Basic Computer Concepts
UNIT -2

Computer Organization and Architecture:

Computer architecture refers to those attributes of a system visible to a programmer or, put another way, those attributes that have a direct impact on the logical execution of a program. A term that is often used interchangeably with computer architecture is **Instruction Set Architecture (ISA)**. The ISA defines instruction formats, instruction opcodes, registers, instruction and data memory; the effect of executed instructions on the registers and memory; and an algorithm for controlling instruction execution.

Computer organization refers to the operational units and their interconnections that realize the architectural specifications. Examples of architectural attributes include the instruction set, the number of bits used to represent various data types, I/O mechanisms, and techniques for addressing memory. Organizational attributes include those hardware details transparent to the programmer, such as control signals; interfaces between the computer and peripherals; and the memory technology used.

The different models in the family have different price and performance characteristics. A prominent example of both these phenomena is the IBM System/370 architecture. This architecture was first introduced in 1970 and included a number of models. The System/370 architecture, with a few enhancements, has survived to this day as the architecture of IBM's mainframe product line. There is more interplay between organizational and architectural design decisions. An intriguing example of this is the reduced instruction set computer (RISC).

Cache Memory : It is a memory that is smaller and faster than main memory and that is interposed between the processor and main memory. The cache acts as a buffer for recently used memory locations.

Basic Computer Concepts
UNIT -2

Structure and Function :

The hierarchical nature of complex systems is essential to both their design and their description. The behavior at each level depends only on a simplified, abstracted characterization of the system at the next lower level. At each level, the designer is concerned with structure and function:
Structure: The way in which the components are interrelated.
Function: The operation of each individual component as part of the structure.

Function : There are only four basic functions that a computer can perform:

- **Data processing:** Data may take a wide variety of forms, and the range of processing requirements is broad.
- **Data storage:** Even if the computer is processing data on the fly the computer must temporarily store at least those pieces of data that are being worked on at any given moment. There is at least a short-term data storage function. The computer performs a long-term data storage function. Files of data are stored on the computer for subsequent retrieval and update.
- **Data movement:** The computer's operating environment consists of devices that serve as either sources or destinations of data. When data are received from or delivered to a device that is directly connected to the computer, the process is known as input–output (I/O), and the device is referred to as a peripheral. When data are moved over longer distances the process is known as data communications.
- **Control:** Within the computer, a control unit manages the computer's resources and orchestrates the performance of its functional parts in response to instructions.

Structure :

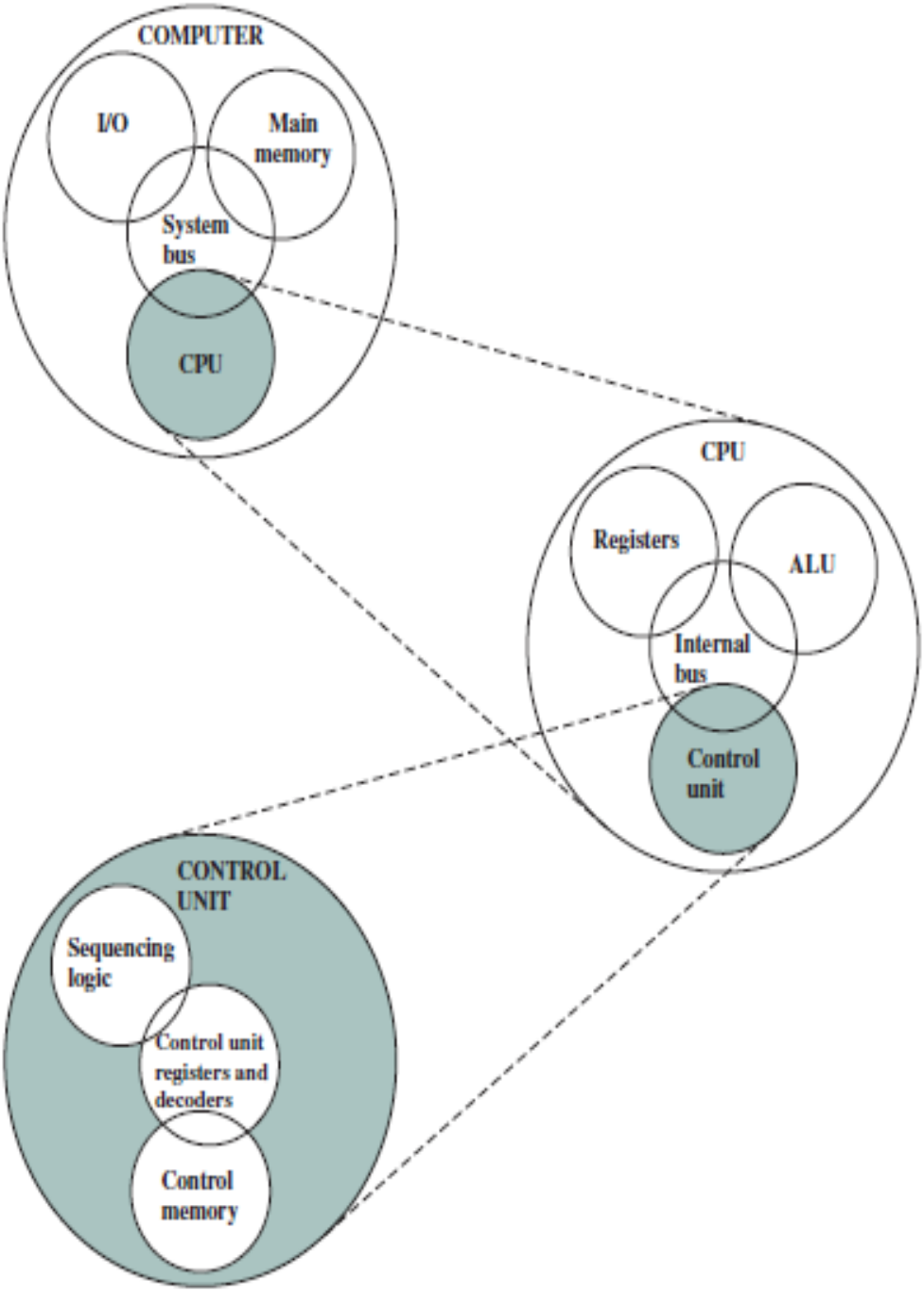
Simple Single-Processor computer

There are four main structural components:

1. **Central processing unit (CPU):** Controls the operation of the computer and performs its data processing functions; often simply referred to as processor.
2. **Main memory:** Stores data.
3. **I/O:** Moves data between the computer and its external environment.
4. **System interconnection:** A common example of system interconnection is by means of a system bus, consisting of a number of conducting wires to which all the other components attach.

Major Components of CPU :

a. **CU(Control unit):** Controls the operation of the CPU and hence the computer.
b. **Arithmetic and logic unit (ALU):** Performs the computer's data processing functions.
c. **Registers:** Provides storage internal to the CPU.
d. **CPU interconnection:** Some mechanism that provides for communication among the control unit, ALU, and registers.



The Computer: Top-Level Structure

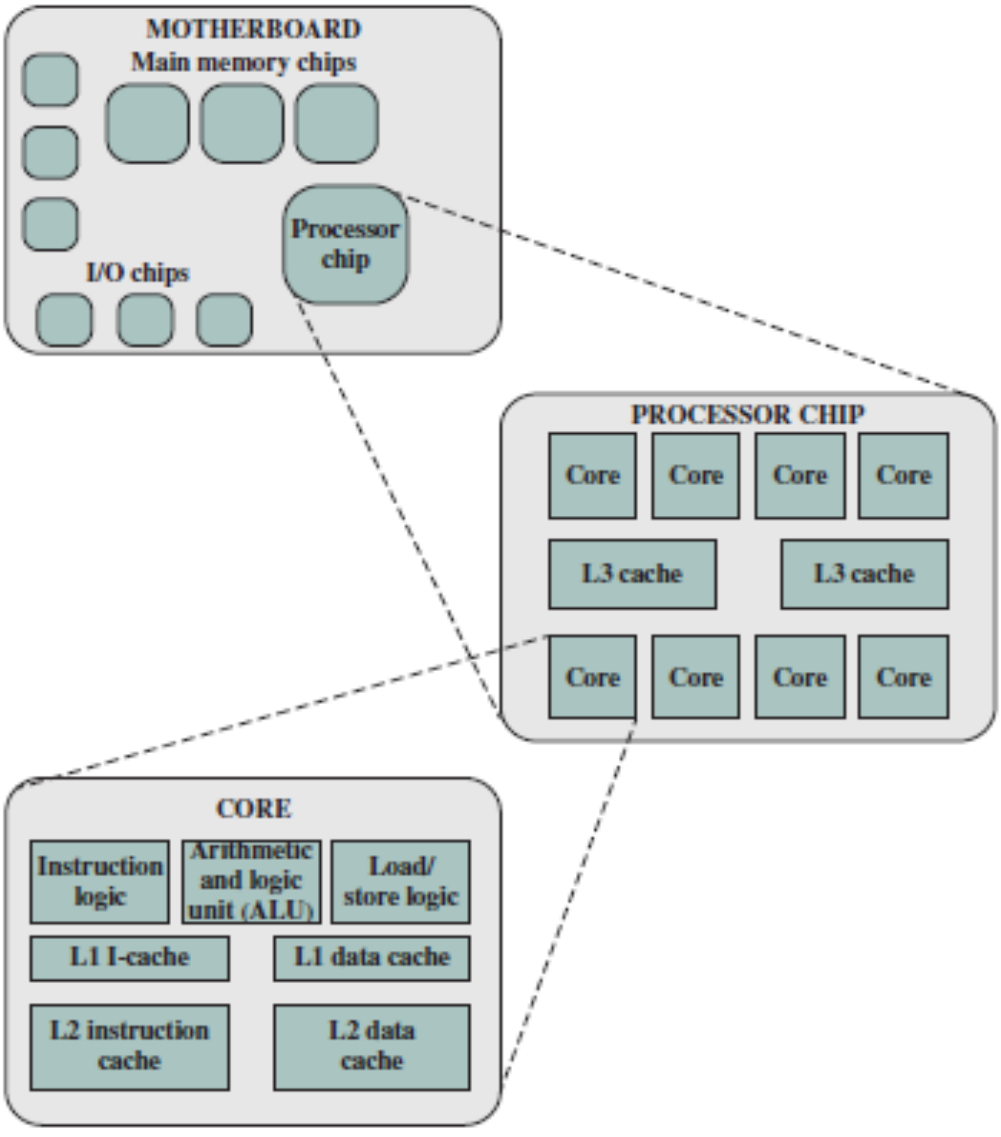
multicore computer structure :

Contemporary computers generally have multiple processors. When these processors all reside on a single chip, the term multicore computer is used, and each processing unit (consisting of a control unit, ALU, registers, and perhaps cache) is called a core.

