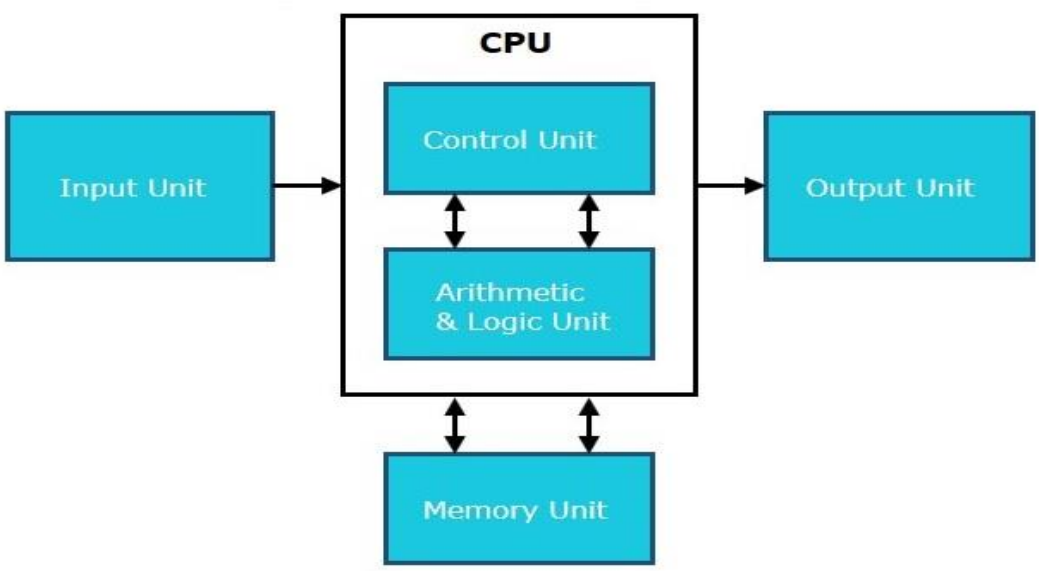
Computer Architecture
Subject code: BTE301-18
Module 1
Functional block of a computer

CPU

CPU stands for Central Processing Unit; it is also known as "the brain of the computer". A CPU is a primary component of a computer that performs most of the processing and controls the operation of all components running inside a computer.



Control Unit

As its name implies, a control unit acts as the "brain" of the CPU. A major role of a control unit is to manage and execute instructions to perform the tasks specified by a computer program. v. A CPU executes instructions by fetching them from memory, decodes them, and then executes them. So, it plays a vital role in **fetch-decode-execute** instructions.

Functions of Control Unit

- Instruction Fetch − A CU fetches instructions from RAM (Random Access Memory).
- Instruction Decoding − It decodes the fetched instructions to operate.
- Instruction Execution − A CU sends control signals to perform operations like ALU for arithmetic and logical operations.
- Control Flow Management − It controls flow by updating the programme counter.
- Exception Handling − A control unit effectively manages exceptions and interruptions like hardware failures, system calls, and external events, by appropriately diverting the control flow of the CPU to the planned procedure for managing such exceptions.
- Synchronization − A CU plays a crucial role in facilitating the coordination of instruction execution across several cores.

Arithmetic Logic Unit (ALU)

The Arithmetic Logic Unit (ALU) is a component that has been extensively optimised and engineered to do multiple tasks concurrently. It is commonly built to execute operations

speedily. It works in conjunction with other CPU components, such as registers, memory, and control units, to execute complex instructions.
Functions of an ALU

- Arithmetic Operations − The ALU can perform basic arithmetic operations.
- Logic Operations − The ALU can also perform logical operations like AND, OR, NOT, XOR, and bit-shifting operations.

Memory Unit
A memory is a hardware component which is used to store and access the data whenever required. Majorly; computer memory is categorised into two parts Primary Memory (RAM) and Secondary Memory (Hard Disk). RAM is used for short-term, fast data access and is essential for active program execution. On the other hand, storage or secondary memory provides permanent data storage.
Hence, memory and storage units both are critical components of a computer system.

Functions of memory

Primary Memory

- RAM is also known as primary or temporary memory; it is a type of volatile memory used for temporarily storing data.
- The contents inside the RAM are erased when the computer's power gets off or restarted.
- RAM is actively used for program or instruction execution.
- Once we start the computer; system necessary files, programs and operating system files are loaded into the RAM for the smooth running of the computer.
- The more RAM a computer has, the better it can handle multitasking and the faster it can run applications since data can be accessed more quickly.

Storage (Hard Drives, SSDs, Flash Drives, etc.)

- Storage devices are used to store the data permanently, even when the computer is powered off.
- They are non-volatile; the data remains intact even when the power is turned off or the system restarts.
- The most popular and commonly used storage devices are Hard disks (HDs), Solid-State Drives (SSDs), USB flash drives, and optical disks (e.g., DVDs), pen drives.
- The data storage capacity of these devices in gigabytes (GB) to terabytes (TB) and more, depending on the type and size of the storage device.

Functions of the CPU
The key functions of a CPU are as follows −

- The CPU performs arithmetic and logic operations.
- It directs the operation of the processor.
- It directs Input and output units that how to respond to the instructions that have been communicated to the processor.
- A CPU contains registers which are considered small storage locations within the CPU to hold data temporarily during execution of a program.
- A CPU executes instructions by fetching them from memory, decodes them, and then executes them.

Memory

Computer memory is just like the human brain. It is used to store data/information and instructions. It is a data storage unit or a data storage device where data is to be processed and instructions required for processing are stored. It can store both the input and output can be stored here.

Characteristics of Computer Memory

- It is faster computer memory as compared to secondary memory.
- It is semiconductor memories.
- It is usually a volatile memory, and main memory of the computer.
- A computer system cannot run without primary memory.

Types of Computer Memory

In general, computer memory is of three types:

- Primary memory
- Secondary memory
- Cache memory

1. Primary Memory

It is also known as the main memory of the computer system. It is used to store data and programs or instructions during computer operations. It uses semiconductor technology and hence is commonly called semiconductor memory. Primary memory is of two types:

- **RAM (Random Access Memory):** It is a volatile memory. Volatile memory stores information based on the power supply. If the power supply fails/ interrupted/stopped, all the data and information on this memory will be lost. RAM is used for booting up or start the computer. It temporarily stores programs/data which has to be executed by the processor. RAM is of two types:
    - **S RAM (Static RAM):** S RAM uses transistors and the circuits of this memory are capable of retaining their state as long as the power is applied. This memory consists of the number of flip flops with each flip flop storing 1 bit. It has less access time and hence, it is faster.
    - **D RAM (Dynamic RAM):** D RAM uses capacitors and transistors and stores the data as a charge on the capacitors. They contain thousands of memory cells. It needs refreshing of charge on capacitor after a few milliseconds. This memory is slower than S RAM.

- **ROM (Read Only Memory):** It is a non-volatile memory. Non-volatile memory stores information even when there is a power supply failed/ interrupted/stopped. ROM is used to store information that is used to operate the system. As its name refers to read-only memory, we can only read the programs and data that is stored on it. It contains some electronic fuses that can be programmed for a piece of specific information. The information stored in the ROM in binary format. It is also known as permanent memory. ROM is of four types:
    - **MROM(Masked ROM):** Hard-wired devices with a pre-programmed collection of data or instructions were the first ROMs. Masked ROMs are a type of low-cost ROM that works in this way.
    - **PROM (Programmable Read Only Memory):** This read-only memory is modifiable once by the user. The user purchases a blank PROM and uses a PROM program to put the required contents into the PROM. Its content can’t be erased once written.

- **EPROM (Erasable Programmable Read Only Memory):** EPROM is an extension to PROM where you can erase the content of ROM by exposing it to Ultraviolet rays for nearly 40 minutes.
- **EEPROM (Electrically Erasable Programmable Read Only Memory):** Here the written contents can be erased electrically. You can delete and reprogramme EEPROM up to 10,000 times. Erasing and programming take very little time, i.e., nearly 4 -10 ms(milliseconds). Any area in an EEPROM can be wiped and programmed selectively.

2. Secondary Memory

It is also known as auxiliary memory and backup memory. It is a non-volatile memory and used to store a large amount of data or information. The data or information stored in secondary memory is permanent, and it is slower than primary memory. A CPU cannot access secondary memory directly. The data/information from the auxiliary memory is first transferred to the main memory, and then the CPU can access it.

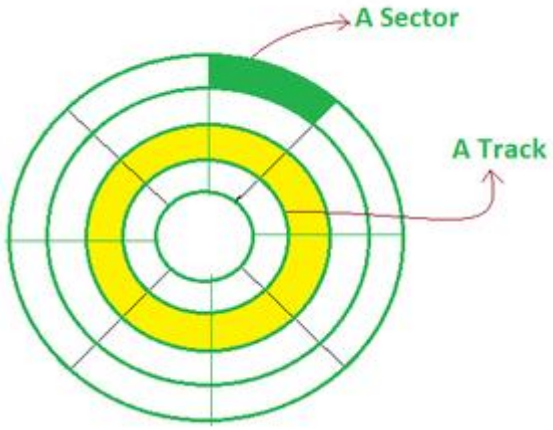
Characteristics of Secondary Memory

- It is a slow memory but reusable.
- It is a reliable and non-volatile memory.
- It is cheaper than primary memory.
- The storage capacity of secondary memory is large.
- A computer system can run without secondary memory.
- In secondary memory, data is stored permanently even when the power is off.

