

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



+ Chapter 3

A Top-Level View of Computer Function and Interconnection



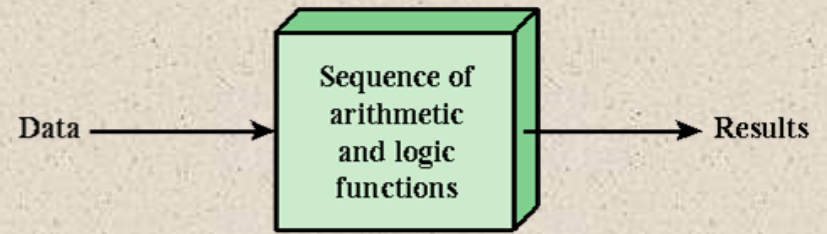
Computer Components



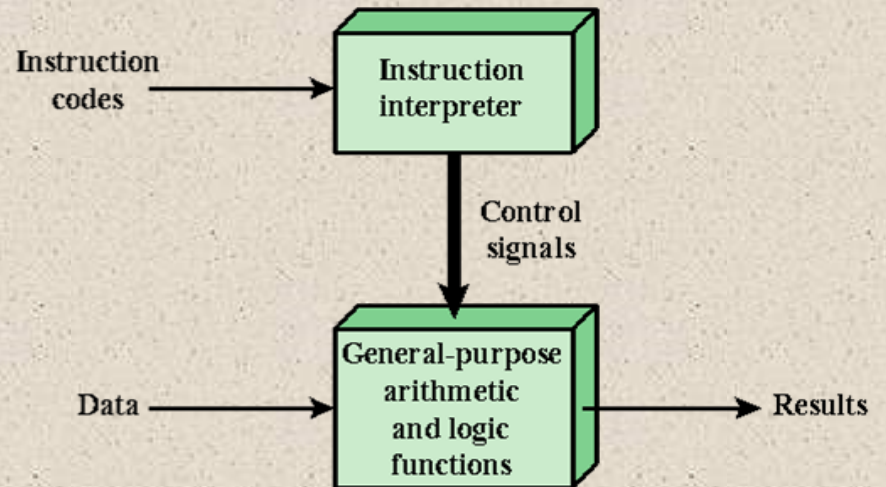
- Contemporary computer designs are based on concepts developed by John von Neumann at the Institute for Advanced Studies, Princeton
- Referred to as the *von Neumann architecture* and is based on three key concepts:
 - Data and instructions are stored in a single read-write memory
 - The contents of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

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

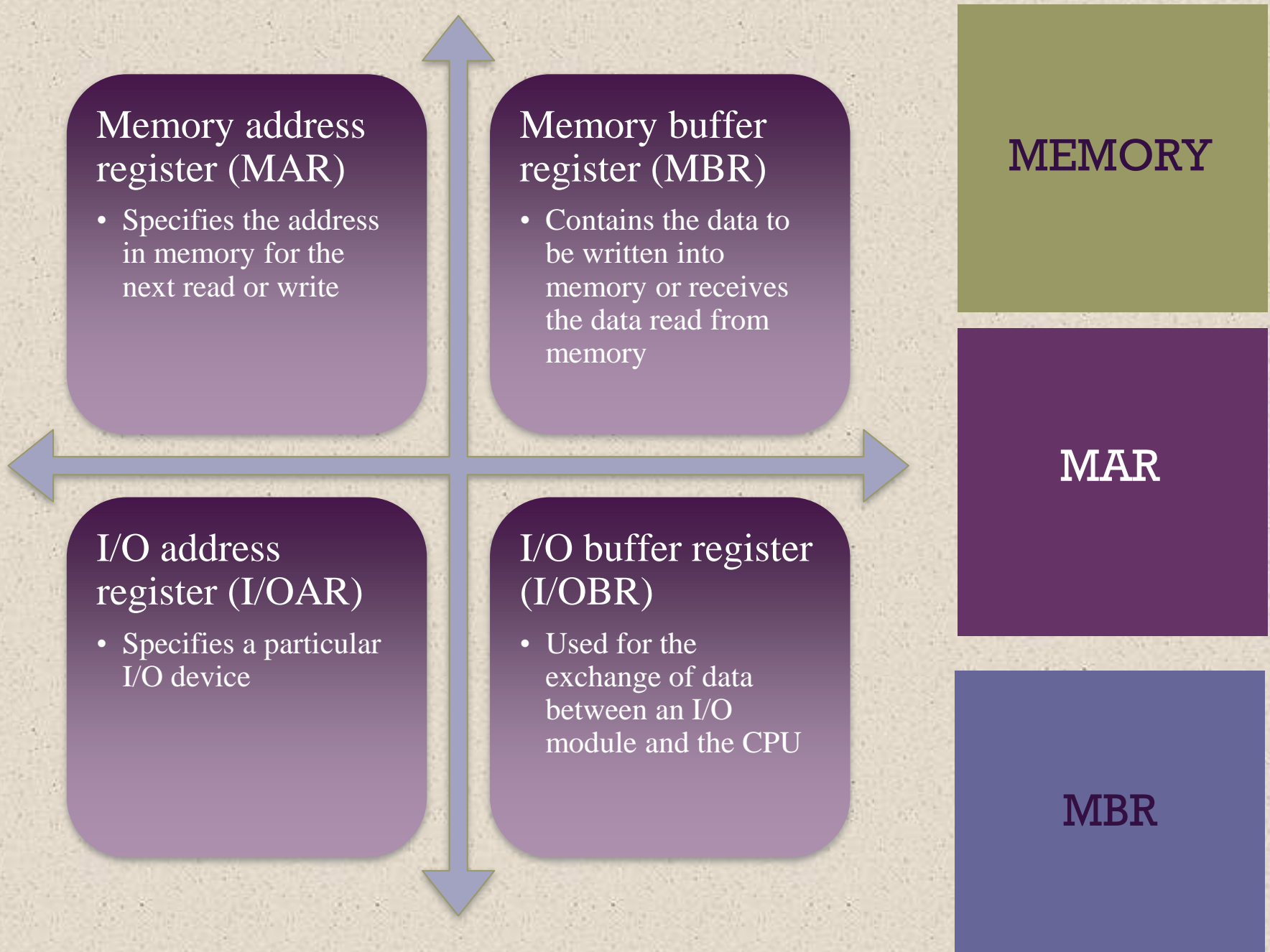
Major components:

- 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





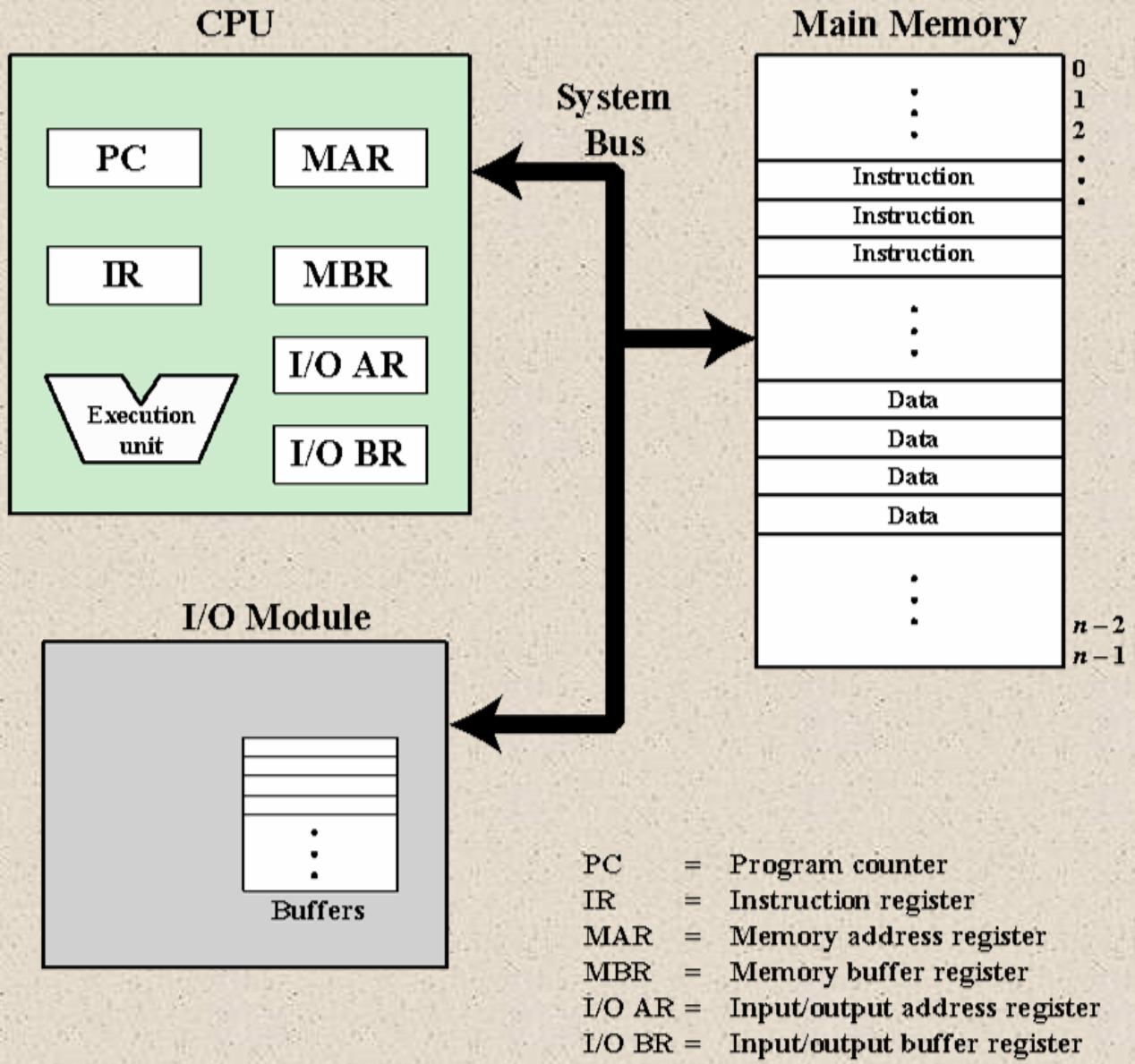


Figure 3.2 Computer Components: Top-Level View

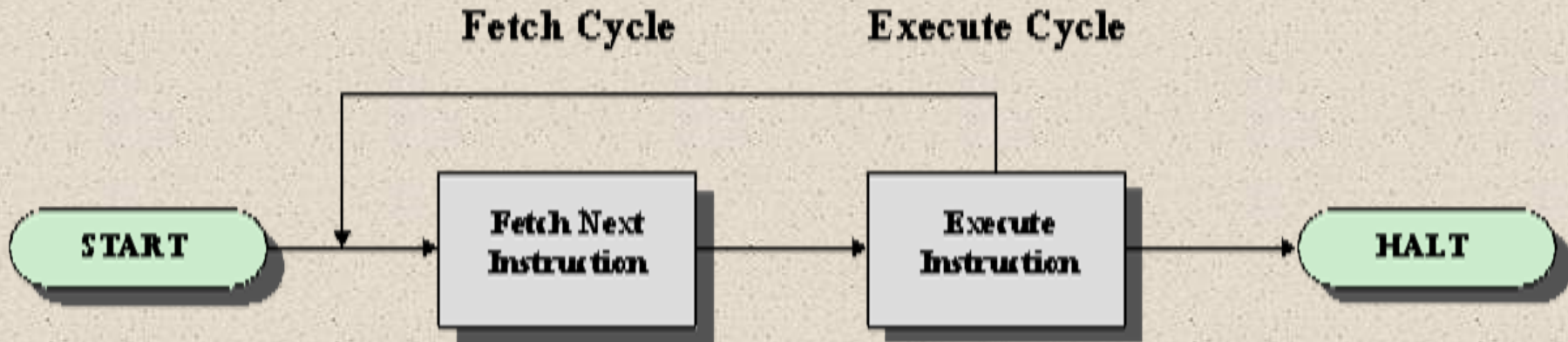


Figure 3.3 Basic Instruction Cycle



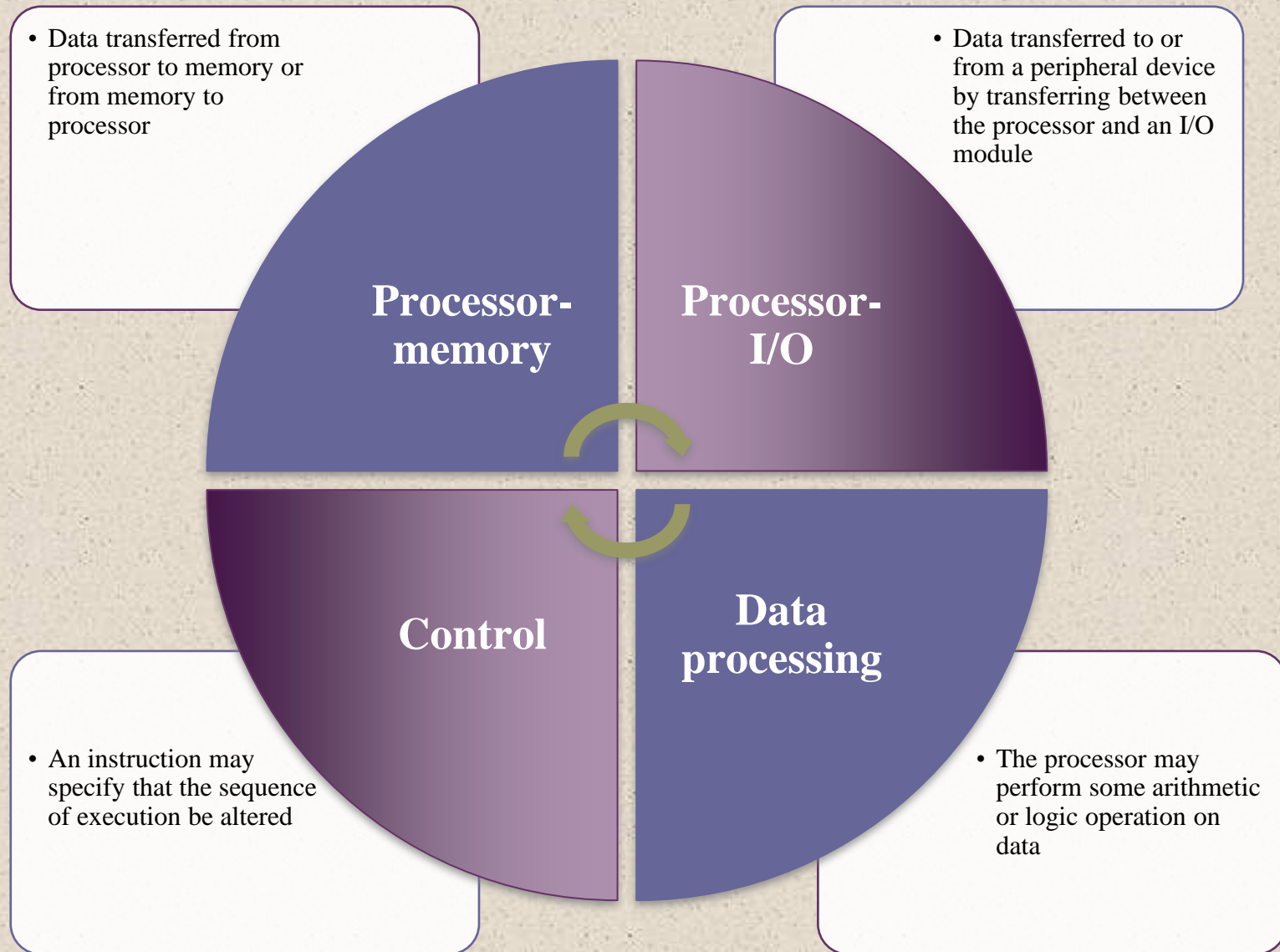
Fetch Cycle



- 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

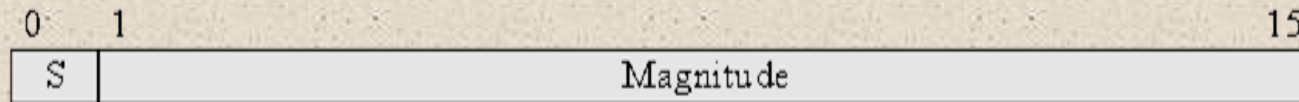