Central processing unit (CPU): That portion of a computer that fetches and executes instructions. It consists of an ALU, a control unit, and registers. It is often simply referred to as a it is often simply referred to as a processor.

Core: An individual processing unit on a processor chip. A core may be equivalent in functionality to a CPU on a single-CPU system. Other specialized processing units, such as one optimized for vector and matrix operations, are also referred to as cores.

Processor: A physical piece of silicon containing one or more cores. The processor is the computer component that interprets and executes instructions. If a processor contains multiple cores, it is referred to as a multicore processor.



Simplified View of Major Elements of a Multicore Computer

Use of multiple layers of memory is called **cache memory** between the processor and main memory. Cache memory is smaller and faster than main memory and is 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 **printed circuit board (PCB)** is a rigid, flat board that holds and interconnects chips and other electronic components. The board is made of layers, typically two to ten, that interconnect components via copper pathways that are etched into the board. The main printed circuit board in a computer is called **a system board or motherboard**, while smaller ones that plug into the slots in the main board are called expansion boards.

A **chip** is a single piece of semiconducting material, typically silicon, upon which electronic circuits and logic gates are fabricated. The resulting product is referred to as **an integrated circuit.**

The motherboard contains a slot or socket for the processor chip, which typically contains multiple individual cores, in what is known as a **multicore processor.**

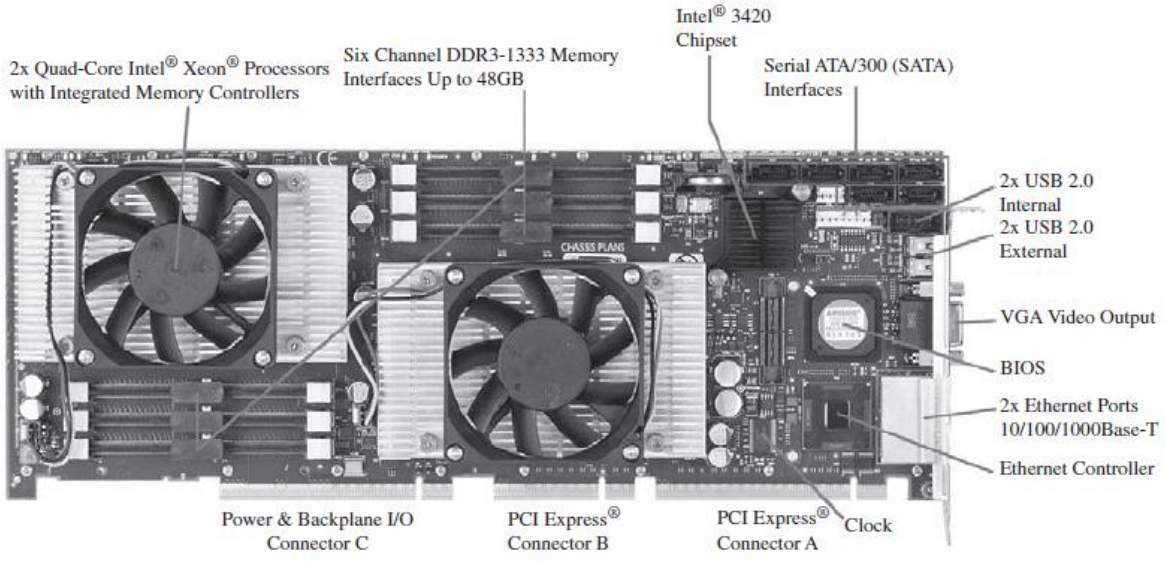
The functional elements of a core are:

A. **Instruction logic:** This includes the tasks involved in fetching instructions, and decoding each instruction to determine the instruction operation and the memory locations of any operands.
B. **Arithmetic and logic unit (ALU):** Performs the operation specified by an instruction.
C. **Load/store logic:** Manages the transfer of data to and from main memory via cache.

The core also contains an L1 cache, split between an instruction cache (I-cache)that is used for the transfer of instructions to and from main memory, and an L1 data cache, for the transfer of operands and results.

Example : Motherboard with Two Intel Quad-Core Xeon Processors.

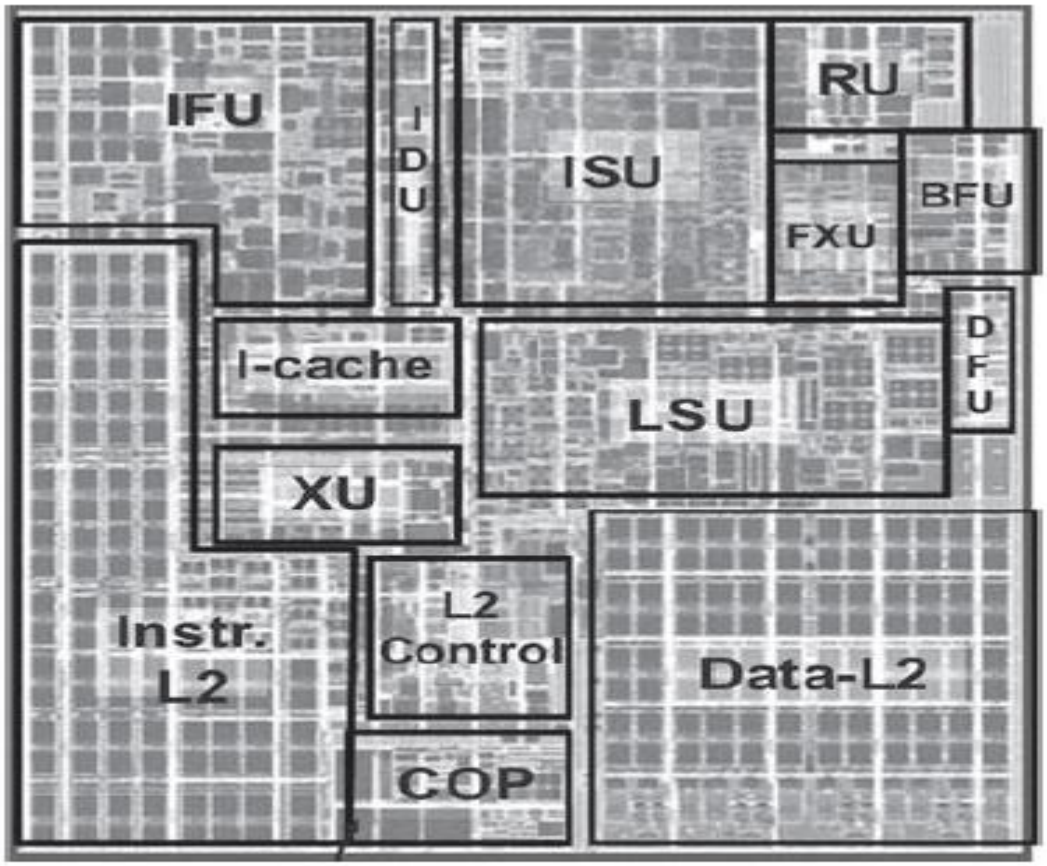
PCI(Peripheral Components Interconnection)- Express slots for a high-end display adapter and for additional peripherals.



Motherboard with Two Intel Quad-Core Xeon Processors
Source: Chassis Plans, www.chassis-plans.com

Serial Advanced Technology Attachment (SATA) sockets for connection to disk memory.

The internal structure of a single core is shown in figure below.



zEnterprise EC12 Core layout
Source: IBM zEnterprise EC12 Technical Guide, December 2013, SG24-8049-01. IBM, Reprinted by Permission

ISU (Instruction Sequence Unit): Determines the sequence in which instructions are executed.
IFU (Instruction Fetch Unit): Logic for fetching instructions.
IDU (Instruction Decode Unit): The IDU is fed from the IFU buffers, and is responsible for the parsing and decoding of all z/Architecture operation codes.
LSU (Load-Store Unit): The LSU contains the 96-kB L1 data cache,1 and manages data traffic between the L2 data cache and the functional execution units. It is responsible for handling all types of operand accesses of all lengths, modes and formats.
XU (Translation Unit): This unit translates logical addresses from instructions into physical addresses in main memory. The XU also contains a translation lookaside buffer (TLB) used to speed up memory access.
FXU (Fixed-Point Unit): The FXU executes fixed-point arithmetic operations.
BFU (Binary Floating-Point Unit): The BFU handles all binary and hexadecimal floating-point operations, as well as fixed-point multiplication operations.
DFU (Decimal Floating-Point Unit): The DFU handles both fixed-point and floating-point operations on numbers that are stored as decimal digits.
RU (Recovery Unit): The RU keeps a copy of the complete state of the system that includes all registers, collects hardware fault signals, and manages the hardware recovery actions.

COP (Dedicated Co-Processor): The COP is responsible for data compression and encryption functions for each core.
I-cache: This is a 64-kB L1 instruction cache, allowing the IFU to prefetch instructions before they are needed.
L2 control: This is the control logic that manages the traffic through the two L2 caches.
Data-L2: A 1-MB L2 data cache for all memory traffic other than instructions.
Instr-L2: A 1-MB L2 instruction cache.

