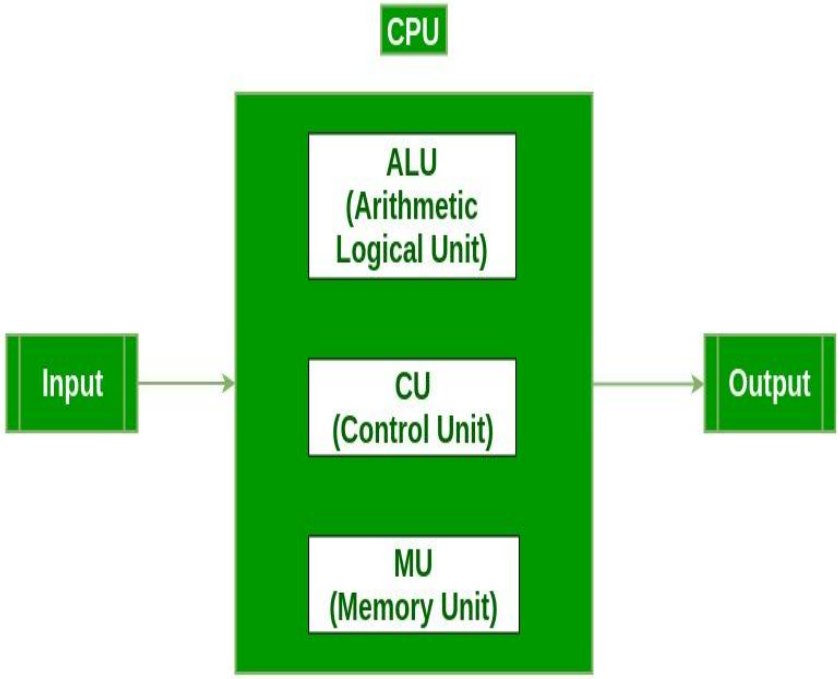
Introduction to Computer Organization & Architecture

1.1 Functional blocks of a computer
1.2 Computer Organization and Architecture
1.3 Structure and Function
1.4 Computer Components
1.5 Computer Functions

1.1 Functional blocks of a computer

- Mainly computer system consists of three parts, that are central processing unit (CPU), Input Devices, and Output Devices.
- The Central Processing Unit (CPU) is divided into three parts again: Arithmetic logic unit (ALU), Memory Unit (MU) and the control unit (CU).



1.1 Functional blocks of a computer

- Computer:
  - A computer is a combination of **hardware and software** resources which integrate together and provides various functionalities to the user.
  - Hardware are the physical components of a computer like the processor, memory devices, monitor, keyboard etc. while software is the set of programs or instructions that are required by the hardware resources to function properly.
  - There are a few basic components that aids the working-cycle of a computer i.e. the Input- Process- Output Cycle and these are called as the functional components of a computer.

1.1 Functional blocks of a computer

- Computer:
  - It needs certain input, processes that input and produces the desired output.
  - The input unit takes the input, the central processing unit does the processing of data and the output unit produces the output.
  - The memory unit holds the data and instructions during the processing.

1.1 Functional blocks of a computer

- Input Unit :
  - The input unit consists of input devices that are attached to the computer.
  - These devices take input and convert it into binary language that the computer understands.
  - Some of the common input devices are keyboard, mouse, joystick, scanner etc.
  - We can enter the data from the outside world into the primary storage as the input through input devices.
  - The input devices are the medium of communication between the outside world and the computer system.

1.1 Functional blocks of a computer

- Central Processing Unit (CPU):
  - Once the information is entered into the computer by the input device, the processor processes it.
  - The CPU is called the brain of the computer because it is the control center of the computer.
  - It first fetches instructions from memory and then interprets them so as to know what is to be done.
  - Thereafter CPU executes or performs the required computation and then either stores the output or displays on the output device.
  - The CPU has three main components which are responsible for different functions – Arithmetic Logic Unit (ALU), Control Unit (CU) and Memory Unit(MU).

1.1 Functional blocks of a computer

- Arithmetic and Logic Unit (ALU) :
  - The ALU, as its name suggests performs mathematical calculations and takes logical decisions.
  - Arithmetic calculations include addition, subtraction, multiplication and division.
  - Logical decisions involve comparison of two data items to see which one is larger or smaller or equal.

1.1 Functional blocks of a computer

- Control Unit :
  - The Control unit coordinates and controls the data flow in and out of CPU and also controls all the operations of ALU, memory registers and also input/output units.
  - It is also responsible for carrying out all the instructions stored in the program.
  - It decodes the fetched instruction, interprets it and sends control signals to input/output devices until the required operation is done properly by ALU and memory.

1.1 Functional blocks of a computer

- Memory:
  - Memory attached to the CPU is used for storage of data and instructions and is called internal memory.
  - The internal memory is divided into many storage locations, each of which can store data or instructions.
  - Each memory location is of the same size and has an address.
  - With the help of the address, the computer can read any memory location easily without having to search the entire memory.
  - When a program is executed, it's data is copied to the internal memory and is stored in the memory till the end of the execution.

1.1 Functional blocks of a computer

- Memory:
  - The internal memory is also called the Primary memory or Main memory. This memory is also called as RAM, i.e. Random Access Memory.
  - We can store the data and programs on a long-term basis in the secondary memory.
  - The hard disks and the optical disks are the common secondary devices. It is slow and cheap memory as compare to primary memory. This memory is not connected to the processor directly.
  - It has a large capacity to store the data. The secondary storage is direct access by the CPU; that's why it is different from the primary storage.

1.1 Functional blocks of a computer

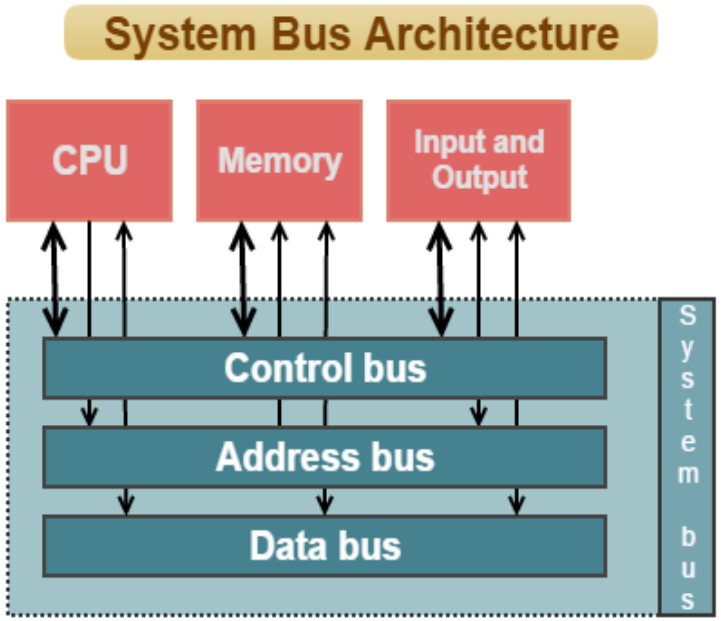
- Output Unit:
  - The output unit consists of output devices that are attached with the computer.
  - It converts the binary data coming from CPU to human understandable form.
  - The common output devices are monitor, printer, projectors etc.

1.1 Functional blocks of a computer

- Bus connection:
  - A computer consists of input unit that takes input, a CPU that processes the input and an output unit that produces output.
  - All these devices communicate with each other through a common bus.
  - A bus is a transmission path, made of a set of conducting wires over which data or information in the form of electric signals, is passed from one component to another in a computer.
  - The bus can be of three types – Address bus, Data bus and Control Bus.
  - The address bus – It carries the address location of the data or instruction.

1.1 Functional blocks of a computer

- Bus connection:
  - The data bus – It carries data from one component to another
  - The control bus – It carries the control signals.
  - The system bus – It is the common communication path that carries signals to/from CPU, main memory and input/output devices.