Types of Secondary Memory

1. Magnetic Tapes: Magnetic tape is a long, narrow strip of plastic film with a thin, magnetic coating on it that is used for magnetic recording. Bits are recorded on tape as magnetic patches called RECORDS that run along many tracks. Typically, 7 or 9 bits are recorded concurrently. Each track has one read/write head, which allows data to be recorded and read as a sequence of characters. It can be stopped, started moving forward or backward, or rewound.

2. Magnetic Disks: A magnetic disk is a circular metal or a plastic plate and these plates are coated with magnetic material. The disc is used on both sides. Bits are stored in magnetized surfaces in locations called tracks that run in concentric rings. Sectors are typically used to break tracks into pieces.



Hard discs are discs that are permanently attached and cannot be removed by a single user.

3. Optical Disks: It's a laser-based storage medium that can be written to and read. It is reasonably priced and has a long lifespan. The optical disc can be taken out of the computer by occasional users.

Types of Optical Disks

CD – ROM

- It's called compact disk. Only read from memory.
- Information is written to the disc by using a controlled laser beam to burn pits on the disc surface.
- It has a highly reflecting surface, which is usually aluminium.
- The diameter of the disc is 5.25 inches.
- 16000 tracks per inch is the track density.
- The capacity of a CD-ROM is 600 MB, with each sector storing 2048 bytes of data.
- The data transfer rate is about 4800KB/sec. & the new access time is around 80 milliseconds.

WORM-(WRITE ONCE READ MANY)

- A user can only write data once.
- The information is written on the disc using a laser beam.
- It is possible to read the written data as many times as desired.
- They keep lasting records of information but access time is high.
- It is possible to rewrite updated or new data to another part of the disc.
- Data that has already been written cannot be changed.
- Usual size – 5.25 inch or 3.5 inch diameter.
- The usual capacity of 5.25 inch disk is 650 MB,5.2GB etc.

DVDs

- The term "DVD" stands for "Digital Versatile/Video Disc," and there are two sorts of DVDs:
    - DVDR (writable)
    - DVDRW (Re-Writable)
- **DVD-ROMS (Digital Versatile Discs)**: These are read-only memory (ROM) discs that can be used in a variety of ways. When compared to CD-ROMs, they can store a lot more data. It has a thick polycarbonate plastic layer that serves as a foundation for the other layers. It's an optical memory that can read and write data.
- **DVD-R**: DVD-R is a writable optical disc that can be used just once. It's a DVD that can be recorded. It's a lot like WORM. DVD-ROMs have capacities ranging from 4.7 to 17 GB. The capacity of 3.5 inch disk is 1.3 GB.

3. Cache Memory

It is a type of high-speed semiconductor memory that can help the CPU run faster. Between the CPU and the main memory, it serves as a buffer. It is used to store the data and programs that the CPU uses the most frequently.

Advantages of Cache Memory

- It is faster than the main memory.
- When compared to the main memory, it takes less time to access it.
- It keeps the programs that can be run in a short amount of time.
- It stores data in temporary use.

Disadvantages of Cache Memory

- Because of the semiconductors used, it is very expensive.
- The size of the cache (amount of data it can store) is usually small.

Input output subsystems

Input/Output Subsystem
The I/O subsystem of a computer provides an efficient mode of communication between the central system and the outside environment. It handles all the input-output operations of the computer system.

Peripheral Devices
Input or output devices that are connected to computer are called peripheral devices. These devices are designed to read information into or out of the memory unit upon command from the CPU and are considered to be the part of computer system. These devices are also called peripherals.
For example: Keyboards, display units and printers are
common peripheral devices.

There are three types of peripherals:

1. Input peripherals : Allows user input, from the outside world to the computer.
   Example: Keyboard, Mouse etc.

2. Output peripherals: Allows information output, from the computer to the outside world.
   Example: Printer, Monitor etc

3. Input-Output peripherals: Allows both input(from outised world to computer) as well as, output(from computer to the outside world). Example: Touch screen etc.

Interfaces
Interface is a shared boundary between two separate components of the computer system which can be used to attach two or more components to the system for communication purposes.

There are two types of interface:
1. CPU Inteface
2. I/O Interface

Input-Output Interface
Peripherals connected to a computer need special communication links for interfacing with CPU. In computer system, there are special hardware components between the
CPU and peripherals to control or manage the input-output transfers. These components are called input-output interface units because they provide communication links
between processor bus and peripherals. They provide a method for transferring information between internal system and input-output devices.

Modes of I/O Data Transfer

Data transfer between the central unit and I/O devices can be handled in generally three types of modes which are given below:

1. Programmed I/O
2. Interrupt Initiated I/O
3. Direct Memory Access

Programmed I/O

Programmed I/O instructions are the result of I/O instructions written in computer program. Each data item transfer is initiated by the instruction in the program.Usually the program controls data transfer to and from CPU and peripheral. Transferring data under programmed I/O requires constant monitoring of the peripherals by the CPU.

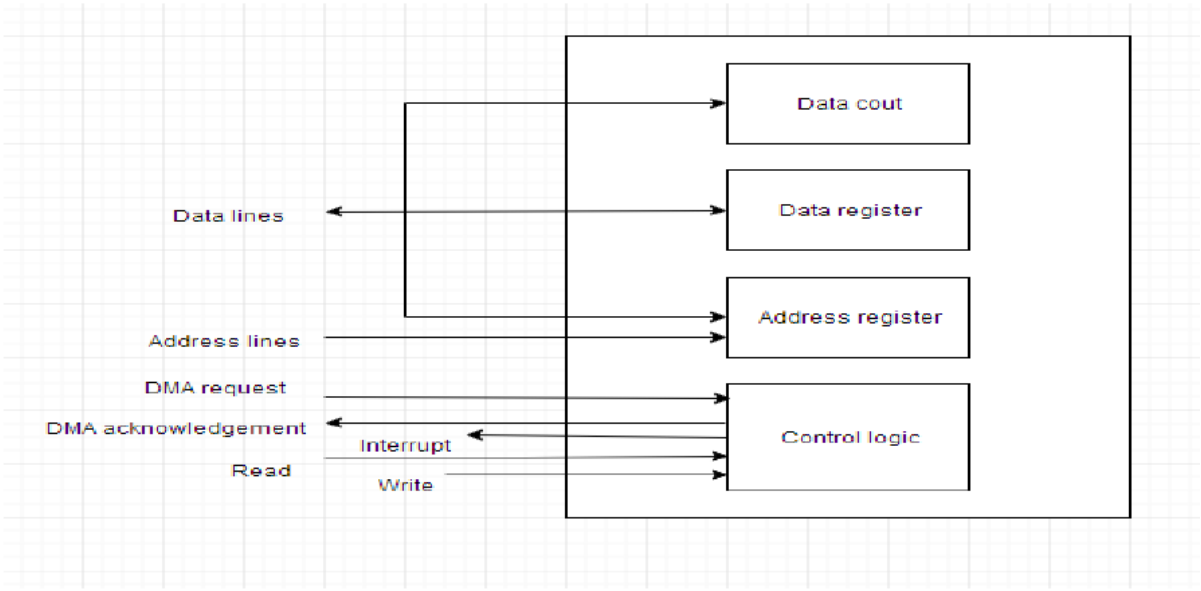
Interrupt Initiated I/O

In the programmed I/O method the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer. This is time consuming process because it keeps the processor busy needlessly.This problem can be overcome by using interrupt initiated I/O. In this when the interface determines that the peripheral is ready for data transfer, it generates an interrupt. After receiving the interrupt signal, the CPU stops the task which it is processing and service the I/O transfer and then returns back to its previous processing task.

Direct Memory Access

Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve thespeed of transfer. This technique is known as DMA.In this, the interface transfer data to and from the memory through memory bus. A DMA controller manages to transfer data between peripherals and memory unit.

