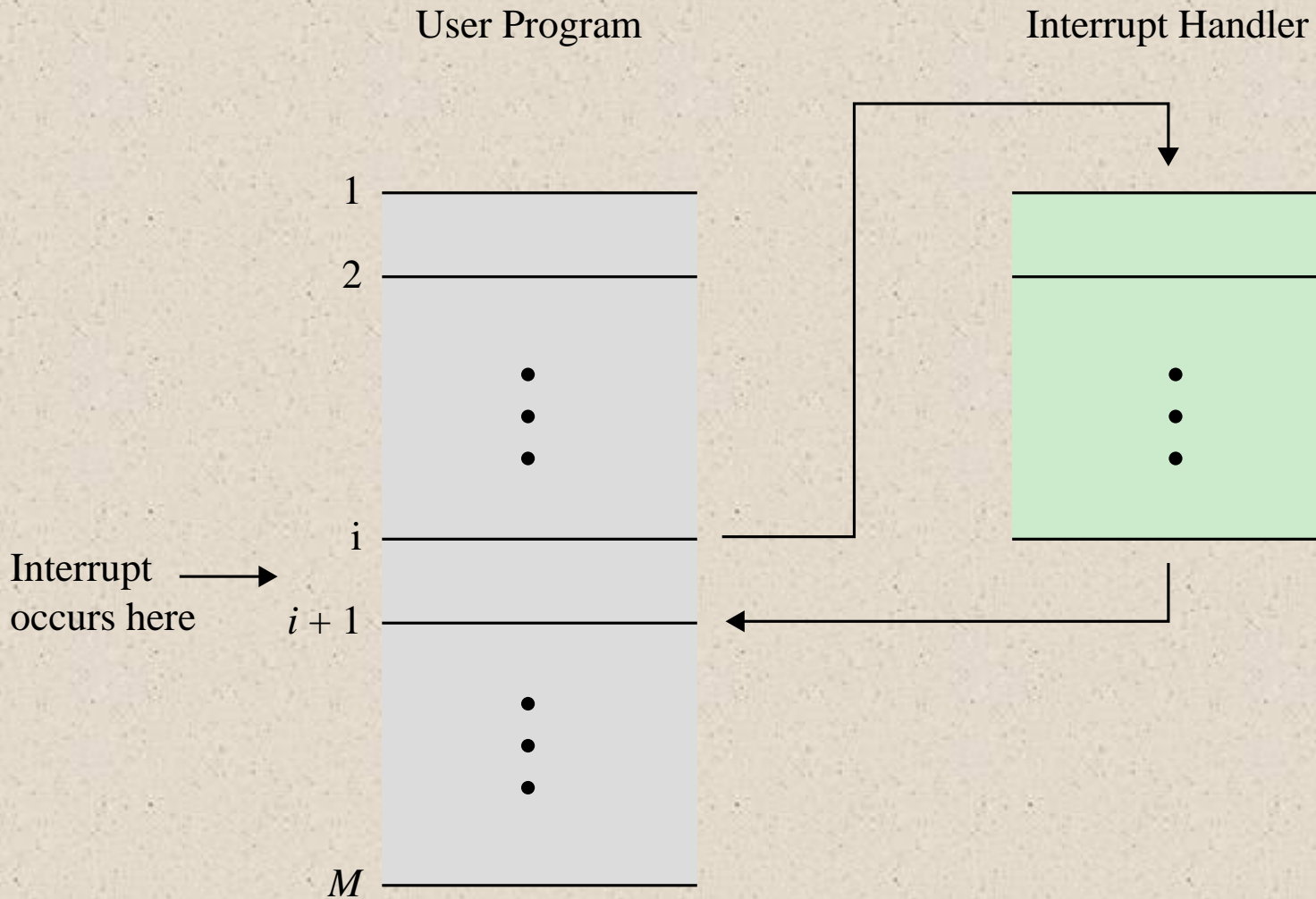


Figure 3.8 Transfer of Control via Interrupts

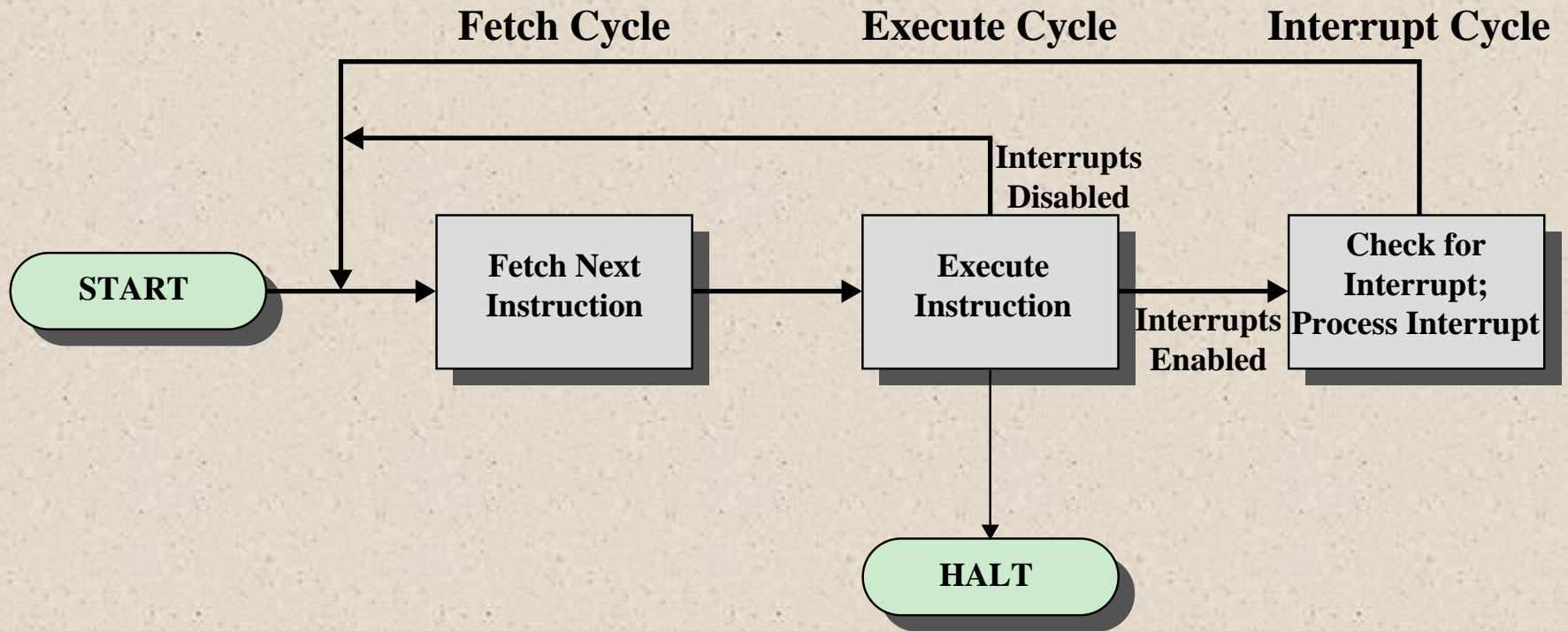


Figure 3.9 Instruction Cycle with Interrupts

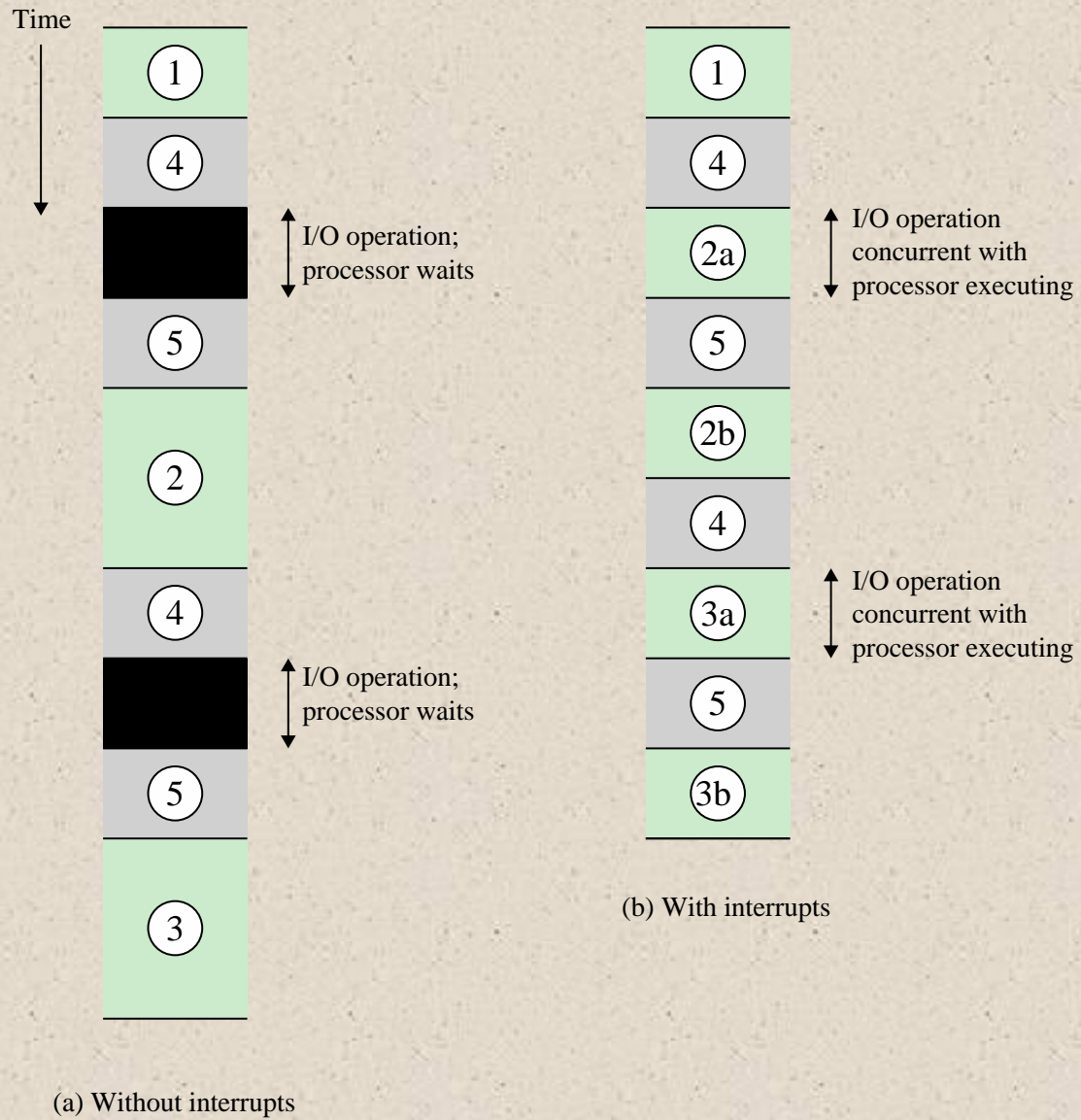


Figure 3.10 Program Timing: Short I/O Wait

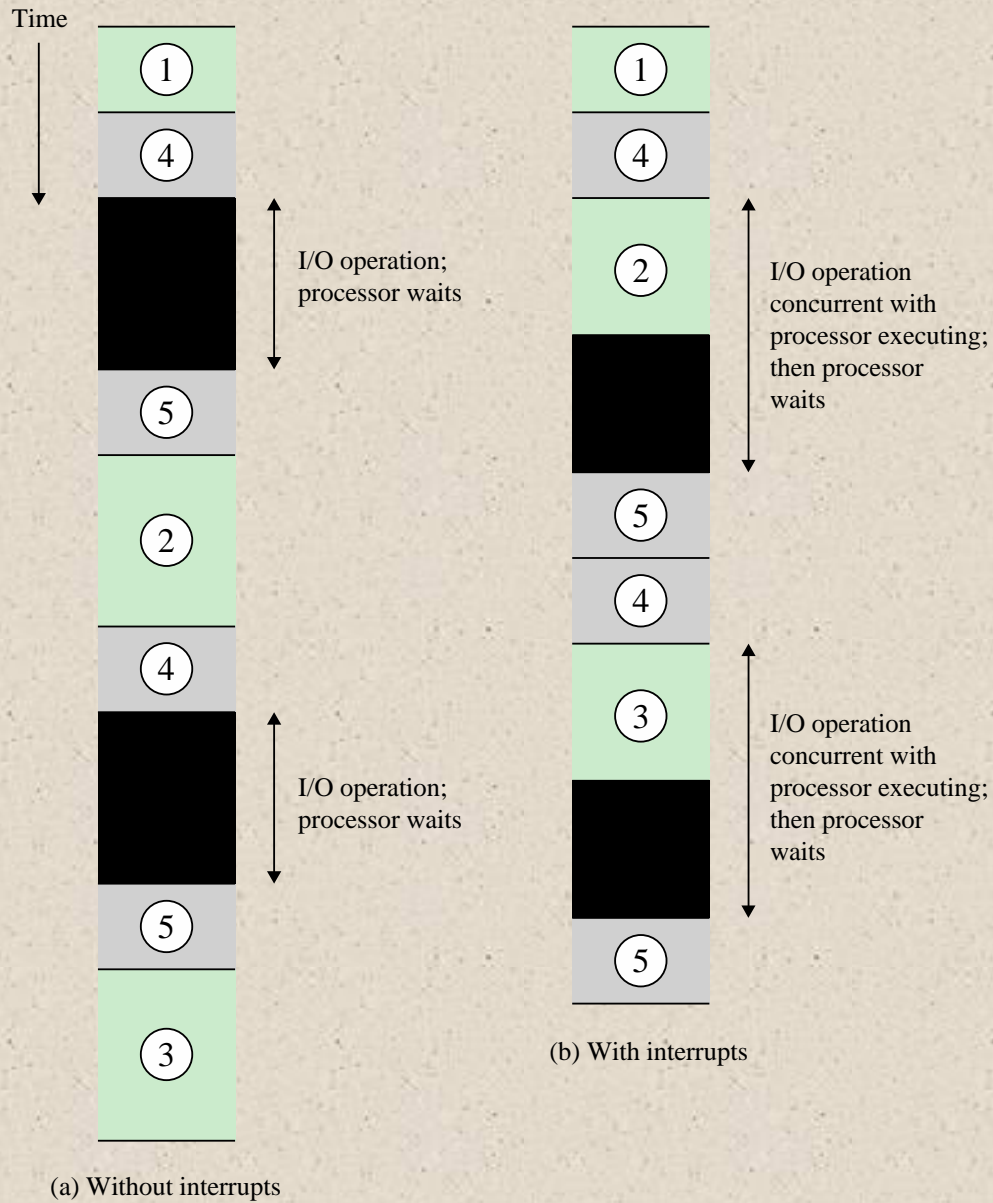


Figure 3.11 Program Timing: Long I/O Wait

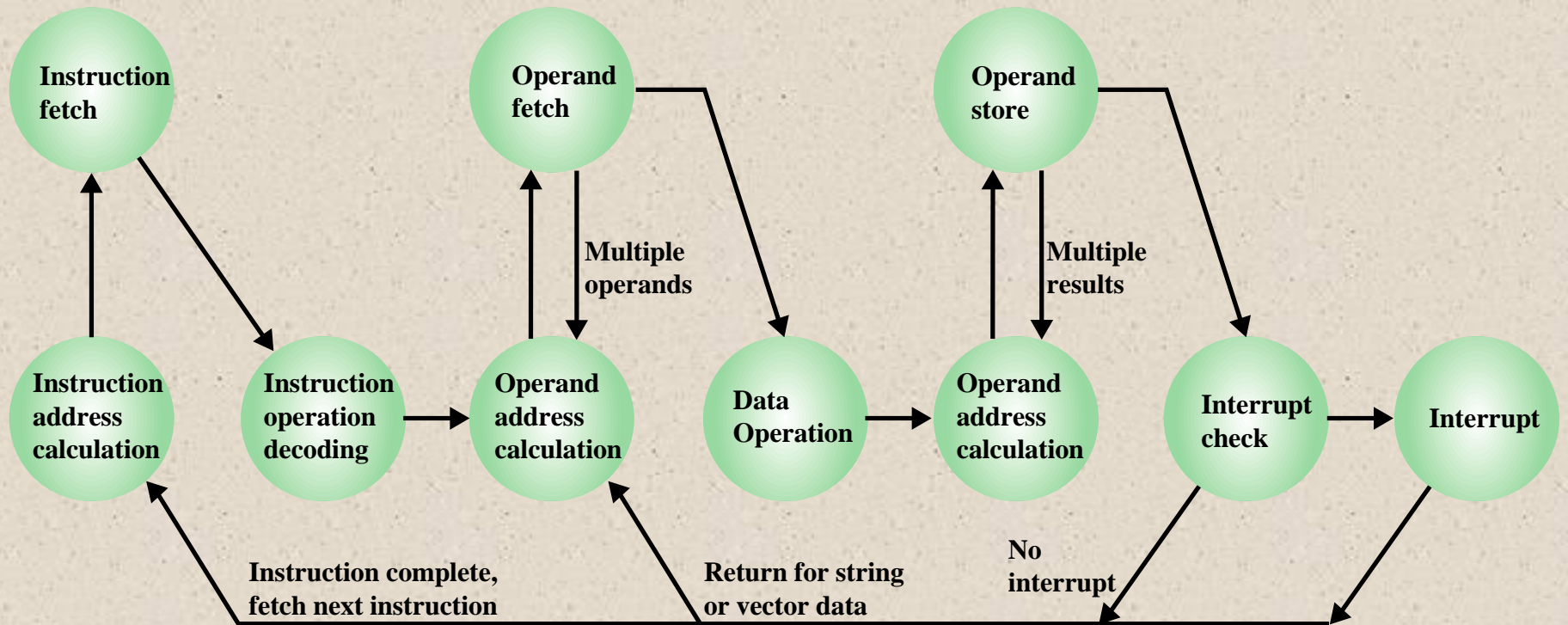


Figure 3.12 Instruction Cycle State Diagram, With Interrupts

+ Multiple Interrupts

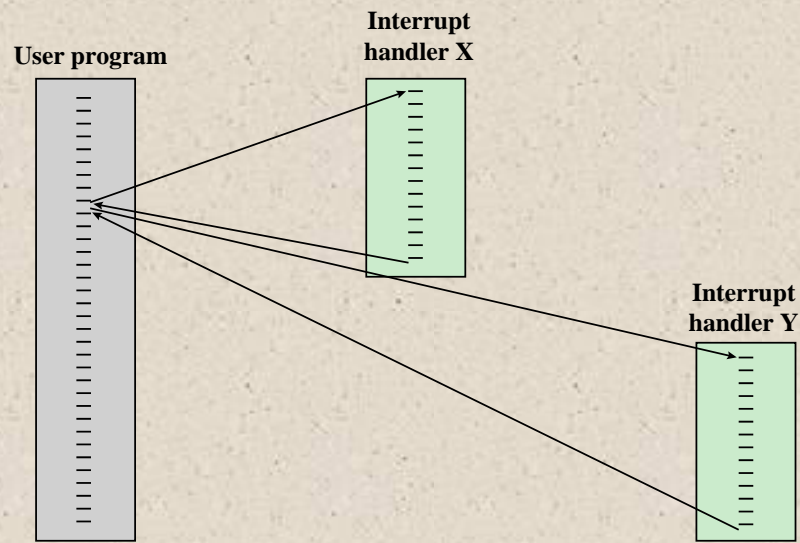


■ Disable interrupt

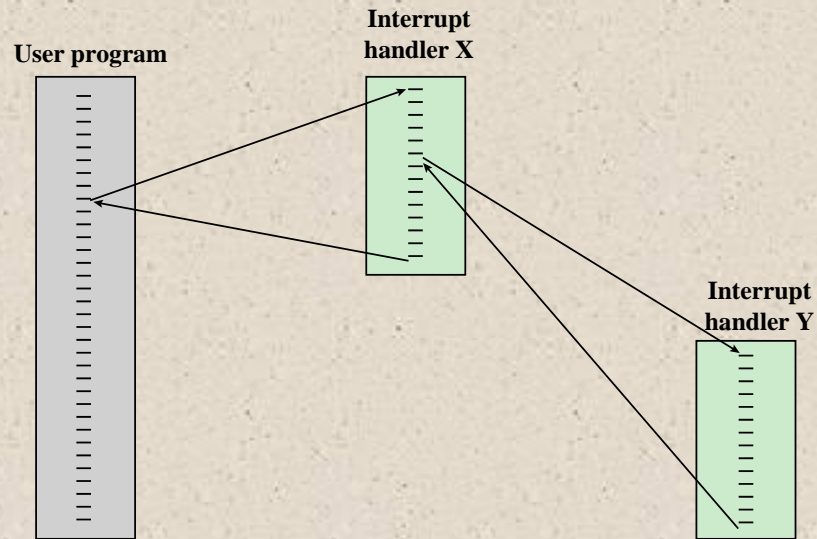
- ✓ processor will ignore further interrupts whilst processing one interrupt
- ✓ Interrupts remain pending and are checked after first interrupt has been processed
- ✓ Interrupts handled in sequence as they occur

■ Define priorities

- ✓ Low priority interrupts can be interrupted by higher priority interrupts
- ✓ When higher priority interrupt has been processed processor returns to previous interrupt



(a) Sequential interrupt processing



(b) Nested interrupt processing

Figure 3.13 Transfer of Control with Multiple Interrupts

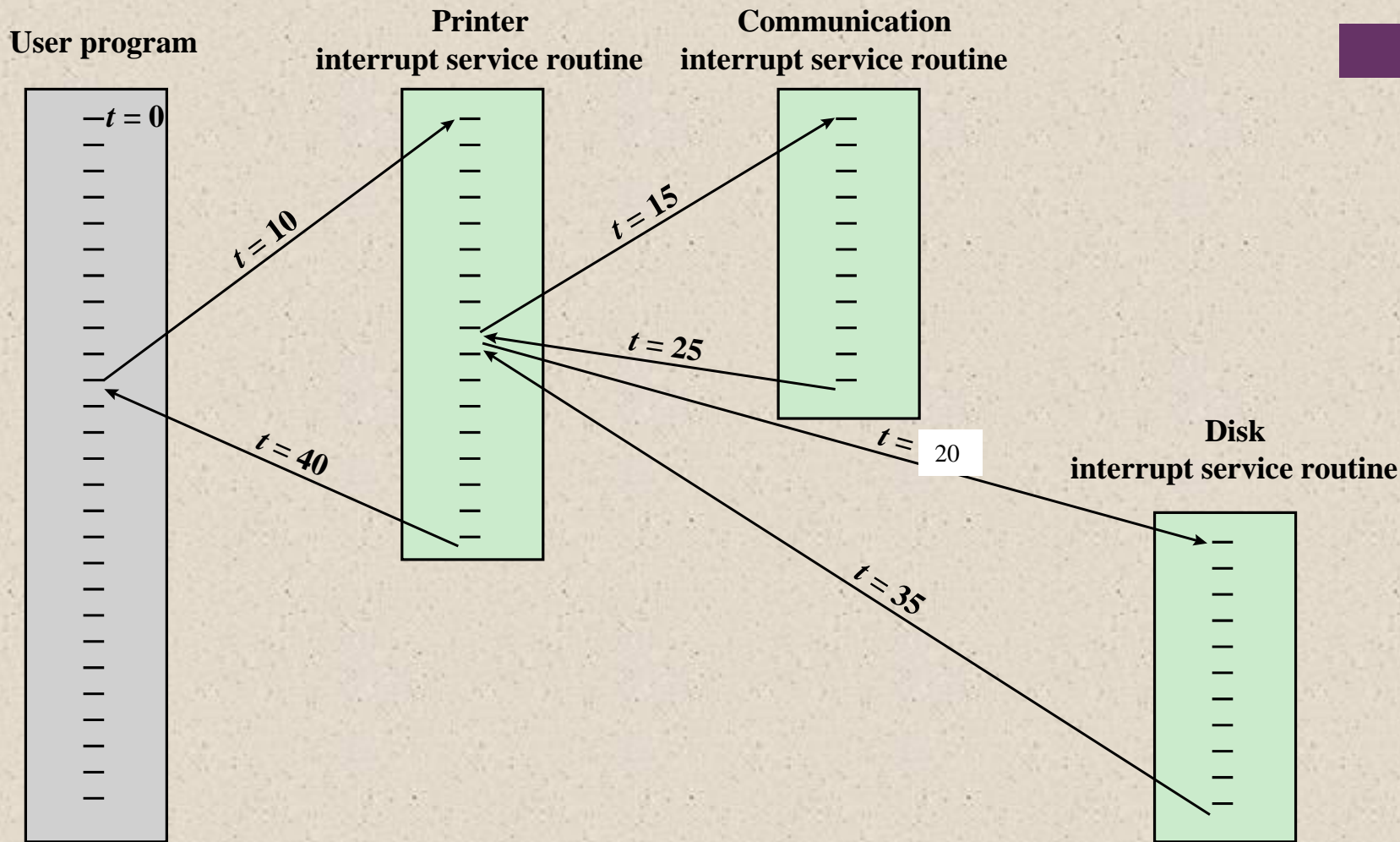