Action Categories





(a) Instruction format



(b) Integer format

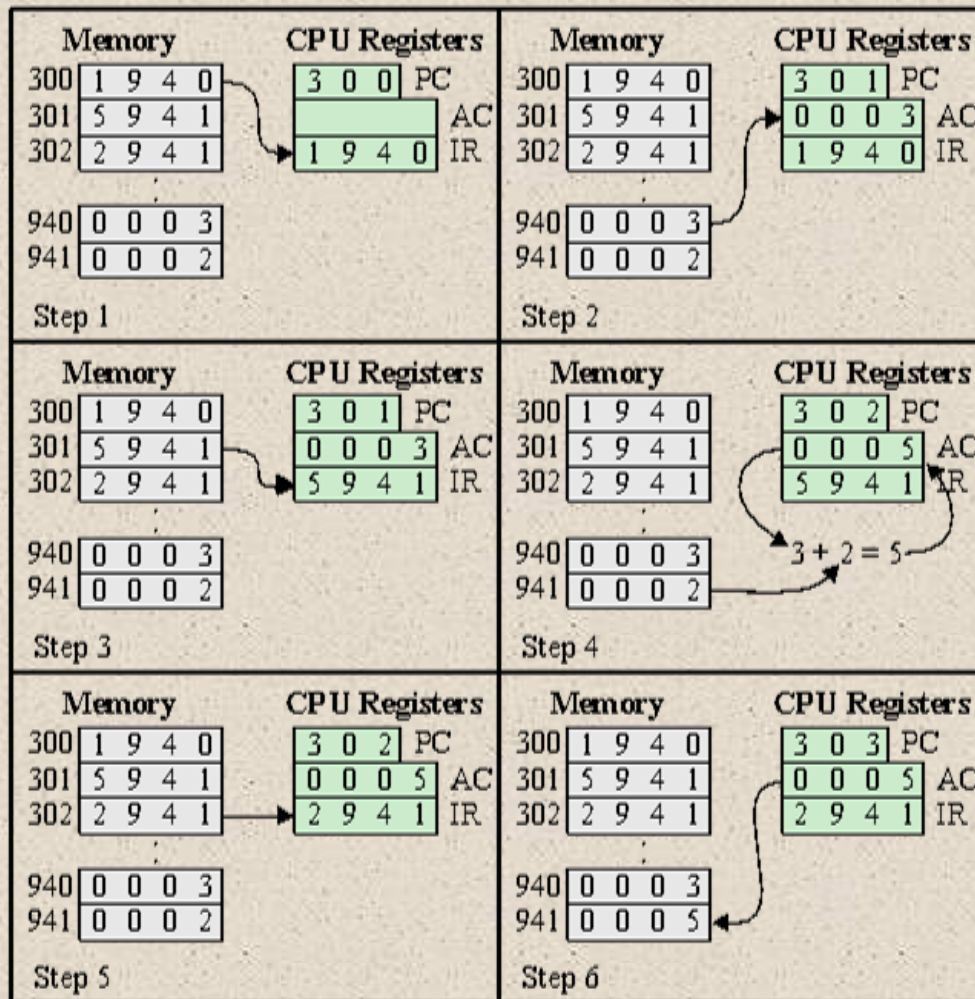
Program Counter (PC) = Address of instruction
 Instruction Register (IR) = Instruction being executed
 Accumulator (AC) = Temporary storage

(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 3.5 Example of Program Execution
(contents of memory and registers in hexadecimal)**