The Evolution of the Intel x86 Architecture :

The 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).**

The evolution of the Intel product line:

8080: The world's first general-purpose microprocessor. This was an 8-bit machine, with an 8-bit data path to memory. The 8080 was used in the first personal computer.

8086: It is 16-bit machine. In addition to a wider data path and larger registers, the 8086 sported an instruction cache, or queue, that prefetches a few instructions before they are executed. The 8086 is the first appearance of the x86 architecture.

80286: This extension of the 8086 enabled addressing a 16-MB memory instead of just 1MB.

80386: Intel's first 32-bit machine. This was the first Intel processor to support multitasking, meaning it could run multiple programs at the same time.

80486: The 80486 introduced the use of much more sophisticated and powerful cache technology and sophisticated instruction pipelining.

Pentium: With the Pentium, Intel introduced the use of superscalar techniques, which allow multiple instructions to execute in parallel.

Pentium Pro: The Pentium Pro continued the move into superscalar organization begun with the Pentium, with aggressive use of register renaming, branch prediction, data flow analysis, and speculative execution.

Pentium II: The Pentium II incorporated Intel MMX technology, which is designed specifically to process video, audio, and graphics data efficiently.

Pentium III: The Pentium III incorporates additional floating-point instructions.

Pentium 4: The Pentium 4 includes additional floating-point and other enhancements for multimedia.

Core: This is the first Intel x86 microprocessor with a dual core, referring to the implementation of two cores on a single chip.

Core 2: The Core 2 extends the Core architecture to 64 bits.

Basic Computer Concepts
UNIT -2

Computer Components:

At a top level, a computer consists of CPU (central processing unit), memory, and I/O components. 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.

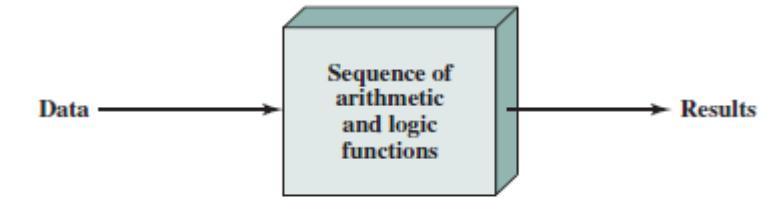
Contemporary computer designs are based on concept of Von Neumann architecture, and the three key concepts are:

- Data and instructions are stored in a single read–write memory.
- The contents of this memory are addressable by location.
- Execution occurs in a sequential fashion.

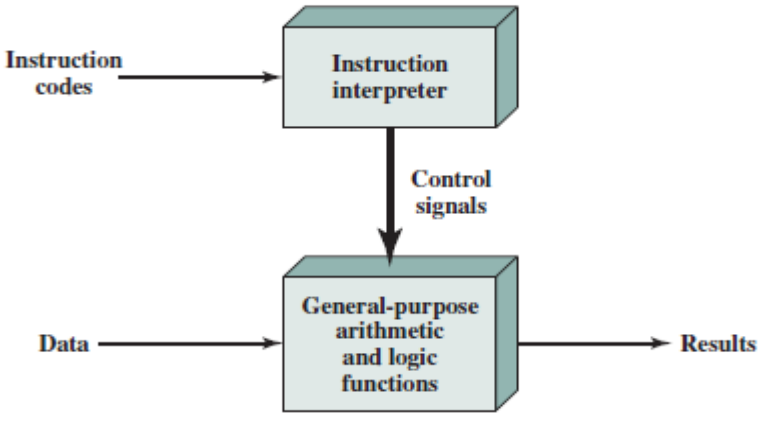
There is a small set of basic logic components that can be combined in various ways to store binary data and perform arithmetic and logical operations on that data. The process of connecting the various components in the desired configuration as a form of programming, the resulting "program" is in the form of hardware and is termed a hardwired program.

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.

The entire program is actually a sequence of steps. Instead of rewiring the hardware for each new program, all we need to do is provide a new sequence of codes. Each code is, in effect, an instruction, and part of the hardware interprets each instruction and generates control signals. To distinguish this new method of programming, a sequence of codes or instructions is called software.



(a) Programming in hardware



(b) Programming in software

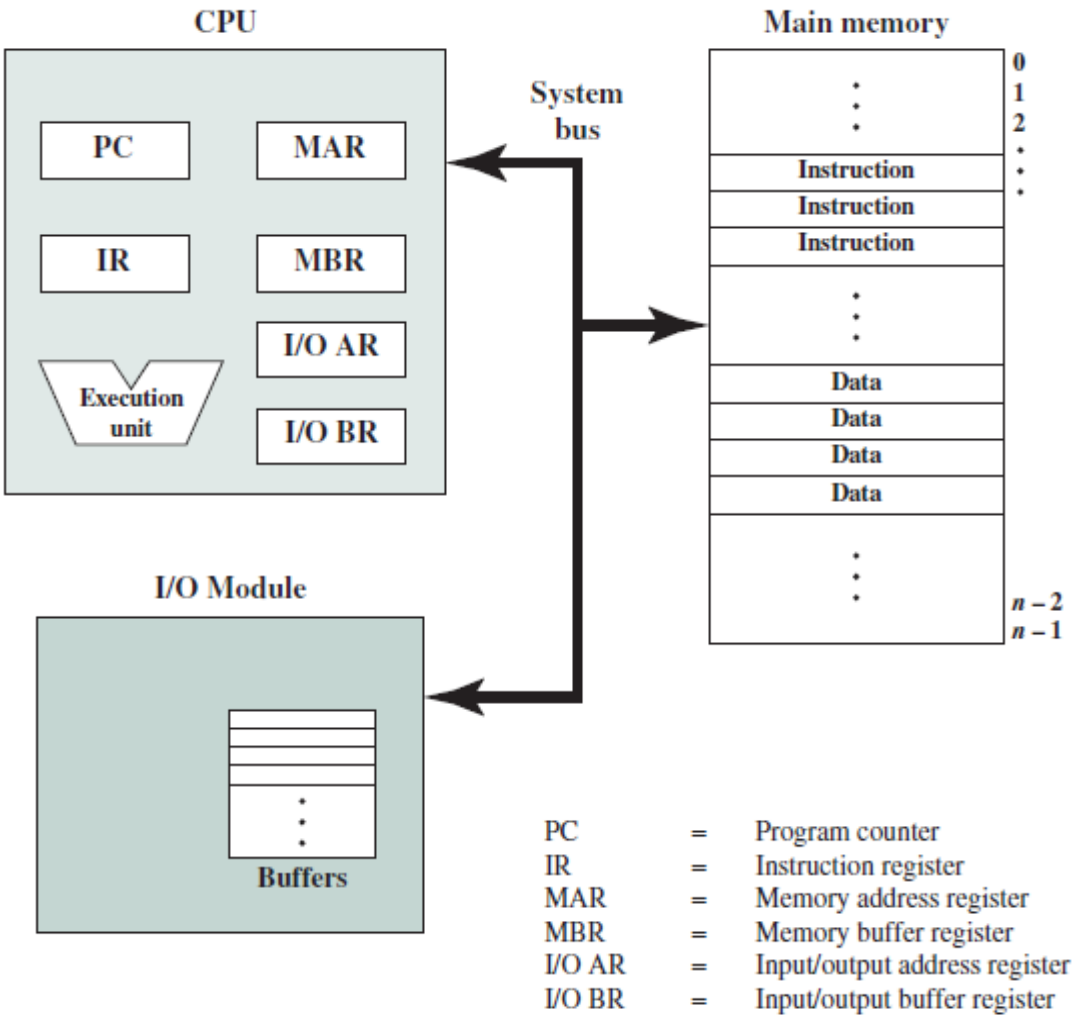
Hardware and Software Approaches

Basic Computer Concepts
UNIT -2

Data and instructions must be put into the system. For this we need some sort of input module. This module contains basic components for accepting data and instructions in some form and converting them into an internal form of signals usable by the system. 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.

There must be a place to temporarily store both instructions and data. That module is called memory or main memory. 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, and a memory buffer register (MBR), which contains the data to be written into memory or receives the data read from memory. Similarly, 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.

Computer Function:



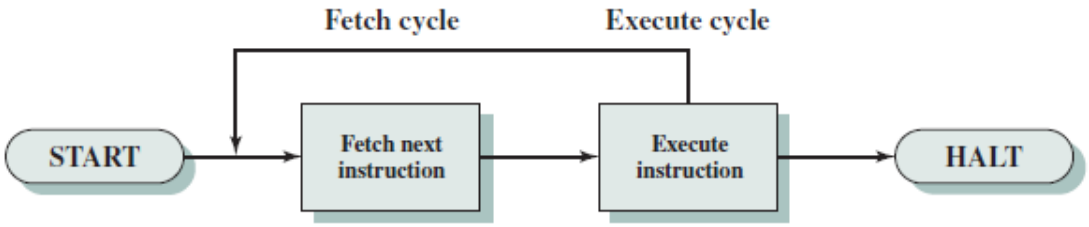
Computer Components: Top-Level View

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 instruction execution may involve several operations and depends on the nature of the instruction.

Instruction Fetch and Execute

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

In a typical processor, a register called 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.