Figure 3.14 Example Time Sequence of Multiple Interrupts



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)

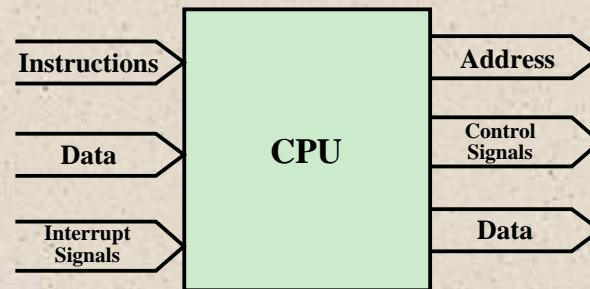
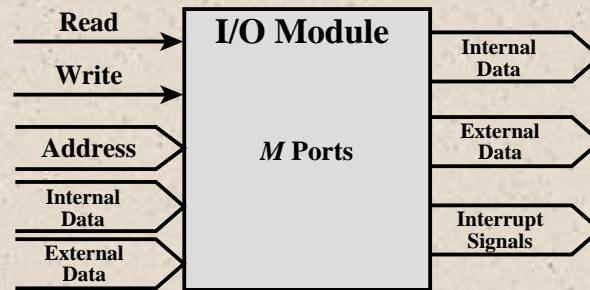
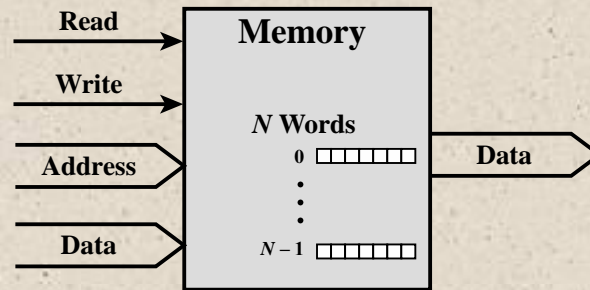


Figure 3.15 Computer Modules

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**

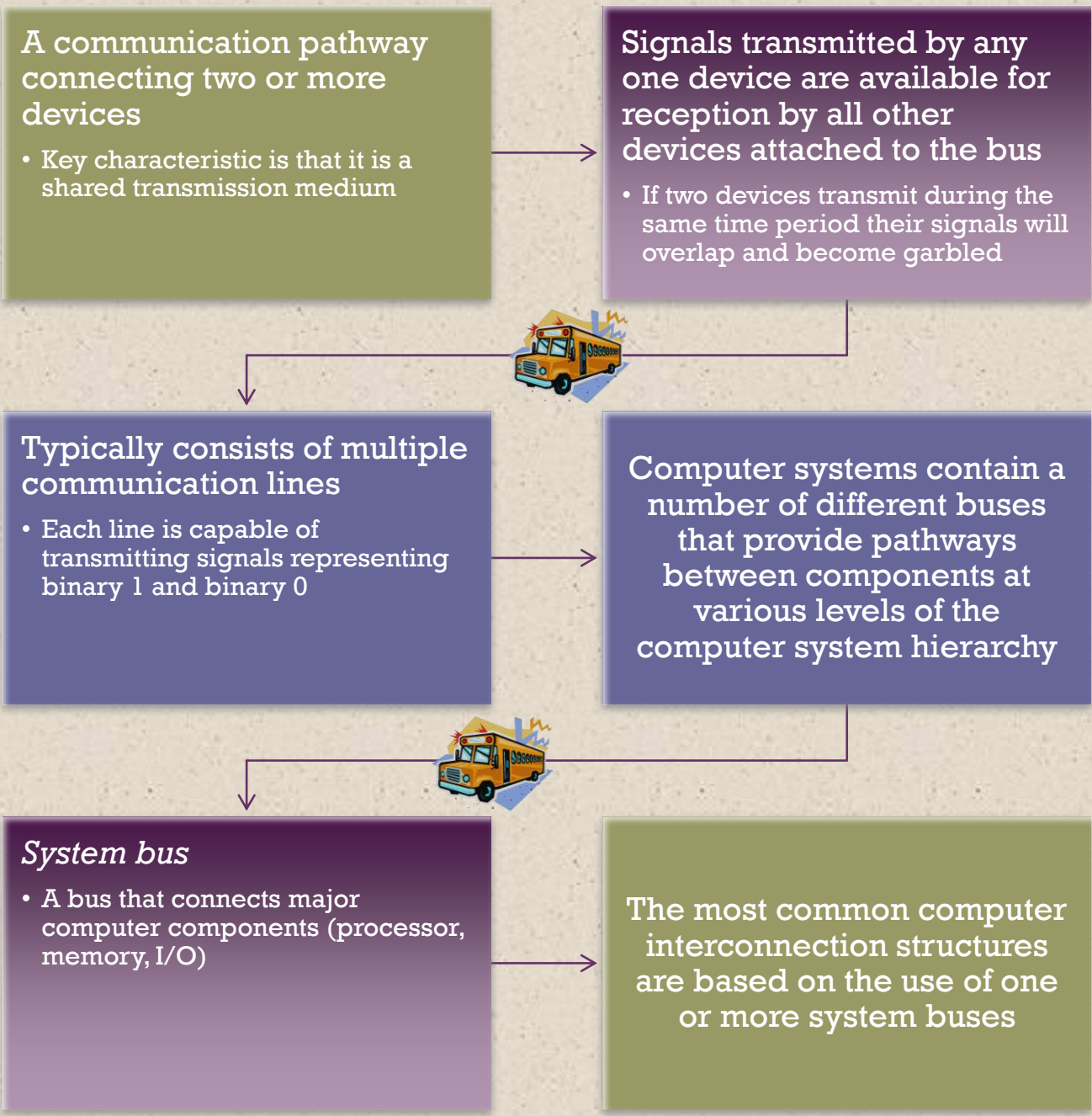
**Processor
to I/O**

**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**

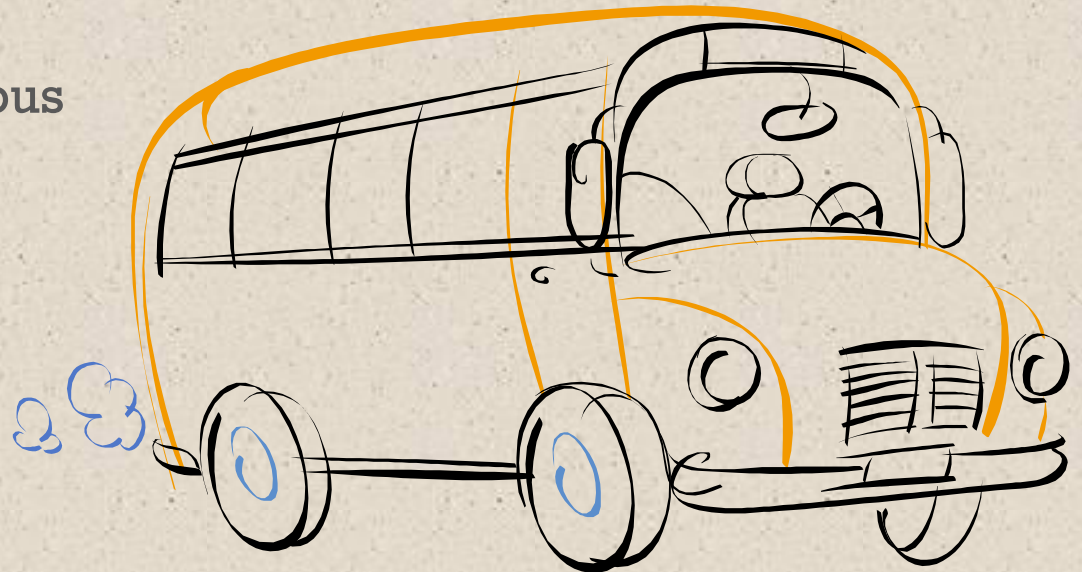
Interconnectivity Bus Connections



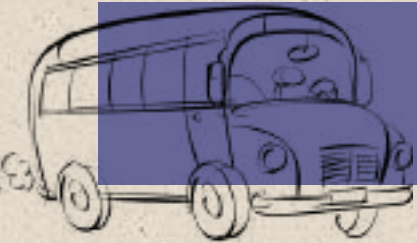
Data Bus



- 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

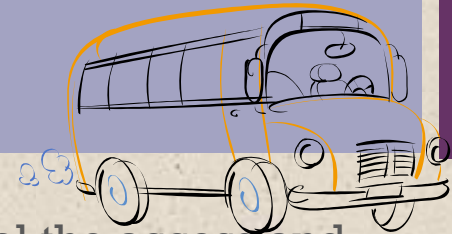


+ Address Bus



- 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

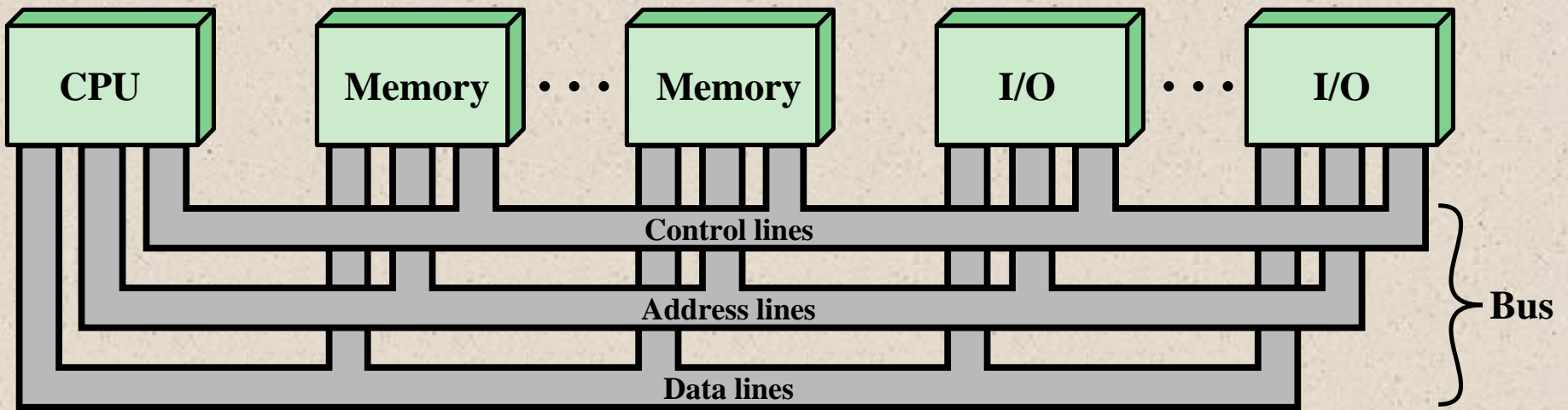


Figure 3.16 Bus Interconnection Scheme



Point-to-Point Interconnect



Principal reason for 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

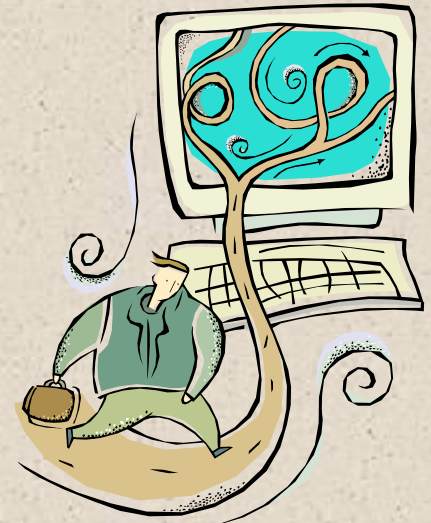
A conventional shared bus on the same chip magnified the difficulties of increasing bus data rate and reducing bus latency to keep up with the processors