1.2 Computer Organization and Architecture

- Computer Architecture refers to those attributes of a system that have a direct impact on the logical execution of a program. Examples:
  - The instruction set
  - The number of bits used to represent various data types
  - I/O mechanisms
  - Memory addressing techniques
- Computer Organization refers to the operational units and their interconnections that realize the architectural specifications. Examples are things that are transparent to the programmer:
  - Control signals
  - Interfaces between computer and peripherals
  - The memory technology being used

1.2 Computer Organization and Architecture

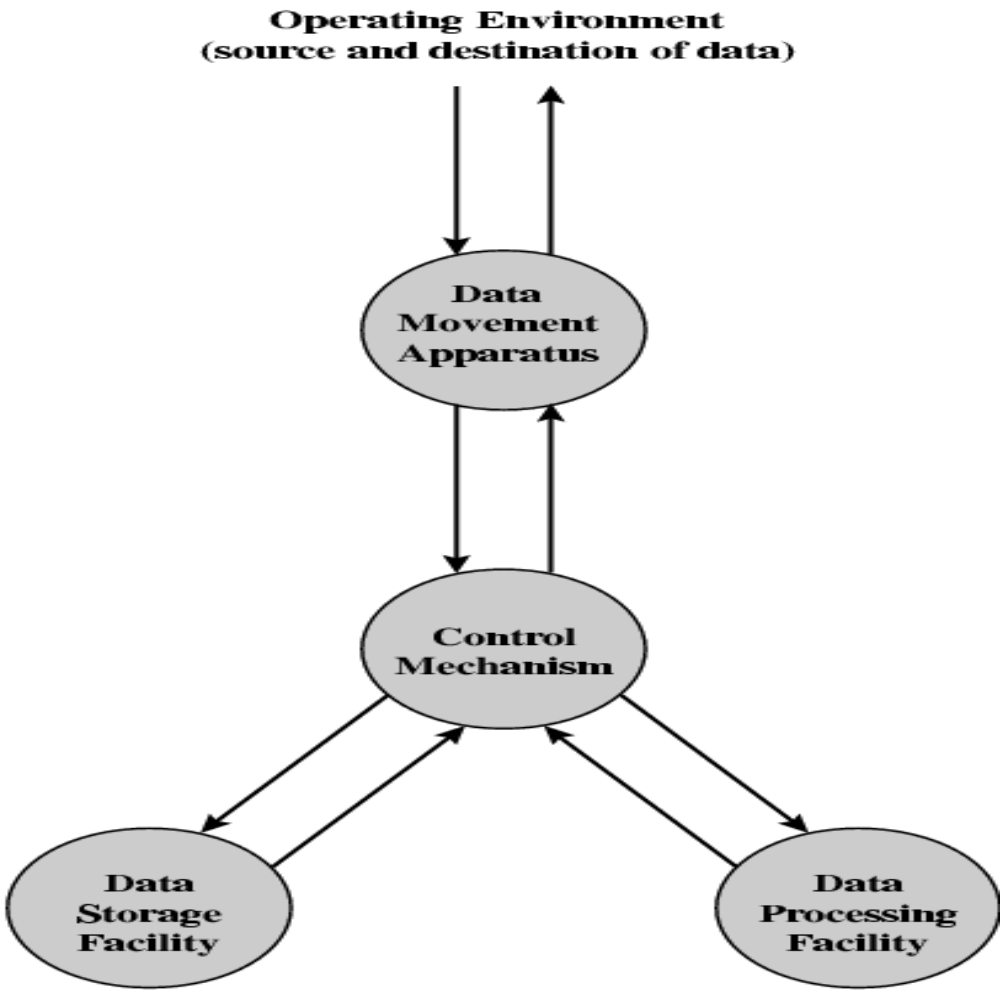
- So, for example, the fact that a multiply instruction is available is a computer architecture issue.
- How that multiply is implemented is a computer organization issue.
- Architecture is those attributes visible to the programmer
  - Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques.
  - e.g. Is there a multiply instruction?
- Organization is how features are implemented
  - Control signals, interfaces, memory technology.
  - e.g. Is there a hardware multiply unit or is it done by repeated addition?

1.3 Structure and Function

- Structure is the way in which components relate to each other.
- Function is the operation of individual components as part of the structure
- All computer functions are:
  - **Data processing:** Computer must be able to process data which may take a wide variety of forms and the range of processing.
  - **Data storage:** Computer stores data either temporarily or permanently.
  - **Data movement:** Computer must be able to move data between itself and the outside world.
  - **Control:** There must be a control of the above three functions.

1.3 Structure and Function

- Functional view of a computer:



1.3 Structure and Function

- Structure: Simplest possible view of a computer:

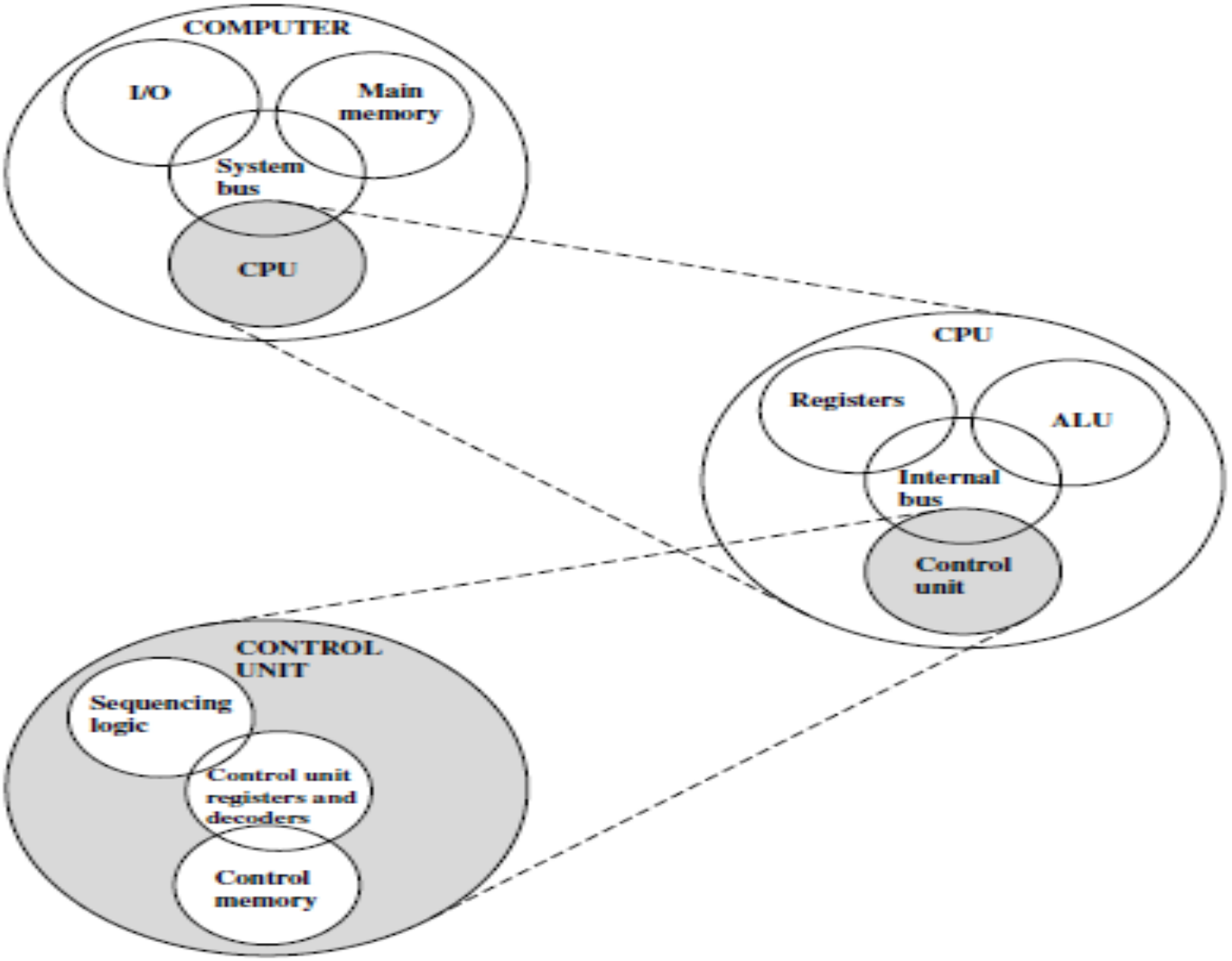


Figure 1.4 The Computer: Top-Level Structure

1.3 Structure and Function

- Internal Structure of the Computer Itself:
  - Central Processing Unit (CPU):
    - Controls the operation of the computer and performs its data processing functions. Often simply referred to as processor.
  - 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.