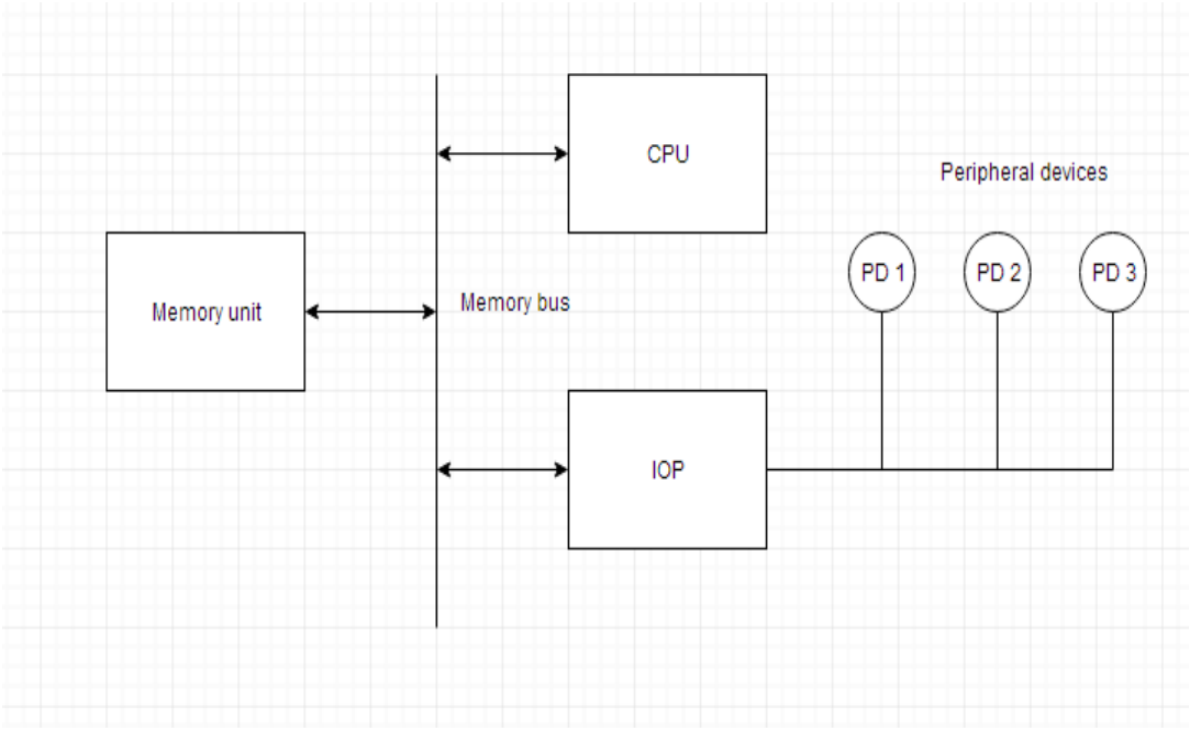
Many hardware systems use DMA such as disk drive controllers, graphic cards, network cards and sound cards etc. It is also used for intra chip data transfer in multicore processors. In DMA, CPU would initiate the transfer, do other operations while the transfer is in progress and receive an interrupt from the DMA controller when the transfer has been completed.



Computer Architecture: Input/Output Processor
An input-output processor (IOP) is a processor with direct memory access capability. In this, the computer system is divided into a memory unit and number of processors.

Each IOP controls and manage the input-output tasks. The IOP is similar to CPU except that it handles only the details of I/O processing. The IOP can fetch and execute its own instructions. These IOP instructions are designed to manage I/O transfers only.

Block Diagram Of I/O Processor
Below is a block diagram of a computer along with various I/OProcessors. The memory unit occupies the central position and can communicate with each processor.
The CPU processes the data required for solving the computational tasks. The IOP provides a path for transfer of data between peripherals and memory. The CPU assigns the task of initiating the I/O program.The IOP operates independent from CPU and transfer data between peripherals and memory.



The communication between the IOP and the devices is similar to the program control method of transfer. And the communication with the memory is similar to the direct memory access method. In large scale computers, each processor is independent of other processors and any processor can initiate the operation.
The CPU can act as master and the IOP act as slave processor. The CPU assigns the task of initiating operations but it is the IOP, who executes the instructions, and not the CPU. CPU instructions provide operations to start an I/O transfer. The IOP asks for CPU through interrupt. Instructions that are read from memory by an IOP are also

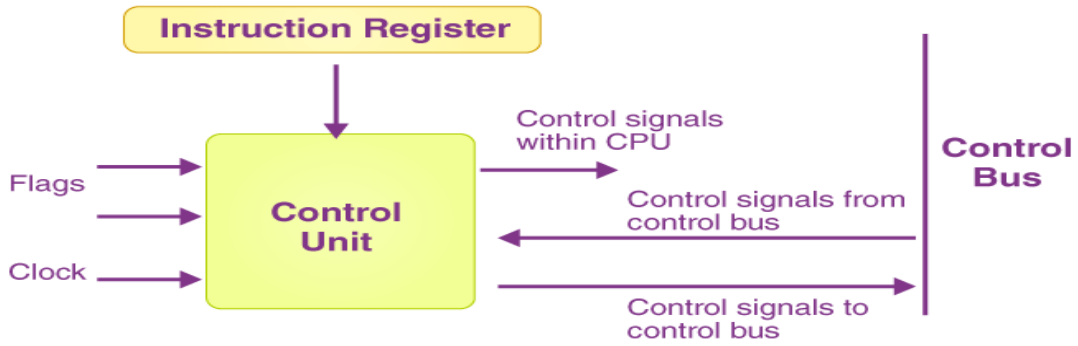
called commands to distinguish them from instructions that are read by CPU. Commands are prepared by programmers and are stored in memory. Command words make the program for IOP. CPU informs the IOP where to find the commands in memory.

Control unit

The control unit, abbreviated as CU, is an integral part of the computer's processor. It plays a pivotal role in managing the operations of the computer. The control unit gives instructions to the memory, logic unit, and both the output and input devices on how to react to a program's commands. Devices such as CPUs and GPUs are examples that utilize a control unit.

The control unit, a vital part of the computer's CPU (central processing unit), orchestrates the processor's operation. The concept of a control unit was first introduced by John von Neumann in his Von Neumann Architecture . The control unit's responsibility is to guide the computer's arithmetic/logic unit, memory, and input and output devices on how to react to the commands given to the processor.

The control unit retrieves internal program commands from the main memory, moves them to the processor instruction register, and produces a control signal based on the register's contents to manage the execution of these commands.



Block Diagram of the Control Unit

Understanding the Working of a CPU Control Unit

A control unit gets data from the user, converts it into control signals, and then passes these signals to the central processor. The computer's processor then guides the connected hardware on the tasks to perform. Since CPU architecture varies from one manufacturer to another, the functions carried out by a control unit in a computer depend on the type of CPU. Examples of devices that require a control unit include:

- CPUs or Central Processing Units
- GPUs or Graphics Processing Units

The Role of the Control Unit

- It manages the flow of data into, out of, and between the different subunits of a processor.
- It can interpret commands and instructions.
- It regulates the flow of data within the processor.
- It can accept external commands or instructions, which it transforms into a series of control signals.
- It oversees the multiple execution units of a CPU (such as ALUs, data buffers, and registers).
- It also performs various operations, including fetching, decoding, managing execution, and storing results.

Different Types of Control Units

The design of the control unit depends on the type of control unit being used. Here are the types of control units:

Hardwired Control Unit

In a Hardwired Control Unit, the necessary control signals for instruction execution control are generated by specially designed hardware logical circuits. The signal production mechanism cannot be changed without physically altering the circuit structure.

Micro Programmable Control Unit

The main difference between the unit structure and the hardwired control unit structure is the presence of the control store. The control store is used to store words containing encoded control signals necessary for instruction execution.

Instruction set architecture of a CPU – registers

An Instruction Set Architecture (ISA) defines the communication rules between the hardware and software of the computer. The ISA is a design principle (conceptual) and not stored in a computer's memory.

Some things an ISA defines:

- How binary instructions are formatted
- What instructions are available to be processed on a specific hardware setup
- How computer memory, (volatile and non-volatile) is accessed.

Complex Instruction Set Computers (CISC)

CISC (Complex Instruction Set Computer) is an ISA design practice that focuses on multi-step instructions and complex, power-consuming hardware. These designs primarily focus on hardware components and binary instruction complexity. Processing components are typically not interchangeable with RISC-designed systems.

CISC Instructions Attributes:

- Single instructions take more than one CP cycle to complete
- Instruction length varies based on the instruction type
- Hardware must be designed to accept more complicated instructions

CISC architecture materialized during the nascent stages of computing to simplify programming through a diverse set of instructions capable of performing intricate tasks within a single command. In contrast to RISC architecture, which focuses on minimizing instruction

count and streamlining execution, CISC processors adopt a broader approach by encompassing a range of instructions tailored for various operations.