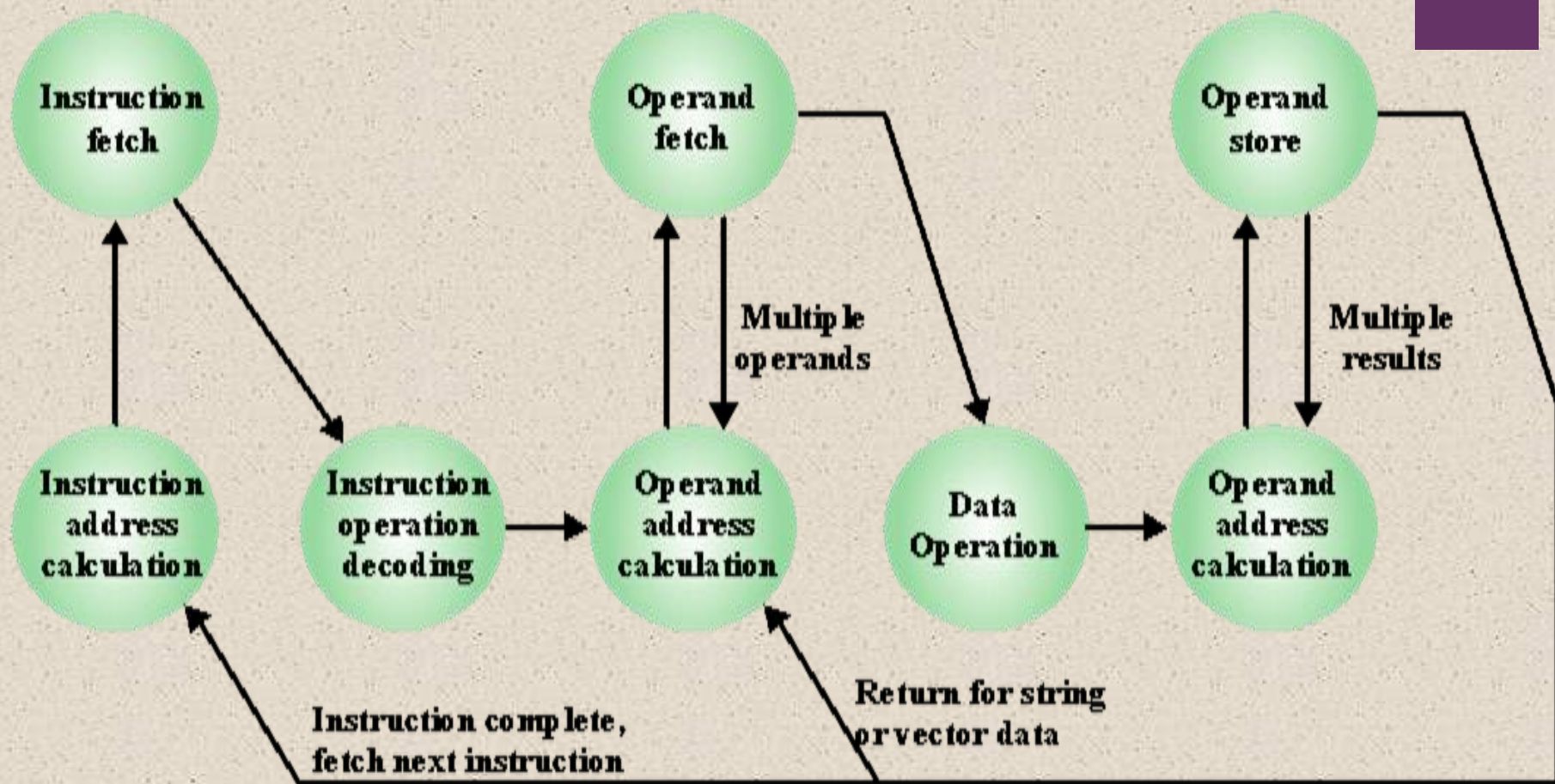


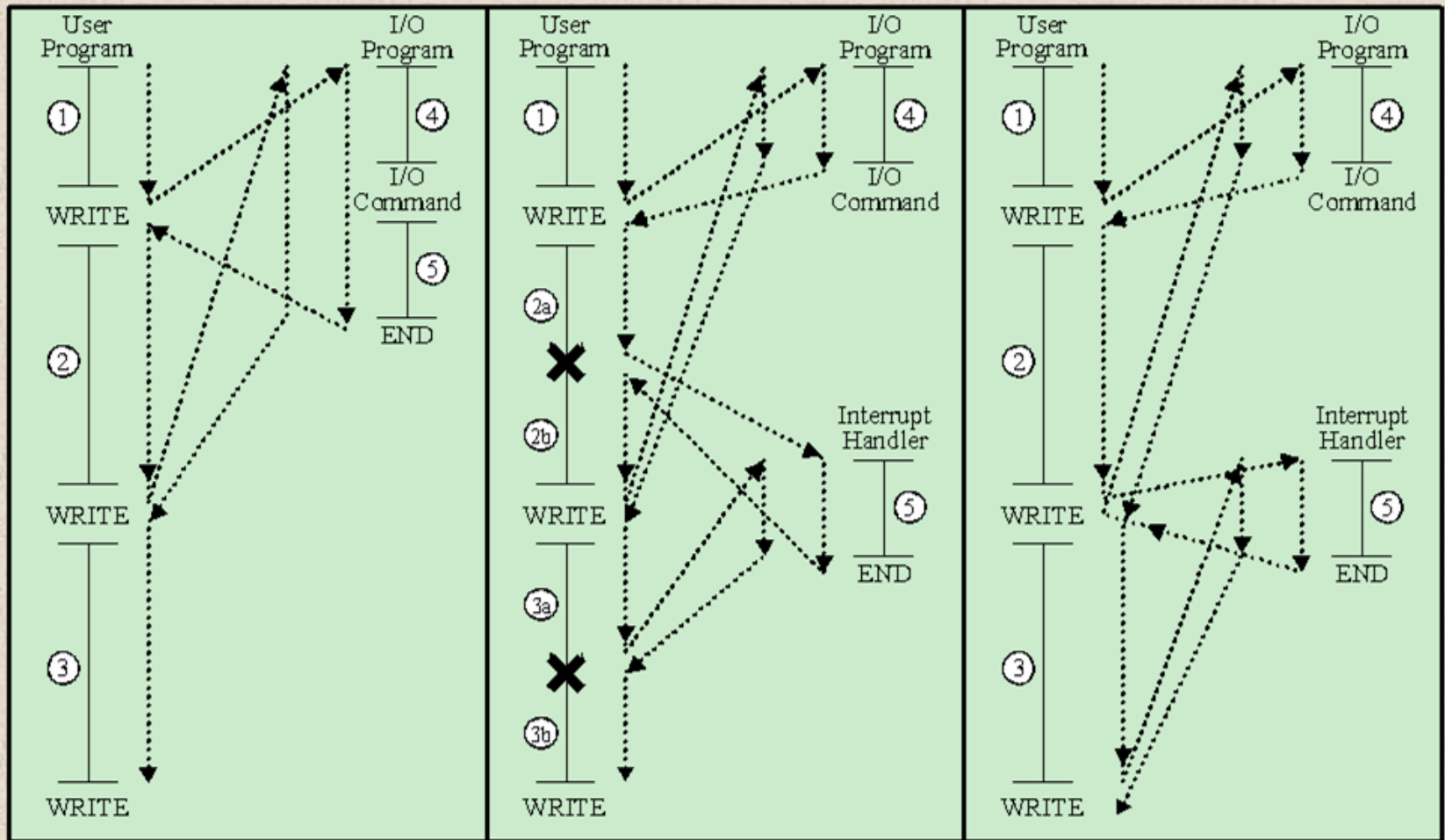
Figure 3.6 Instruction Cycle State Diagram



