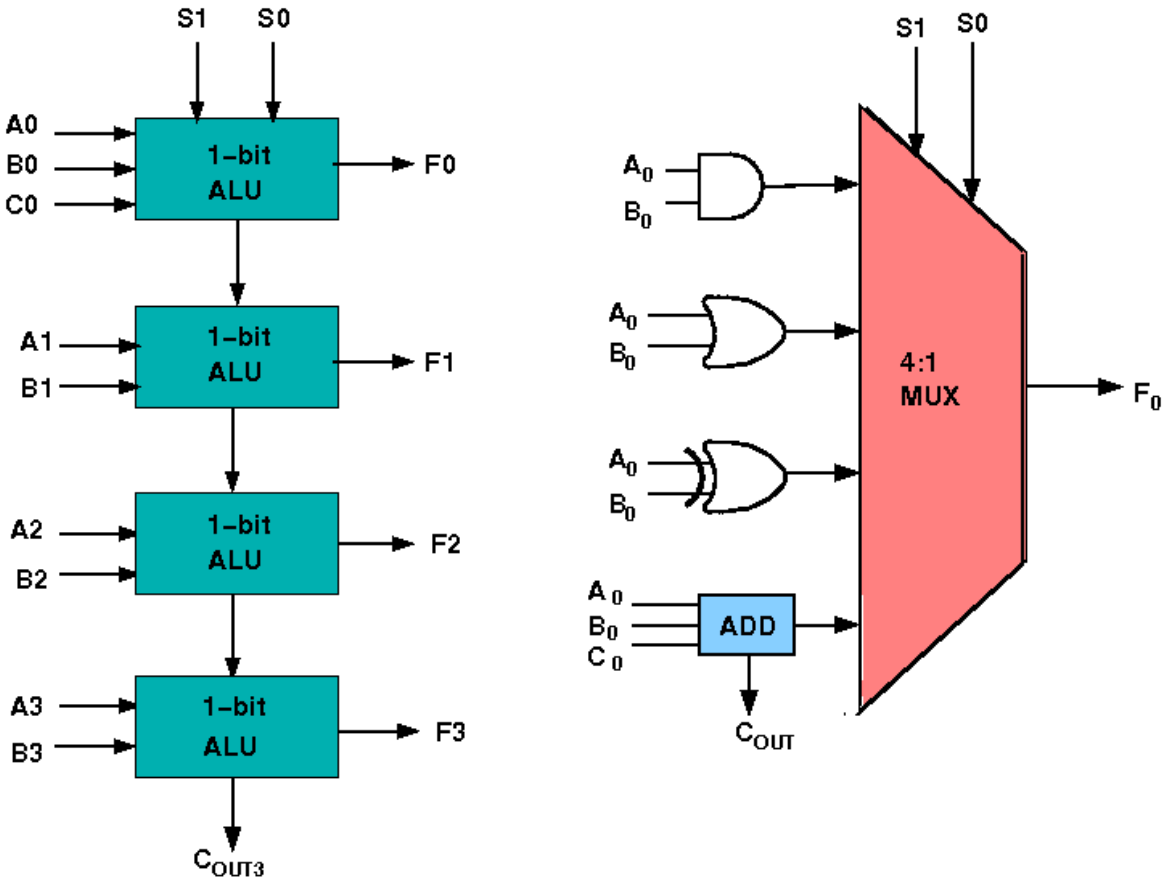
7.1 Design of 4-bit ALU

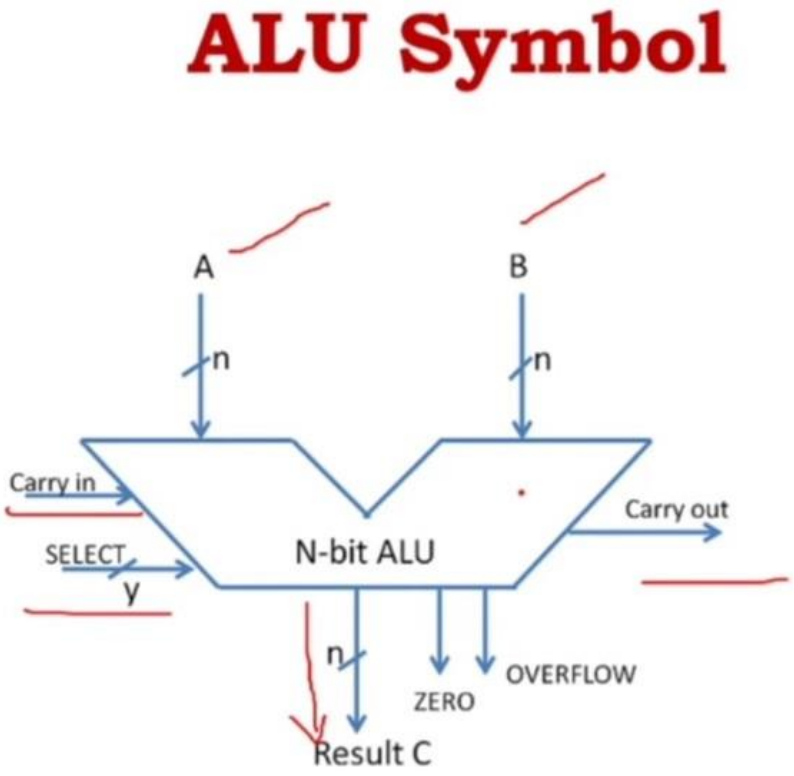
Design of ALU

ALU or Arithmetic Logical Unit is a digital circuit to do arithmetic operations like addition, subtraction, division, multiplication and logical operations like and, or, XOR, NAND, NOR etc. A simple block diagram of a 4-bit ALU for operations AND, OR, XOR and Add is shown here:

Block Diagram of ALU



ALU Symbol:



Design of 4-bit ALU

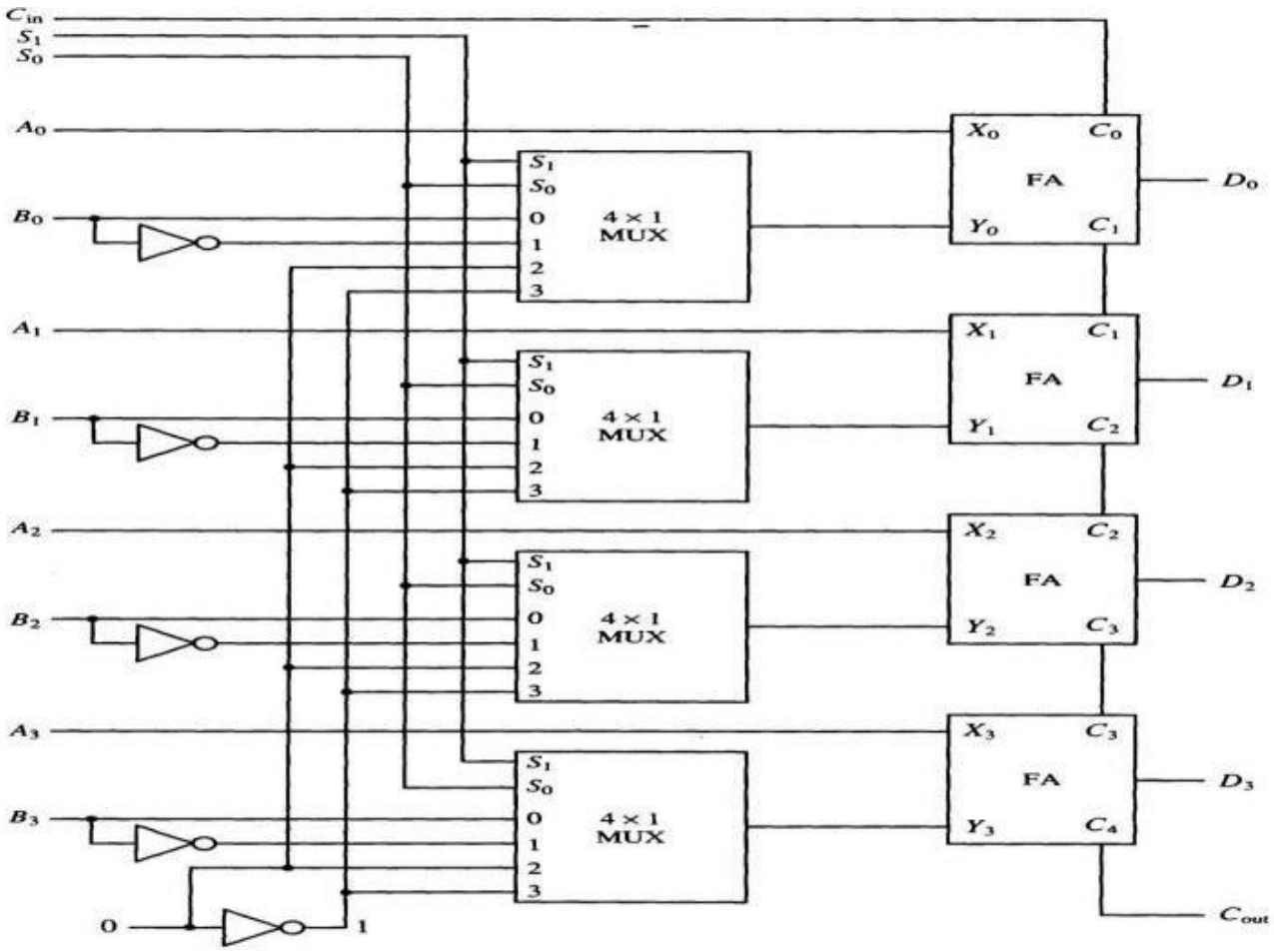


Fig: The 4-bit ALU block is combined using 4 1-bit ALU block

Design Issues:

The circuit functionality of a 1-bit ALU is shown here, depending upon the control signal S_1 and S_0 the circuit operates as follows:

for Control signal **S_1 = 0** , S_0 = 0, the output is **A And B**,

for Control signal **S_1 = 0** , S_0 = 1, the output is **A Or B**,

for Control signal **S_1 = 1** , S_0 = 0, the output is **A Xor B**,

for Control signal **S_1 = 1** , S_0 = 1, the output is **A Add B**.

The truth table for 4-bit ALU

SELECT			INPUT	OUT
S_1	S_0	Cin	Y	A+Y+Cin
0	0	0	B	D= A +B
0	0	1	B	D = A +B +1
0	1	0	$\overline{B}$	D= A+ $\overline{B}$
0	1	1	$\overline{B}$	D = A + $\overline{B}$ +1
1	0	0	0	D = A
1	0	1	0	D = A + 1
1	1	0	1	D = A - 1
1	1	1	1	D = A

7.2 Accumulator

An accumulator is a special-purpose register within the CPU's Arithmetic Logic Unit (ALU) that stores intermediate results of calculations. It effectively "accumulates" values during a sequence of operations, hence the name. While modern CPUs often use general-purpose registers for greater flexibility, accumulators were a common feature in older architectures and are still used in some specialized processors. The Term accumulator is used in refence to contemporary CPUs, having been replaced around the turn of the millennium by the term “register”.

Here's a more detailed explanation:

- **Purpose:**

The accumulator's primary function is to hold the output of arithmetic and logical operations performed by the ALU.

- **Example:**

Imagine calculating 3 + 4 + 5. The accumulator would first hold 3, then 7 (after adding 3 and 4), and finally 12 (after adding 5).

- **Historical Context:**

Accumulators were prominent in earlier CPU designs, such as those used in the PDP-8, a popular processor for process control and laboratory applications.

- **Modern CPUs:**

While accumulators are less common in modern general-purpose CPUs, they can still be found in specialized processors or in certain instruction sets.