Has lower latency, higher data rate, and better scalability

+ Quick Path Interconnect

- Introduced in 2008
- 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



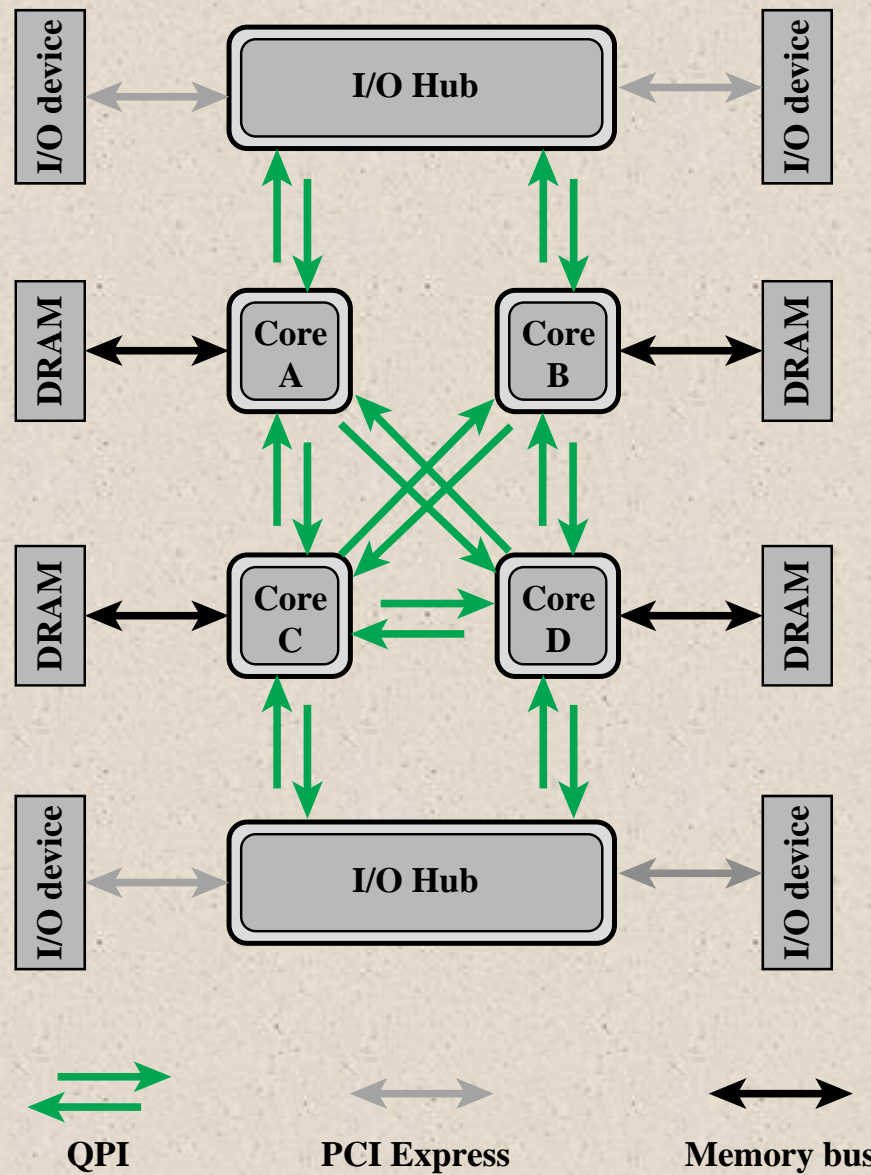


Figure 3.17 Multicore Configuration Using QPI

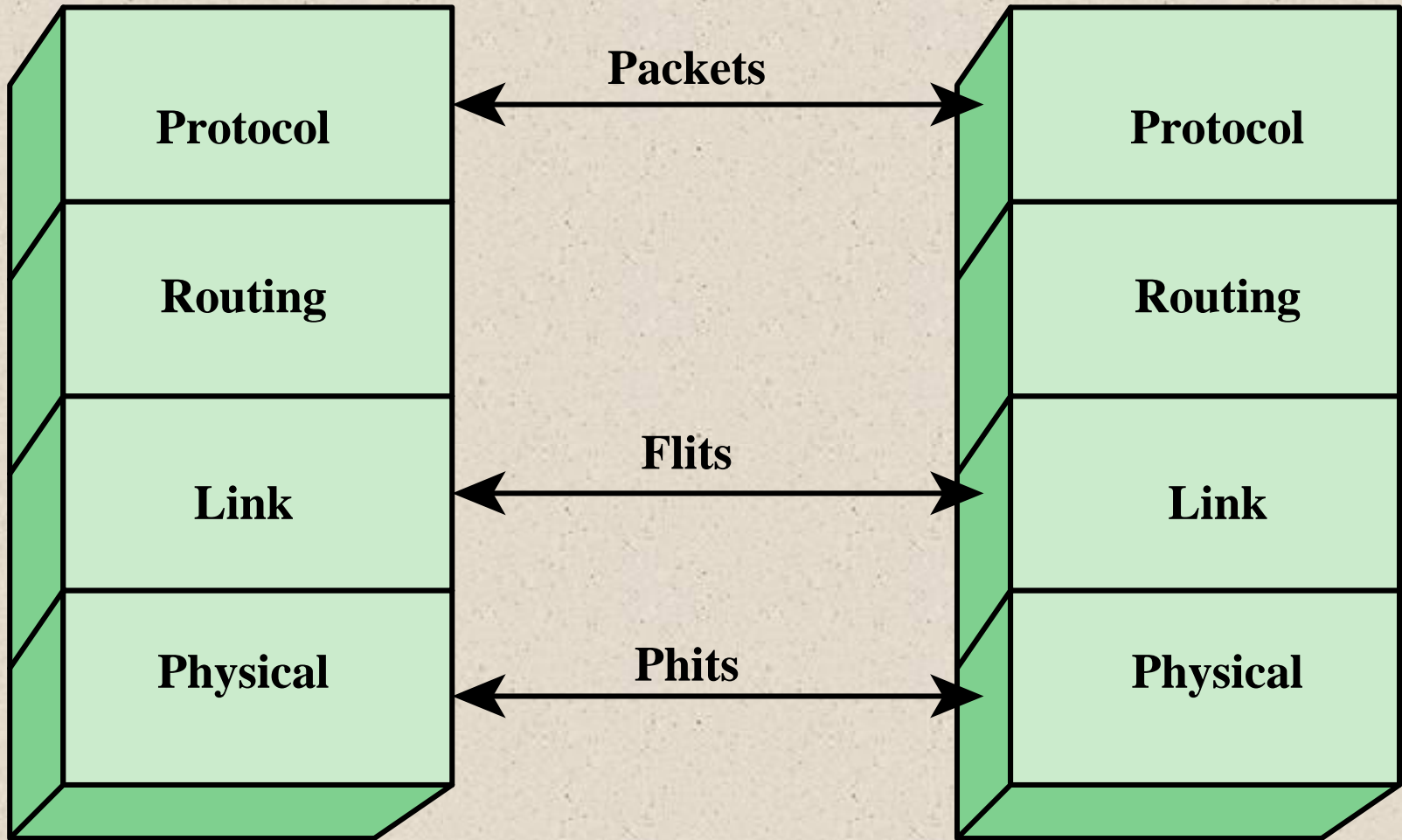


Figure 3.18 QPI Layers

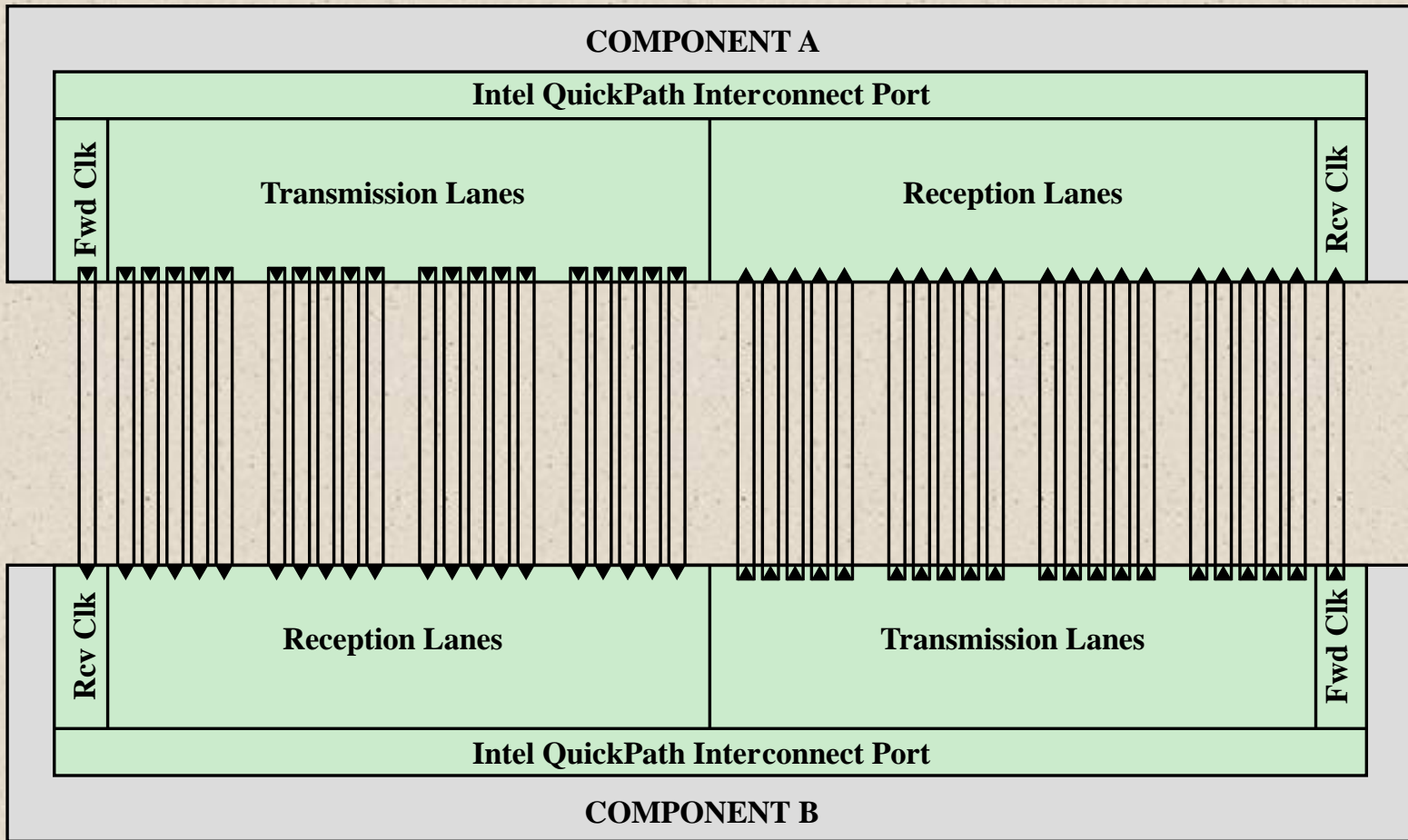


Figure 3.19 Physical Interface of the Intel QPI Interconnect

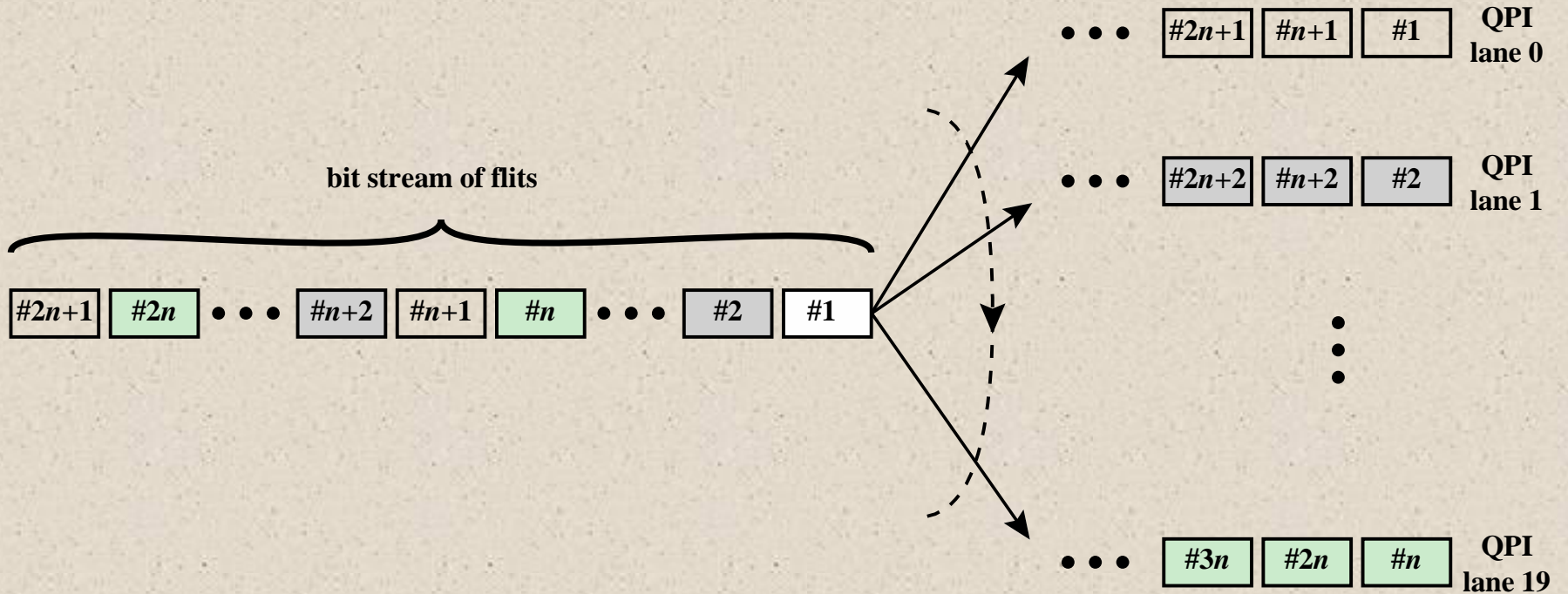


Figure 3.20 QPI Multilane Distribution



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

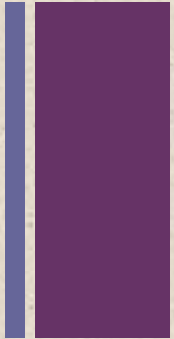
- Used to determine the course that a packet will traverse across the available system interconnects
- Defined by firmware and describe the possible paths that a packet can follow