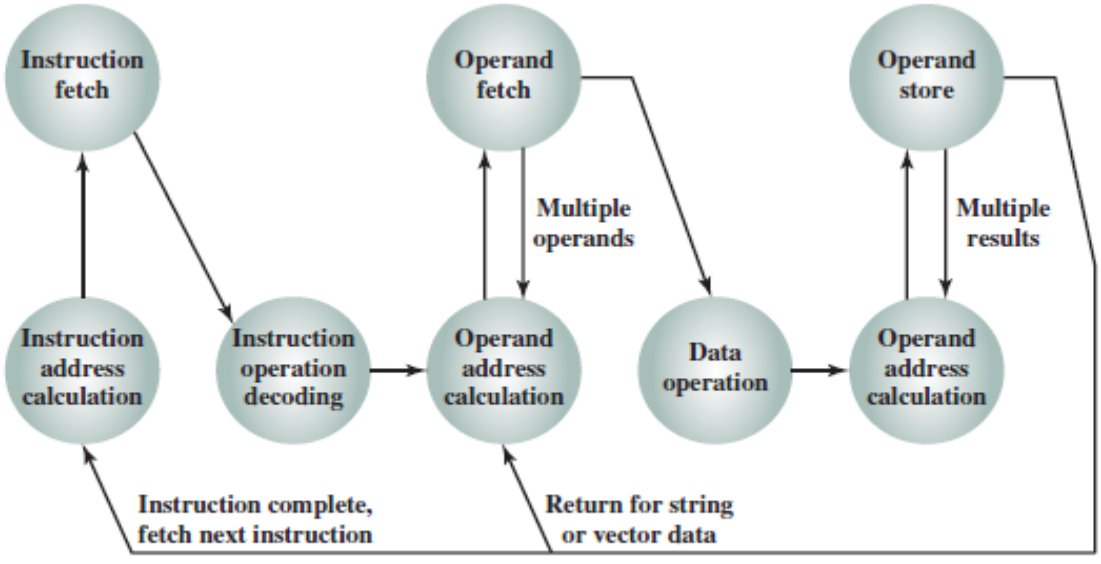


Basic Instruction Cycle

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. 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.

The processor contains a single data register, called an accumulator (AC). Both instructions and data are 16 bits long. Thus, it is convenient to organize memory using 16-bit words. The execution cycle for a particular instruction may involve more than one reference to memory. Instead of memory references, an instruction may specify an I/O operation.



Instruction Cycle State Diagram

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.
States in the upper part of above Figure involve an exchange between the processor and either memory or an I/O module. States in the lower part of the diagram involve only internal processor operations. The OAC (Operand Address Calculation) state appears twice, because an instruction may involve a read, a write, or both.
On some machines, a single instruction can specify an operation to be performed on a vector (one-dimensional array) of numbers or a string (one-dimensional array) of characters.

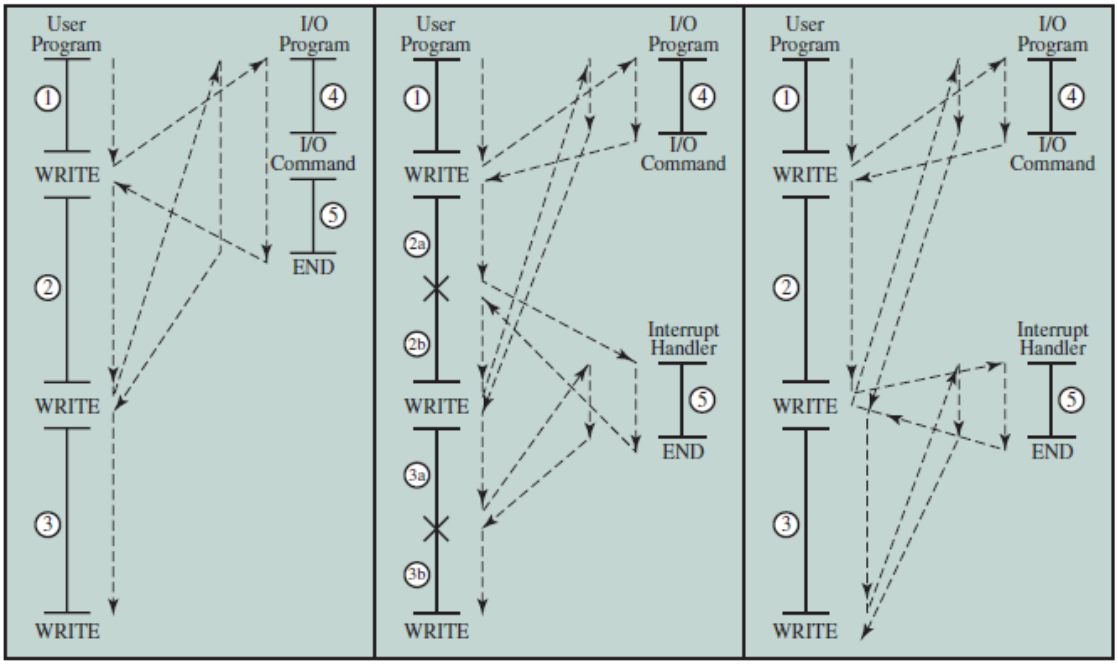
Interrupts :

Virtually all computers provide a mechanism by which other modules (I/O, memory) may interrupt the normal processing of the processor.
An interrupt is a signal emitted by a device attached to a computer or from a program within the computer.

Classes of Interrupts

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

Interrupts are provided primarily as a way to improve processing efficiency. After each write operation, the processor must pause and remain idle until the printer catches up. The length of this pause may be on the order of many hundreds or even thousands of instruction cycles that do not involve memory. Clearly, this is a very wasteful use of the processor.



Program Flow of Control without and with Interrupts

The I/O program consists of three sections: [above figure (a) Program Flow of Control with **No interrupts**]

- A sequence of instructions, labeled 4 in the figure, to prepare for the actual I/O operation. This may include copying the data to be output into a special buffer and preparing the parameters for a device command.
- The actual I/O command, without the use of interrupts, once this command is issued, the program must wait for the I/O device to perform the requested function (or periodically poll the device). The program might wait by simply repeatedly performing a test operation to determine if the I/O operation is done.
- A sequence of instructions, labeled 5 in the figure, to complete the operation. This may include setting a flag indicating the success or failure of the operation.

Because the I/O operation may take a relatively long time to complete, the I/O program is hung up waiting for the operation to complete; hence, the user program is stopped at the point of the WRITE call for some considerable period of time.

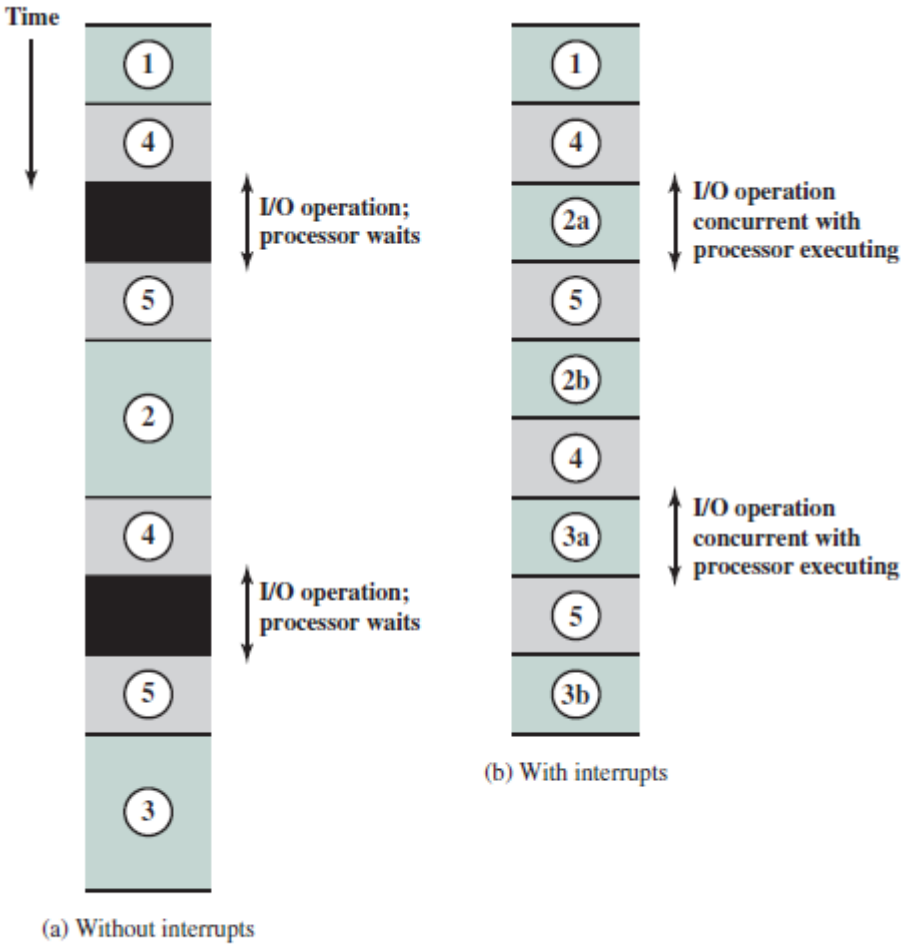
Interrupts And The Instruction Cycle:

With interrupts, the processor can be engaged in executing other instructions while an I/O operation is in progress.

The I/O program that is invoked in this case consists only of the preparation code and the actual I/O command. After these few instructions have been executed, control returns to the user program. Meanwhile, the external device 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.