1.3 Structure and Function

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

1.3 Structure and Function

- (Micro programmed) Control Unit Structure :
  - Sequencing logic:
    - Sequential logic is used to construct finite state machines, a basic building block in all digital circuitry, as well as memory circuits and other devices.
    - Digital sequential logic circuits are divided into synchronous and asynchronous types.
    - In synchronous sequential circuits, the state of the device changes only at discrete times in response to a clock signal.
    - In asynchronous circuits the state of the device can change at any time in response to changing inputs..

1.3 Structure and Function

- (Micro programmed) Control Unit Structure :
  - Control unit, register and decoder:
    - These are the instruction register and the program counter and are used together with the instruction decoder in the execution cycle.
    - As was said earlier, the system program is stored in ROM. The program counter is used for storing the position of the next instruction.
    - The instruction register is used for storing the current instruction.
    - The instruction decoder decodes the instruction (ie; it determines, from the instruction, what operation to perform).

1.3 Structure and Function

- (Micro programmed) Control Unit Structure :
  - Control memory:
    - The set of microinstructions is stored in the control memory.
    - To execute an instruction, the sequencing logic unit issues a READ command to the control memory.

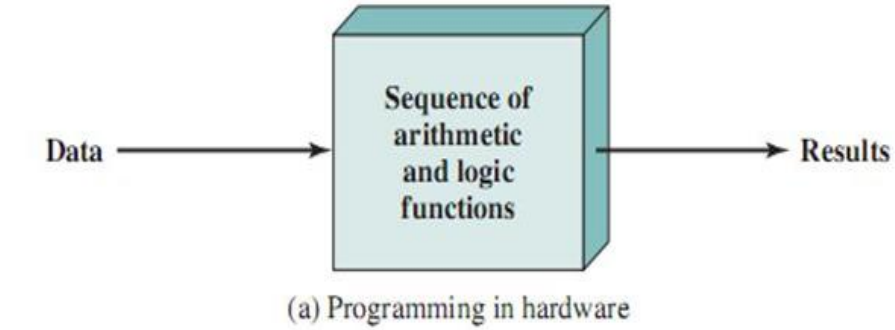
1.4 Computer Components

- Computer components:
  - Virtually all contemporary computer designs are based on concepts developed by John von
  - Neumann at the Institute for Advanced Studies, Princeton. Such a design is 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.

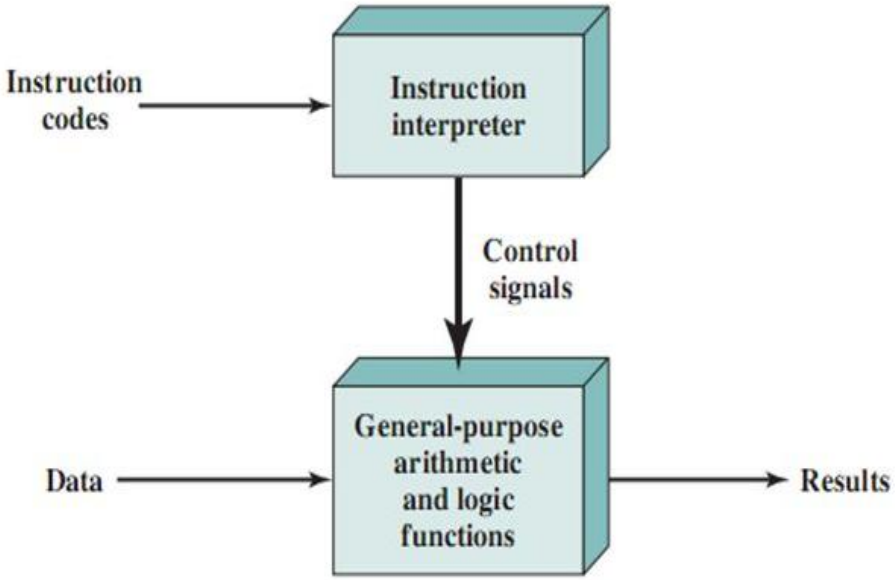
1.4 Computer Components

- Computer components:
  - If there is a particular computation to be performed, a configuration of logic components designed specifically for that computation could be constructed.
  - The resulting “program” is in the form of hardware and is termed a hardwired program.
  - Now consider this alternative. 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 Fig. a.

1.4 Computer Components



(a) Programming in hardware



(b) Programming in software

Hardware and Software Approaches

1.4 Computer Components

- Computer components:
  - 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 by providing a unique code for each possible set of control signals, and let us add to the general-purpose hardware a segment that can accept a code and generate control signals Fig. b.
  - To distinguish this new method of programming, a sequence of codes or instructions is called software.

1.4 Computer Components

- Computer components:
  - Fig. b, indicates two major components of the system: an instruction interpreter and a module of general-purpose arithmetic and logic functions. These two constitute the CPU.
  - Data and instructions must be put into the system. For this we need some sort of input module.
  - A means of reporting results is needed, and this is in the form of an output module. Taken together, these are referred to as I/O components.