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

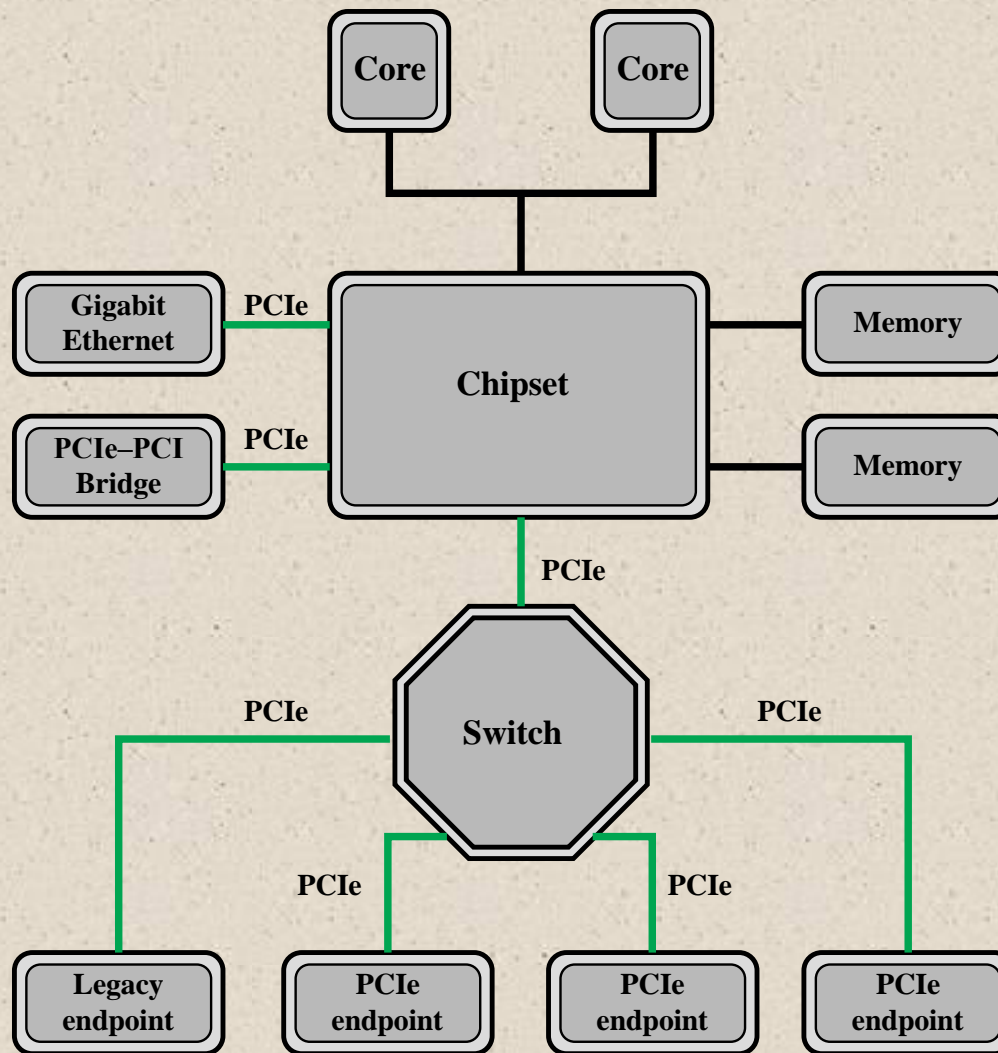


Figure 3.21 Typical Configuration Using PCIe

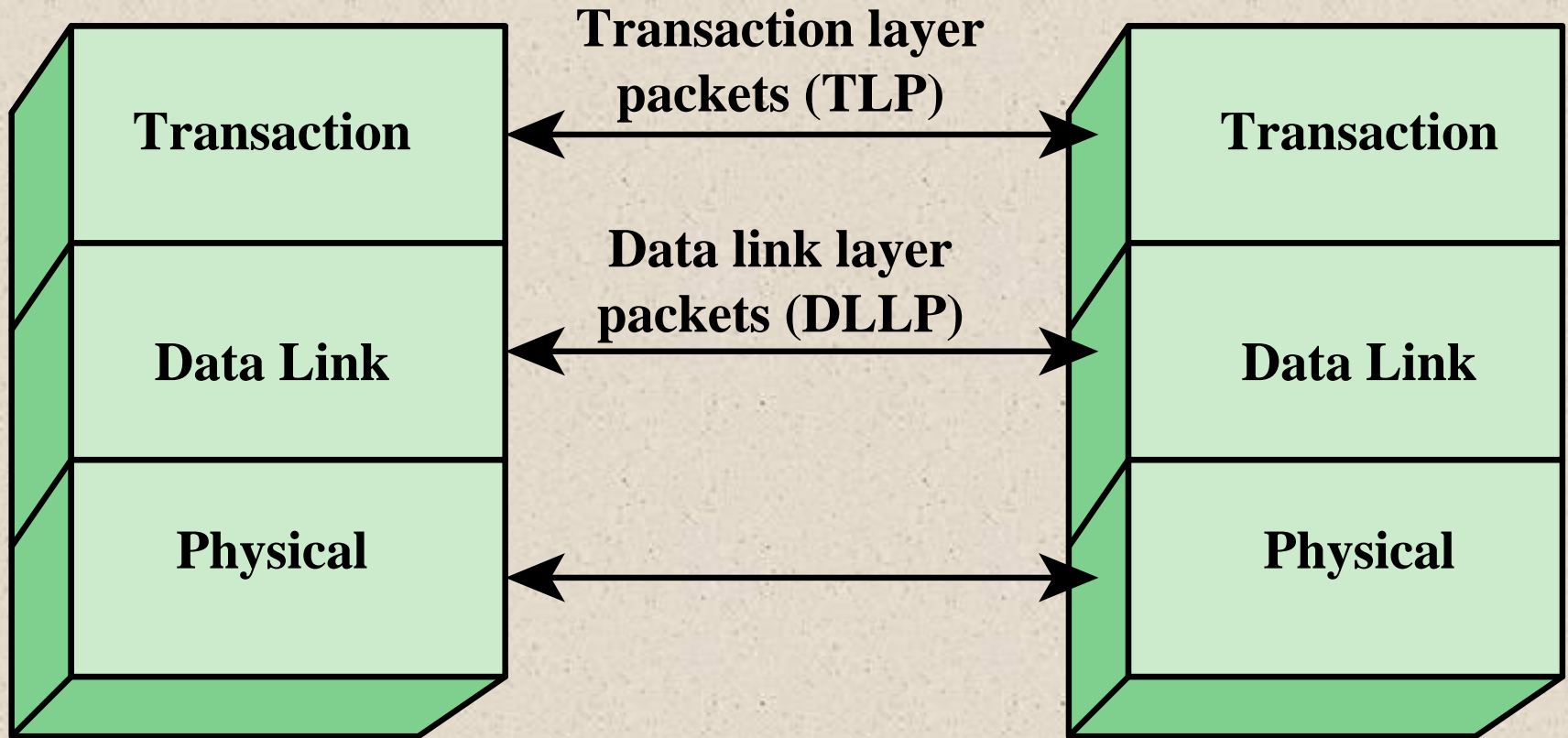


Figure 3.22 PCIe Protocol Layers

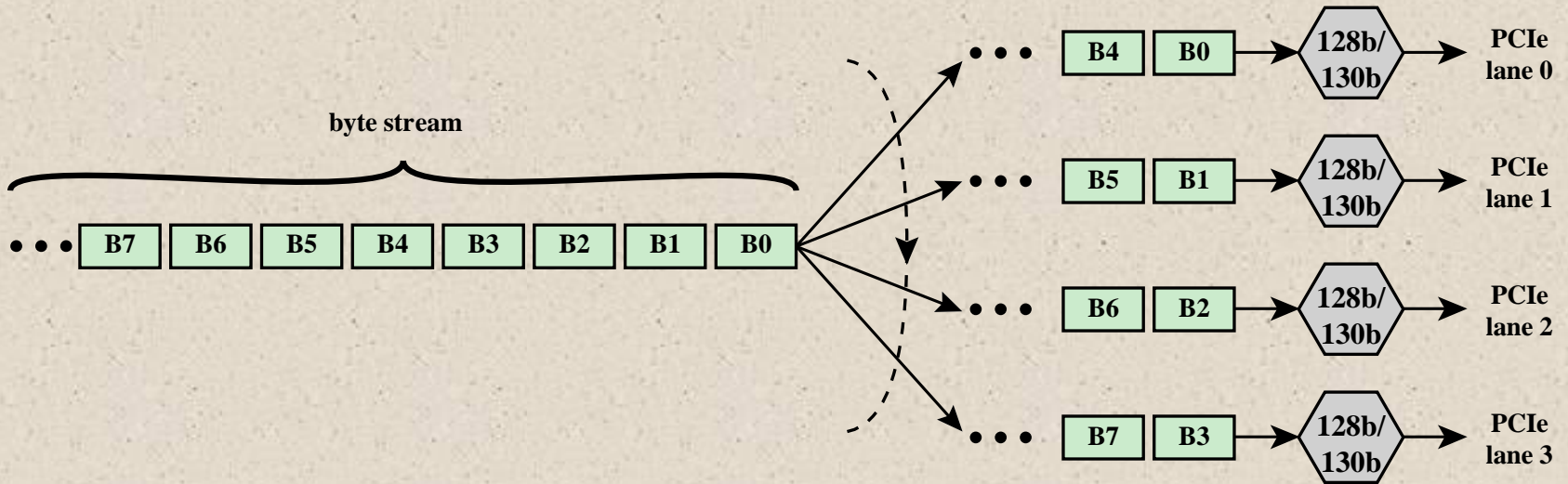


Figure 3.23 PCIe Multilane Distribution

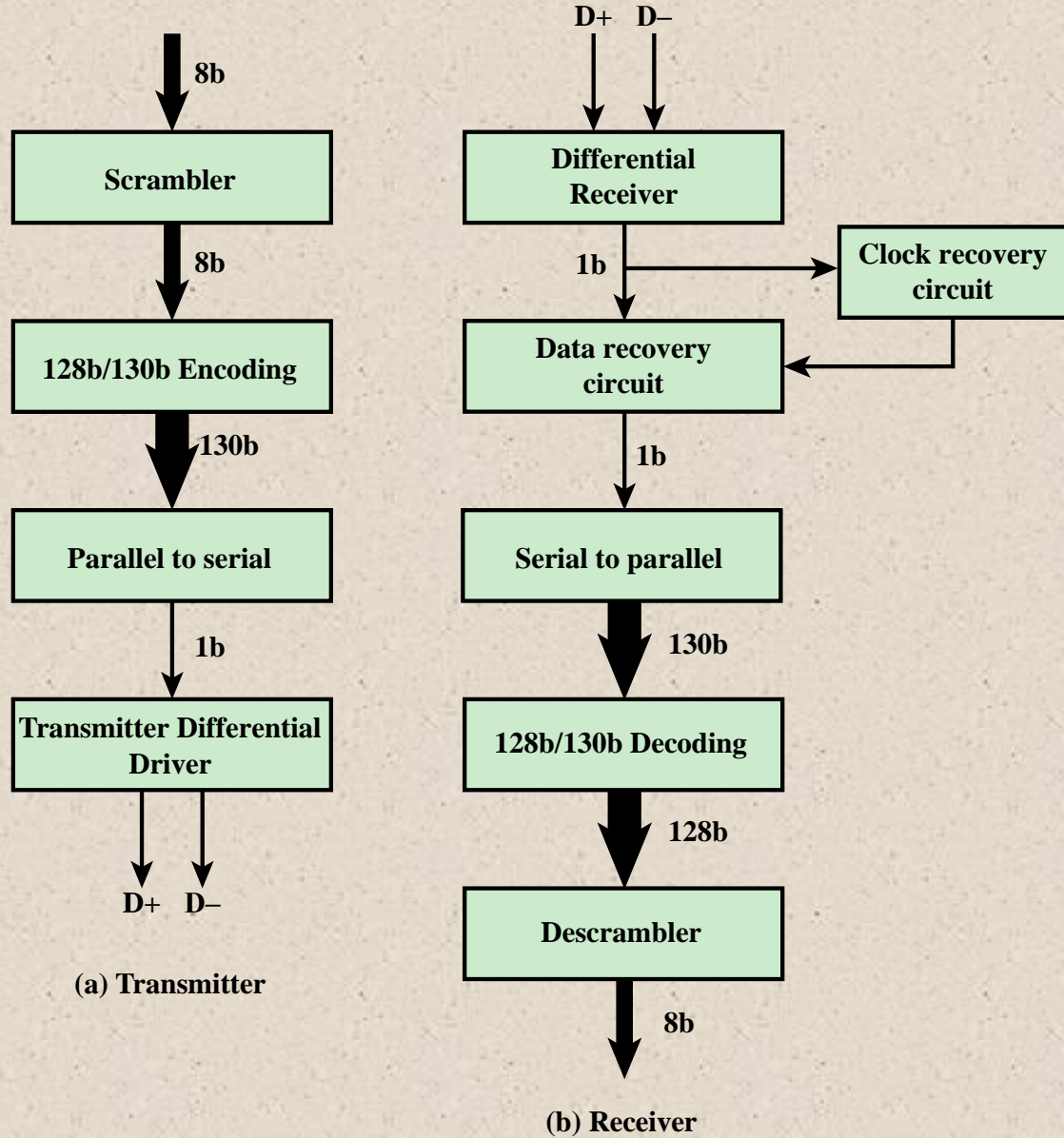


Figure 3.24 PCIe Transmit and Receive Block Diagrams



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



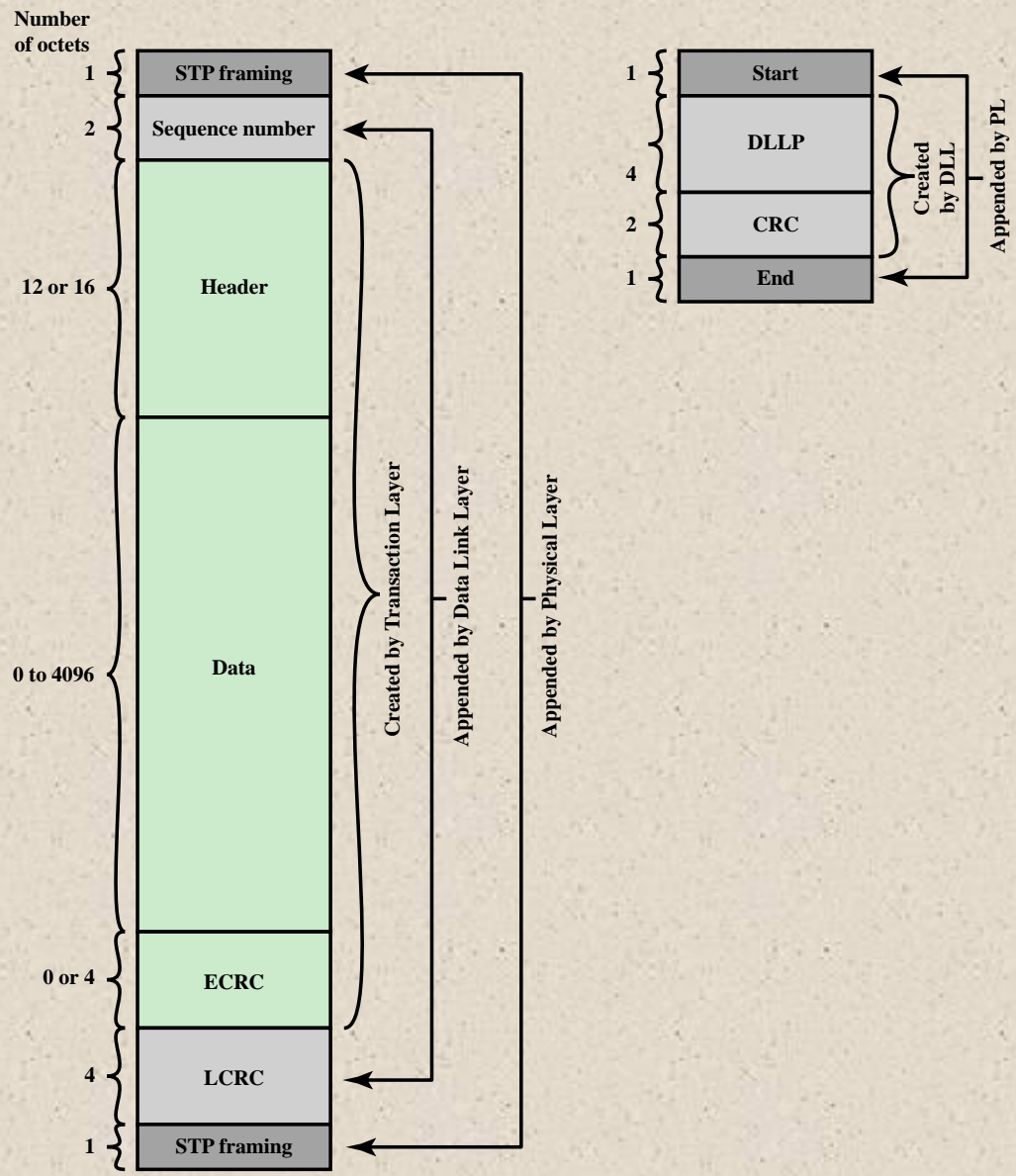
The TL supports four address spaces:

- **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

Table 3.2

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	



(a) Transaction Layer Packet

(b) Data Link Layer Packet

Figure 3.25 PCIe Protocol Data Unit Format

+ Summary

Chapter 3

- Computer components
- Computer function
 - Instruction fetch and execute
 - Interrupts
 - I/O function
- Interconnection structures
- Bus interconnection

A Top-Level View of Computer Function and Interconnection

- Point-to-point interconnect
 - QPI physical layer
 - QPI link layer
 - QPI routing layer
 - QPI protocol layer
- PCI express
 - PCI physical and logical architecture
 - PCIe physical layer
 - PCIe transaction layer
 - PCIe data link layer



**William Stallings
Computer Organization
and Architecture
10th Edition**



+ Chapter 4

Cache Memory



<p>Location</p> <ul style="list-style-type: none">Internal (e.g. processor registers, cache, main memory)External (e.g. optical disks, magnetic disks, tapes) <p>Capacity</p> <ul style="list-style-type: none">Number of wordsNumber of bytes <p>Unit of Transfer</p> <ul style="list-style-type: none">WordBlock <p>Access Method</p> <ul style="list-style-type: none">SequentialDirectRandomAssociative	<p>Performance</p> <ul style="list-style-type: none">Access timeCycle timeTransfer rate <p>Physical Type</p> <ul style="list-style-type: none">SemiconductorMagneticOpticalMagneto-optical <p>Physical Characteristics</p> <ul style="list-style-type: none">Volatile/nonvolatileErasable/nonerasable <p>Organization</p> <ul style="list-style-type: none">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:

- Whenever any access request arrives, the memory is searched from the beginning till the required data is found
- Search begins at the first memory location and proceeds sequentially moving ahead one step at a time till the desired location is reached
- Time taken to access any location is the variable.
- Simple but slow technique
- Tapes is an example of sequential access.