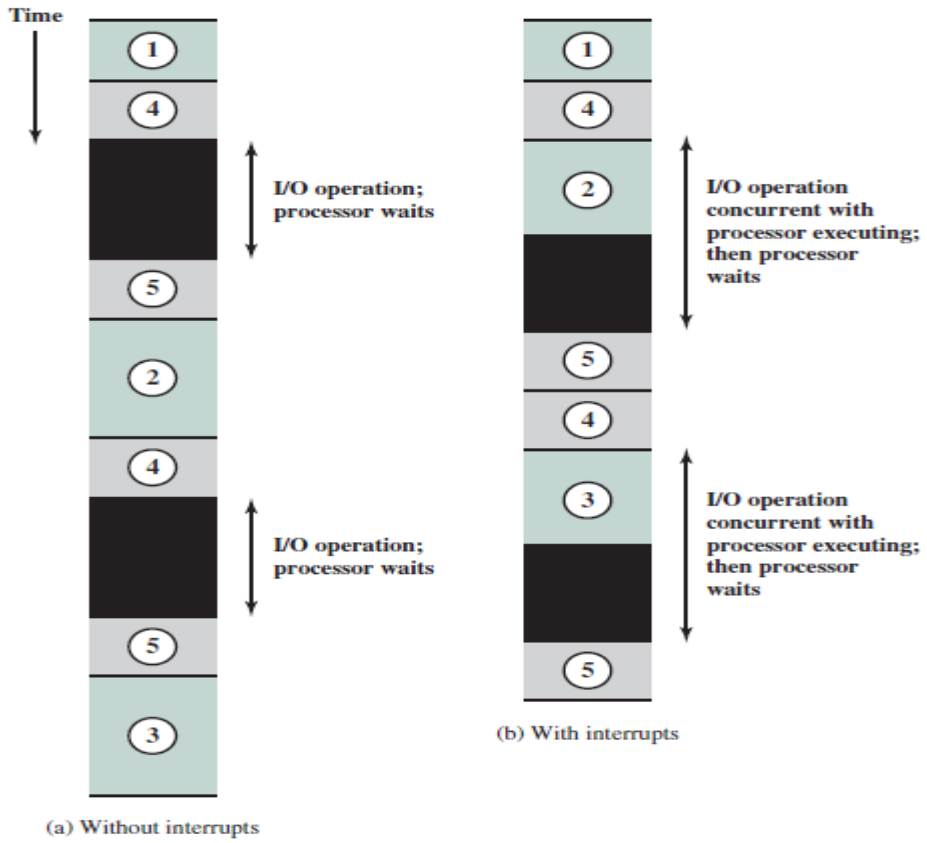
When the external device becomes ready to be serviced the I/O module for that external device sends an interrupt request signal to the processor. The processor responds by

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. [figure (b) and (c)]



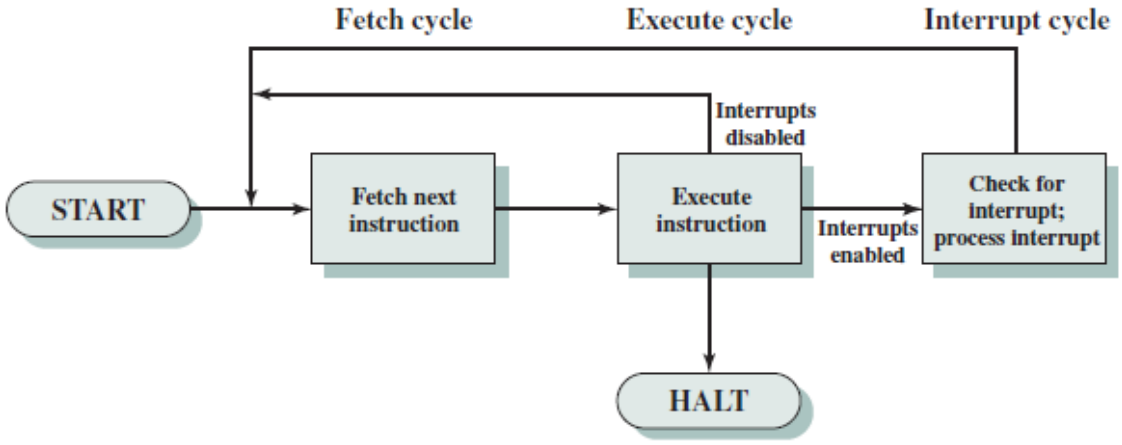
Program Timing: Short I/O Wait

From the point of view of the user program, an interrupt is an interruption of the normal sequence of execution. When the interrupt processing is completed, execution resumes.



Program Timing: Long I/O Wait

From the point of view of the user program, an interrupt is an interruption of the normal sequence of execution. When the interrupt processing is completed, execution resumes.



Instruction Cycle with Interrupts

To accommodate interrupts, an interrupt cycle is added to the instruction cycle. In the interrupt cycle, the processor checks to see if any interrupts have occurred, indicated by the presence of an interrupt signal. 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.

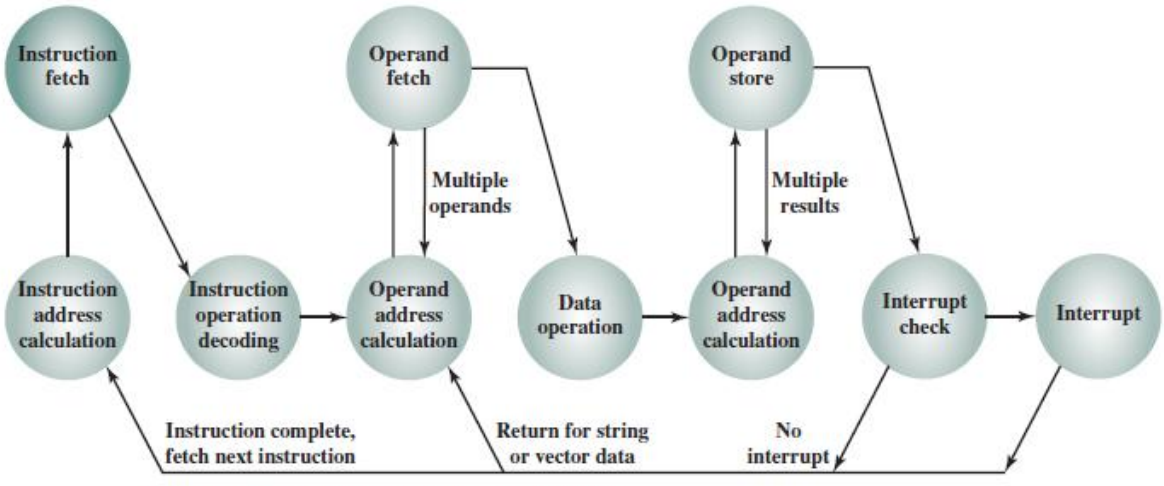
The processor now proceeds to the fetch cycle and fetches the first instruction in the interrupt handler program, which will service the interrupt. The interrupt handler program is generally part of the operating system. Typically, this program determines the nature of the interrupt and performs whatever actions are needed.

The handler determines which I/O module generated the interrupt and may branch to a program that will write more data out to that I/O module. When the interrupt handler routine is completed, the processor can resume execution of the user program at the point of interruption.

Suppose, that multiple interrupts can occur. For example, a program may be receiving data from a communications line and printing results. The printer will generate an interrupt every time it completes a print operation. The communication line controller will generate an interrupt every time a unit of data arrives. The unit could either be a single character or a block, depending on the nature of the communications discipline. In any case, it is possible for a communications interrupt to occur while a printer interrupt is being processed.

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. If an interrupt occurs during this time, it generally remains pending and will be checked by the processor after the processor has enabled interrupts. Thus, when a user program is executing and an interrupt occurs then interrupts are disabled immediately.

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. When the communications ISR is complete (t = 25), the previous processor state is restored, which is the execution of the printer ISR.



Instruction Cycle State Diagram, with Interrupts

I/O Function :

An I/O module (e.g., a disk controller) can exchange data directly with the processor. Just as the processor can initiate a read or write with memory, designating the address of a specific location, the processor can also read data from or write data to an I/O module. In this latter case, the processor identifies a specific device that is controlled by a particular I/O module.

In some cases, it is desirable to allow I/O exchanges to occur directly with memory. In such a case, 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. During such a transfer, 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).

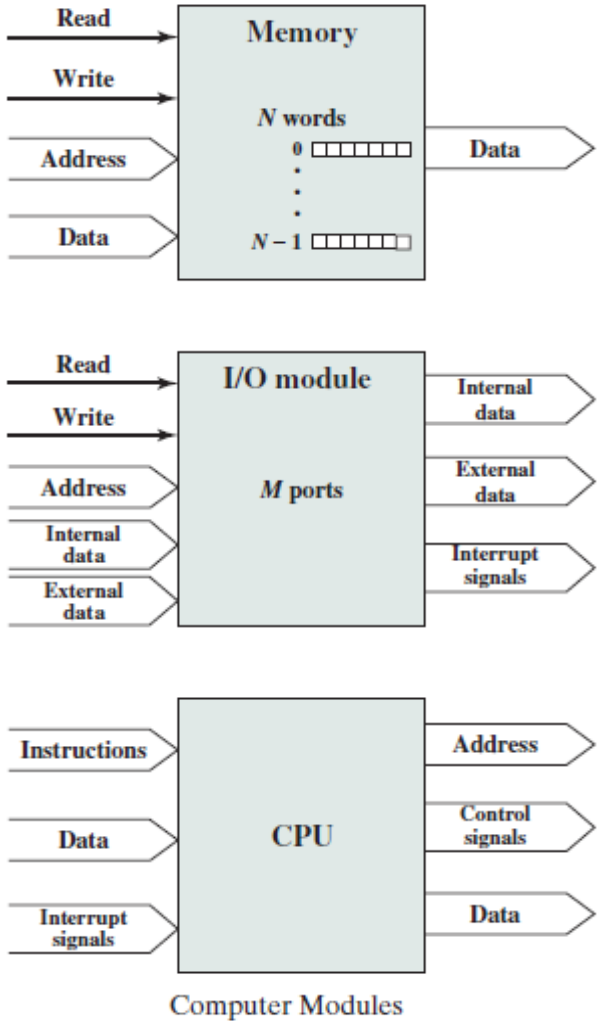
Interconnection Structures:

A computer consists of a set of components or modules of three basic types (processor, memory, I/O) that communicate with each other. The collection of paths connecting the various modules is called the interconnection structure. The design of this structure will depend on the exchanges that must be made among modules.

Memory: Typically, a memory module will consist of N words of equal length. Each word is assigned a unique numerical address (0, 1, c, N-1). A word of data can be read from or written into the memory. The nature of the operation is indicated by read and write control signals. The location for the operation is specified by an address.

I/O module: From an internal (to the computer system) point of view, I/O is functionally similar to memory. There are two operations; read and write. Further, an I/O module may control more than one external device. There are external data paths for the input and output of data with an external device. Finally, an I/O module may be able to send interrupt signals to the processor.

Processor: The processor reads in instructions and data, writes out data after processing, and uses control signals to control the overall operation of the system. It also receives interrupt signals.