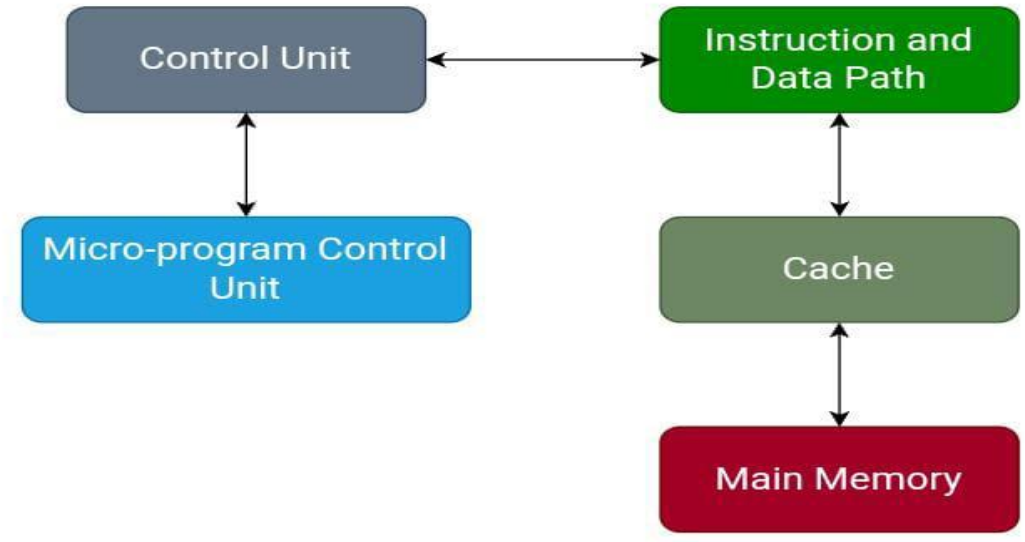
Core principles of CISC architecture

Several guiding principles underlie CISC:

1. **Complex instruction set:** CISC architecture relies on intricate instructions to carry out multifaceted low-level tasks or multi-step processes in solitary instruction. This approach reduces the number of instructions needed to achieve a particular task.
2. **Variable-length instructions:** CISC architectures employ a variable-length instruction format, allowing for instructions of different sizes. This flexibility facilitates the inclusion of more intricate commands.
3. **Direct operand manipulation:** CISC architectures enable direct manipulation of operands within memory, minimizing the overall number of memory accesses required.
4. **Extensive instruction set:** The abundance of instructions within CISC architectures spans a spectrum of operations, addressing modes, and data types. This inclusivity permits the execution of intricate tasks with a single instruction, minimizing the necessity for multiple commands and potential errors.
5. **Addressing modes:** CISC processors often support memory-to-memory operations, where data can be directly transferred between memory locations. This characteristic reduces the reliance on registers, augments the intricacy of memory management and addressing, and encompasses diverse addressing modes for enhanced memory access adaptability.

CISC architecture in comparison

CISC architecture is characterized by its intention to offer a comprehensive array of intricate and adaptable instructions within a singular instruction set for computer processors. This contrasts with RISC architecture, which prioritizes simplicity and efficient execution. The multi-step instructions inherent to CISC systems render them suitable for diverse tasks.



CISC architecture

CISC instruction set

CISC processors boast an extensive and varied instruction set tailored to handle diverse tasks. This encompasses arithmetic, logical, data movement, and control flow operations. The length of CISC instruction formats can vary due to the complexity of the operations. Each instruction comprises fields specifying the operation, operands, and addressing modes.

Advantages of CISC architecture

CISC architecture presents several advantages:

- **Versatility and rich instruction set:** A primary strength of CISC architecture lies in its comprehensive and diverse instruction set. This wealth of instructions streamlines coding for intricate operations, avoiding the need to deconstruct them into multiple simpler instructions.
- **Reduced code size:** Integrating multi-functional instructions can lead to shorter code sequences than RISC architectures. This can be advantageous in memory-constrained scenarios, as fewer instructions are necessary to accomplish a given task.
- **Programmer-friendly:** CISC architecture is often lauded for its capacity to execute complex operations using single instructions. This simplifies the programming process for developers engaged in intricate calculations or data manipulations.
- **Efficient memory use:** CISC processors facilitate memory-to-memory operations, allowing direct data manipulation without intermediate register transfers. This enhances memory space efficiency and potentially diminishes the demand for supplementary instructions.

Disadvantages of CISC Architecture

CISC architecture presents certain drawbacks:

- **Complex instruction decoding:** CISC architecture's extensive, variable-length instruction set necessitates intricate decoding logic. This complexity involves additional circuitry and time, potentially impeding overall processor performance.
- **Execution time variability:** Due to the varying lengths of CISC instructions, execution times for distinct instructions can differ significantly. This hinders accurate prediction of program execution time, affecting real-time applications requiring precise timing.
- **Pipelining challenges:** Effective pipelining, a technique that enhances processor performance by overlapping instruction execution, can be intricate in CISC architectures due to irregular instruction lengths and complex instructions.
- **Increased power consumption:** The intricacy of decoding and executing intricate instructions demands more power in CISC processors. This can result in higher energy consumption than simpler, more streamlined RISC architectures.
- **Limited compiler optimization:** The rich instruction set of CISC architectures can constrain compiler optimization efficacy. Compilers might encounter challenges in generating optimal code for such processors, potentially influencing overall performance.
- **Manufacturing complexity:** The intricate nature of CISC architecture, characterized by variable-length instructions and complex execution paths, can elevate the intricacy of processor design, manufacturing, and testing. This could lead to augmented costs and potential manufacturing hurdles.

Reduced Instruction Set Computers (RISC)

RISC (Reduced Instruction Set Computer) is an ISA design practice of ISAs that focuses on simple, quickly executed instructions to improve efficiency and reduce power consumption. These designs primarily focus on simple hardware components and reducing binary instruction complexity. Processing components are typically not interchangeable with CISC-designed systems.

General RISC Instructions Attributes:

- Single instructions take only one CPU cycle to complete
- Instruction lengths are fixed, regardless of the instruction type
- Reduced complexity of hardware leads to less power consumption at the expense of overall processing times.

RISC architecture emerged as a response to the challenges posed by the increasingly complex and lengthy instruction sets in traditional CISC architectures. In the late 1970s and early 1980s, computer scientists began questioning the necessity of incorporating complex instructions into processors. This led to the development of a new design philosophy that aimed to simplify instruction sets while maintaining or improving performance. RISC architecture was born from this idea.

Core principles of RISC architecture

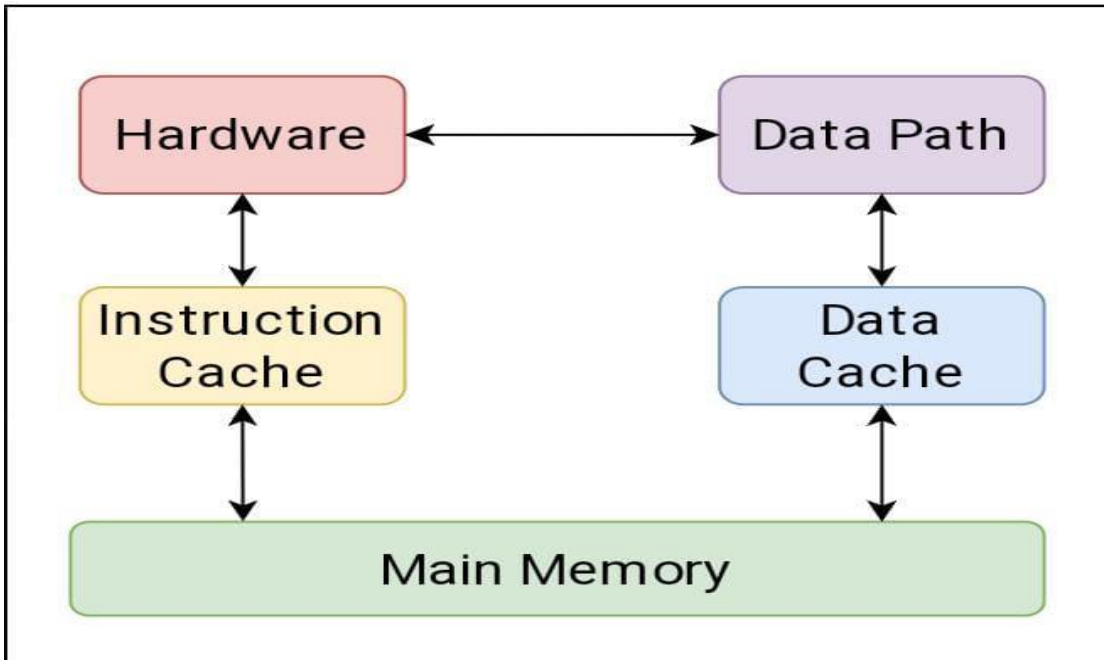
The hallmark principles of RISC architecture are summarized in these five points:

- **Single-cycle execution:** In most traditional central processing unit (CPU) designs, the peak possible execution rate is one instruction per basic machine cycle. For a given technology, the cycle time is determined by the slowest instruction in the instruction set. RISC designs aim to execute most instructions in a single cycle, increasing the processor's overall speed.
- **Load-store architecture:** RISC architectures use a load-store architecture, meaning only load and store instructions can access memory. All other instructions must operate on data in registers. This simplifies the instruction set and reduces the number of memory accesses required.
- **Simple instructions:** RISC architectures use simple instructions that can be executed quickly. This reduces the complexity of the processor and allows it to operate at a higher clock speed.
- **Large register set:** RISC architectures have many registers, which reduces the number of memory accesses required and allows for more efficient use of the processor's resources.
- **Pipelining:** RISC architectures use pipelining to increase the speed of instruction execution. Pipelining allows multiple instructions to be executed simultaneously, increasing the processor's overall throughput.

The RISC architecture

The hardware of RISC architecture is designed to execute the instruction quickly, which is possible because of the more precise and smaller number of instructions and a large number of registers.

In RISC, the data path is used to store and manipulate data in a computer. It is responsible for managing data within the processor and its movement between the processor and the memory. The processor uses a cache to reduce the access time to the main memory. The instruction cache is beneficial for retrieving and storing the data of frequently used instructions. It speeds up the process of instruction execution. The data cache provides storage for frequently used data from the main memory.