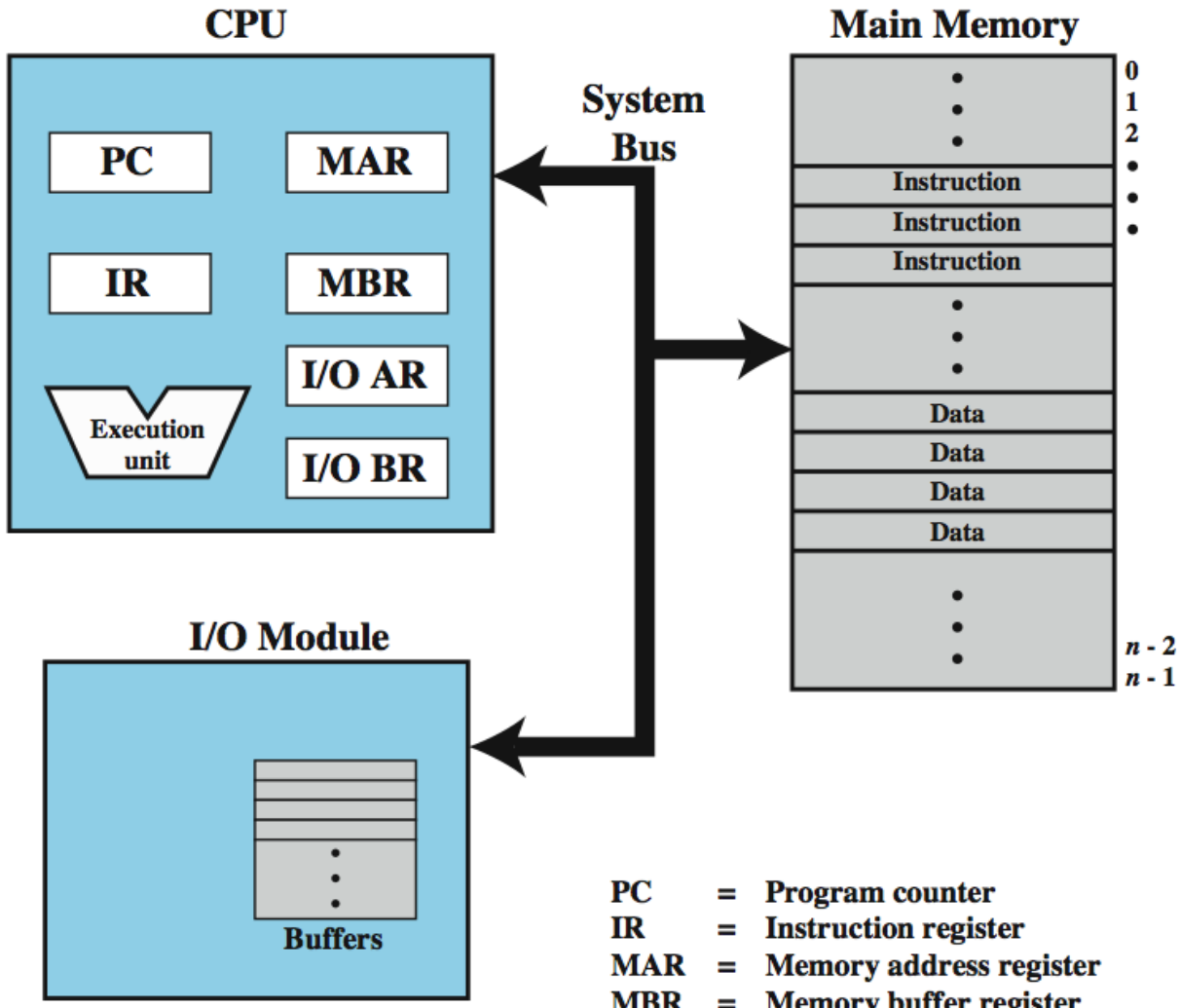
1.4 Computer Components

- Computer components:
  - There must be a place to store temporarily both instructions and data. That module is called memory, or main memory to distinguish it from external storage or peripheral devices.
  - The CPU exchanges data with memory. For this purpose, it typically makes use of two internal (to the CPU) registers:
    - A memory address register (MAR): Which specifies the address in memory for the next read or write.
    - A memory buffer register (MBR): Which contains the data to be written into memory or receives the data read from memory.
    - An I/O address register (I/OAR): Specifies a particular I/O device.
    - An I/O buffer register (I/OBR): Is used for the exchange of data between an I/O module and the CPU.

1.4 Computer Components

- Computer components:
  - Each location contains a binary number that can be interpreted as either an instruction or data.
  - An I/O module transfers data from external devices to CPU and memory, and vice versa.
  - It contains internal buffers for temporarily holding these data until they can be sent on.

1.4 Computer Components

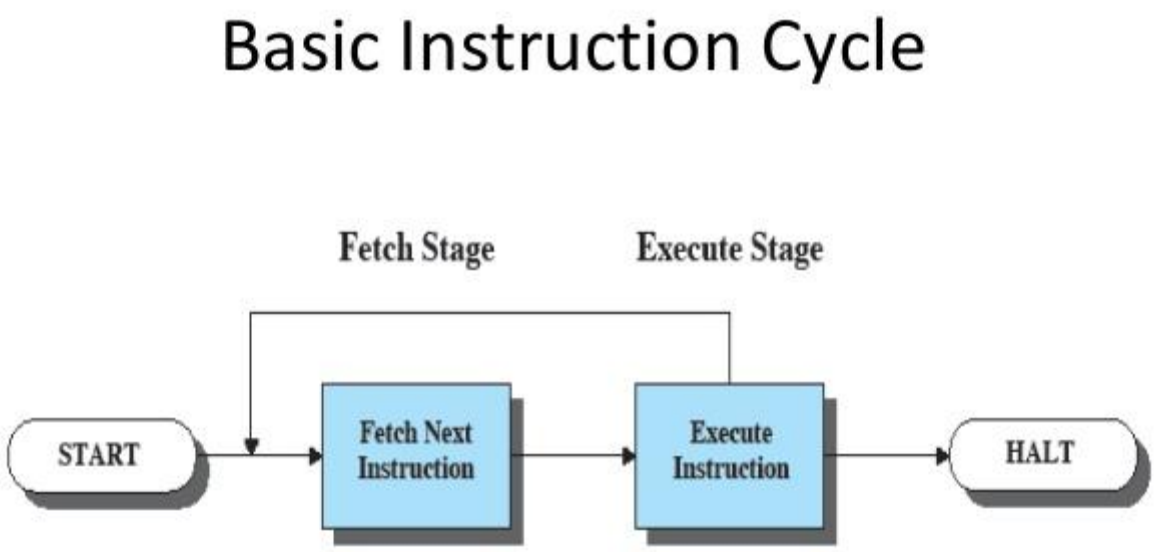


1.5 Computer Functions

- Basic Instruction Cycle:
  - The basic function performed by a computer is execution of a program, which consists of a set of instructions stored in memory.
  - Instruction processing consists of two steps:
    - The processor reads (fetches) instructions from memory one at a time and executes each instruction.
    - Program execution consists of repeating the process of instruction fetch and instruction execution.
  - The processing required for a single instruction is called an instruction cycle.
  - The two steps are referred to as the fetch cycle and the execute cycle.

1.5 Computer Functions

- Basic Instruction Cycle:
  - Program execution halts only if the machine is turned off, some sort of unrecoverable error occurs, or a program instruction that halts the computer is encountered.



1.5 Computer Functions

- Instruction Fetch and Execute:
  - 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 always increments the PC after each instruction fetch so that it will fetch the next instruction in sequence.
  - The fetched instruction is loaded into a register in the processor known as the instruction register (IR).
  - The processor interprets the instruction and performs the required action.

1.5 Computer Functions

- Instruction Fetch and Execute:
  - In general, these actions fall into four categories:
    - Processor-memory: Data may be transferred from processor to memory or from memory to processor.
    - Processor-I/O: Data may be transferred to or from a peripheral device by transferring between the processor and an I/O module.
    - Data processing: The processor may perform some arithmetic or logic operation on data.
    - Control: An instruction may specify that the sequence of execution be altered.

1.5 Computer Functions

- Instruction Fetch and Execute:
  - For example, the processor may fetch an instruction from location 149, which specifies that the next instruction be from location 182.
  - The processor will remember this fact by setting the program counter to 182.
  - Thus, on the next fetch cycle, the instruction will be fetched from location 182 rather than 150.

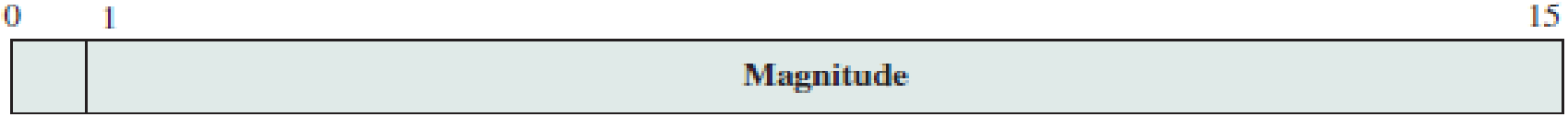
1.5 Computer Functions

- Instruction Fetch and Execute:
  - An instruction's execution may involve a combination of these actions:
    - The program counter (PC) holds the address of the instruction to be fetched next, the processor always increments the PC after each instruction fetch so that it will fetch the next instruction in sequence.
    - The fetched instruction is loaded into a register in the processor known as the instruction register (IR).
    - The processor also contains a single data register, called an accumulator (AC).