Computer Modules

The interconnection structure must support the following types of transfers : Memory to Processor ,Processor to Memory, I/O to processor, Processor to I/O,I/O to or from memory

Most common interconnection structures are :

A. Bus interconnection
B. Point to Point interconnection

Bus Interconnection :

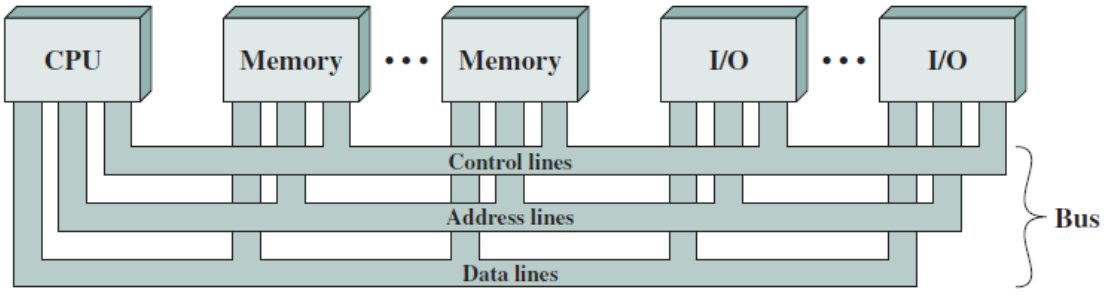
A bus is a communication pathway connecting two or more devices. A bus consists of multiple communication pathways, or lines. Each line is capable of transmitting signals representing binary 1 and binary 0. Several lines of a bus can be used to transmit binary digits simultaneously (in parallel).

A bus that connects major computer components (processor, memory, I/O) is called a **system bus**. A system bus consists of from about fifty to hundreds of separate lines. Each line is assigned a particular meaning or function. On any bus the lines can be classified into three functional groups : data, address, and control lines. In addition, there may be power distribution lines that supply power to the attached modules.

The **data lines** provide a path for moving data among system modules. These lines are called the **data bus**.

The **address lines** are used to designate the source or destination of the data on the data bus. The width of the **address bus** determines the maximum possible memory capacity of the system. The address lines are also used to address I/O ports.

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



Bus Interconnection Scheme

Typical control lines include:

Memory write, Memory read, I/O write, I/O read, Transfer ACK, Bus request, Bus grant, Interrupt request, Interrupt ACK, Clock, Reset.

The operation of the bus is as follows. If one module wishes to send data to another, it must do two things: (1) obtain the use of the bus, and (2) transfer data via the bus.

If one module wishes to request data from another module, it must (1) obtain the use of the bus, and (2) transfer a request to the other module over the appropriate control and address lines. It must then wait for that second module to send the data.

Point-to-Point Interconnect :

Compared to the shared bus, the point-to point interconnect has lower latency, higher data rate, and better scalability.

Example of the point-to-point interconnect approach is Intel's **QuickPath Interconnect (QPI)**.

Characteristics of QPI and other point-to-point interconnect schemes :

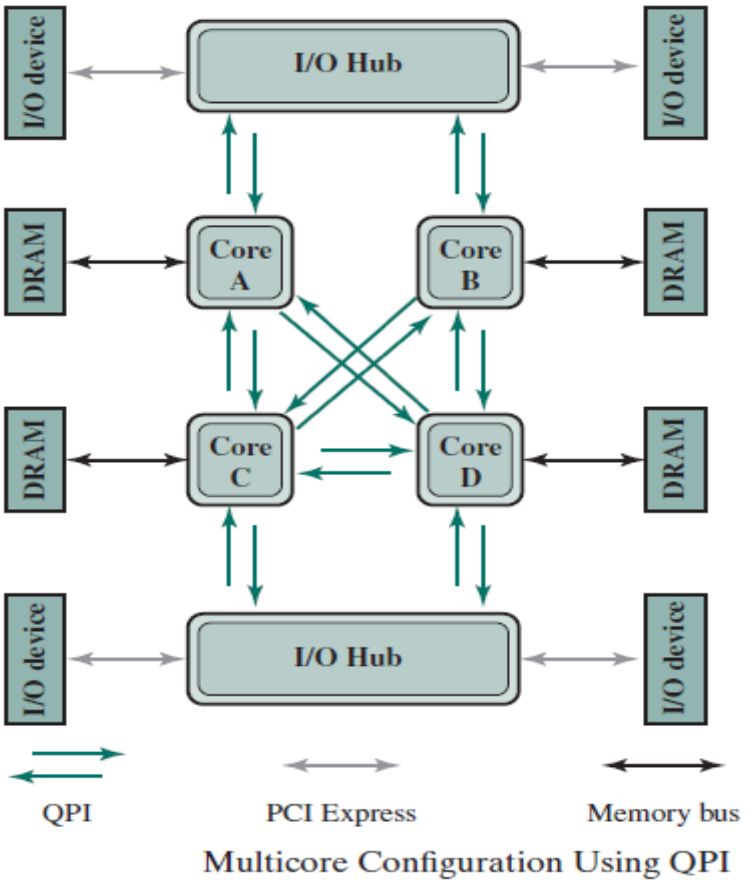
Multiple direct connections: Multiple components within the system enjoy direct pairwise connections to other components.

Layered protocol architecture: The 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 not sent as a raw bit stream. Rather, data are sent as a sequence of packets, each of which includes control headers and error control codes.

The QPI links form a switching fabric that enables data to move throughout the network. Direct QPI connections can be established between each pair of core processors.

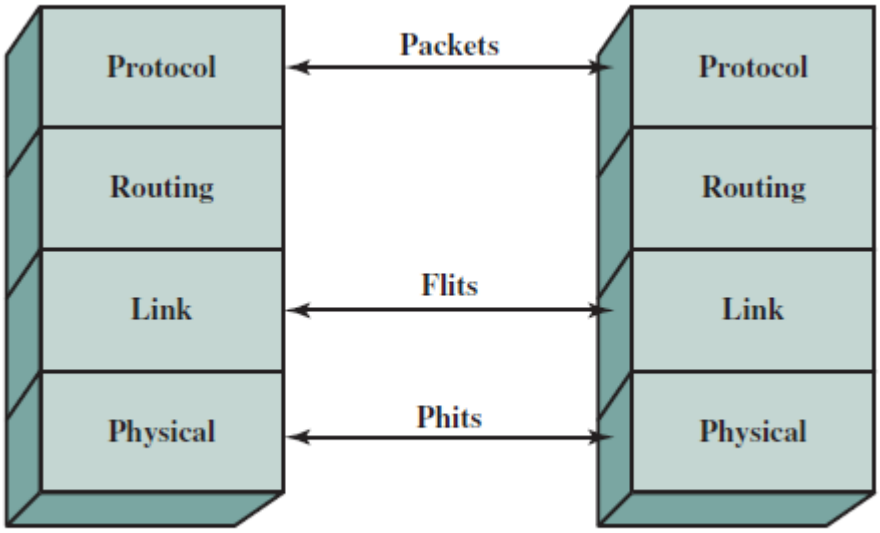
In addition, QPI is used to connect to an I/O module, called an I/O hub (IOH). The IOH acts as a switch directing traffic to and from I/O devices.



Multicore Configuration Using QPI

The link from the IOH to the I/O device controller uses an interconnect technology called PCI Express (PCIe). The IOH translates between the QPI protocols and formats and the PCIe protocols and formats.

QPI is defined as a four-layer protocol architecture.



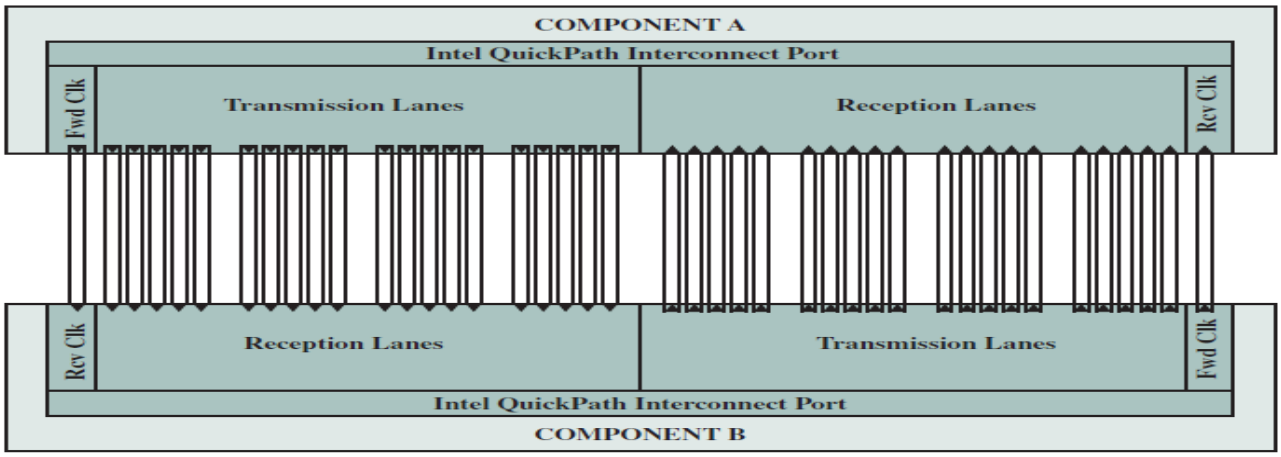
QPI Layers

Physical: Consists of the actual wires carrying the signals, as well as circuitry and logic to support ancillary features required in the transmission and receipt of the 1s and 0s. The unit of transfer at the Physical layer is 20 bits, which is called a Phit (physical unit).
Link: Responsible for reliable transmission and flow control. The Link layer's unit of transfer is an 80-bit Flit (flow control unit).
Routing: Provides the framework for directing packets through the fabric.
Protocol: The high-level set of rules for exchanging packets of data between devices. A packet is comprised of an integral number of Flits.