Method of Accessing Units of Data



■ Random Access:

- Data represent at any memory location can be accessed directly
- Time taken to access any location is the same.
- Faster way to retrieve data.
- RAM is an example of this kind of access.

■ Direct Access:

- it's semi-random mode of operation in which data is stored in blocks /tracks which can be accessed randomly.
- Unlike random access the time taken to access any location may not be the same
- Used by magnetic disk

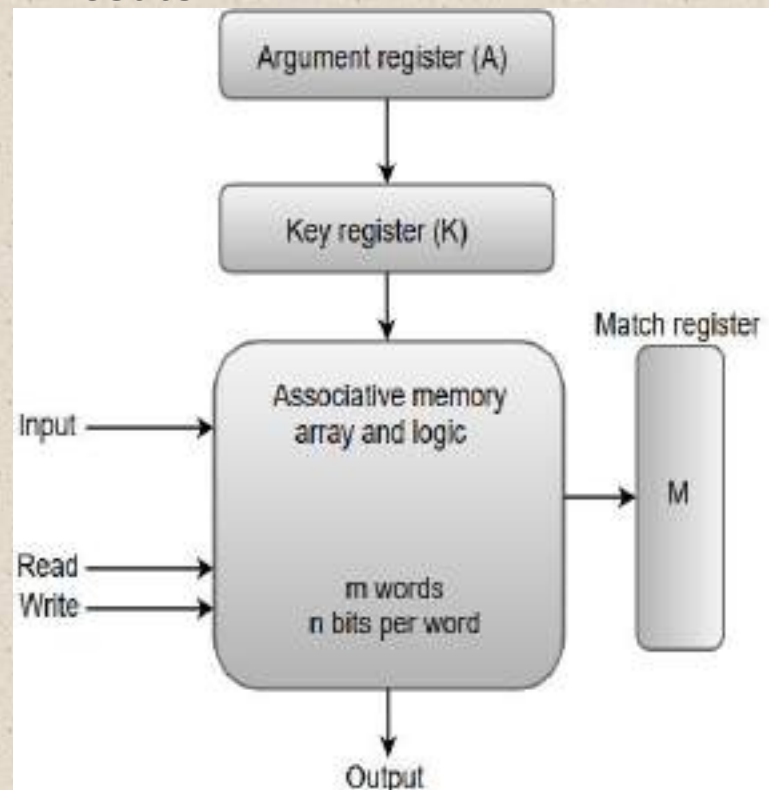
+ Method of Accessing Units of Data

■ Associative:

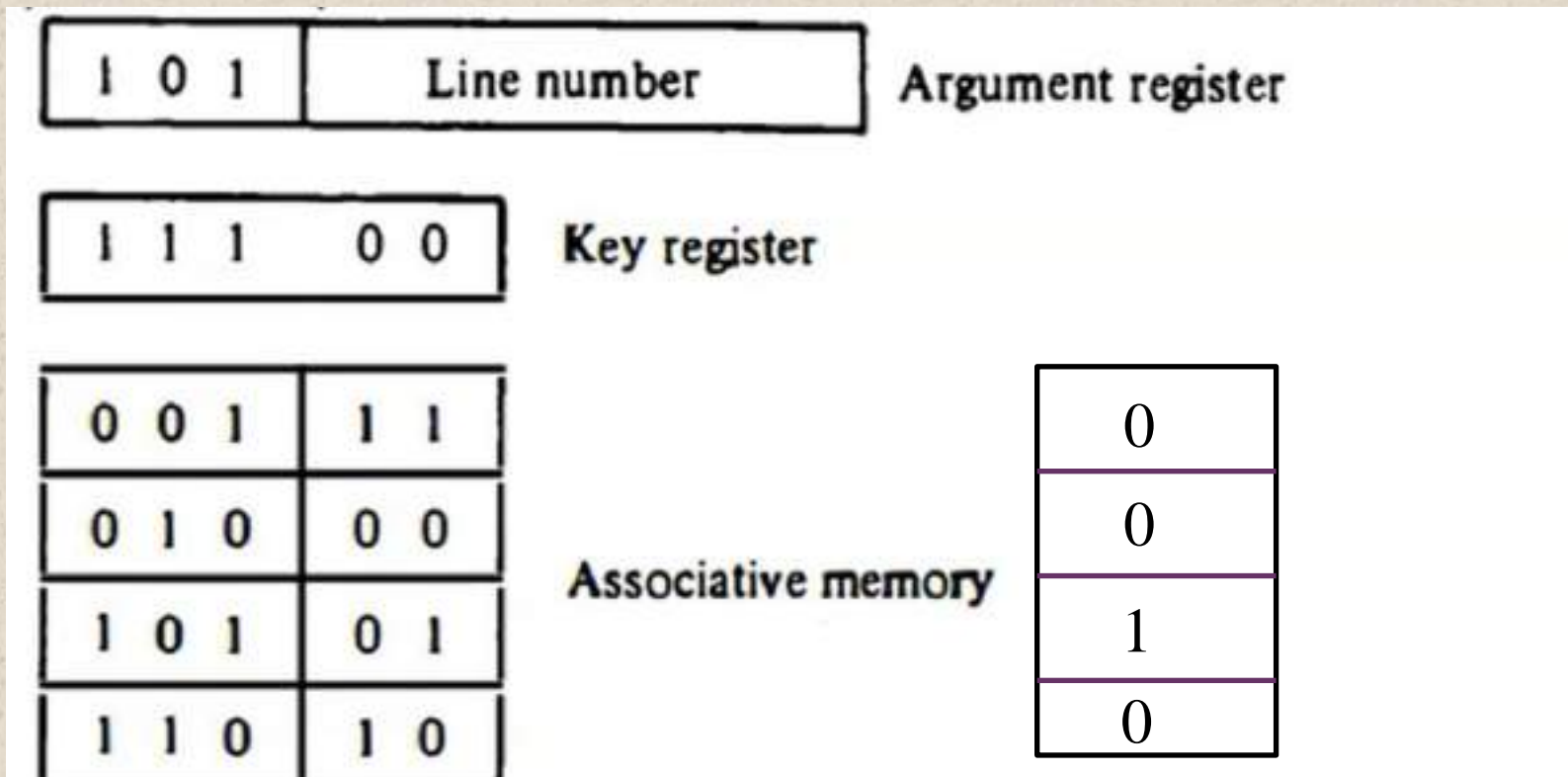
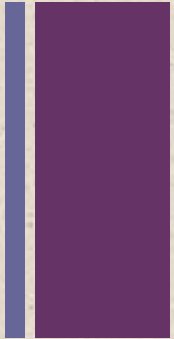
- When data is accessed by data content rather than data address, then the memory is referred to as associative memory or content addressable memory (CAM).
- Data is stored at the very first empty location found in memory.
- In associative memory when data is stored at a particular location then no address is stored along with it.
- When the stored data need to be searched then only the key(i.e. data or part of data) is provided.
- A sequential search is performed in the memory using the specified key to find out the matching key from the memory.

+ Block Diagram of Associative

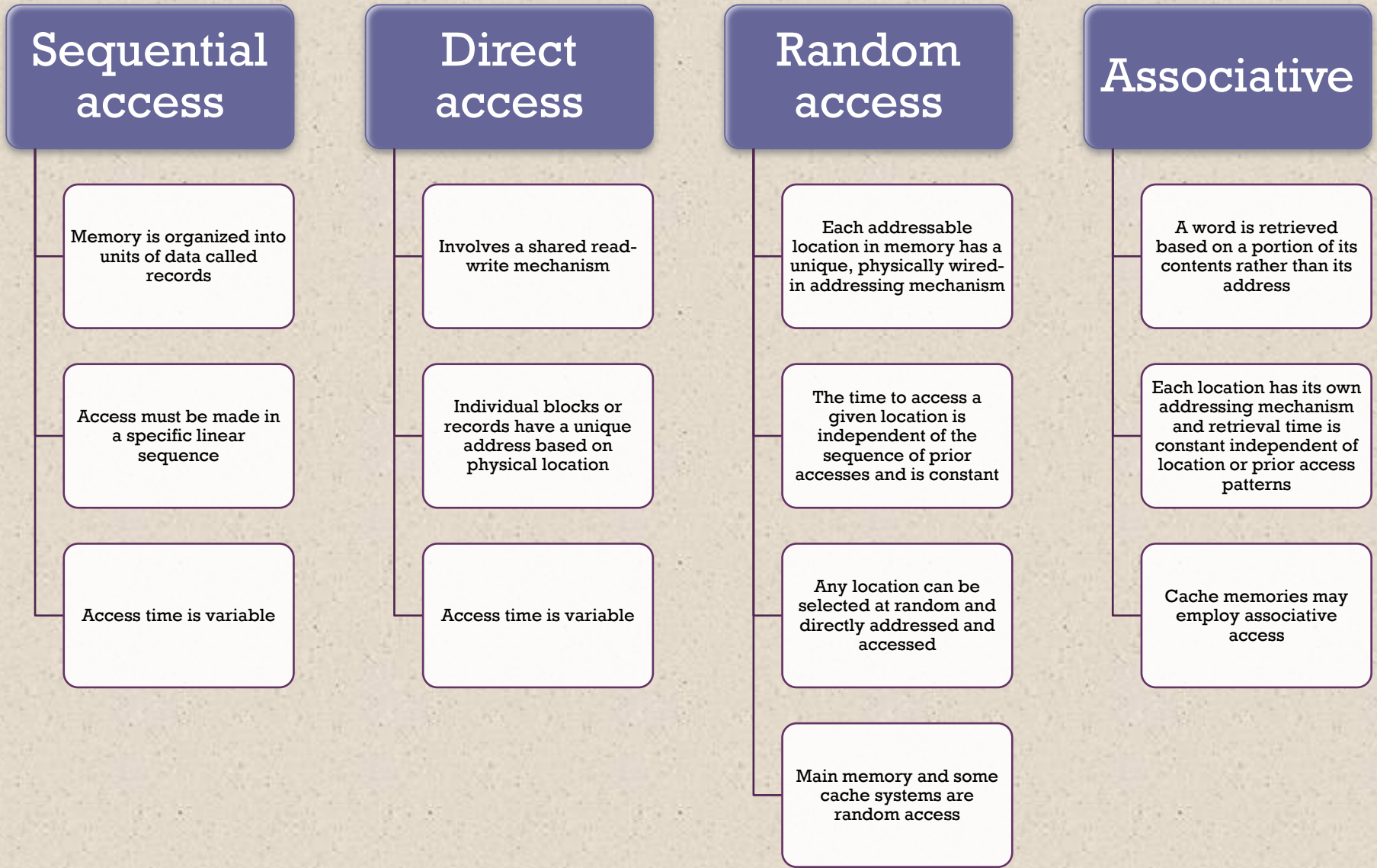
- **Argument Register (A):** It contains the word to be searched.
- **Key register (K):** This specifies which part of the argument word needs to be compared with words in memory. If all bits in register are 1, the entire word should be compared else only the bits having 1 will be compared.
- **Associative memory array:** it contains the words which are to be compared with the argument word
- **Match register (M):** After the matching process the bits corresponding to matching words in match register are set to 1



+ Associative example



Method of Accessing Units of Data



Capacity and Performance:

The two most important characteristics of memory

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

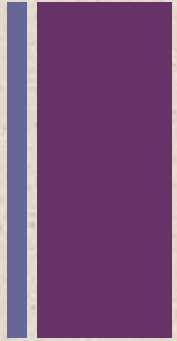
Memory cycle time

- 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 $1/(\text{cycle time})$

+ Memory

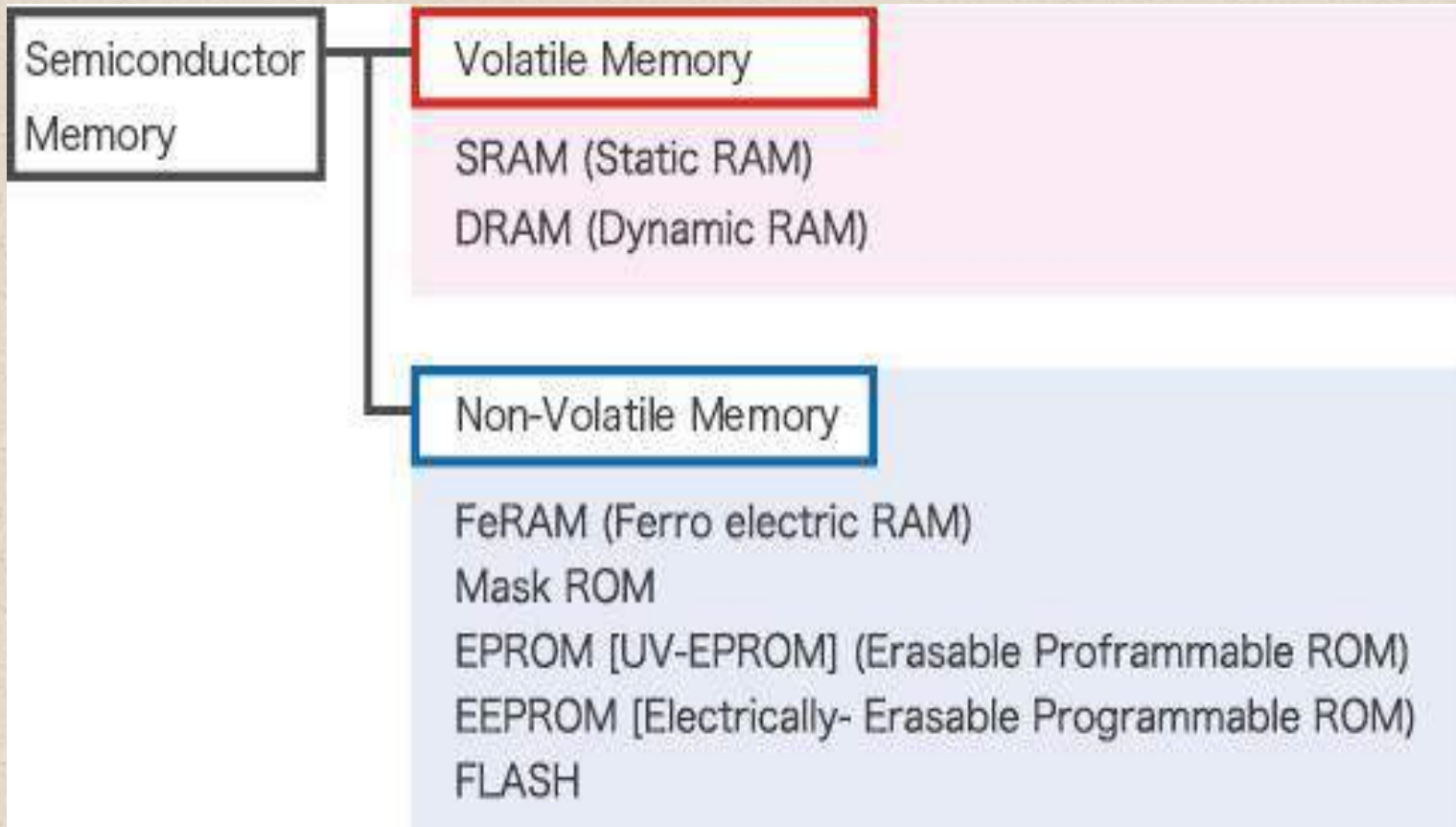


- The most common forms are:
 - Semiconductor memory
 - Magnetic surface memory
 - Optical
 - Magneto-optical

- Several physical characteristics of data storage are important:
 - 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)