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

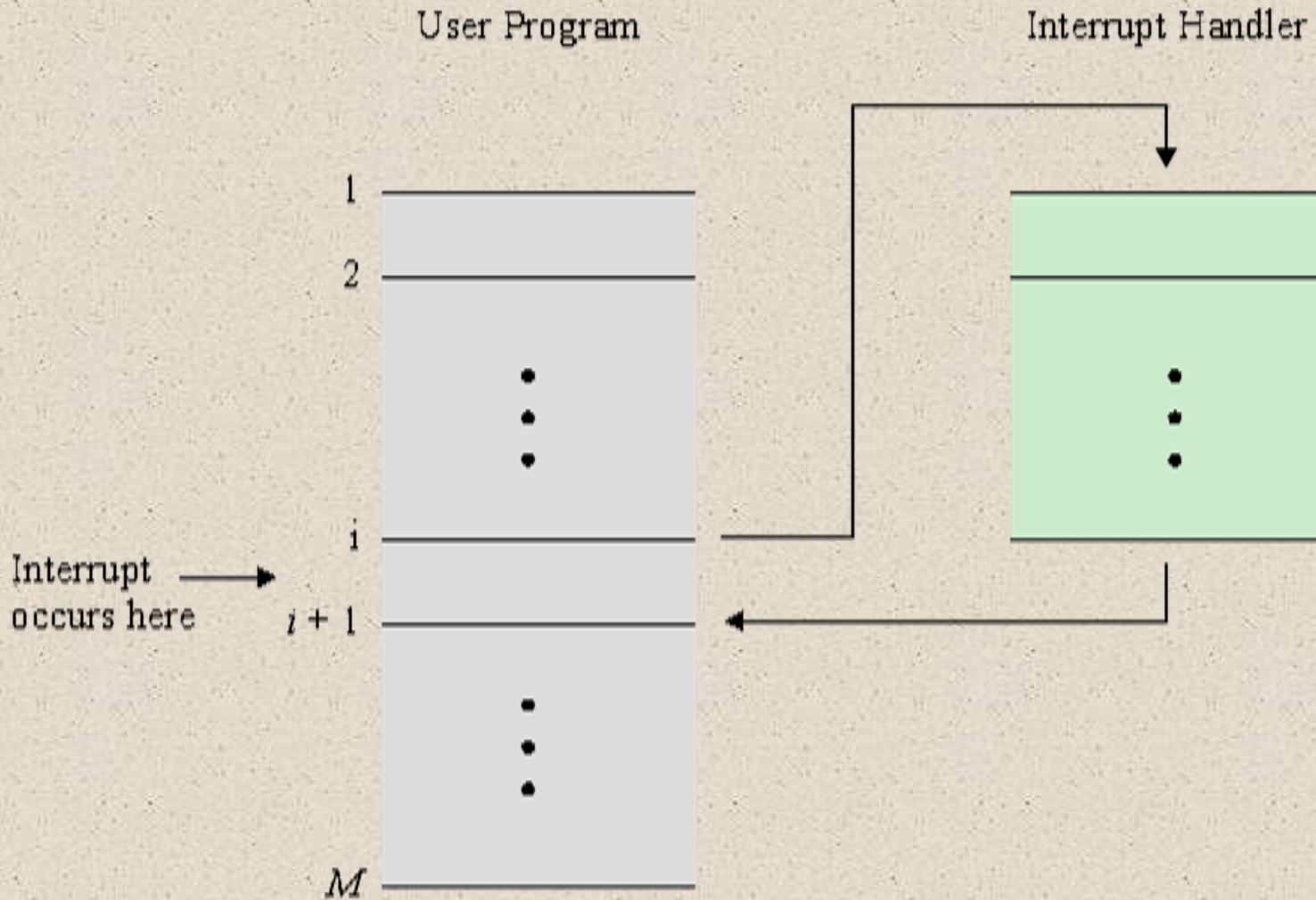


Figure 3.8 Transfer of Control via Interrupts

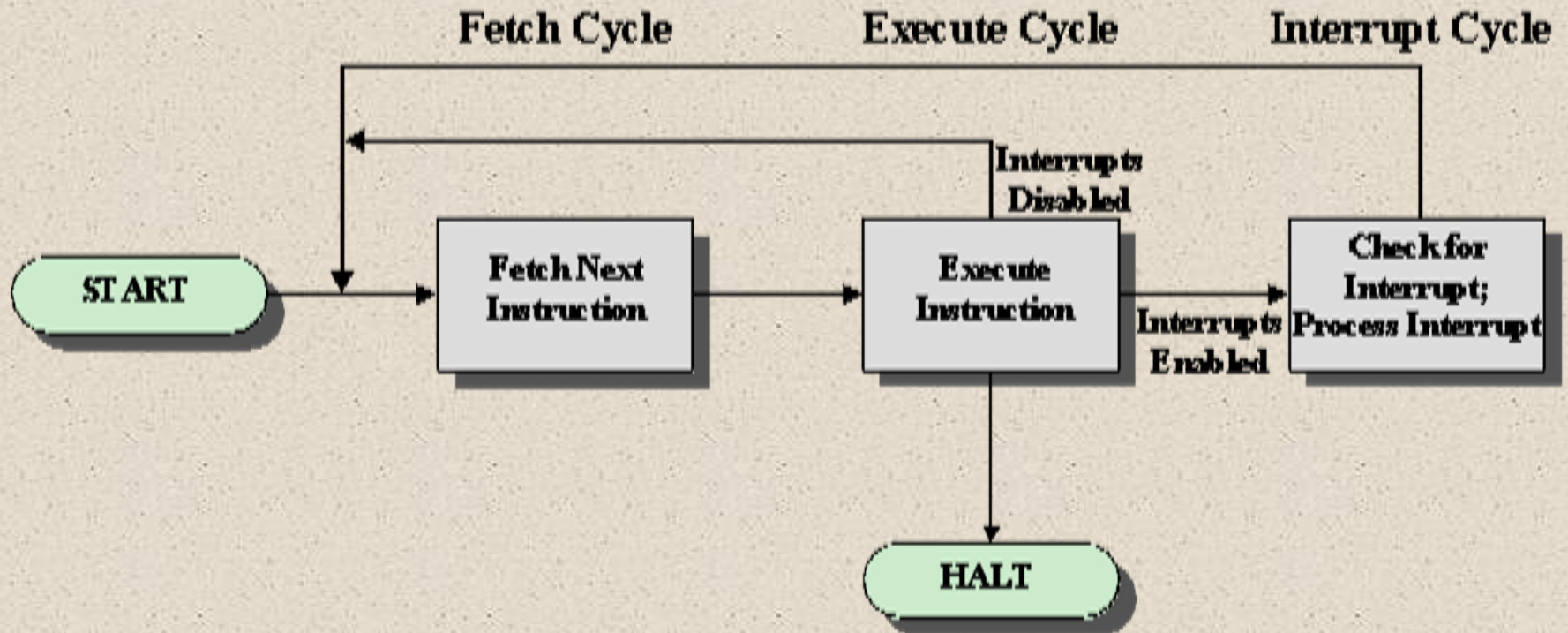


Figure 3.9 Instruction Cycle with Interrupts

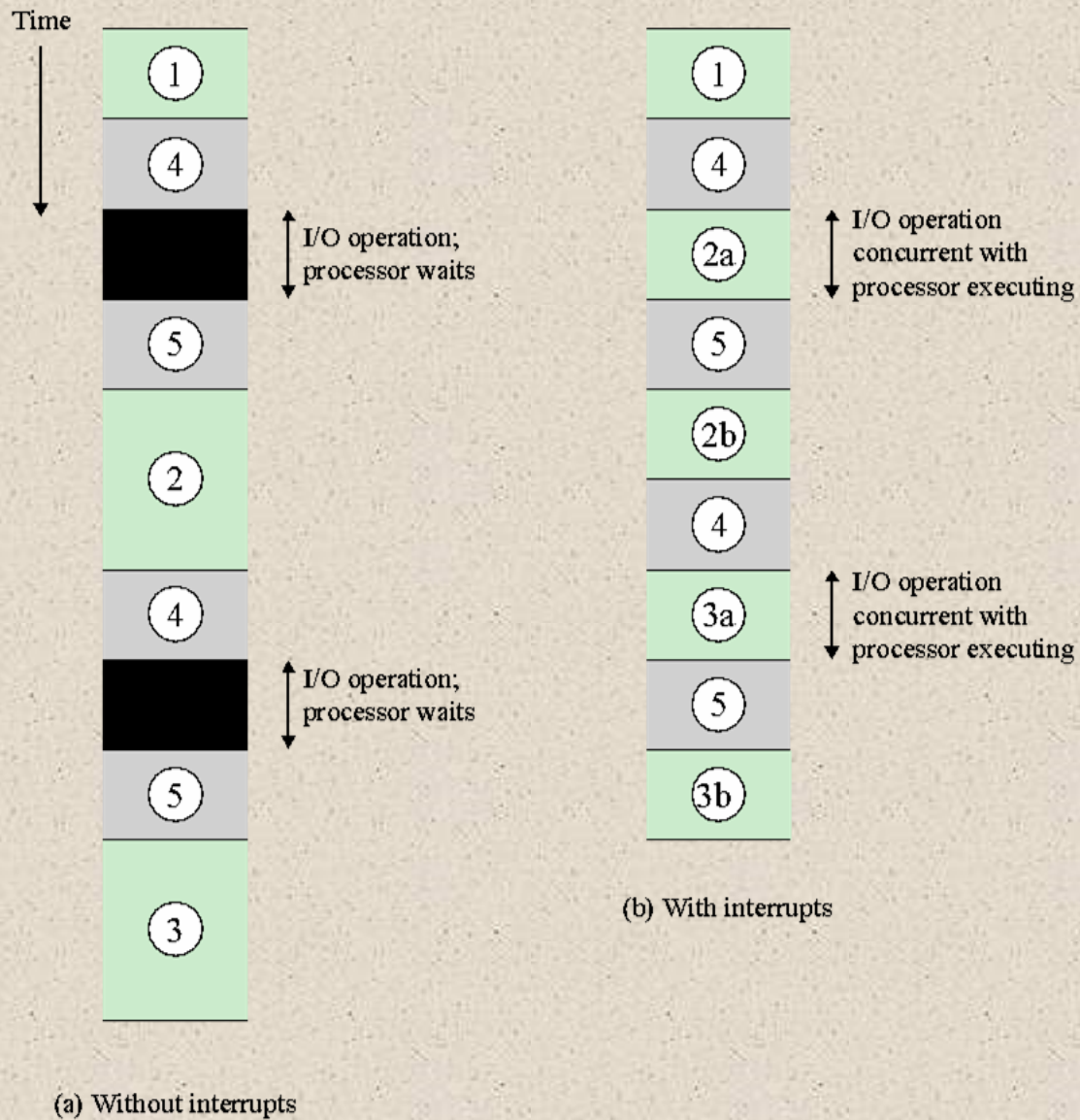