QPI Physical Layer

The QPI port consists of 84 individual links grouped as follows. Each data path consists of a pair of wires that transmits data one bit at a time; the pair is referred to as a lane. There are 20 data lanes in each direction (transmit and receive), plus a clock lane in each direction. Thus, QPI is capable of transmitting 20 bits in parallel in each direction. The 20-bit unit is referred to as a phit. QPI links involve dedicated bidirectional pairs, the total capacity is 32 GB/s.
The lanes in each direction are grouped into four quadrants of 5 lanes each. The form of transmission on each lane is known as **differential signaling**, or **balanced transmission.**
The technique used by QPI is known as low-voltage differential signaling (LVDS).

The physical layer manages the translation between 80-bit flits and 20-bit phits using a technique known as **multilane distribution**. This approach enables QPI to achieve very high



Physical Interface of the Intel QPI Interconnect

data rates by implementing the physical link between two ports as multiple parallel channels.

QPI Link Layer

The QPI link layer performs two key functions: flow control and error control. These functions are performed as part of the QPI link layer protocol, and operate on the level of the flit (flow control unit).

The **flow control function** is 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.

The **error control function** at the link layer detects and recovers from such bit errors, and so isolates higher layers from experiencing bit errors.

QPI Routing Layer

The routing layer is used to determine the course that a packet will traverse across the available system interconnects. Routing tables are defined by firmware and describe the possible paths that a packet can follow.

QPI Protocol Layer

In this layer, the 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.

Universal Serial Bus (USB)

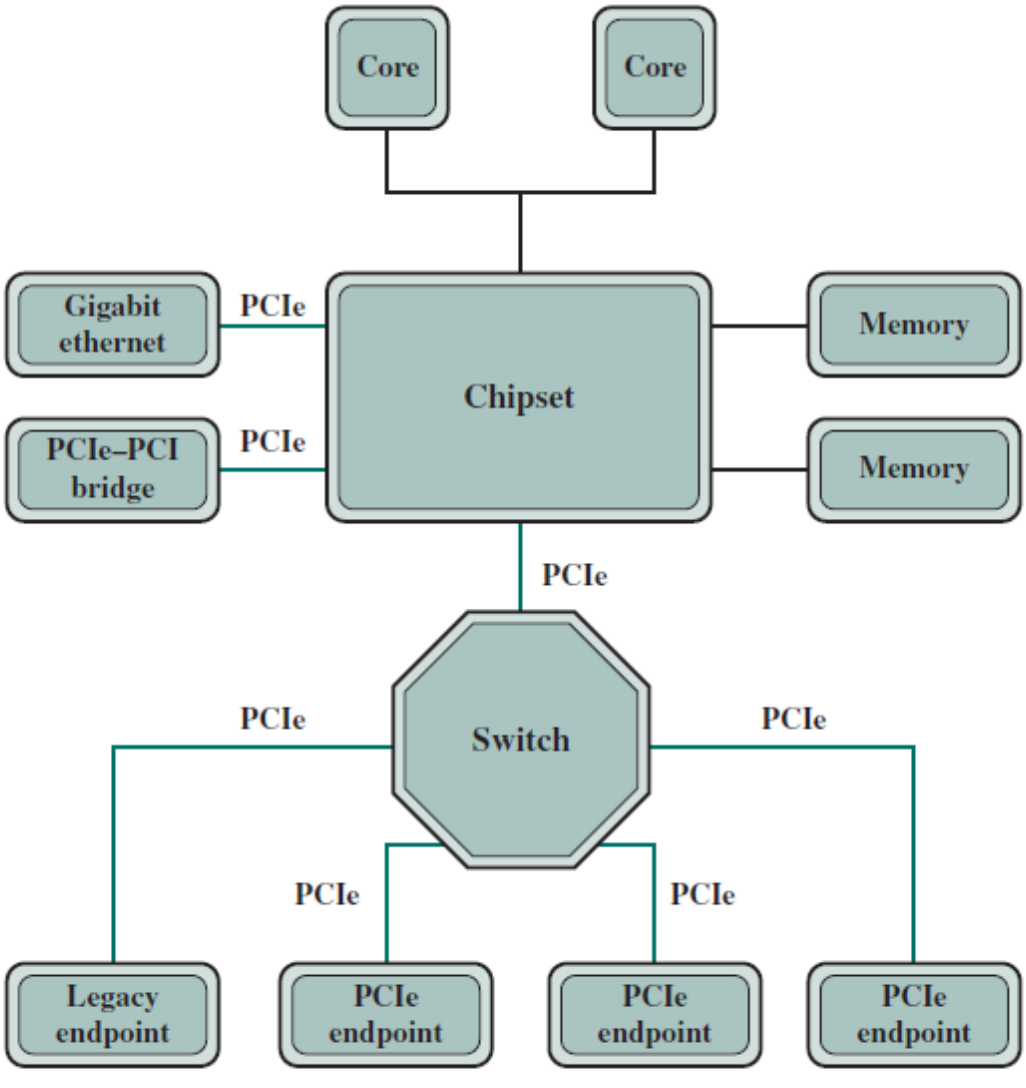
USB is widely used for peripheral connections.
It is the default interface for slower speed devices, such as keyboard and pointing devices, but is also commonly used for high-speed I/O, including printers, disk drives, and network adapters.
USB has gone through multiple generations. The first version, USB 1.0, defined a Low Speed data rate of 1.5 Mbps and a Full Speed rate of 12 Mbps. USB 2.0 provides a data rate of 480 Mbps. USB 3.0 includes a new, higher speed bus called SuperSpeed in parallel with the USB 2.0 bus. The signaling speed of Super-Speed is 5 Gbps, but due to signaling overhead, the usable data rate is up to 4 Gbps. The most recent specification is USB 3.1, which includes a faster transfer mode called SuperSpeed+. This transfer mode achieves a signaling rate of 10 Gbps and a theoretical usable data rate of 9.7 Gbps.
A USB system is controlled by a root host controller, which attaches to devices to create a local network with a hierarchical tree topology.

PCI Express :

The peripheral component interconnect (PCI) is a popular high-bandwidth, processor-independent bus that can function as a peripheral bus.
The result is that PCI has been widely adopted and is finding increasing use in personal computer, workstation, and server systems.

PCI Express (PCIe) is a point-to-point interconnect scheme intended to replace bus-based schemes such as PCI. A key requirement for PCIe is high capacity to support the needs of higher data rate I/O devices, such as Gigabit Ethernet.

PCI Physical and Logical Architecture



Typical Configuration Using PCIe

The figure shows the configration that supports the use of PCIe. A root complex device, also referred to as a chipset or a host bridge, connects the processor and memory subsystem to the PCI Express switch fabric comprising one or more PCIe and PCIe switch devices. PCIe links from the chipset may attach to the following kinds of devices that implement PCIe:

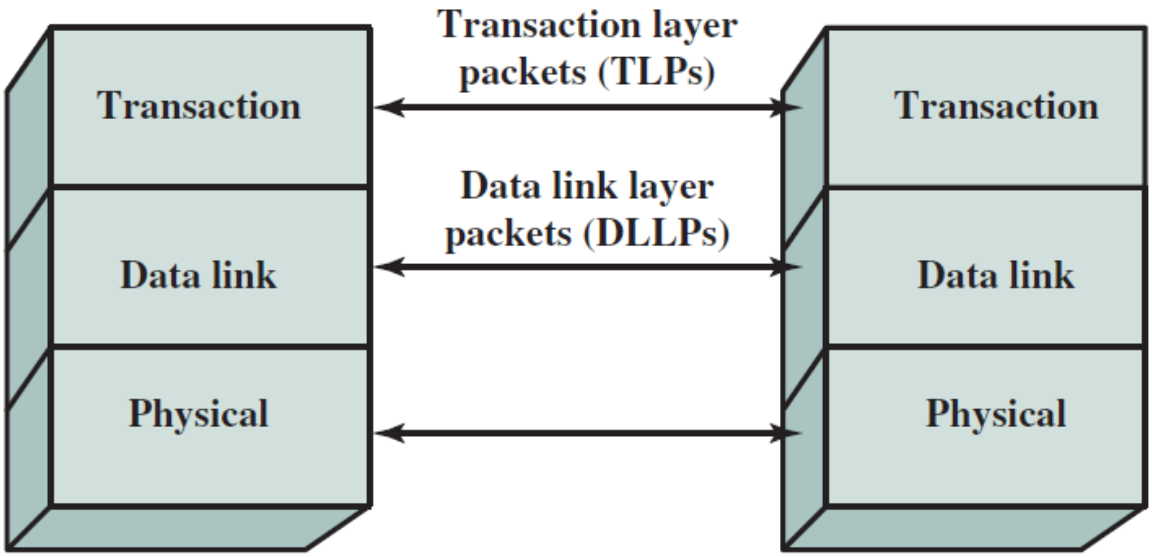
Switch: The switch manages multiple PCIe streams.
PCIe endpoint: An I/O device or controller that implements PCIe, such as a Gigabit ethernet switch, a graphics or video controller, disk interface, or a communications controller.
Legacy endpoint: Legacy endpoint category is intended for existing designs that have been migrated to PCI Express, and it allows legacy behaviors such as use of I/O space and locked transactions.
PCIe/PCI bridge: Allows older PCI devices to be connected to PCIe-based systems.

PCIe Protocol Layers :

The PCIe protocol architecture encompasses the following layers :
Physical: Consists of the actual wires carrying the signals, as well as circuitry and logic to support ancillary features required in the transmission and receipt of the 1s and 0s.
Data link: Is responsible for reliable transmission and flow control. Data packets generated and consumed by the DLL are called Data Link Layer Packets (DLLPs).
Transaction: Generates and consumes data packets used to implement load/store data transfer mechanisms and manages the flow control of those packets. Data packets generated and consumed by the TL are called Transaction Layer Packets (TLPs).

PCIe Physical Layer

PCIe is a point-to-point architecture. Each PCIe port consists of a number of bidirectional lanes. PCIe uses a multilane distribution technique. Figure Shows PCIe port consisting of four lanes.