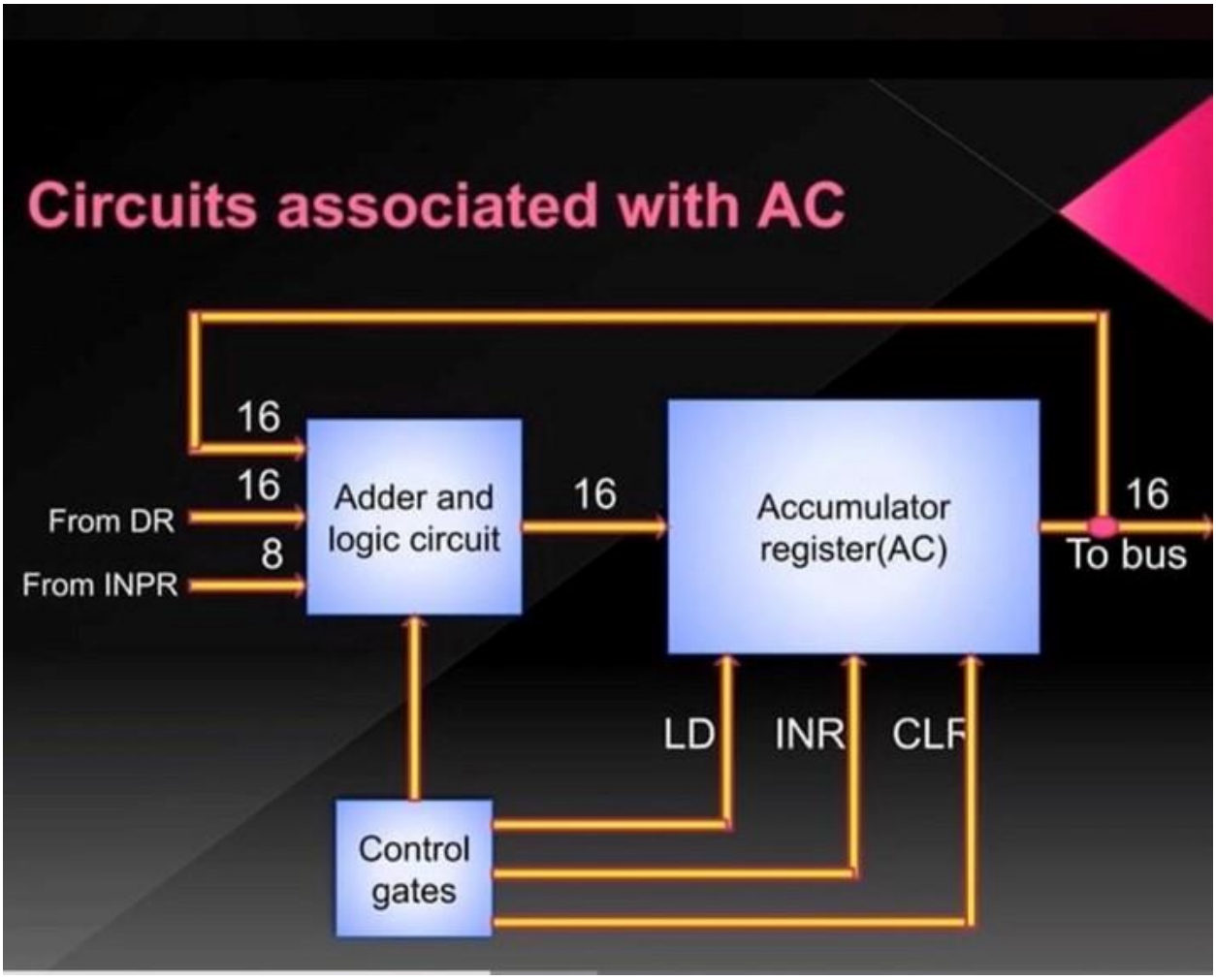
- **Alternatives:**

General-purpose registers offer more flexibility as they can be used for various purposes beyond accumulating values.