1.5 Computer Functions

- Instruction Fetch and Execute:
  - Characteristics of a Hypothetical machine:

(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

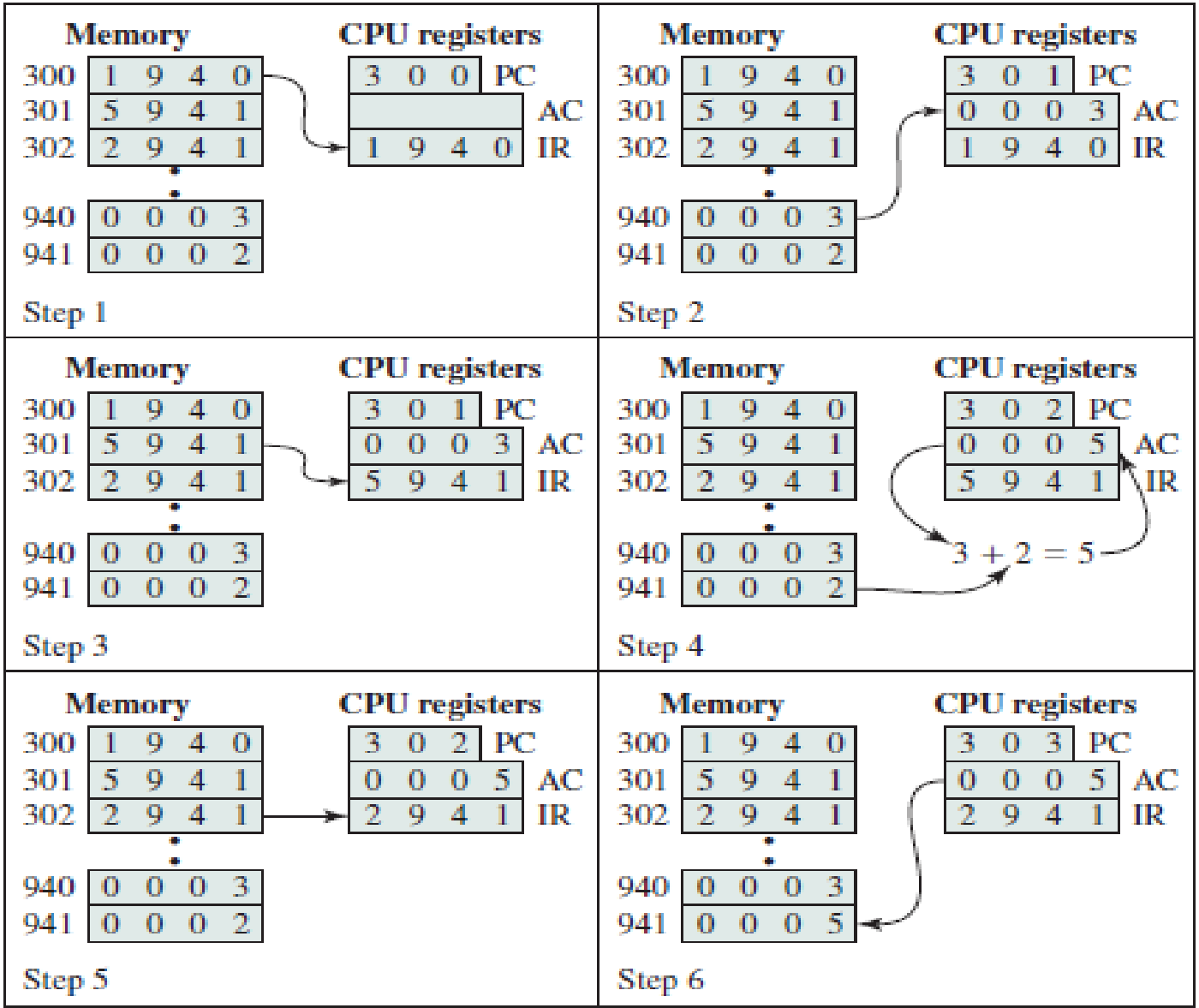
(d) Partial list of opcodes

1.5 Computer Functions

- Instruction Fetch and Execute:
  - Example given in figure illustrates a partial program execution, showing the relevant portions of memory and processor registers.
  - The program fragment shown adds the contents of the memory word at address 940 to the contents of the memory word at address 941 and stores the result in later location.

1.5 Computer Functions

- Instruction Fetch and Execute:



1.5 Computer Functions

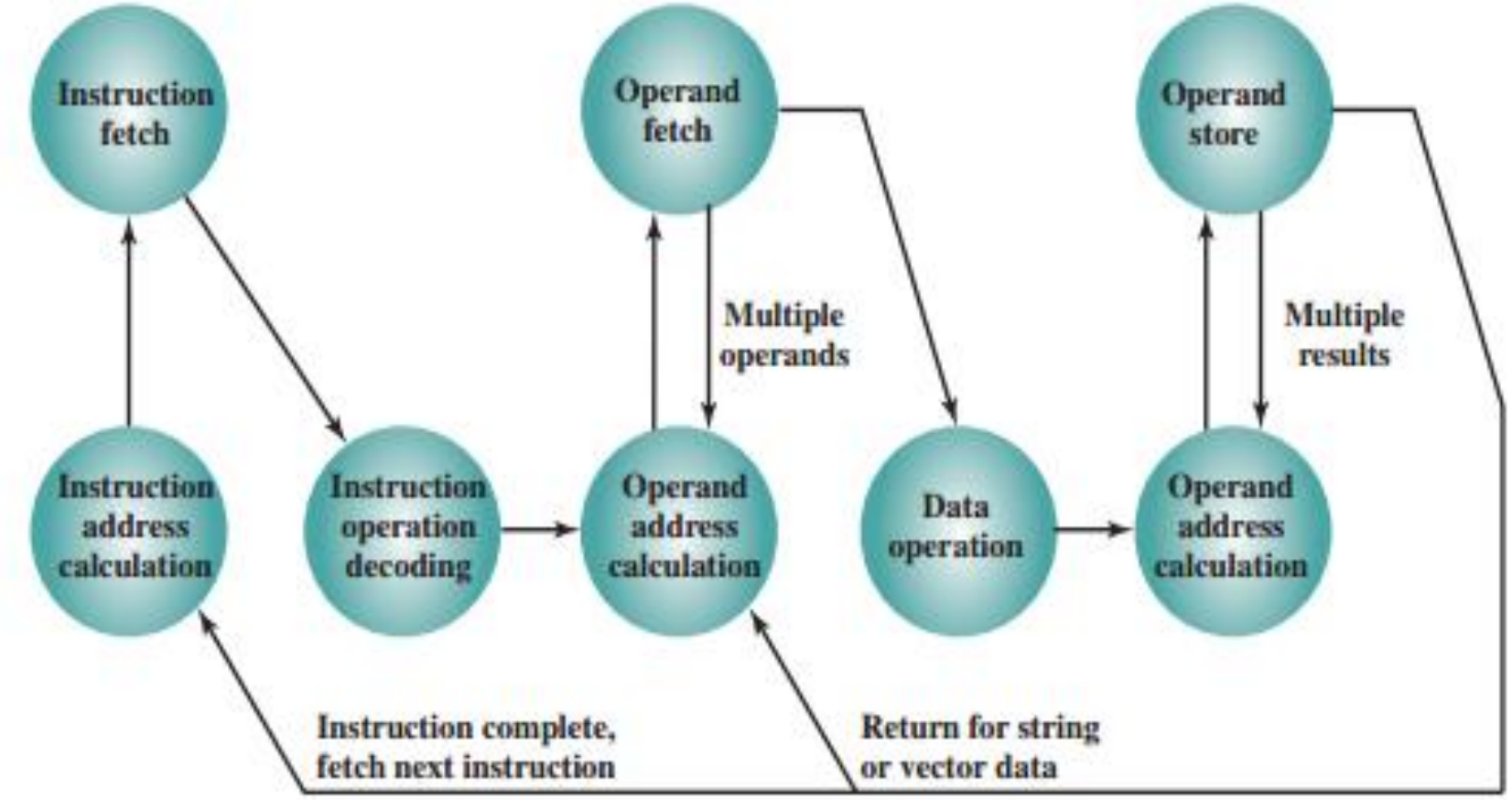
- Instruction Fetch and Execute:
  - Three instructions are required:
    1. The PC contains 300, the address of the first instruction. This instruction (the value 1940 in hexadecimal) is loaded into the instruction register IR and the PC is incremented.
    2. The first hexadecimal digit in the IR indicate that the AC is to be loaded. The remaining three hexadecimal digits specify the address (940) from which data are to be loaded.
    3. The next instruction (5941) is fetched from location 301 and the PC is incremented.
    4. The old contents of the AC and the contents of location 941 are added and the result is stored in the AC.
    5. The next instruction (2941) is fetched from location 302 and the PC is incremented.
    6. The contents of the AC are stored in location 941.

1.5 Computer Functions

- Instruction Fetch and Execute:
  - Some processor includes an instruction, expressed symbolically as ADD B,A, that stores the sum of the contents of memory locations B and A into memory location A. A single instruction cycle with the following steps occurs:
    - Fetch the ADD instruction.
    - Read the contents of memory location A into the processor.
    - Read the contents of memory location B into the processor.
    - In order that the contents of A are not lost, the processor must have at least two registers for storing memory values, rather than a single accumulator.
    - Add the two values.
    - Write the result from the processor to memory location A.