RISC architecture

RISC instruction set

In RISC, the length of the instruction format is fixed. It took one-word memory. The fixed size of the instruction format benefits the program counter as it knows that the next instruction starts from where due to the fixed length of all instructions. In RISC, each instruction requires only one clock execution cycle. In addition, RISC architectures are designed to be highly scalable and accommodate a more significant number of instructions.

Advantages of RISC architecture

The advantages of RISC architecture are as follows:

- **Simplified instruction set:** RISC architecture uses a small instruction set that is highly optimized and simple; instruction executes quickly.
- **Format length is fixed:** In RISC architecture format length of instruction is fixed, which makes the execution or decoding of instructions faster.
- **Register-based architecture:** It stores data in the register within the processor, which is frequently used. This improves the performance because it reduces the number of memory access.
- **Fewer cycles:** In RISC architecture, the instruction requires less number of cycles to execute because of the simple instruction set.
- **Load-Store architecture:** RISC architecture uses load-store architecture, which means memory access differs from logical and arithmetic operations. Therefore, the processor's resources are used more efficiently, improving performance.

Disadvantages of RISC architecture

The disadvantages of RISC architecture are as follows:

- **Complex instructions and addressing modes:** It isn't easy to process complex instructions and complex addressing modes in the RISC architecture.
- **Direct memory-to-memory transfer:** It uses load-store architecture. Hence, it doesn't allow a direct memory-to-memory transfer.
- **Increase in the program length:** RISC architecture has a small and simple instruction set. However, it requires more instruction to operate CISC architecture, increasing the program's length.

Control Unit (CU)

The Control Unit (CU) on a CPU receives information from the software; then, it distributes and directs the data to the relevant hardware components.

Some functions of the CU:

- Determine what/where the next instruction must go for processing
- Send clock signals to all hardware to force synchronous operations
- Send memory taskings if appropriate

Arithmetic and Logic Unit (ALU)

An Arithmetic Logic Unit (ALU) is a digital circuit used to perform arithmetic and logic operations. It is the fundamental building block of the CPU.

Some ALU functions:

- Addition & subtraction
- Determining equality
- AND/OR/XOR/NOR/NOT/NAND logic gates and more

Computer Instructions

Computer instructions are written in binary, also known as machine code. Computer hardware operates on a series of these binary instructions through pulsating power signals that signify either OFF or ON based on the binary digits 0 and 1 respectively.

Registers

A register is a volatile memory system that provides the CPU with rapid access to information it is immediately using.

Functions of a register:

- Store temporary data for immediate processing by the ALU
- Hold "flag" information if an operation results in overflow or triggers other flags
- Hold the location of the next instruction to be processed by the CPU

Addressing Modes

The operands of the instructions can be located either in the main memory or in the CPU registers. If the operand is placed in the main memory, then the instruction provides the location address in the operand field. Many methods are followed to specify the operand address. The different methods/modes for specifying the operand address in the instructions are known as addressing modes.

Types of Addressing Modes
There are various types of Addressing Modes which are as follows –

Implied Mode − In this mode, the operands are specifiedimplicitly in the definition of the instruction. For example, theinstruction "complement accumulator" is an implied-mode instruction because the operand in the accumulator register isimplied in the definition of the instruction. All register referenceinstructions that use an accumulator are implied-mode instructions.Instruction format with mode field-

Opcode	Mode	Address

Immediate Mode − In this mode, the operand is specified in the instruction itself. In other words, an immediate-mode instruction has an operand field instead of an address field. The operand field includes the actual operand to be used in conjunction with the operation determined in the instruction.

Immediate-mode instructions are beneficial for initializing registers to a constant value.

Register Mode − In this mode, the operands are in registers that reside within the CPU. The specific register is selected from a register field in the instruction. A k-bit field can determine any one of the 2 k registers.

Register Indirect Mode − In this mode, the instruction defines aregister in the CPU whose contents provide the address of the operand in memory. In other words, the selected register includes the address of the operand rather than the operand itself.
A reference to the register is then equivalent to specifying a memory address. The advantage of a register indirect mode instruction is that the address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly.

Autoincrement or Autodecrement Mode
& minuend; This is similar to the register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory. When the address stored in the register defines a table of data in memory, it is necessary to increment or decrement the register after every access to the table. This can be obtained by using the increment or decrement instruction.

Direct Address Mode − In this mode, the effective address isequal to the address part of the instruction. The operand resides in memory and its address is given directly by the address field of the instruction. In a branch-type instruction, the address field specifies the actual branch address.

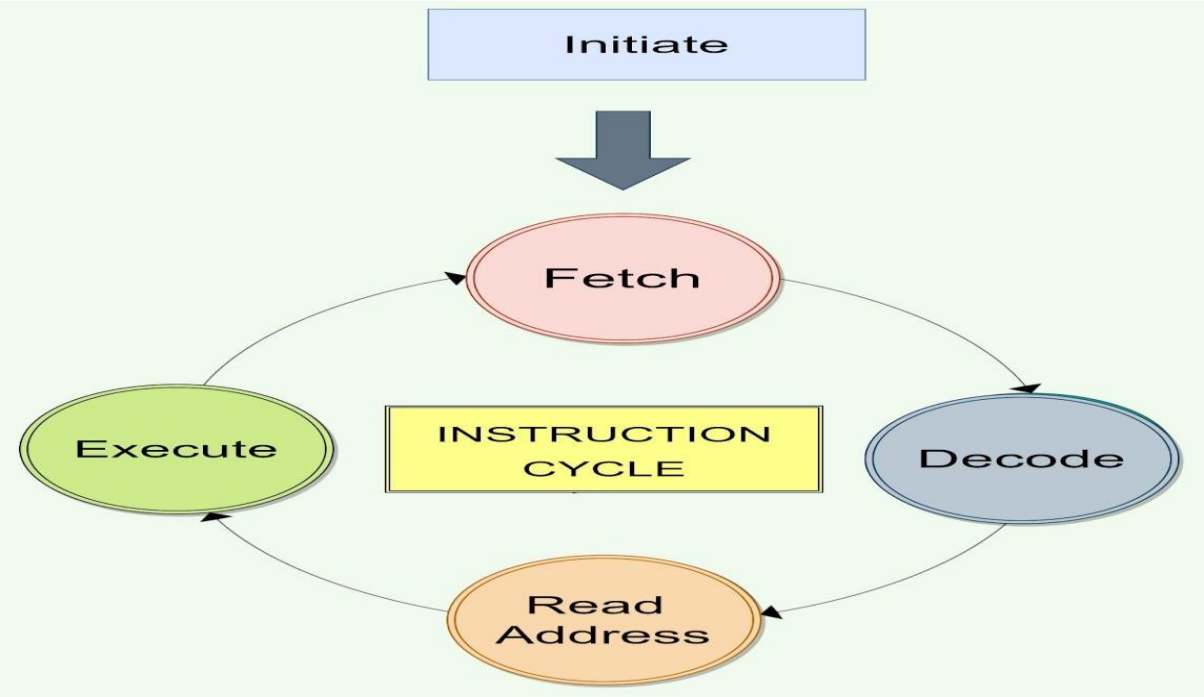
Indirect Address Mode − In this mode, the address field of the instruction gives the address where the effective address is stored in memory. Control fetches the instruction from memory and uses its address part to access memory again to read the effective address.

Indexed Addressing Mode − In this mode, the content of an index register is added to the address part of the instruction to obtain the effective address. The index register is a special CPU register that contains an index value. The address field of the instruction defines the beginning address of a data array in memory.

Instructions execution cycle

The instruction cycle is defined as the basic cycle in which a computer system fetches an instruction from memory, decodes it, and then executes it. Fetch-Execute-Cycle is another name for it. All instructions in a computer system are executed in the RAM of the computer system. The CPU is in charge of carrying out the instruction.
Each instruction cycle in a basic computer includes the following procedures:

- It has the ability to retrieve instructions from memory.
- It's used to decode the command.
- If the instruction has an indirect address, it can read the effective address from memory.
- It is capable of carrying out the command.



Initiating Cycle
During this phase, the computer system boots up and the Operating System loads into the central processing unit's main memory. It begins when the computer system starts.
Fetching of Instruction
The first phase is instruction retrieval. Each instruction executed in a central processing unit uses the fetch instruction. During this phase, the central processing unit sends the PC to MAR and then the READ instruction to a control bus. After sending a read instruction on the data bus, the memory returns the instruction that was stored at that exact address in the memory. The CPU then copies data from the data bus into MBR, which it then copies to registers. The pointer is incremented to the next memory location, allowing the next instruction to be fetched from memory.
Decoding of Instruction
The second phase is instruction decoding. During this step, the CPU determines which instruction should be fetched from the instruction and what action should be taken on the

instruction. The instruction's opcode is also retrieved from memory, and it decodes the related operation that must be performed for the instruction.

Read of an Effective Address

The third phase is the reading of an effective address. The operation's decision is made during this phase. Any memory type operation or non-memory type operation can be used. Direct memory instruction and indirect memory instruction are the two types of memory instruction available.