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

+ Semiconductor memory



+ 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

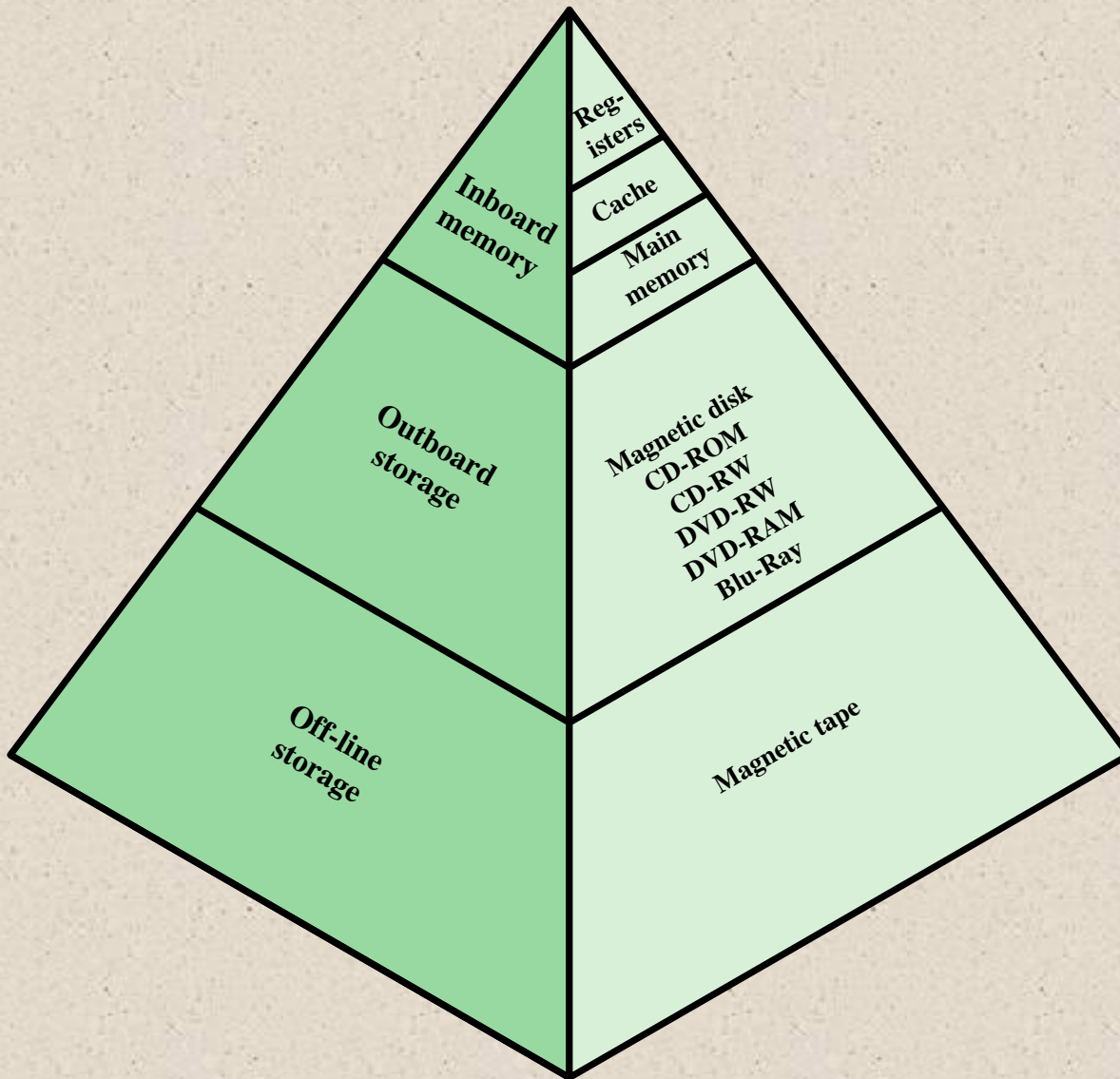
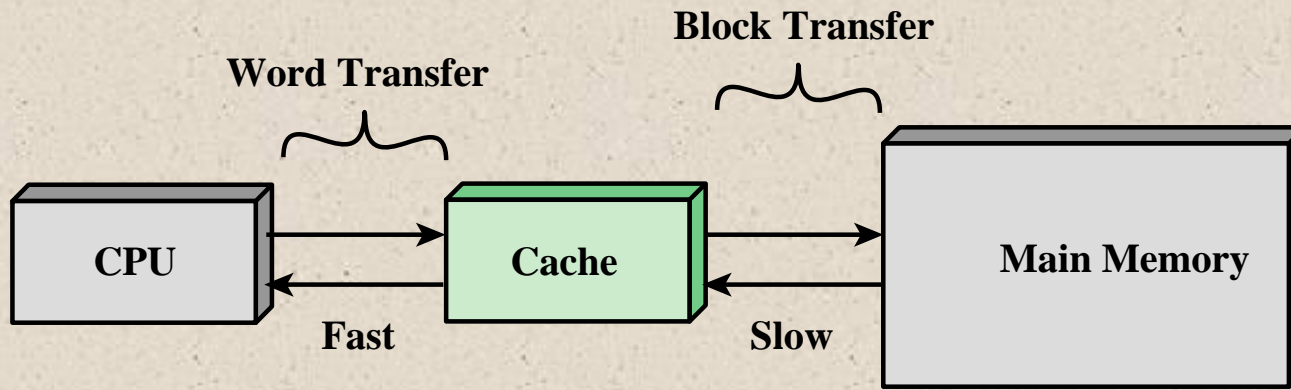


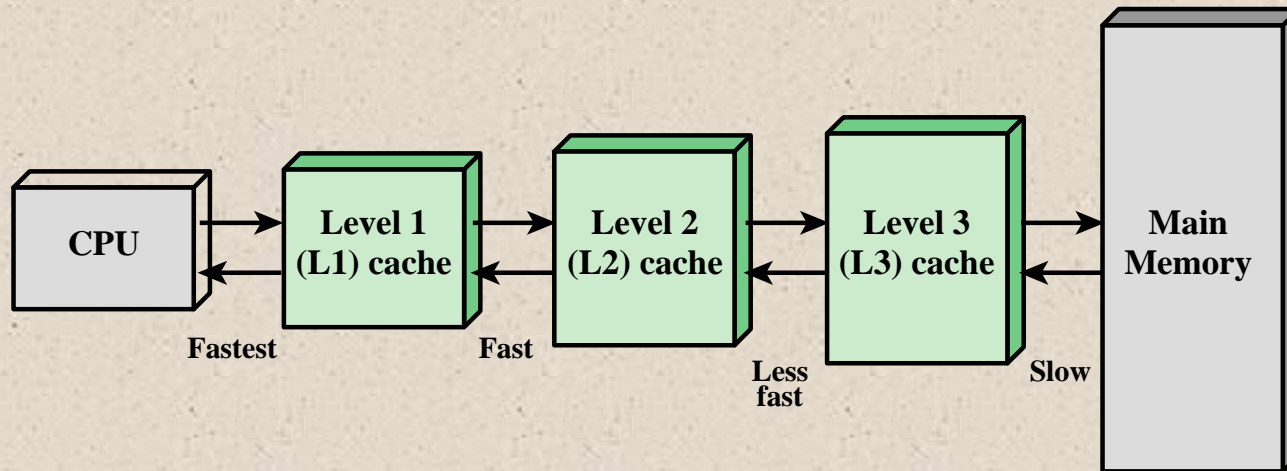
Figure 4.1 The Memory Hierarchy

+ Memory

- The use of three levels exploits the fact that semiconductor memory comes in a variety of types which differ in speed and cost
- Data are stored more permanently on external mass storage devices
- External, nonvolatile memory is also referred to as **secondary** memory or **auxiliary** memory
- Disk cache
 - A portion of main memory can be used as a buffer to hold data temporarily that is to be read out to disk
 - A few large transfers of data can be used instead of many small transfers of data
 - Data can be retrieved rapidly from the software cache rather than slowly from the disk



(a) Single cache



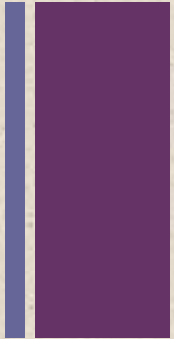
(b) Three-level cache organization

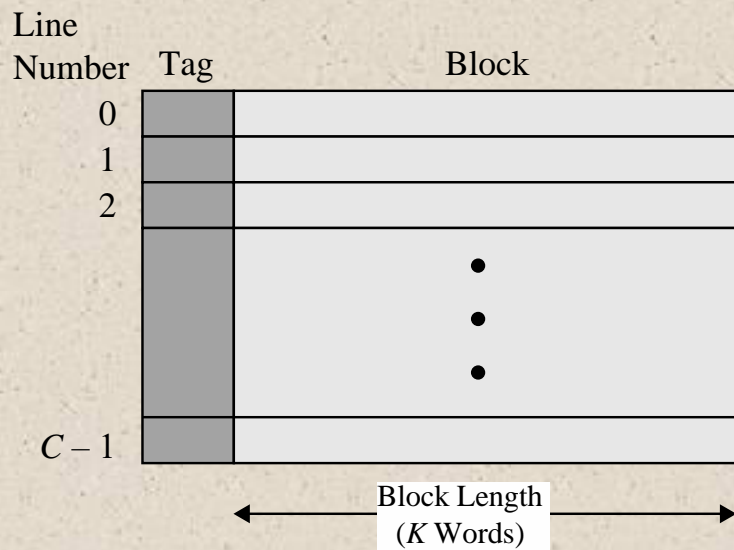
Figure 4.3 Cache and Main Memory



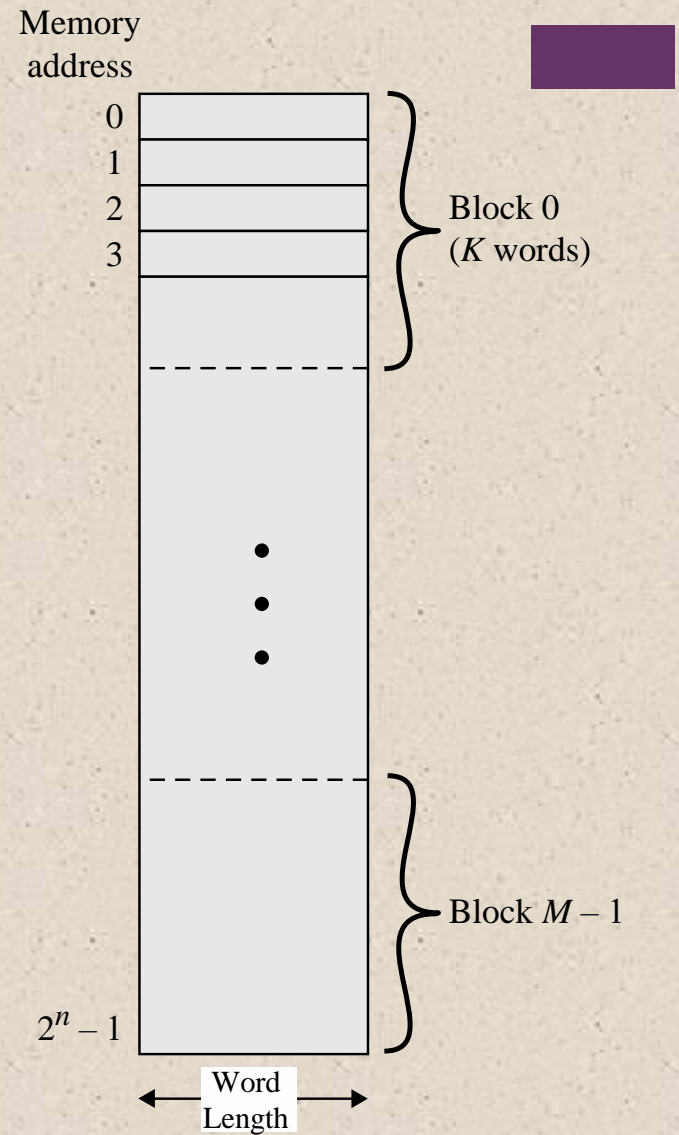
- In Figure 4.3a. The cache contains a copy of portions of main memory. When the processor attempts to read a word of memory, a check is made to determine if the word is in the cache. If so, the word is delivered to the processor. If not, a block of main memory, consisting of some fixed number of words, is read into the cache and then the word is delivered to the processor. Because of the phenomenon of locality of reference, when a block of data is fetched into the cache to satisfy a single memory reference, it is likely that there will be future references to that same memory location or to other words in the block

- Figure 4.3b depicts the use of multiple levels of cache. The L2 cache is slower and typically larger than the L1 cache, and the L3 cache is slower and typically larger than the L2 cache.