1.5 Computer Functions

- Instruction Fetch and Execute:
  - Figure provides a more detailed look at the basic instruction cycle. Figure is in the form of a state diagram.

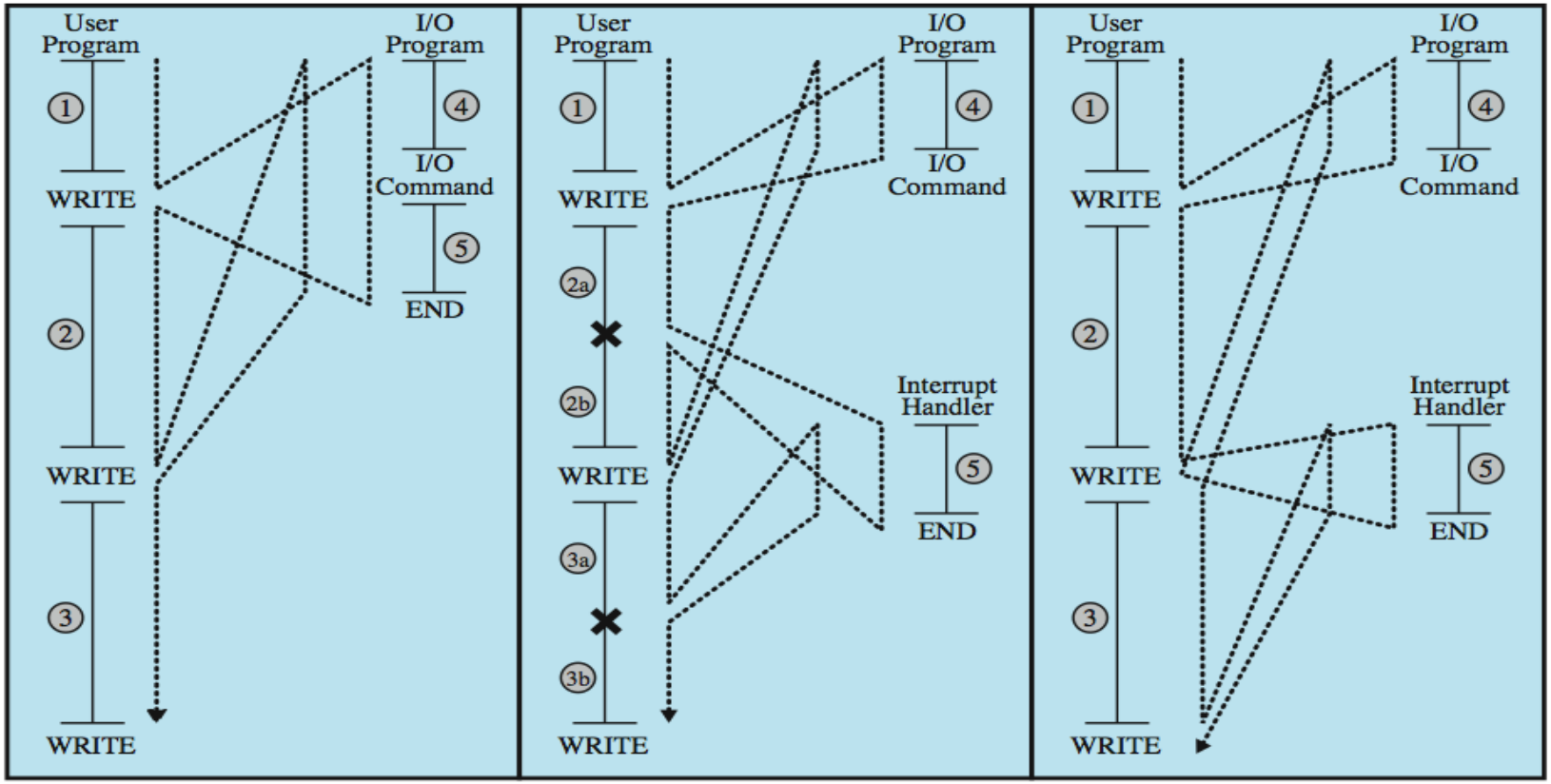


1.5 Computer Functions

- Instruction Fetch and Execute:
  - The states can be described as follows:
    - **Instruction address calculation (iac):** Determine the address of the next instruction to be executed.
    - **Instruction fetch (if):** Read instruction from its memory location into the processor.
    - **Instruction operation decoding (iod):** Analyze instruction to determine type of operation to be performed and operand(s) to be used.
    - **Operand address calculation (oac):** If the operation involves reference to an operand in memory or available via I/O, then determine the address of the operand.
    - **Operand fetch (of):** Fetch the operand from memory or read it in from I/O.
    - **Data operation (do):** Perform the operation indicated in the instruction.
    - **Operand store (os):** Write the result into memory or out to I/O.

1.5 Computer Functions

- Interrupts:
  - Virtually all computers provide a mechanism by which other modules (I/O, memory) may interrupt the normal processing of the processor.



(a) No interrupts (b) Interrupts; short I/O wait (c) Interrupts; long I/O wait

1.5 Computer Functions

- Interrupts:
  - Figure (1.9 a) illustrates this state of affairs. The user program performs a series of WRITE calls.
  - Code segments 1, 2, and 3 refer to sequences of instructions that do not involve I/O.
  - A sequence of instructions, labeled 4 in the figure, to prepare for the actual I/O operation.
  - Once the actual I/O command, without the use of interrupts, is issued, the program must wait for the I/O device to perform the requested function.
  - The program might wait by simply repeatedly performing a test operation to determine if the I/O operation is done.

1.5 Computer Functions

- Interrupts:
  - A sequence of instructions, labeled 5 in the figure, to complete the operation.
  - With interrupts, the processor can be engaged in executing other instructions while an I/O operation is in progress.
  - Consider the flow of control in Figure 1.9b.
  - As before, the user program reaches a point at which it makes a system call in the form of a WRITE call.
  - After these few instructions have been executed, control returns to the user program.