Execution of Instruction

The last step is to carry out the instructions. The instruction is finally carried out at this stage. The instruction is carried out, and the result is saved in the register. The CPU gets prepared for the execution of the next instruction after the completion of each instruction. The execution time of each instruction is calculated, and this information is used to determine the processor's processing speed.

In order for a single instruction to be executed by the CPU, it must go through the **instruction cycle** (also sometimes referred to as the fetch-execute cycle). While this cycle can vary from CPU to CPU, they typically consist of the following stages:

1. Fetch
2. Decode
3. Execute
4. Memory Access
5. Registry Write-Back

In this article, we'll go through the different stages of the instruction cycle to gain a better understanding of how the CPU handles instructions.

Fetching

From the moment we turn our computers on, the CPU is ready to process instructions. As instructions come in, a register in the CPU referred to as the Program Counter (PC) stores the memory address of the instruction that should be processed next. When it's time to start processing the instruction, the CPU copies the instruction's memory address and stores the copied data to another register on the CPU called the Instruction Register (IR). Once the memory of the instruction is available, the instruction gets decoded.

Think of being at a deli. As you come in and give your order, a ticket containing your data (name, number in line, and food order) is created and placed somewhere that the deli staff can easily access and refer to. Once your number comes up, then someone will start working on your order!

Decoding

The next stage in the cycle involves decoding the instruction. During this stage, the Control Unit deciphers what the instruction stored in the IR means. For example, the instruction could have been sent to do an arithmetic operation or to send information to another piece of hardware. As the instruction is decoded, they are turned into a series of control signals that are used to execute the instruction.

Back at the deli, a staff member picks up your order ticket. Before they start making your order, they first need to figure out what you're asking them to make!

Execution

In this stage, the instruction is performed! We noted that during the decoding stage, the instruction is decoded into control signals and sent to the correct part of the ALU to be processed and completed.

In our deli example, this is the part where the order gets made!

So to recap, in order to process an instruction, we need to fetch it from memory, decode the instruction, and execute it. That's all, right? Not quite! Sometimes a few extra stages need to occur before or after execution.

Memory Access

The memory access stage is used to retrieve any required data necessary to execute an instruction. This stage only occurs if the instruction requires data from memory. For example, imagine the following Python code:

```
x = 5
y = x + 3
```

Once the first instruction is complete, a piece of memory is created to store the data x = 5. The second instruction, y = x + 3, is a little trickier to execute because the value of y relies on whatever value was assigned to x. Before y = x + 3 can be executed, we need to access the memory address of the first instruction x = 5 in order to retrieve the data that tells us what the value of x is.

Imagine in your deli order, you ask for honey mustard to be added to one of the two sandwiches you order. Before your order can be created, the staff member needs to make and retrieve honey mustard for the sandwich.

Registry Write-Back

The registry write-back stage is used if the execution of the instruction impacts data. This is another stage that isn't always a part of the cycle.

Let's think back to our previous example:

```
x = 5
y = x + 3
```

As each instruction is executed, we find ourselves needing to save this data. During the registry write-back stage, this new data is stored to one of the register's in the CPU. The registry write-back stage is also necessary if existing data is changed or updated.

As the deli's 10,000th customer, they decide to name your order after you and put it on their "Deli Specials" board. They need to create space and allocate a part of the board's "memory" to store your order.

RTL interpretation of instructions

In symbolic notation, it is used to describe the micro-operations transfer among registers. It is a kind of intermediate representation (IR) that is very close to assembly language, such as that which is used in a compiler.The term “Register Transfer” can perform micro-operations and transfer the result of operation to the same or other register.

Micro-operations:

The operation executed on the data store in registers are called micro-operations. They are detailed low-level instructions used in some designs to implement complex machine instructions.

Register Transfer:

The information transformed from one register to another register is represented in symbolic form by replacement operator is called Register Transfer.

Replacement Operator:

In the statement, R2 <- R1, <- acts as a replacement operator. This statement defines the transfer of content of register R1 into register R2.

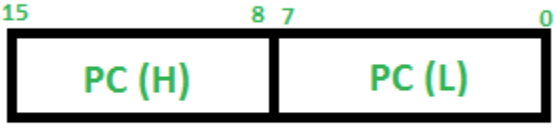
There are various methods of RTL –

1. General way of representing a register is by the name of the register enclosed in a rectangular box as shown in (a).
2. Register is numbered in a sequence of 0 to (n-1) as shown in (b).
3. The numbering of bits in a register can be marked on the top of the box as shown in (c).
4. A 16-bit register PC is divided into 2 parts- Bits (0 to 7) are assigned with lower byte of 16-bit address and bits (8 to 15) are assigned with higher bytes of 16-bit address as shown in (d).

(a)

(b)

(c)



(d)

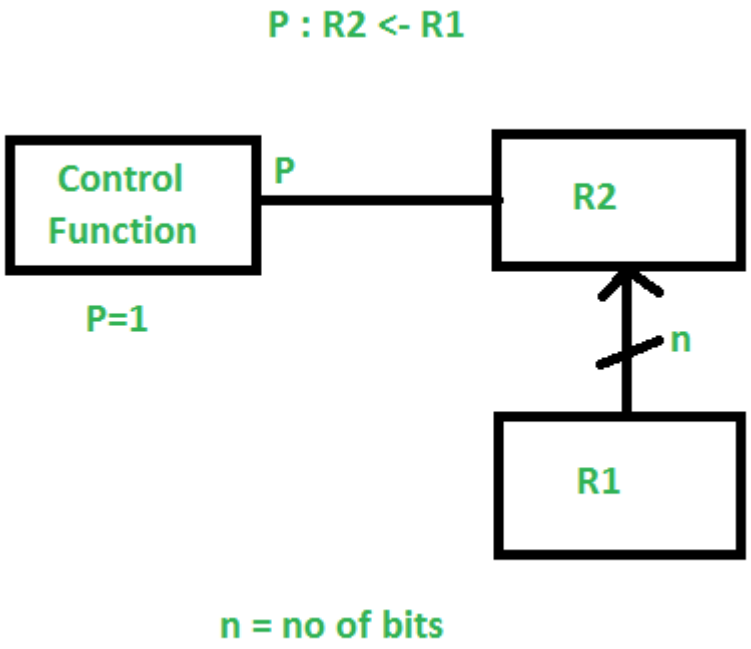
Basic symbols of RTL:

Symbol	Description	Example
Letters and Numbers	Denotes a Register	MAR, R1, R2
( )	Denotes a part of register	R1(8-bit) R1(0-7)
<-	Denotes a transfer of information	R2 <- R1
,	Specify two micro-operations of Register Transfer	R1 <- R2 R2 <- R1
:	Denotes conditional operations	P : R2 <- R1 if P=1
Naming Operator (:=)	Denotes another name for an already existing register/alias	Ra := R1

Register Transfer Operations:
The operation performed on the data stored in the registers are referred to as register transfer operations.
There are different types of register transfer operations:

1. Simple Transfer – R2 <- R1
The content of R1 are copied into R2 without affecting the content of R1. It is an unconditional type of transfer operation.

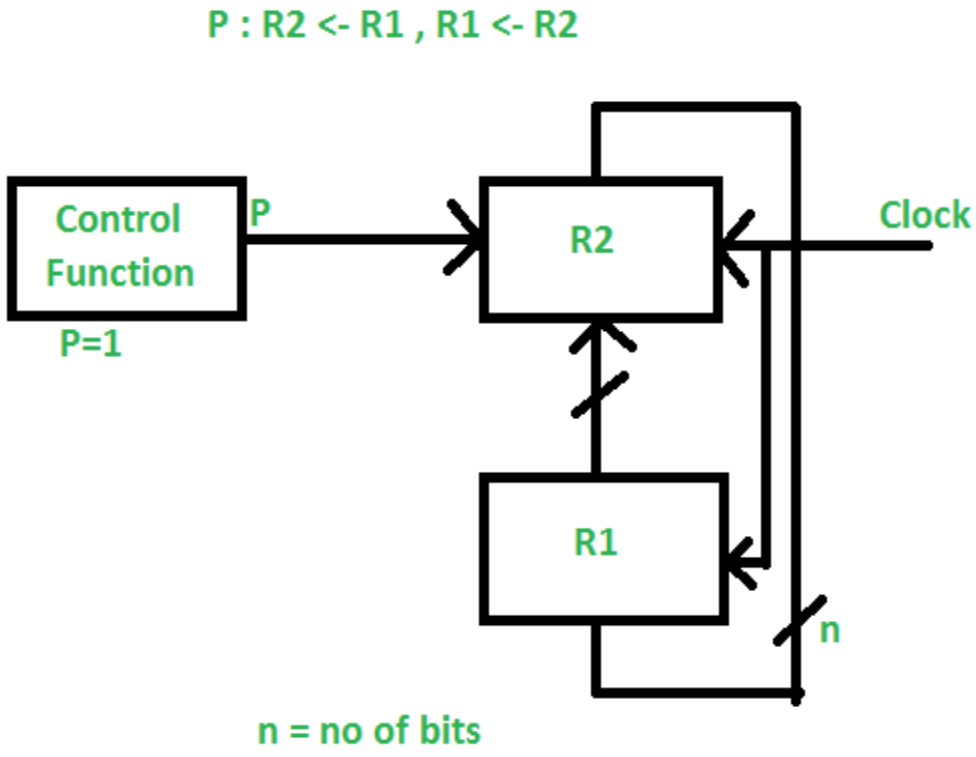
2. Conditional Transfer –



It indicates that if P=1, then the content of R1 is transferred to R2. It is a unidirectional operation.