(a) Cache



(b) Main memory

Figure 4.4 Cache/Main-Memory Structure

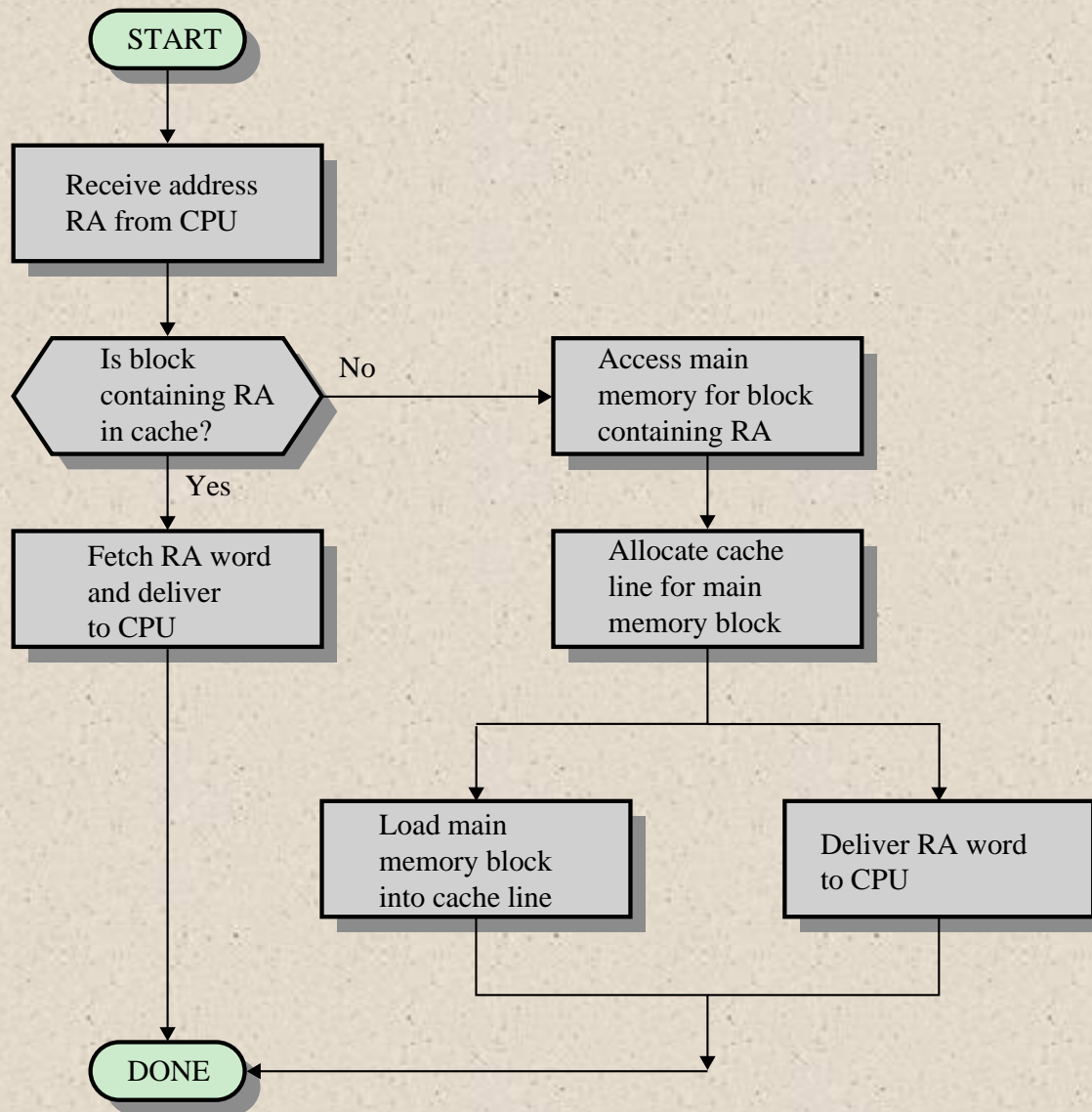


Figure 4.5 Cache Read Operation

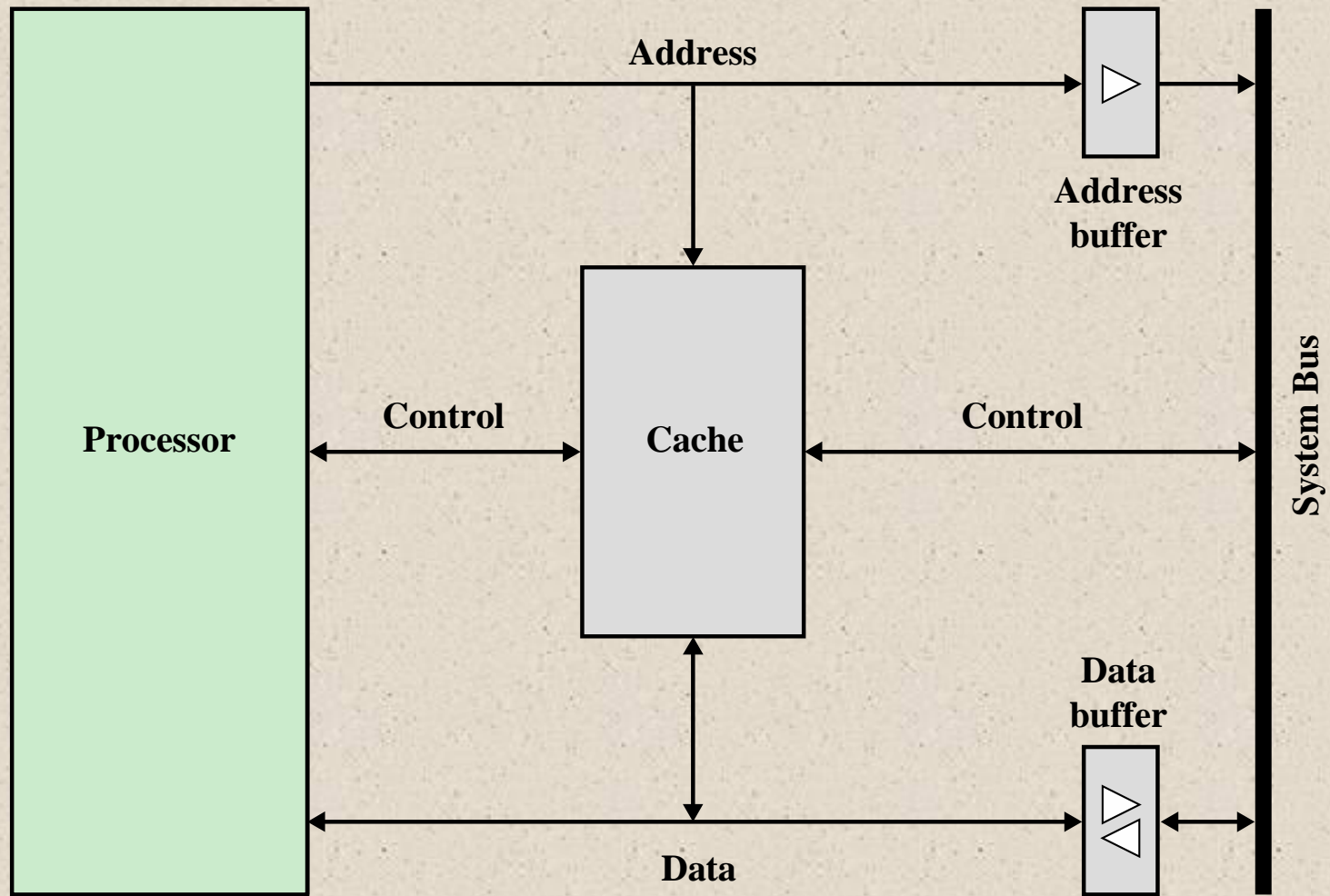


Figure 4.6 Typical Cache Organization

+ Cache organization

- In figure 4.6, the cache connects to the processor via data, control, and address lines. The data and address lines also attach to data and address buffers, which attach to a system bus from which main memory is reached.
- When a **cache hit** occurs, the data and address buffers are disabled and communication is only between processor and cache, with no system bus traffic.
- When a **cache miss** occurs, the desired address is loaded onto the system bus and the data are returned through the data buffer to both the cache and the processor.
- In other organizations, the cache is physically interposed between the processor and the main memory for all data, address, and control lines. In this latter case, for a cache miss, the desired word is first read into the cache and then transferred from cache to processor.

Cache Addresses

Logical

Physical

Cache Size

Mapping Function

Direct

Associative

Set Associative

Replacement Algorithm

Least recently used (LRU)

First in first out (FIFO)

Least frequently used (LFU)

Random

Write Policy

Write through

Write back

Line Size

Number of caches

Single or two level

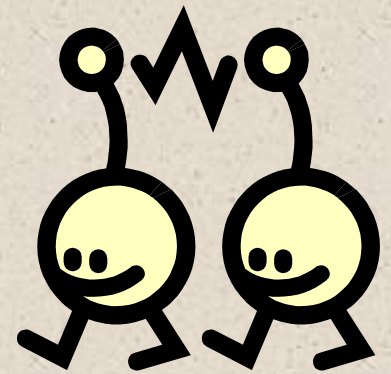
Unified or split

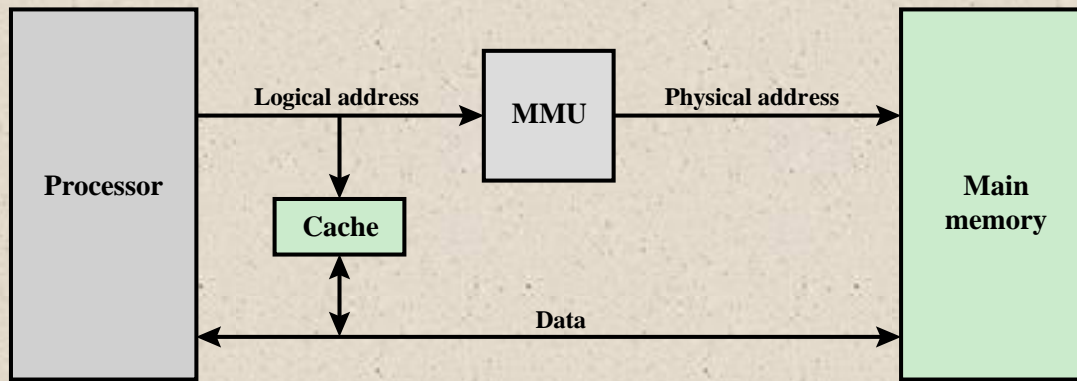
Table 4.2
Elements of Cache Design

+ Cache Addresses

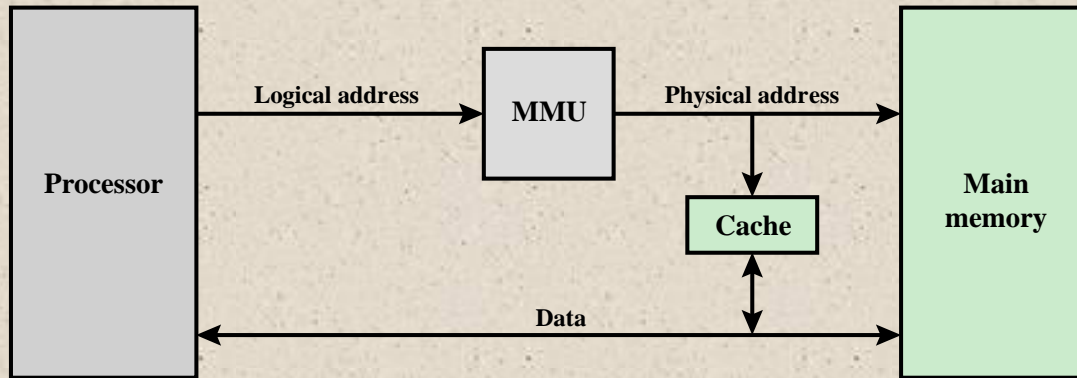
Virtual Memory

- 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





(a) Logical Cache



(b) Physical Cache

Figure 4.7 Logical and Physical Caches

+ Logical and physical caches

- When virtual addresses are used, the system designer may choose to place the cache between the processor and the MMU or between the MMU and main memory (Figure 4.7).
- A **logical cache**, also known as a **virtual cache**, stores data using **virtual addresses**. The processor accesses the cache directly, without going through the MMU.
- A physical cache stores data using main memory **physical addresses**.
- One obvious advantage of the logical cache is that **cache access speed** is faster than for a physical cache, because the cache can respond before the MMU performs an address translation.
- While the disadvantage is, each application sees a virtual memory that starts at address 0. Thus, the same virtual address in two different applications refers to two different physical addresses, The cache memory must therefore be completely flushed with each application context switch



Cache size

- **Small enough** so that the overall average cost per bit is close to that of main memory alone and
- **Large enough** so that the overall average access time is close to that of the cache alone
- **The larger the cache**, the larger the number of gates involved in addressing the cache. The result is that large caches tend to be slightly slower than small ones – even when built with the same integrated circuit technology and put in the same place on chip and circuit board
- The **available chip and board area** also **limits** cache size
- Because the **performance of the cache** is very **sensitive** to the nature of the **workload**, it is **impossible** to arrive at a single '**optimum**' cache size

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:

1. Direct mapping :

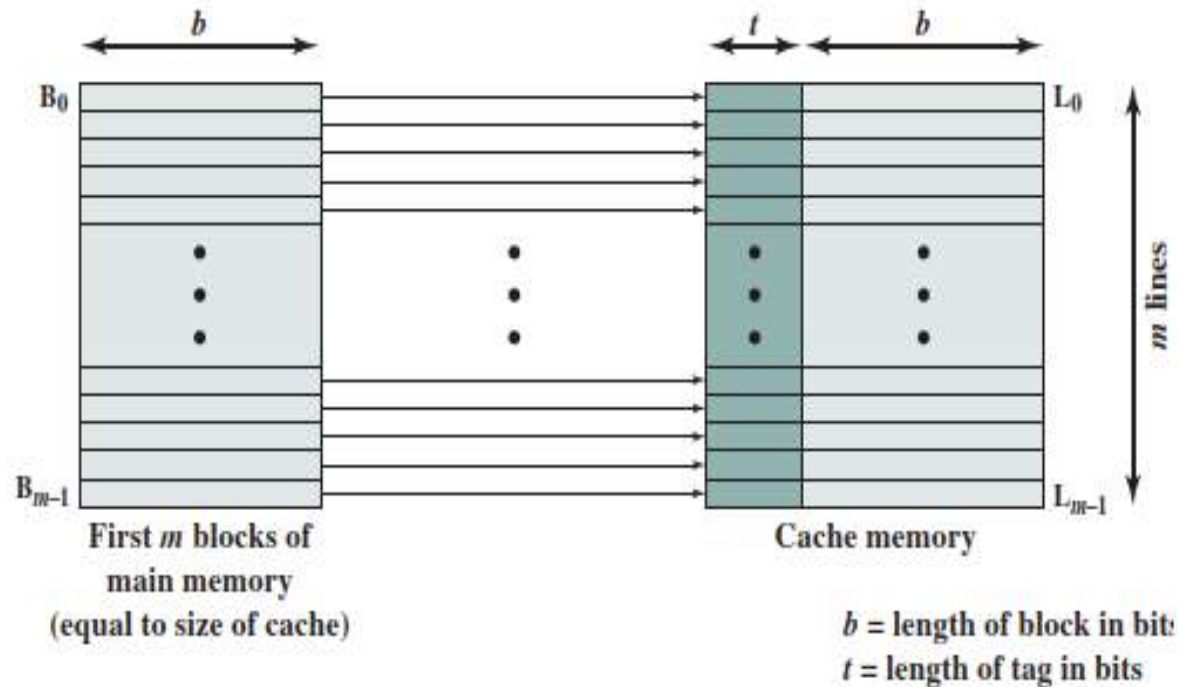
- The simplest technique, known as **direct mapping**, maps each block of main memory into only one possible cache line.
- The mapping is expressed as

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

Where : i = cache line number

j = main memory block number

m = number of lines in the cache



(a) Direct mapping

Figure 4.8 Mapping from Main Memory to Cache: Direct

Figure 4.8a shows the mapping for the first m blocks of main memory. Each block of main memory maps into one unique line of the cache. **The next m blocks** of main memory map into the cache in the same fashion; that is, block B_m of main memory maps into line L_0 of cache, block B_{m+1} maps into line L_1 , and so on.

+ Example



- If Cache size = 64 Kbytes and block size=4 bytes and memory size =16 Mbyte? Find total number of blocks in main memory and total cache lines within cache memory.

+ Direct Mapping

- Each block of main memory maps to only one cache line
- Address is two parts
 - **W**: least significant bits identify **unique word**
 - **S**: Most significant bits identify **one memory block**
 - **S-bits** are split into

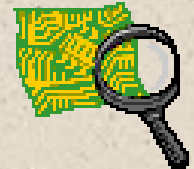
s-r: Tag bits

r: cache line

+ 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



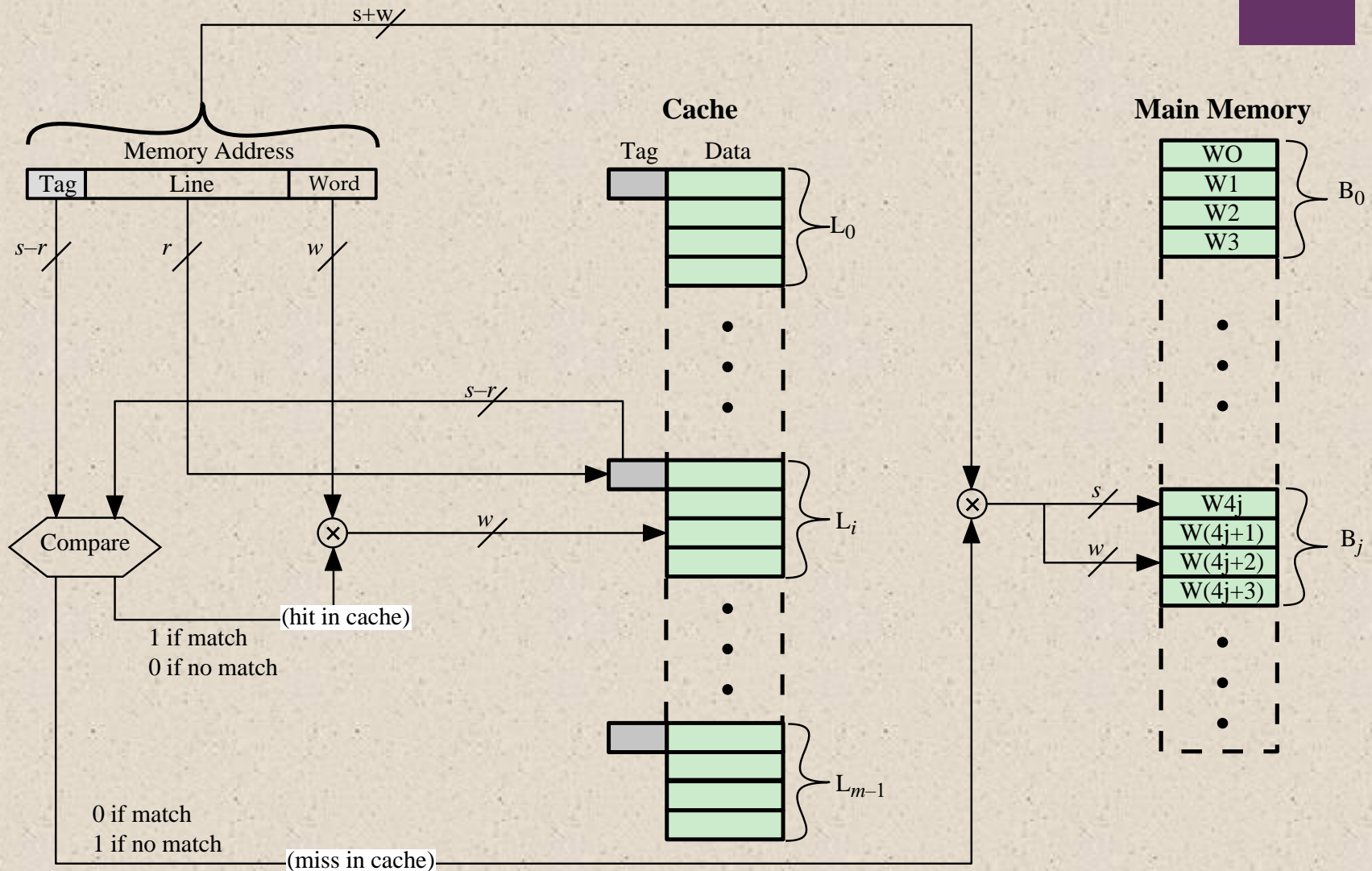


Figure 4.9 Direct-Mapping Cache Organization



Figure 4.9 illustrates the general mechanism .

For purposes of cache access, each main memory address can be viewed as consisting of three fields.

The least significant w bits identify a unique word or byte within a block of main memory; in most contemporary machines, the address is at the byte level .

The remaining s bits specify one of the 2^s blocks of main memory.

The cache logic interprets these s bits as a tag of $s - r$ bits (most significant portion) and a line field of r bits .

This latter field identifies one of the $m = 2^r$ lines of the cache



Advantage and disadvantage Direct mapping



- The direct mapping technique is simple and inexpensive to implement. Its
- main disadvantage is that there is a **fixed cache location** for any given block. Thus, if a program happens to reference words repeatedly from two different blocks that map into the same line, then the blocks will be continually swapped in the cache, and the hit ratio will be low (a phenomenon known as *thrashing*).