1.5 Computer Functions

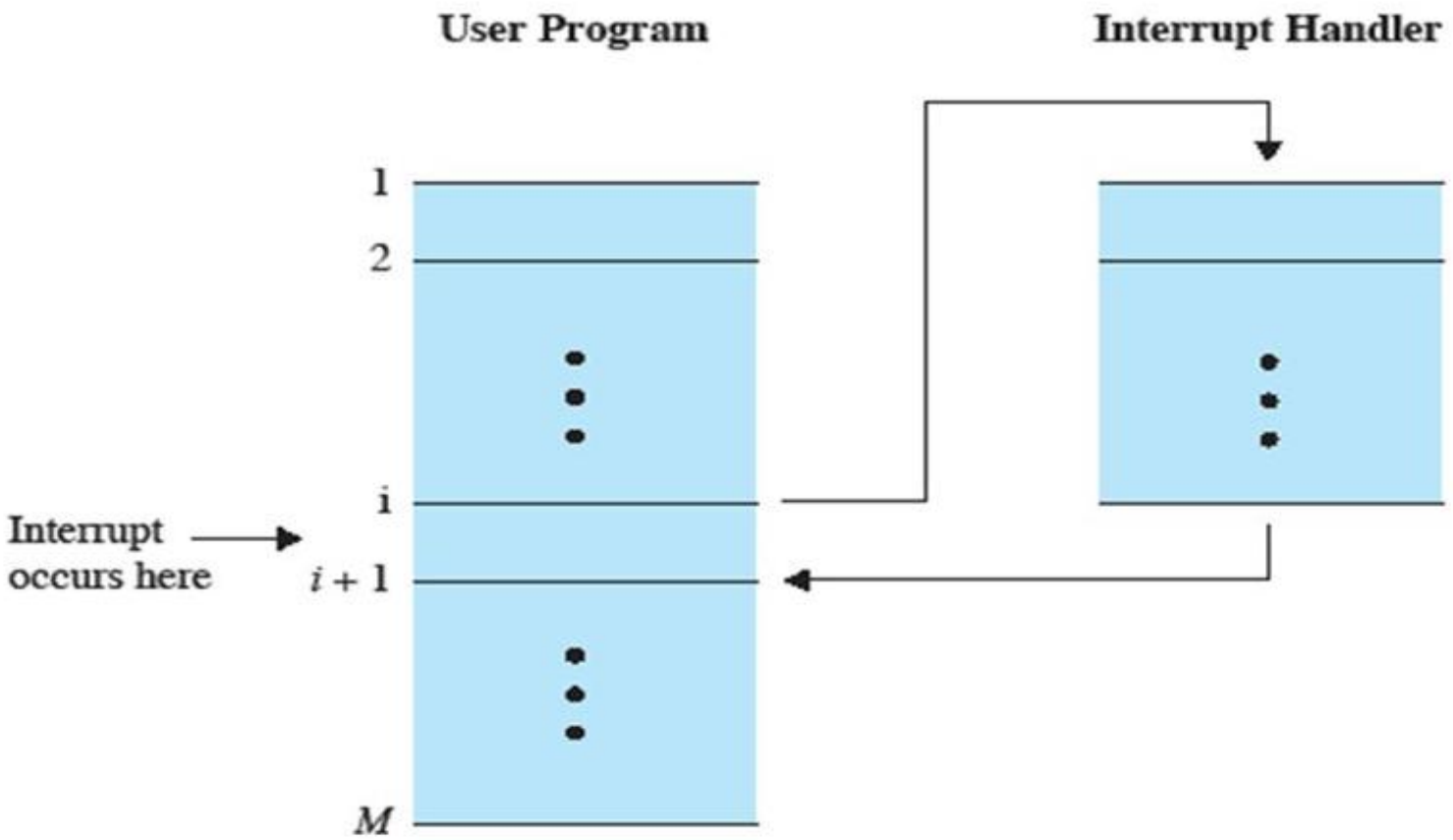
- Interrupts:
  - Meanwhile, the external device (Ex: A printer) is busy accepting data from computer memory and printing it.
  - This I/O operation is conducted concurrently with the execution of instructions in the user program.
  - When the external device becomes ready to accept more data from the processor,— the I/O module for that external device sends an interrupt request signal to the processor.

1.5 Computer Functions

- Interrupts:
  - The processor responds by suspending operation of the current program, branching off to a program to service that particular I/O device, known as an interrupt handler, and resuming the original execution after the device is serviced.
  - The points at which such interrupts occur are indicated by an asterisk in Figure 1.9 b.
  - From the point of view of the user program, an interrupt is just that: an interruption of the normal sequence of execution.

1.5 Computer Functions

- Interrupts:
  - When the interrupt processing is completed, execution resumes as shown in the figure.

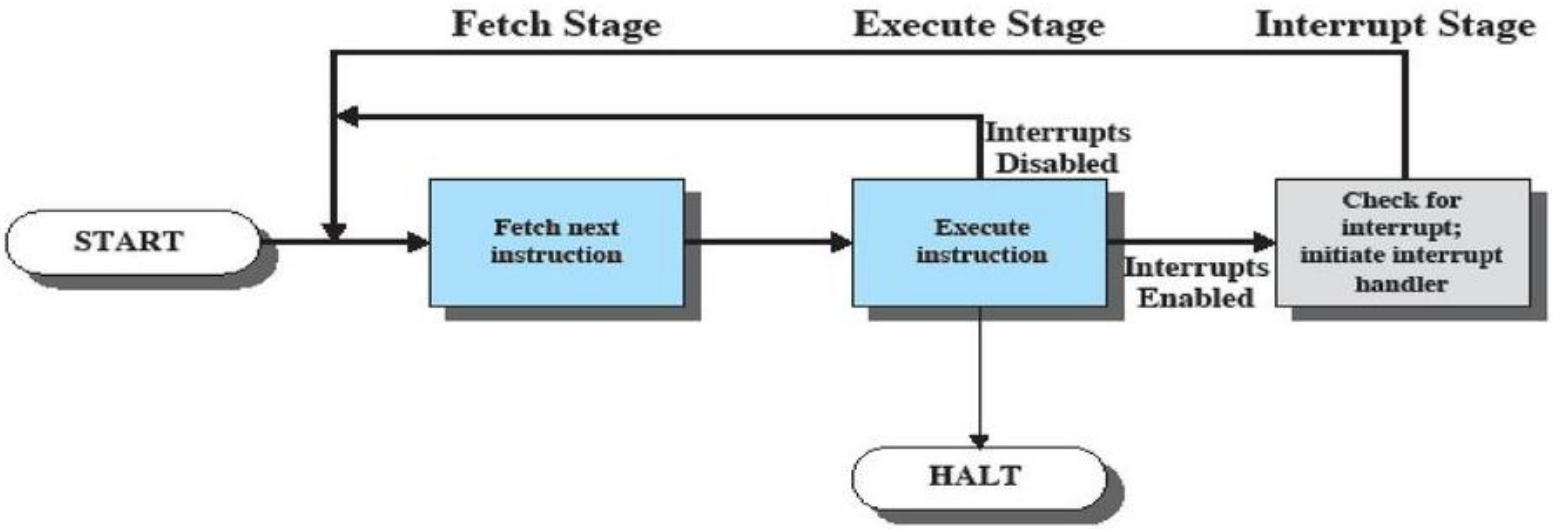


1.5 Computer Functions

- Interrupts:
  - Figure 1.9c indicates this state of affairs.
  - In this case, the user program reaches the second WRITE call before the I/O operation spawned by the first call is complete. The result is that the user program is hung up at that point.
  - When the preceding I/O operation is completed, this new WRITE call may be processed, and a new I/O operation may be started.

1.5 Computer Functions

- Interrupts and the instruction cycle:
  - To accommodate interrupts, an interrupt cycle is added to the instruction cycle, as shown in figure.



1.5 Computer Functions

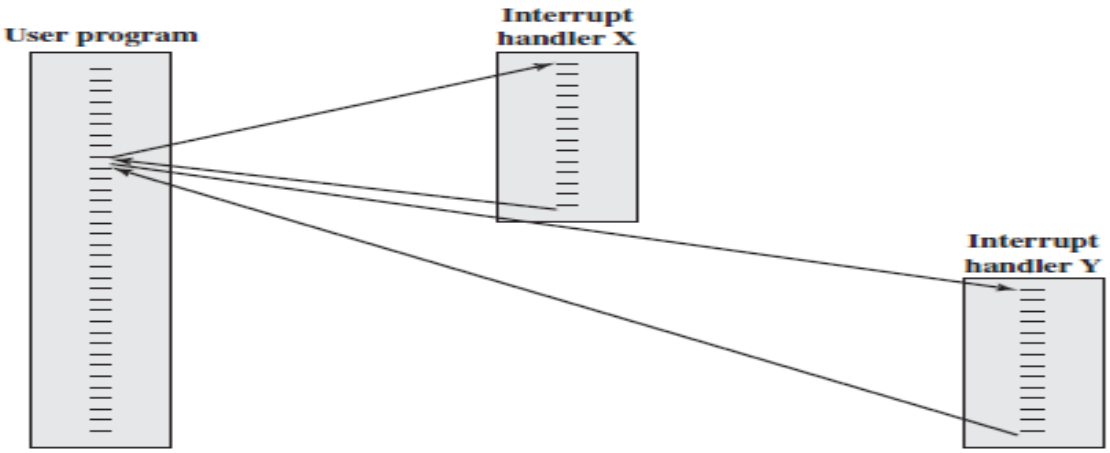
- Interrupts and the instruction cycle:
  - In the interrupt cycle, the processor checks to see if any interrupts have occurred.
  - If no interrupts are pending, the processor proceeds to the fetch cycle and fetches the next instruction of the current program.
  - If an interrupt is pending, the processor does the following:
    - It suspends execution of the current program being executed and saves its context
    - It sets the program counter to the starting address of an interrupt handler routine.
  - The processor now proceeds to the fetch cycle and fetches the first instruction in the interrupt handler program, which will service the interrupt.
  - When the interrupt handler routine is completed, the processor can resume execution of the user program at the point of interruption.