Figure 3.10 Program Timing: Short I/O Wait

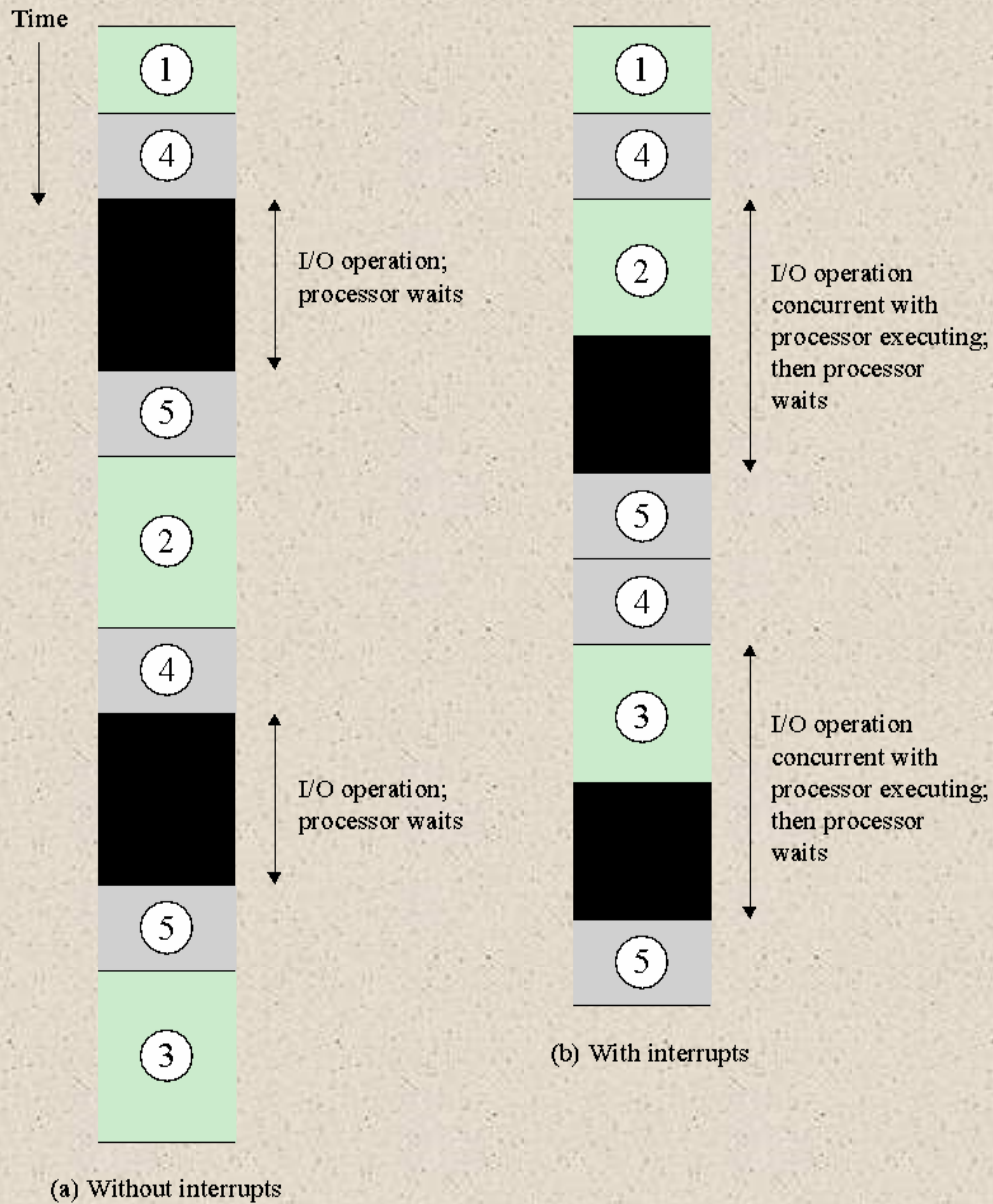


Figure 3.11 Program Timing: Long I/O Wait

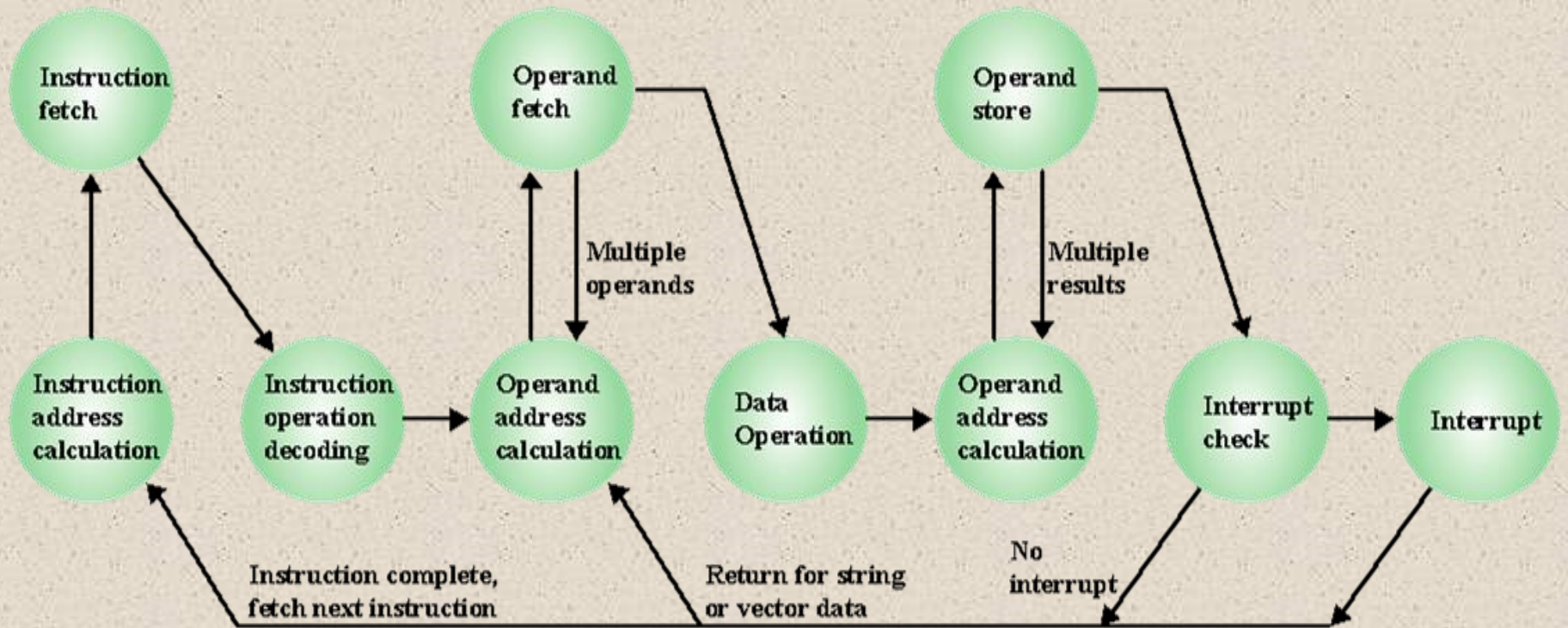
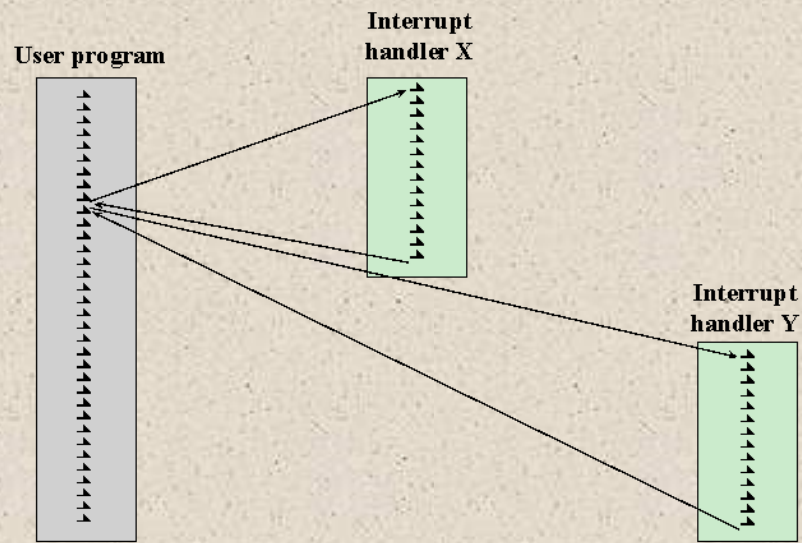
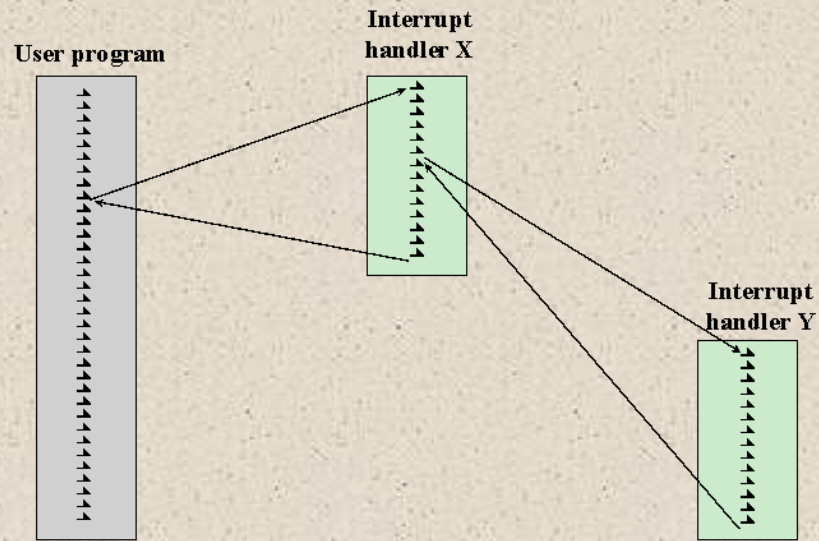