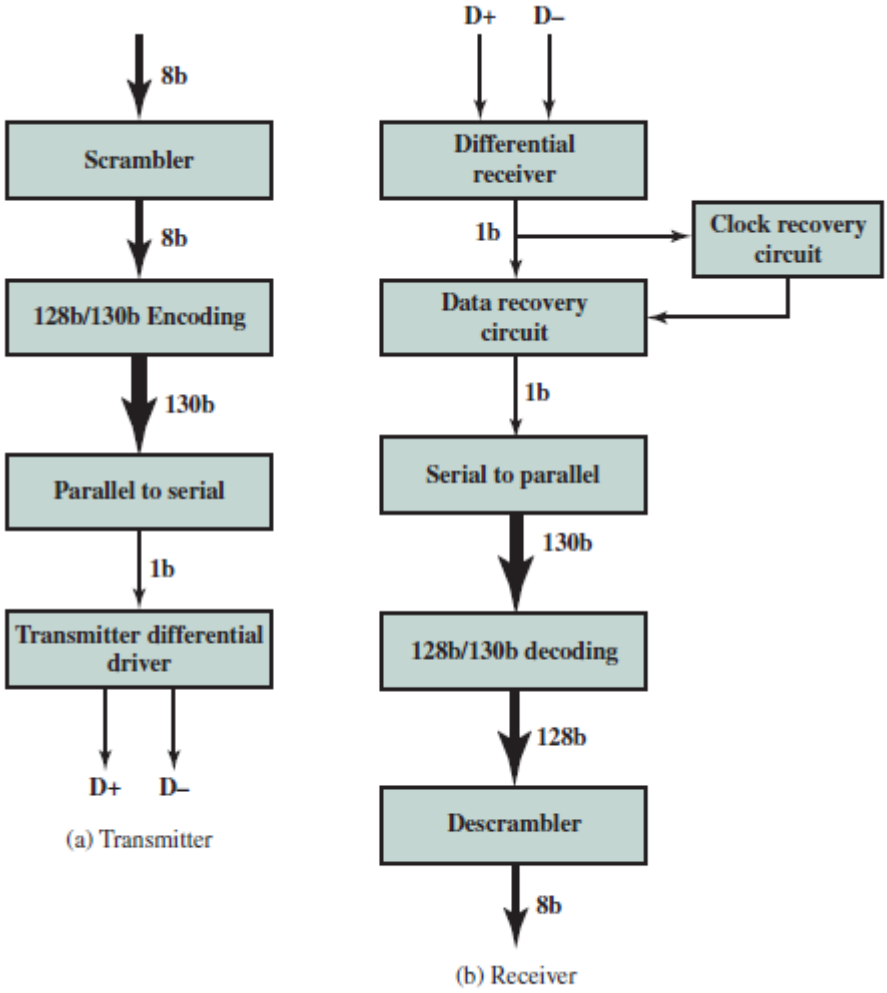
PCIe Protocol Layers

PCIe does not use its clock line to synchronize the bit stream. The clock line is not used to determine the start and end point of each incoming bit; it is used for other signaling purposes only. It is necessary for the receiver to be synchronized with the transmitter.
If there is any drift between the clocks used for bit transmission and reception of the transmitter and receiver, errors may occur. To compensate for the possibility of drift (Pile

up), PCIe relies on the receiver synchronizing with the transmitter based on the transmitted signal.

A common approach to overcome the problem of a long string of bits of one value is scrambling. Scrambling, which does not increase the number of bits to be transmitted, is a mapping technique that tends to make the data appear more random.

The scrambling tends to spread out the number of transitions so that they appear at the receiver more uniformly spaced, which is good for synchronization.



PCIe Transmit and Receive Block Diagrams

The 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, in which a request packet is sent out by a source PCIe device, which then waits for a response, called a completion packet. The completion following a request is initiated by the completer only when it has the data and/or status ready for delivery. Each packet has a unique identifier that enables completion packets to be directed to the correct originator.

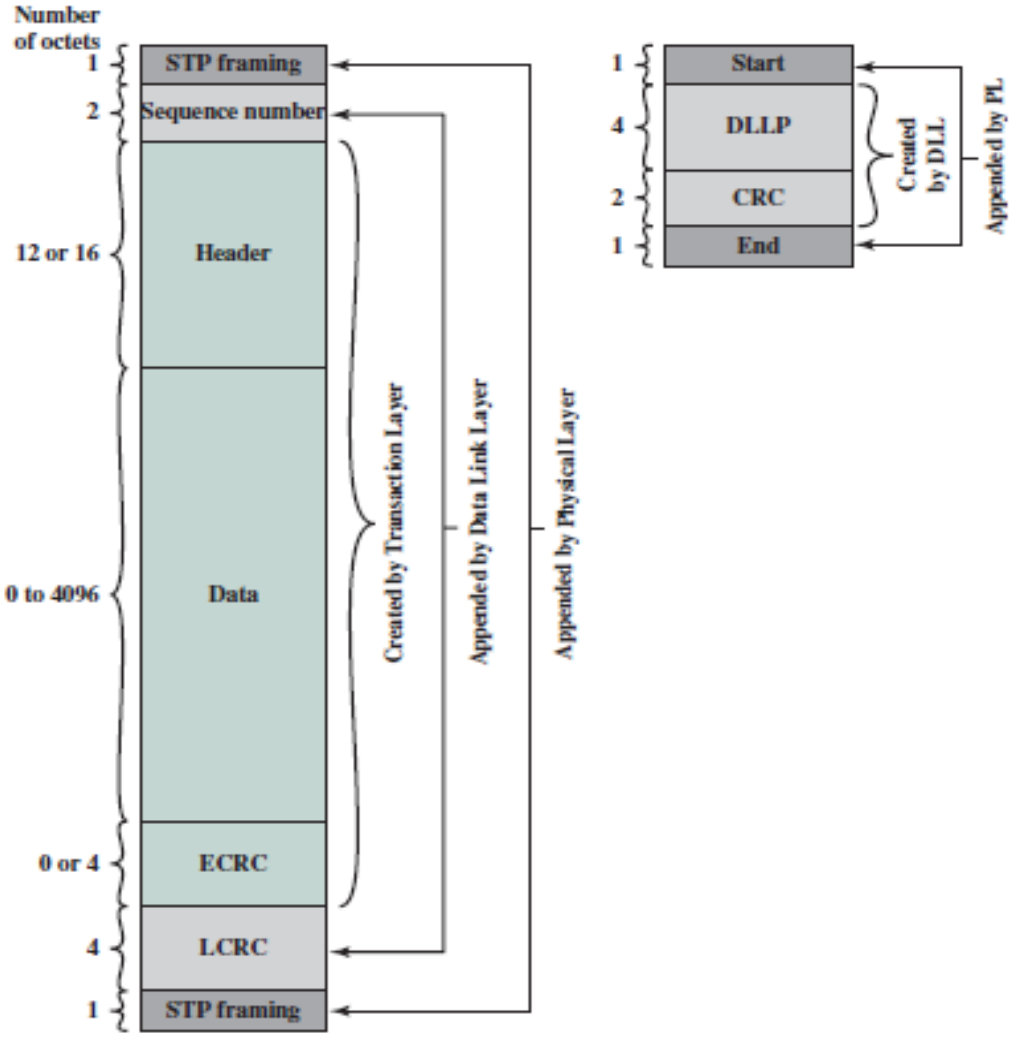
The TL packet format supports 32-bit memory addressing and extended 64-bit memory addressing. Packets also have attributes such as “no-snoop”, “relaxedordering” and “priority”.

Address spaces and transaction types :

The TL supports four address spaces ,

Memory: The memory space includes system main memory. It also includes PCIe I/O devices.
I/O: This address space is used for legacy PCI devices, with reserved memory address ranges used to address legacy I/O devices.
Configuration: This address space enables the TL to read/write configuration registers associated with I/O devices.
Message: This address space is for control signals related to interrupts, error handling, and power management.
For memory, I/O, and configuration address spaces, there are read and write transactions.

TLP packet assembly :



PCIe Protocol Data Unit Format

A TLP originates in the transaction layer of the sending device and terminates at the transaction layer of the receiving device.
Upper layer software sends to the TL the information needed for the TL to create the core of the TLP, which consists of the following fields:
Header: The header describes the type of packet and includes information needed by the receiver to process the packet.

Data: A data field of up to 4096 bytes may be included in the TLP.
ECRC (End to End Cyclic Redundancy Check): An optional end-to-end CRC field enables the destination TL layer to check for errors in the header and data portions of the TLP.

PCIe Data Link Layer :

The purpose of the PCIe data link layer is to ensure reliable delivery of packets across the PCIe link. The DLL participates in the formation of TLPs and also transmits DLLPs.

Data Link Layer Packets: Data link layer packets originate at the data link layer of a transmitting device and terminate at the DLL of the device on the other end of the link. There are three important groups of DLLPs used in managing a link: flow control packets, power management packets, and TLP ACK and NAK packets.

Transaction Layer Packet Processing: The DLL adds two fields to the core of the TLP created by the TL: a 16-bit sequence number and a 32-bit link-layer CRC (LCRC). Whereas the core fields created at the TL are only used at the destination TL, the two fields added by the DLL are processed at each intermediate node on the way from source to destination.

Some Important Notes :

What is PCIe Physical Layer?

Ans. : The Physical Layer is the lowest hierarchical layer for PCIe as shown in the figure below. TLP and DLLP packets are sent from the Data Link Layer to the Physical Layer for transmission over the link. The Physical Layer is divided into 2 languages: the logical one and the electrical one.

What are PCIe protocols?

Ans. : PCIe is a motherboard expansion bus standard introduced in 2003 to enable high speed serial communication between the CPU and its peripheral components. Today, it has become the primary motherboard expansion bus standard and a popular communication method for many other on board applications. Legacy PCI is a parallel data transfer protocol. But PCIe is a serial data transfer protocol.