3.Simultaneous Operations –

If 2 or more operations are to occur simultaneously then they are separated with comma **(,)**.



If the control function P=1, then load the content of R1 into R2 and at the same clock load the content of R2 into R1.

8085 Processor in Computer Architecture

Developed by Intel Corporation in the late 1970s, the 8085 microprocessor revolutionized the field, paving the way for countless advancements in technology. In this article, we delve into the architecture, features, and impact of the 8085 microprocessor, showcasing important points such as 8085 microprocessor architecture,
8085 microprocessor pin diagram, 8085 microprocessor instruction set,
functional unit and more.

The 8085 microprocessor is an 8-bit microprocessor that was introduced by Intel in 1976. It is part of the 8085 family of microprocessors, which includes the 8080, 8085A, and other variants. The 8085 microprocessor became very popular and widely used in various applications, including personal computers, embedded systems, industrial control systems, and more.

Functional Units of 8085 Microprocessor

The 8085 microprocessor consists of several functional units that work together to perform various tasks. These units include the following:

- Arithmetic Logic Unit (ALU)

The ALU is responsible for performing arithmetic and logical operations on data. It can perform operations such as addition, subtraction, logical AND, logical OR, and more. The ALU operates on 8-bit data and provides flags to indicate conditions such as zero, carry, sign, and parity.

- Control Unit (CU)

The Control Unit coordinates and controls the activities of the other functional units within the microprocessor. It generates timing and control signals to synchronize the execution of instructions and manage data transfer between different units.

- Instruction Decoder

The Instruction Decoder decodes the instructions fetched from memory. It determines the type of instruction being executed and generates control signals accordingly. The decoded instructions guide the microprocessor in executing the appropriate operations.

- Registers

The 8085 microprocessor has several registers that serve different purposes:

Accumulator (A): The Accumulator is an 8-bit register used for storing intermediate results during arithmetic and logical operations.

General Purpose Registers (B, C, D, E, H, L): These are six 8-bit registers that can be used for various purposes, including storing data and performing operations.

Special Purpose Registers (SP, PC): The Stack Pointer (SP) is used to manage the stack in memory and the program counter (PC) keeps track of the memory address of the following instruction for fetching.

- Address and Data Bus

The microprocessor uses a bidirectional address bus to specify the memory location or I/O device it wants to access. Similarly, it employs an 8-bit bidirectional data bus for transferring data between the microprocessor and memory or I/O devices.

- Timing and Control Unit

The Timing and Control Unit generates the necessary timing signals to synchronize the activities of the microprocessor. It produces signals such as RD (Read), WR (Write), and various control signals required for instruction execution.

- Interrupt Control Unit

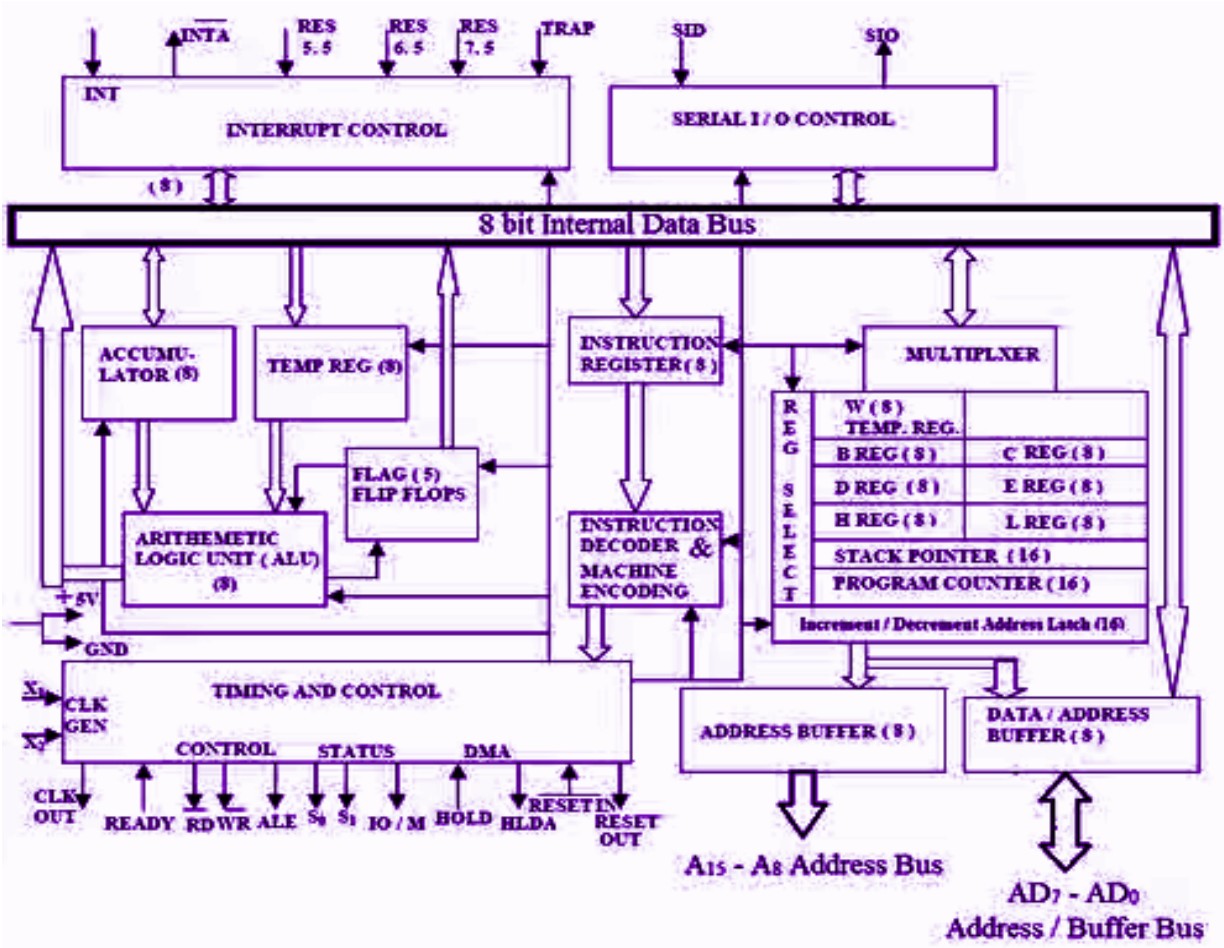
The Interrupt Control Unit manages interrupts in the 8085 microprocessor. It handles external interrupt signals and facilitates interrupt-driven operations by interrupting the normal execution flow of the program and branching to specific interrupt service routines.

- Memory Interface

The Memory Interface connects the microprocessor to the memory system. It manages the address and data transfers between the microprocessor and the memory chips, including Read and Write operations.

8085 Microprocessor Architecture Diagram

The 8085 microprocessor architecture diagram is as follows:



8085 Microprocessor Architecture

The 8085microprocessor architecture has the following essential components:

- Accumulator (A):

The accumulator is an 8-bit register used for arithmetic and logical operations. It holds one of the operands during calculations and stores the result.

- General-Purpose Registers (B, C, D, E, H, L):

The 8085 microprocessor has six general-purpose registers, each of which is 8 bits. These registers can be used for data manipulation and storage.

- Stack Pointer (SP):

The stack pointer is a 16-bit register that points to the top of the stack. The stack is used to store return addresses during subroutine calls, as well as local variables and other data.

- Program Counter (PC):

Program counter is a 16-bit register that holds the memory address of the following instruction for fetching and executing. It automatically increments after each instruction fetch, allowing programs to execute the next instruction quickly.

- Flag Register (F):

The flag register is a 8-bit register that contains several flags that indicate the status of certain conditions after arithmetic and logical operations. The flags are as follows:

Carry Flag (C): Set if there is a carry or borrow during arithmetic operations.

Zero Flag (Z): Set if the result of an operation is zero.

Sign Flag (S): Set if the result of an operation is negative (MSB set).

Parity Flag (P): Set if the result of an operation has even parity.

Auxiliary Carry Flag (AC): Set if there is a carry or borrow from bit 3 to bit 4 during arithmetic operations.

- Address and Data Buses:

The 8085 microprocessor has a 16-bit address bus, allowing it to address up to 64 KB of memory. It also has an 8-bit data bus for transferring data between the microprocessor and memory or I/O devices.

- Instruction Decoder and Control Unit:

The instruction decoder interprets the fetched instructions and generates control signals to execute the corresponding operation. The control unit coordinates the timing and sequencing of instructions and manages the flow of data and control signals within the microprocessor.