Figure 3.12 Instruction Cycle State Diagram, With Interrupts



(a) Sequential interrupt processing



(b) Nested interrupt processing

Figure 3.13 Transfer of Control with Multiple Interrupts

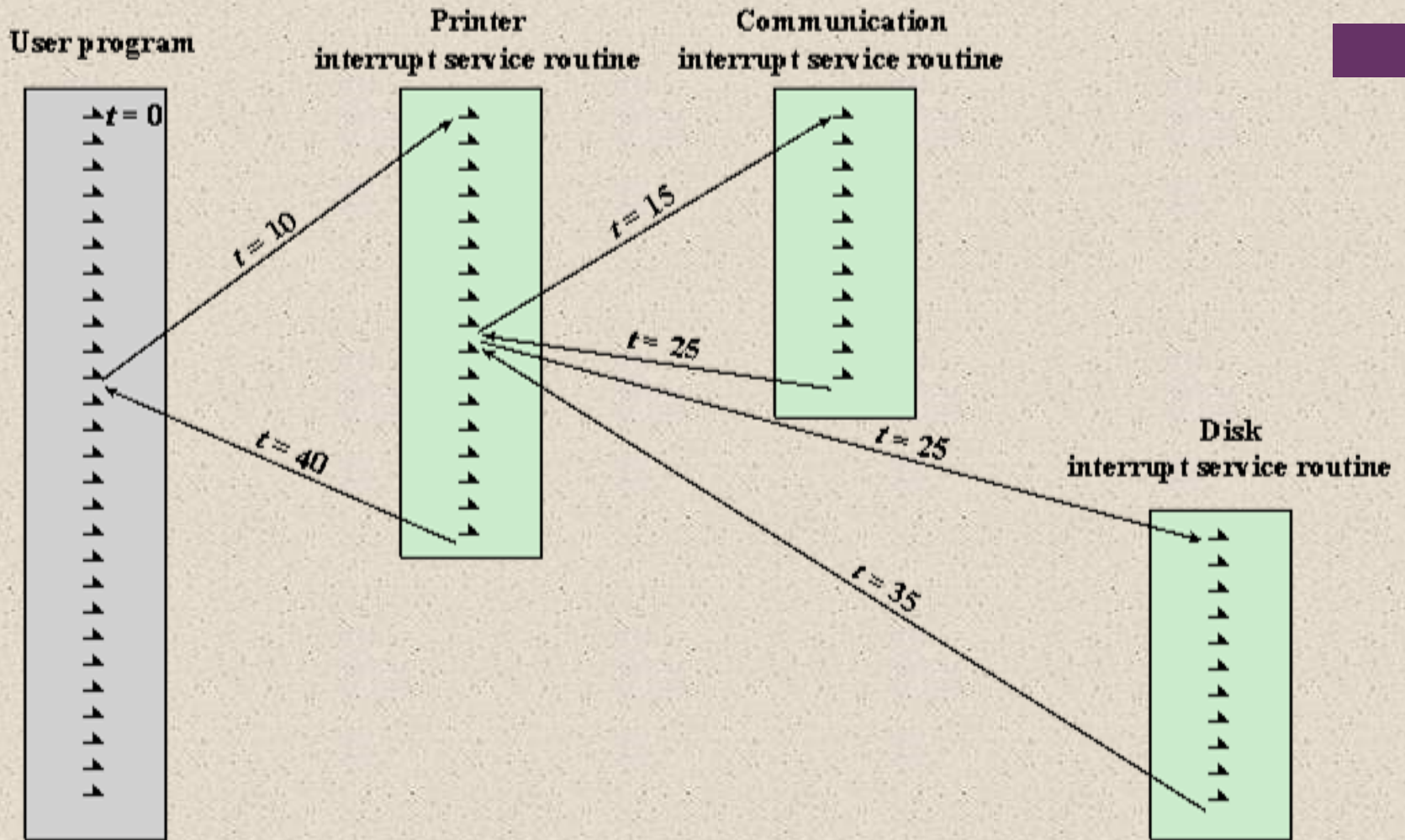


Figure 3.14 Example Time Sequence of Multiple Interrupts



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)

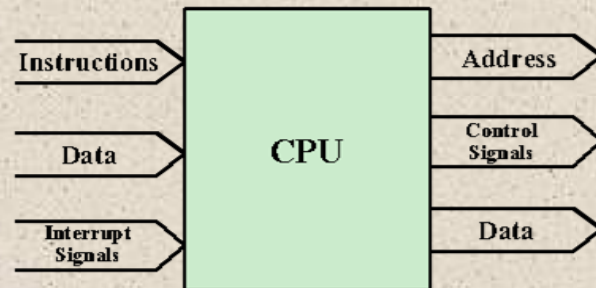
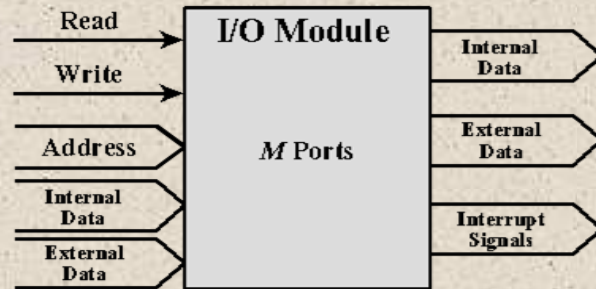
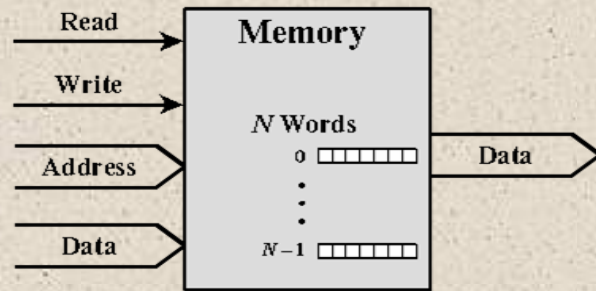