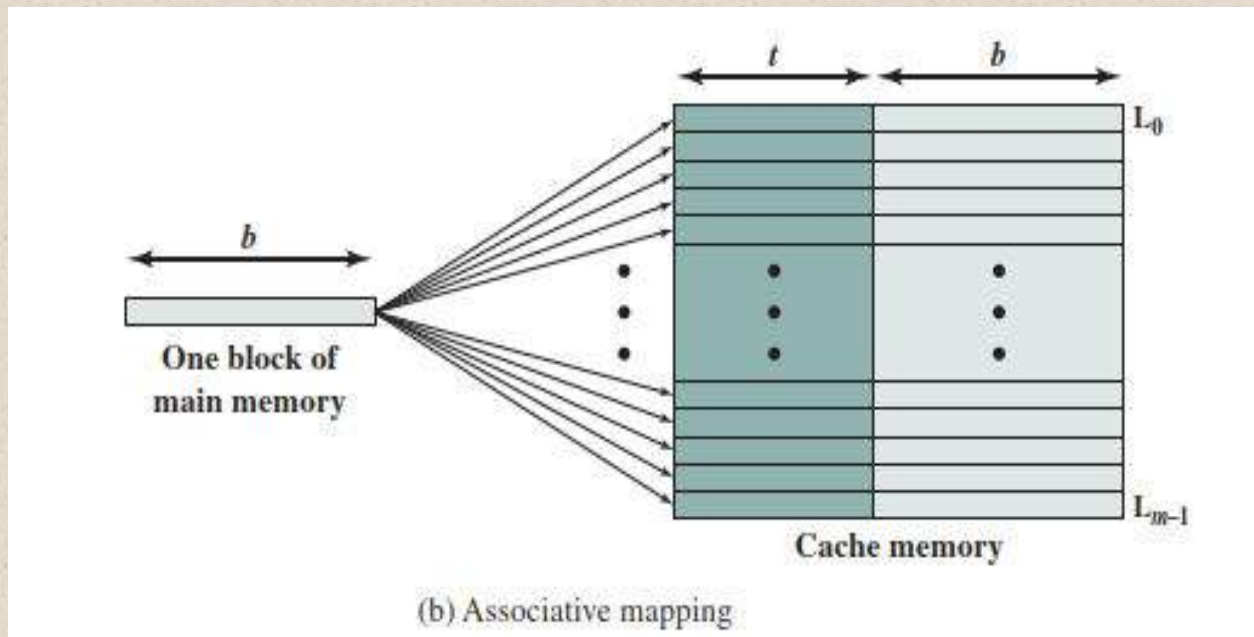
+ Fully associative mapping



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

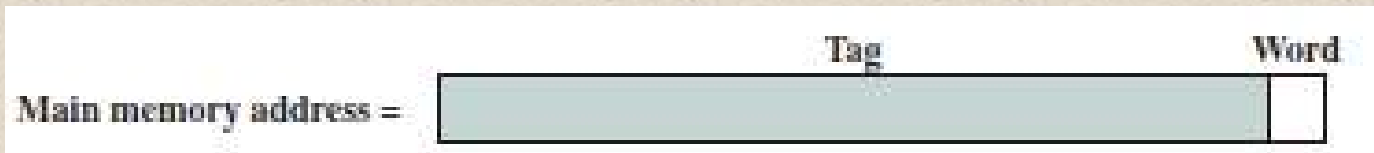
+ Associative mapping

- Overcome the disadvantage of direct mapping by **permitting each main memory block to be loaded into any line of the cache**



+ Associative mapping

- The cache control logic interprets a memory address simply as a **Tag** and **A word** field.
- The Tag field uniquely identifies a block of main memory
- To determine whether a block is in the cache, the cache control logic must simultaneously examine every line's tag for a match



- Word is used for specific access within block
- 2 bit word → 1 out of 4 words
- 4 bit word → 1 out of 16 words

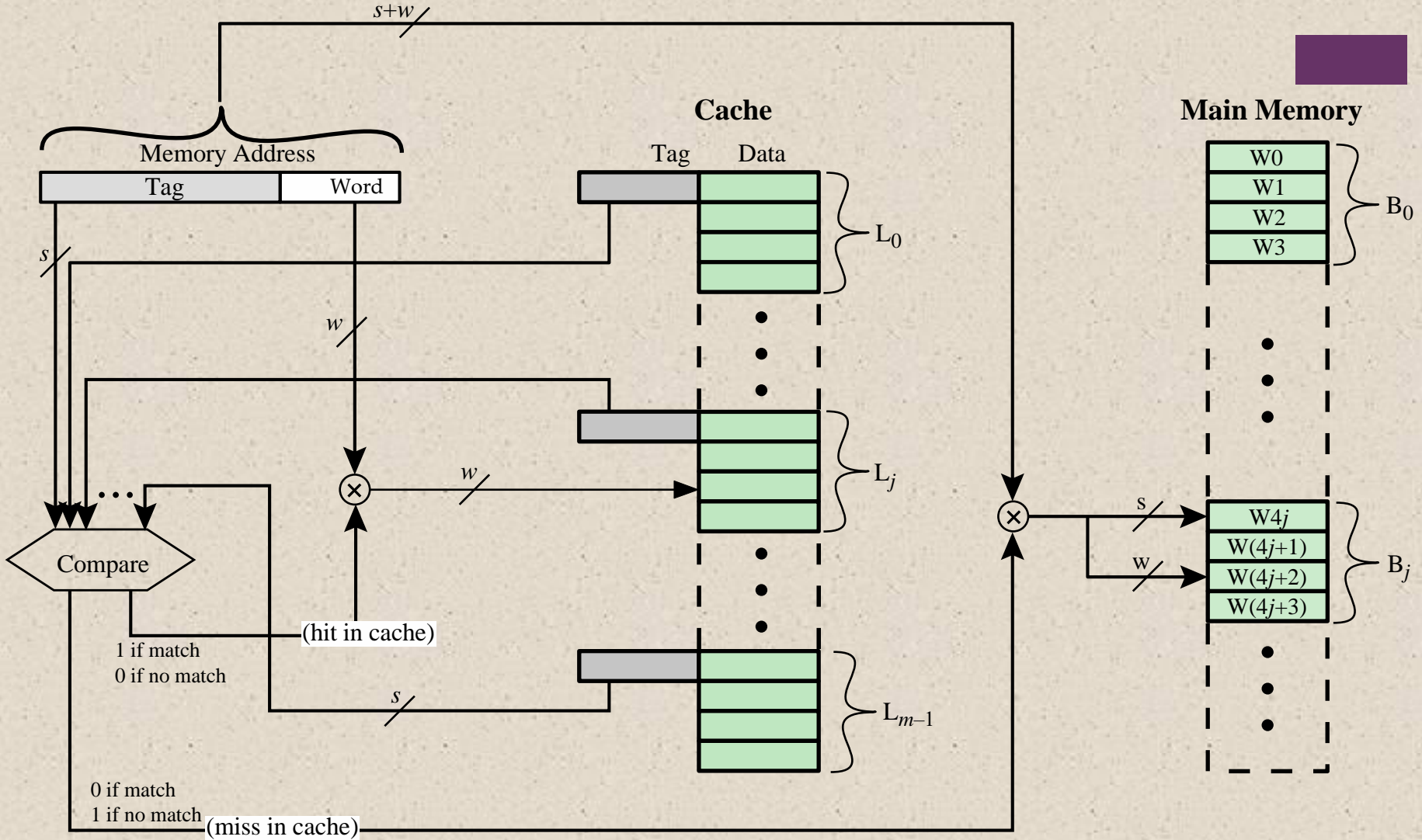


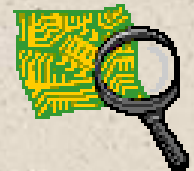
Figure 4.11 Fully Associative Cache Organization



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



+ Example

- Using mapping If Cache size = 64 Kbytes and block size=4 bytes and memory size =16 Mbyte? Find total number of blocks in main memory, and the content of memory address

■ Sol

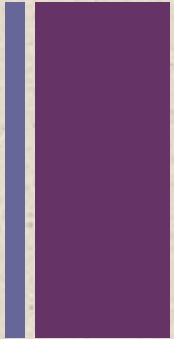
→



Associative Mapping Pros & Cons



- With associative mapping there is **flexibility** as to which block to replace when a new block is read into the cache.
- Note: Replacement algorithms are designed to maximize the hit ratio.
- **Complex circuitry** required to examine the tags of all cache lines in parallel



Set Associative Mapping

- Compromise that exhibits the strengths of both the direct and associative approaches while reducing their disadvantages
- The cache consists of several sets, each of which consists of several lines

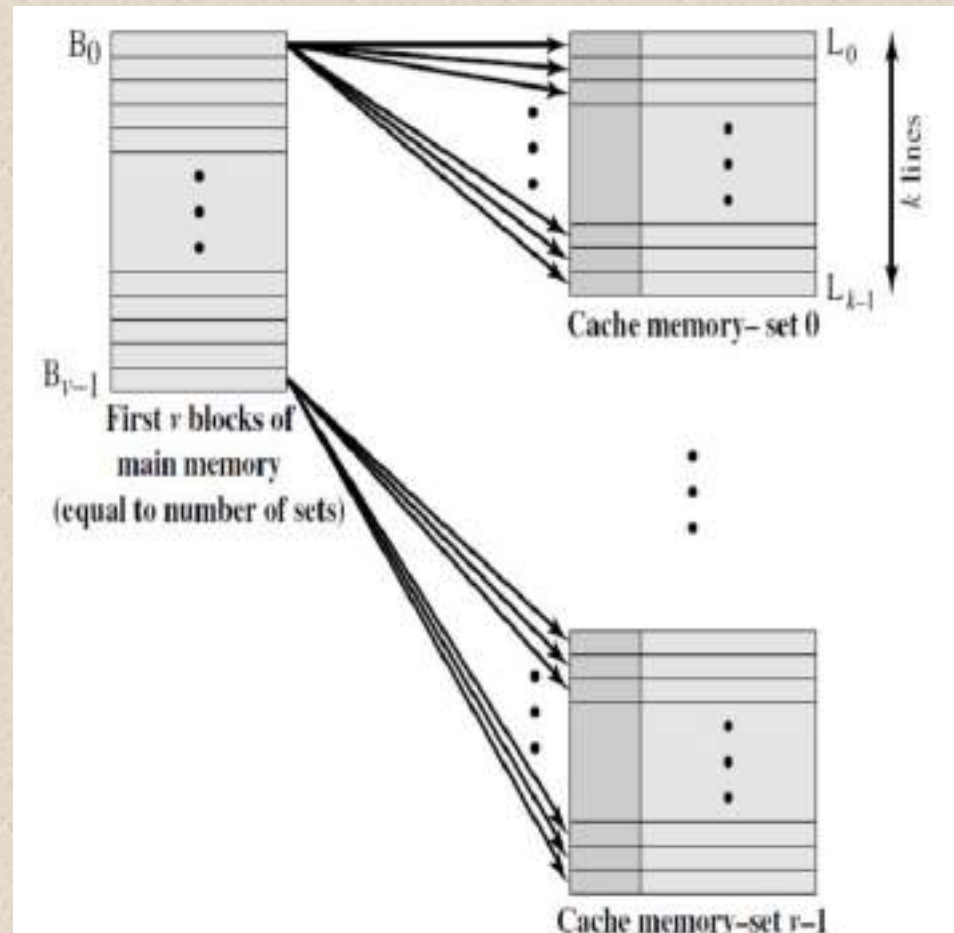
$$m = v \times k$$

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

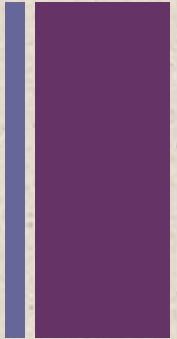
- i cache set number j main memory block number
- m number of lines in the cache v number of sets
- k number of lines in each set

+ Set Associative Mapping

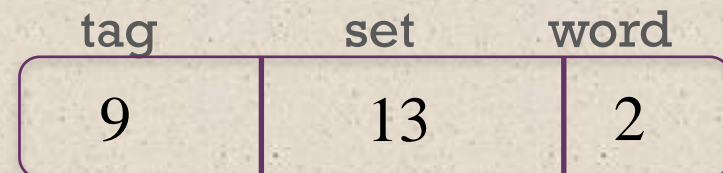
- Each word maps into all the cache lines in a specific set
- Set-associative cache can be physically implemented as associative caches



+ Set Associative Mapping



- Cache size : 64 Kbytes.
- Block size: 4 bytes
- Cache is organized as $16 K = 2^{14}$ **lines** of 4 bytes each.
- Memory Size: 16 Mbytes, with each byte directly addressable by a 24-bit address ($2^{24} = 16 M$).
- Main memory: ($2^{22} = 4M$) blocks of 4 bytes each
- each set contain 2 line then
- # of set(v) = $2^{14} / 2^1 = 2^{13} = 8 K$
- Size of tag = $(s-d) \rightarrow 22-13 = 9$



+ Set Associative Mapping



- Memory address as three fields: **Tag, Set, and Word.**
- **The d set bits specify one of $v = 2^d$ sets**
- **The s bits of the Tag and Set fields specify one of the 2^s blocks of main memory.**
- **Number of sets(v) can be calculate by dividing cache size into k (number of line in each set)**

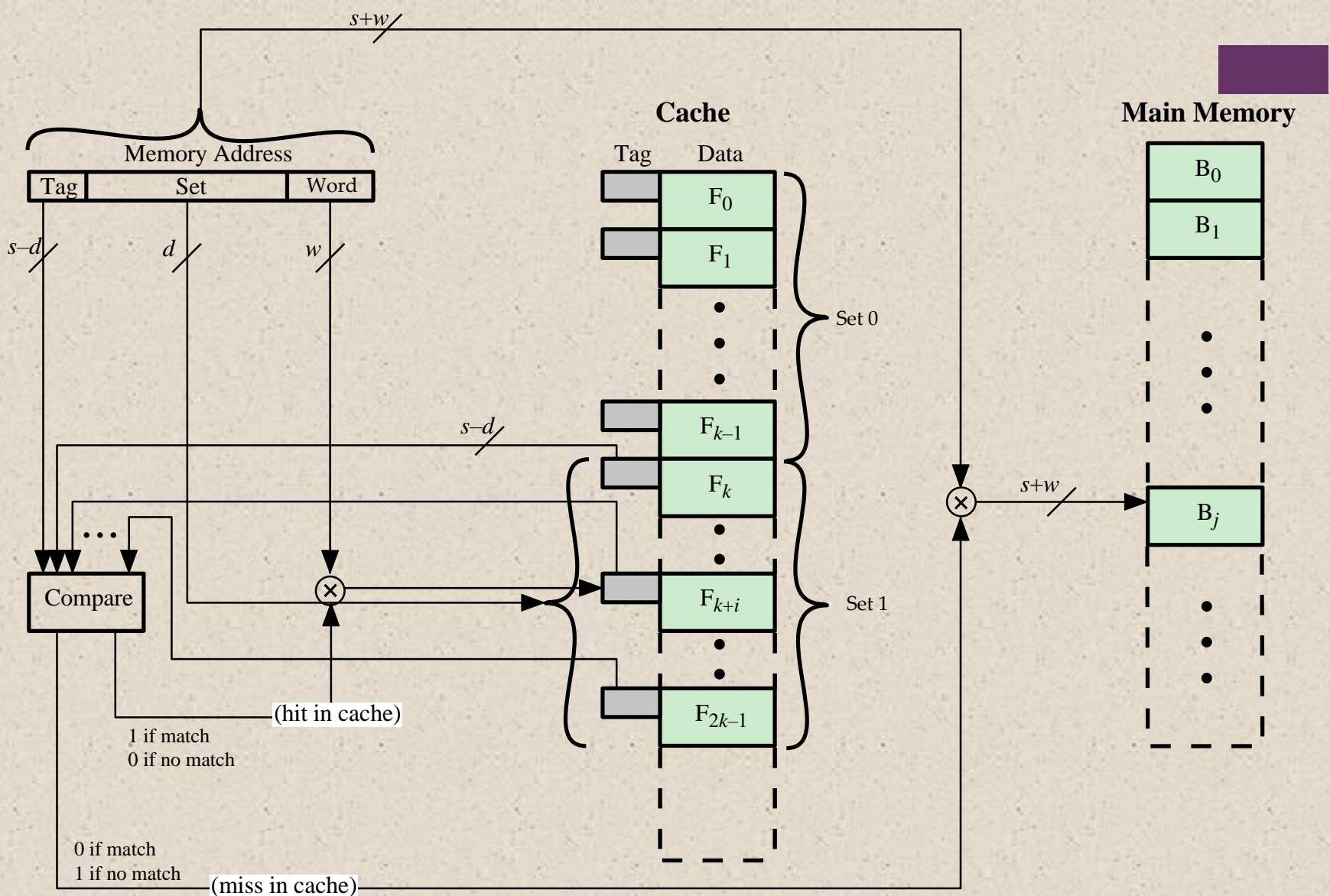


Figure 4.14 k -Way Set Associative Cache Organization




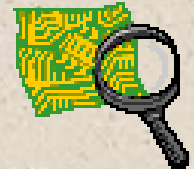
Figure 4.14 illustrates the cache control logic. With **fully associative** mapping, the tag in a memory address is quite large and must be compared to the tag of every line in the cache. With *k-way set-associative mapping*, the tag in a memory address is much smaller and is only compared to the *k tags within a single set*



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



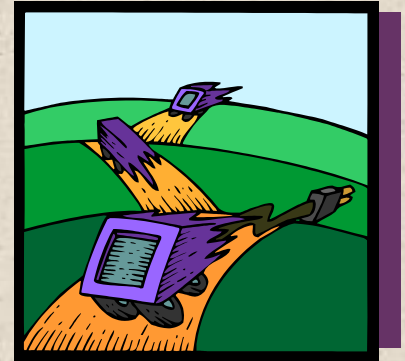
+ Set Associative Mapping



- In the extreme case of $v = m$, $k = 1$, the set-associative technique reduces to direct mapping
- for $v = 1$, $k = m$, it reduces to associative mapping.
- The use of two lines per set ($v = m/2$, $k = 2$) is the most common set-associative organization



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 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
 - When an update occurs, a dirty bit, or use bit, associated with the line is set. Then, when a block is replaced, it is written back to main memory if and only if the dirty bit is set
 - The problem with write back is “cache coherent” different copies for block between cache and main memory

+ Summary

Chapter 4

Cache Memory

- Computer memory system overview
 - Characteristics of Memory Systems
 - Memory Hierarchy
- Cache memory principles
- Pentium 4 cache organization

- Elements of cache design
 - Cache addresses
 - Cache size
 - Mapping function
 - Replacement algorithms
 - Write policy
 - Line size
 - Number of caches