1.5 Computer Functions

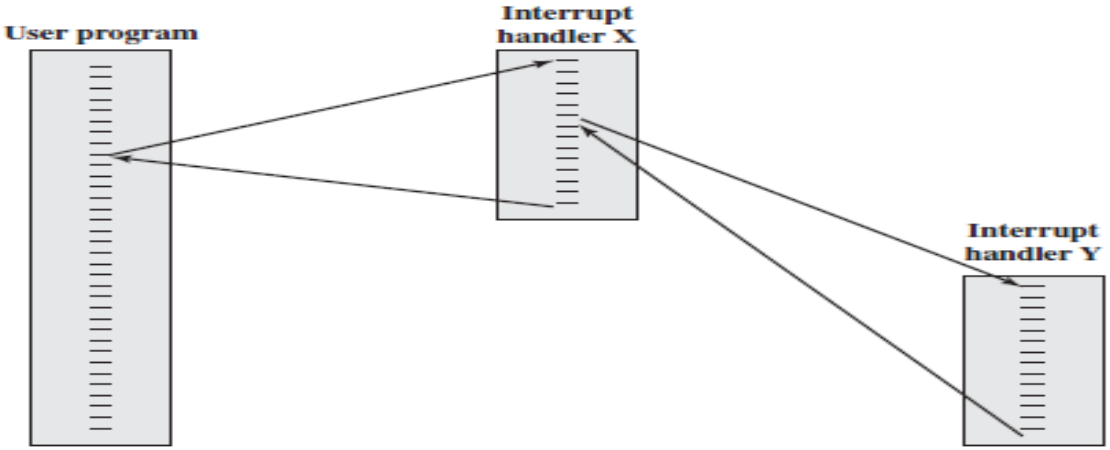
- Multiple interrupts:
  - Multiple interrupts can occur. Two approaches can be taken to dealing with multiple interrupts.
  - The first is to disable interrupts while an interrupt is being processed.
  - A disabled interrupt simply means that the processor can and will ignore that interrupt request signal.
  - Thus, when a user program is executing and an interrupt occurs, interrupts are disabled immediately.
  - After the interrupt handler routine completes, interrupts are enabled before resuming the user program and the processor checks to see if additional interrupts have occurred.
  - This approach is nice and simple, as interrupts are handled in strict sequential order as shown in figure (a).

1.5 Computer Functions

- Multiple interrupts:



(a) Sequential interrupt processing



(b) Nested interrupt processing

Figure 3.13 Transfer of Control with Multiple Interrupts

1.5 Computer Functions

- Multiple interrupts:
  - The drawback to the preceding approach is that it does not take into account relative priority or time-critical needs
  - A second approach is to define priorities for interrupts and to allow an interrupt of higher priority to cause a lower-priority interrupt handler to be itself interrupted, as shown in figure (b).
  - As an example of this second approach, consider a system with three I/O devices: a printer, a disk, and a communications line, with increasing priorities of 2, 4, and 5, respectively.

1.5 Computer Functions

- Multiple interrupts:
  - A user program begins at t = 0.
  - At t = 10, a printer interrupt occurs; user information is placed on the system stack and execution continues at the printer interrupt service routine (ISR).
  - While this routine is still executing, at t = 15, a communications interrupt occurs.
  - Because the communications line has higher priority than the printer, the interrupt is honored.
  - The printer ISR is interrupted, its state is pushed onto the stack and execution continues at the communications ISR.
  - While this routine is executing, a disk interrupt occurs (t = 20), because this interrupt is of lower priority, it is simply held, and the communications ISR runs to completion.

1.5 Computer Functions

- Multiple interrupts:
  - When the communications ISR is complete (t = 25), the previous processor state is restored, which is the execution of the printer ISR.
  - However, before even a single instruction in that routine can be executed, the processor honors the higher-priority disk interrupt and control transfers to the disk ISR.
  - Only when that routine is complete (t = 35) is the printer ISR resumed.
  - When that routine completes (t = 40), control finally returns to the user program.

1.5 Computer Functions

- Multiple interrupts:

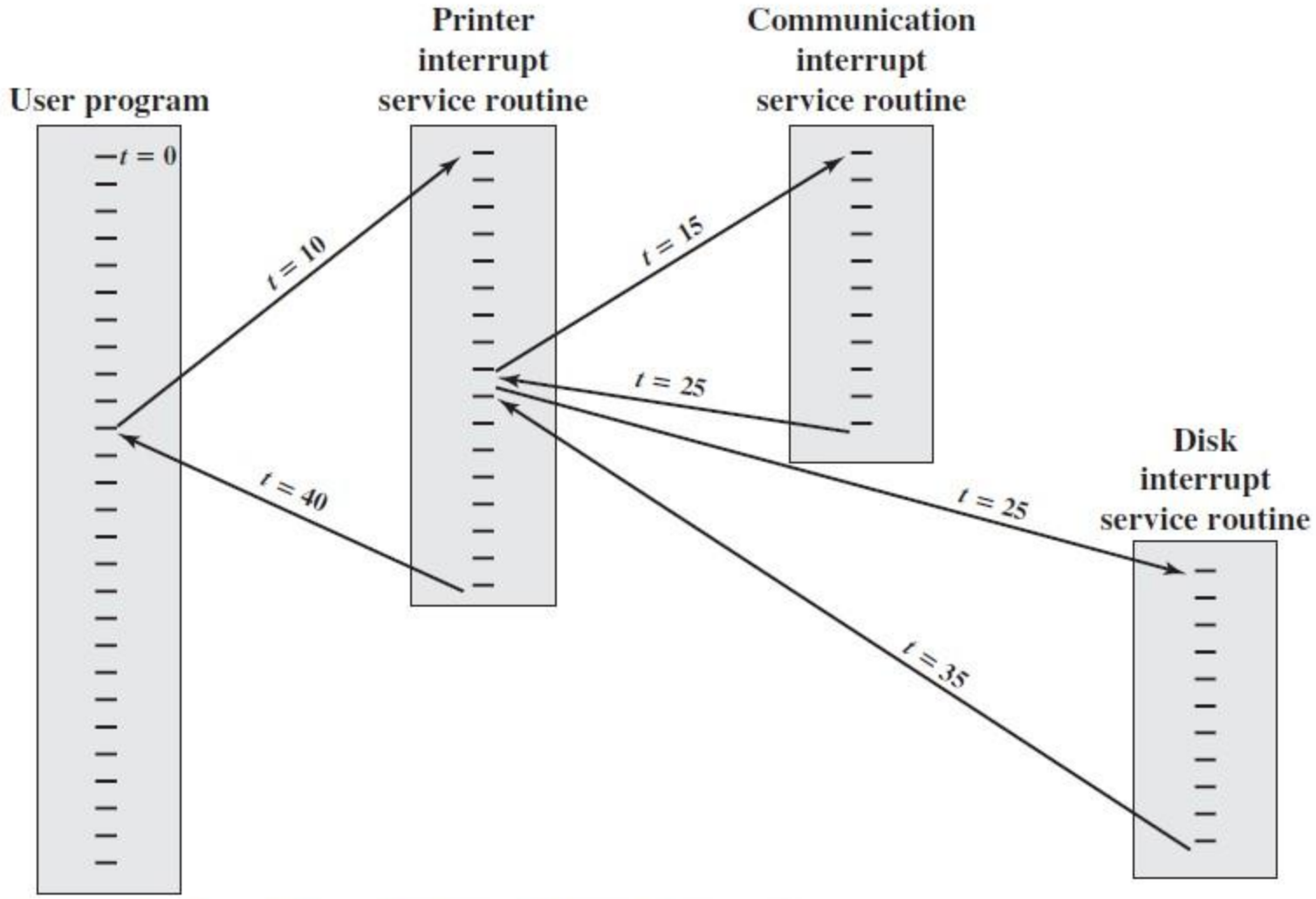


Figure 3.14 Example Time Sequence of Multiple Interrupts

THANK YOU