Figure 3.15 Computer Modules

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

Bus Interconnection

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

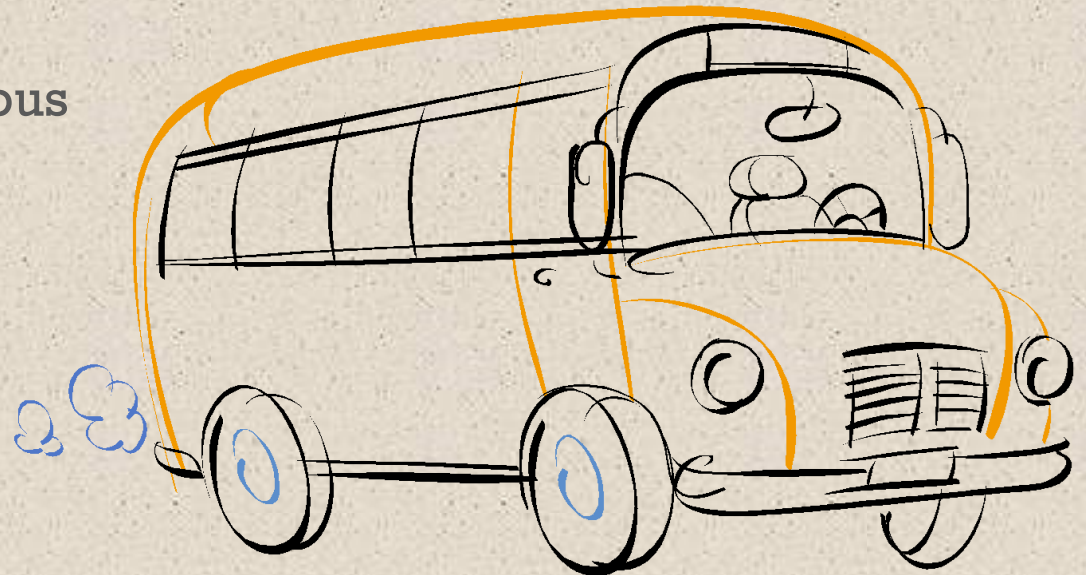
- A bus that connects major computer components (processor, memory, I/O)

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

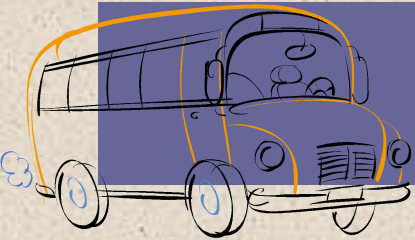
Data Bus



- 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

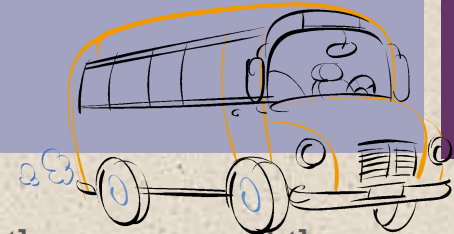


+ Address Bus



- 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

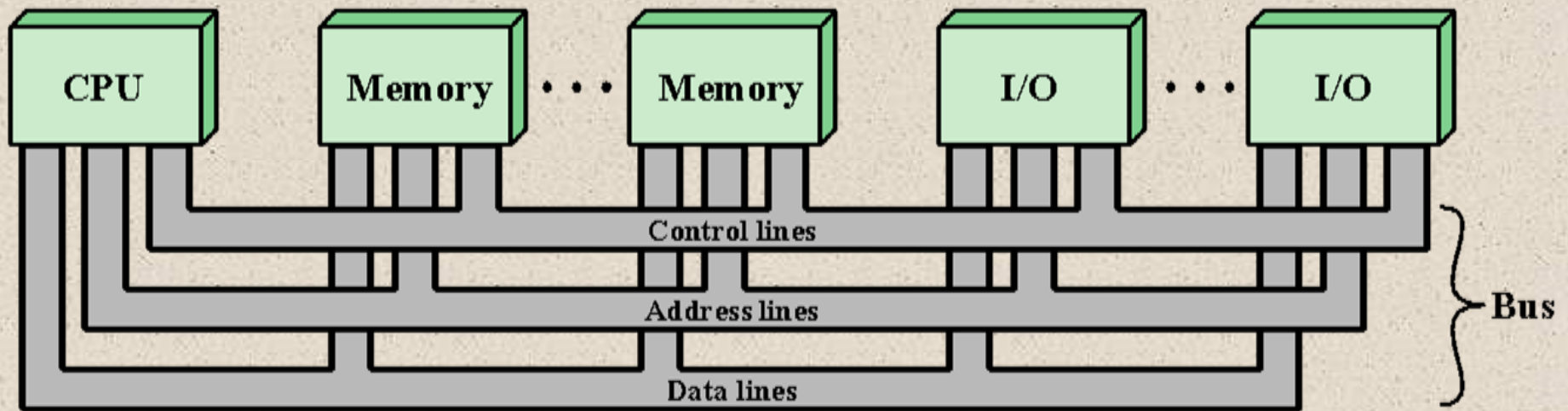
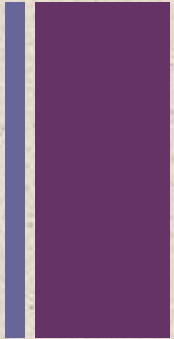


Figure 3.16 Bus Interconnection Scheme



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



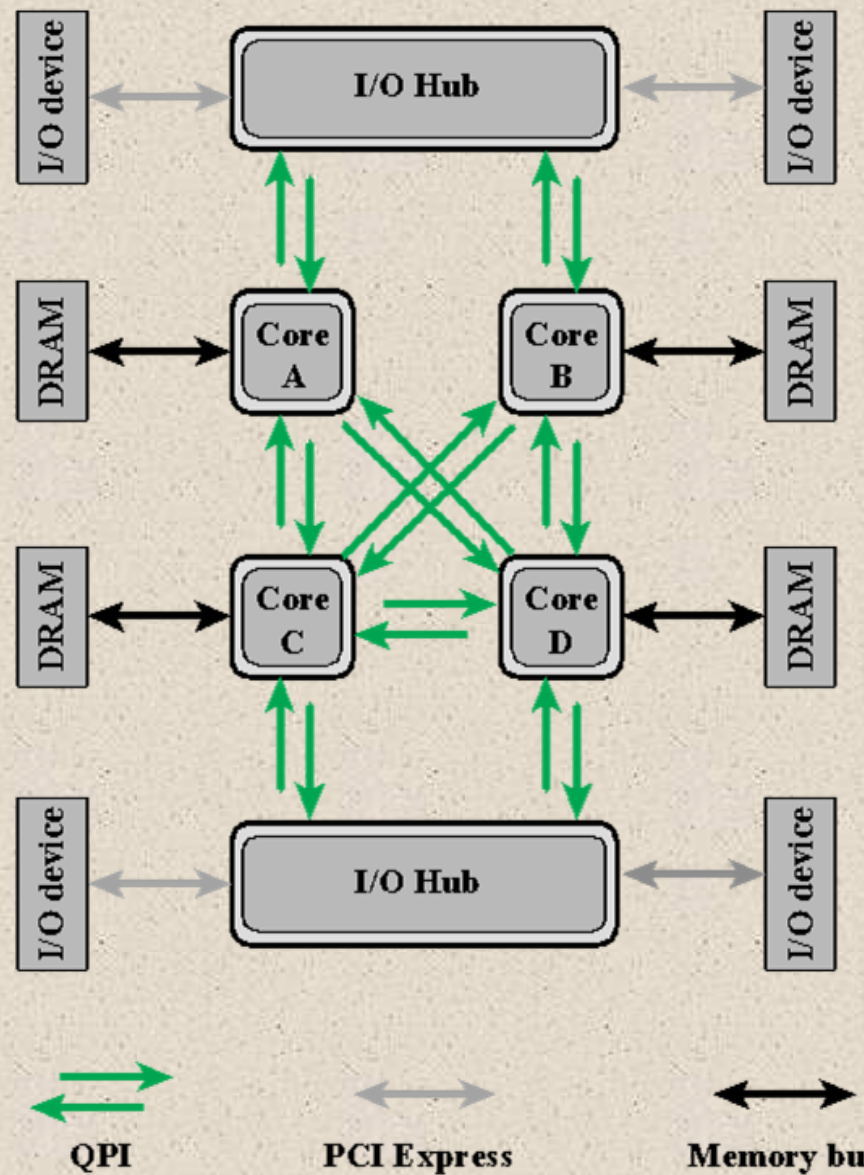
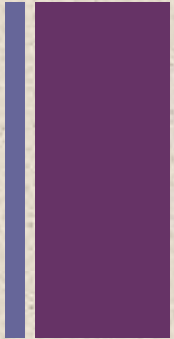


Figure 3.17 Multicore Configuration Using QPI



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