- Interrupts:

The 8085 microprocessor supports 5 hardware interrupts as follows:

1. INTR
2. RST 5.5
3. RST 6.5
4. RST 7.5
5. TRAP

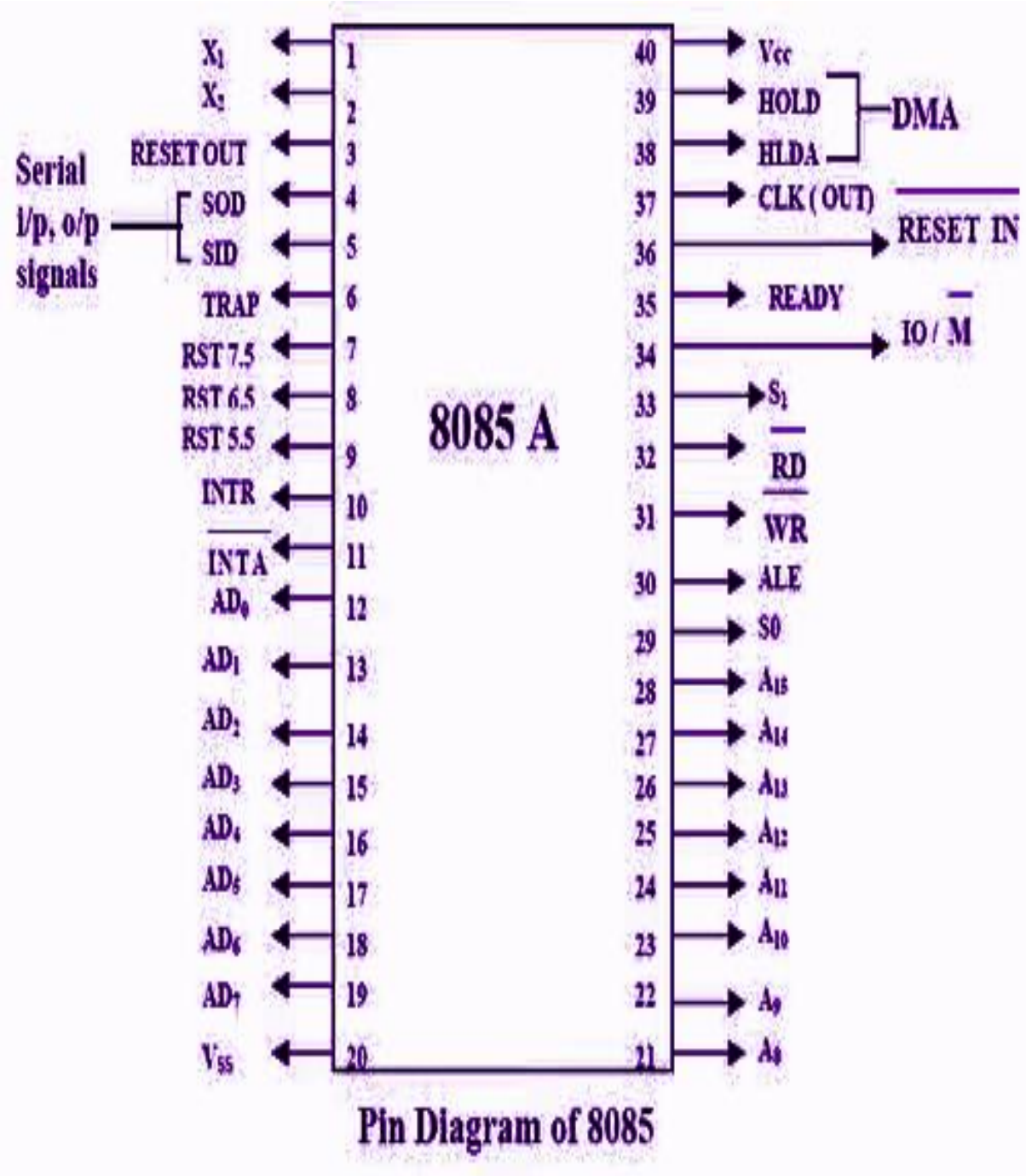
Interrupts allow the microprocessor to respond to external events and handle them in a prioritized manner.

- Input/Output (I/O) Ports:

The 8085 microprocessor has a limited number of I/O pins. It uses I/O ports to connect with external devices for input and output operations.

8085 Microprocessor Pin Diagram

Here's a brief description of some of the important pins in the 8085 microprocessor pin diagram:



Pin Diagram of 8085

- Vcc: Power supply voltage pin (+5V).
- A0-A15: Address bus pins for transmitting the memory address during read and write operations.
- AD0-AD7: Data bus pins for transmitting data between the microprocessor and external memory or I/O devices.
- RD: Read control signal, activated by the microprocessor to initiate a read operation.
- WR: Write control signal, activated by the microprocessor to initiate a write operation.
- /RESET: Active-low reset input, used to reset the microprocessor to its initial state.
- /INT: Active-low interrupt input, used for external hardware interrupts.
- IO/M: Input/output and memory control signal, used to distinguish between I/O and memory operations.
- /IORQ: Active-low input/output request signal, indicating an I/O operation.
- /MREQ: Active-low memory request signal, indicating a memory operation.
- S0 and S1: Status output pins that indicate the current operating mode of the microprocessor.
- HLDA: Hold acknowledge output, used to indicate that the microprocessor has recognized the HOLD request.
- HOLD: Hold input, used to halt the microprocessor's operation temporarily.
- The 8085 microprocessor pin diagram provides a visual representation of the connections and functionalities of the different pins on the 8085 microprocessor.

Addressing Modes in 8085 Microprocessor

The 8085 microprocessor uses various addressing modes to specify the location of data or instructions during program execution. These addressing modes determine how operands are accessed and fetched by the processor. Each addressing mode provides flexibility in accessing and manipulating data and instructions, allowing for efficient program execution.
Here are the commonly used addressing modes in the 8085 microprocessor:

- Immediate Addressing Mode: In this mode, the operand is directly specified in the instruction itself.
- Register Addressing Mode: In this mode, the operand is specified in one of the registers of the microprocessor.
- Direct Addressing Mode: In this mode, the operand is specified by a memory address directly.
- Indirect Addressing Mode: In this mode, the operand is specified indirectly through a register pair, usually HL. The contents of the specified register pair act as a memory address.
- Register Indirect with Displacement Addressing Mode: In this mode, the operand is specified by adding an 8-bit signed displacement value to the contents of a register pair. The resulting sum is treated as a memory address.
- Immediate Indirect Addressing Mode: In this mode, the operand is specified indirectly through an immediate value. The immediate value is added to the program counter (PC), and
- the resulting sum acts as the memory address.

8085 Microprocessor Instruction Format

The 8085 microprocessor follows a specific instruction format for executing instructions. The general instruction format consists of one to three bytes, depending on the instruction type. Here is the breakdown of the 8085 instruction format:

- Opcode (1 byte):

The opcode specifies the operation to be performed by the microprocessor. It represents the instruction mnemonic or code.
Examples of opcodes are MOV, ADD, SUB, JMP, etc.

- Operand(s) (0 to 2 bytes):

The operands provide the necessary data or addresses for the instruction. The number of operands and their sizes depend on the specific instruction. Some instructions do not require any operands, while others may have one or two operands. The operands can be immediate values, registers, memory addresses, or combinations thereof.

Operations on 8085 Microprocessor

The 8085 microprocessor supports a wide range of operations. Here are some of the commonly used operations on the 8085 microprocessor:

- Data Transfer Operations
- Arithmetic Operations
- Logical Operations
- Branching Operations
- I/O Operations
- Stack Operations

8085 Microprocessor Instruction Set

The 8085 microprocessor has a comprehensive instruction set that covers a wide range of operations. Here is an overview of the 8085 microprocessor instruction set categories and some representative instructions within each category:

- Data Transfer Instructions:
  MOV: Move data between registers, memory, and I/O ports.
  MVI: Move immediate data into a register or memory location.
  LXI: Load immediate 16-bit data into a register pair.
  LDA, STA: Load and store data between the accumulator and memory.

- Arithmetic Instructions:
  ADD, ADC: Add data to the accumulator with or without carry.
  SUB, SBB: Subtract data from the accumulator with or without borrow.
  INR, DCR: Increment or decrement the value of a register or memory location.
  DAD: Double the value of a register pair.

- Logical Instructions:
  AND, ORA, XOR: Perform bitwise logical operations on the accumulator with data from registers, memory, or immediate values.
  CMP: Compare the accumulator with data from registers, memory, or immediate values.

- Branching Instructions:
  JMP: Unconditional jump to a specific memory address.
  JC, JNC, JP, JM, JZ, JNZ: Conditional jumps based on the status of the carry flag (C), parity flag (P), sign flag (M), zero flag (Z), etc.
  CALL: Call a subroutine at a specific memory address.
  RET: Return from a subroutine.

- Stack Instructions:
  PUSH, POP: Push and pop data onto and from the stack.
  XTHL: Exchange stack top with HL register pair.

- I/O Instructions:
  IN, OUT: Input and output data from/to I/O ports.
  EI, DI: Enable and disable interrupts.

- Special Instructions:
  HLT: Halt the microprocessor.
  NOP: No operation.

The 8085 microprocessor stands as a remarkable milestone in the history of computing, showcasing the ingenuity and innovation of its creators. Its working principles and architecture have played a crucial role in shaping the landscape of modern technology.