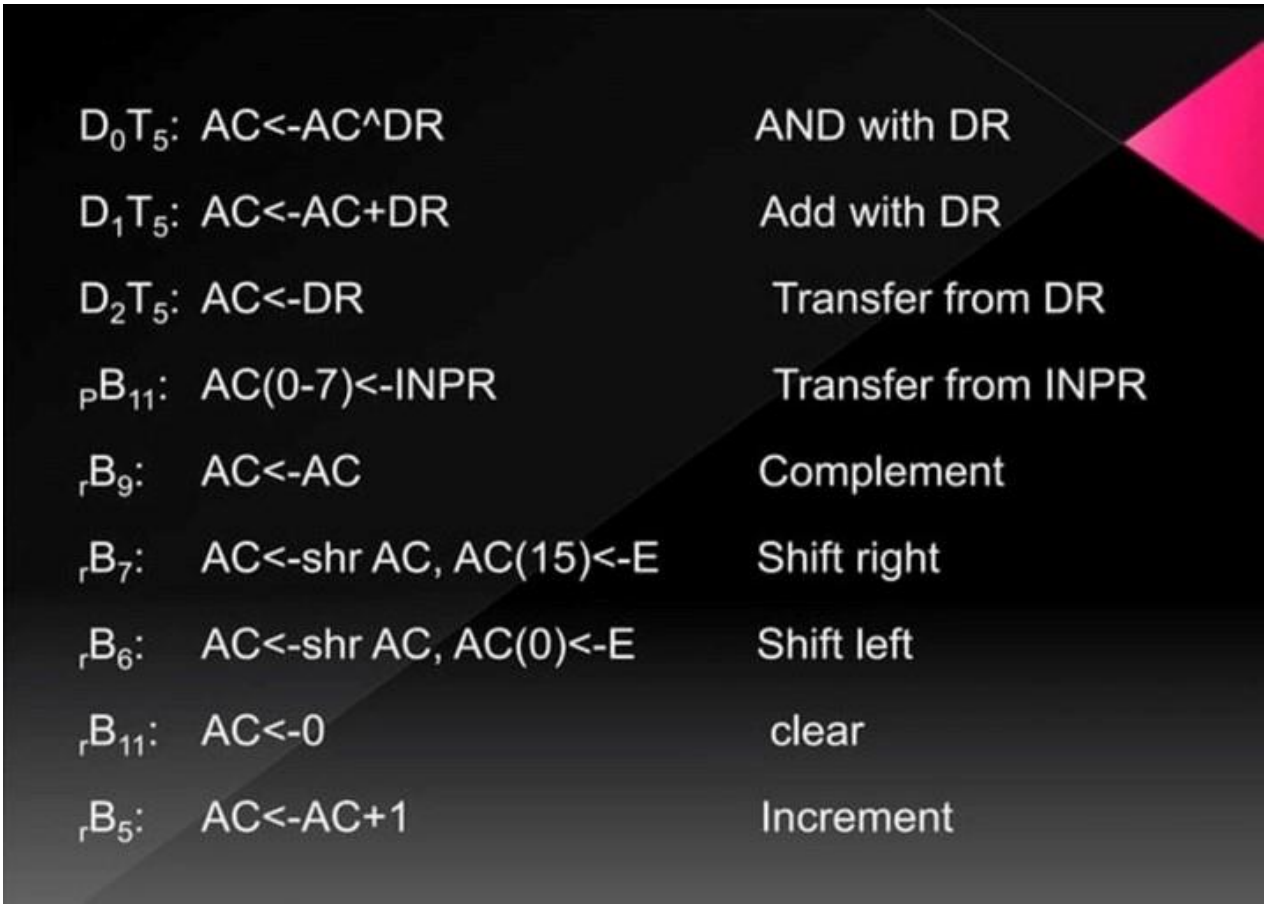
- **Advantages:**

Accumulators can simplify CPU circuitry because the ALU only needs to be wired to the accumulator for its output. They also allow for compact instruction sets as they can hold one of the operands during ALU operations without explicitly referencing it.

Circuit diagram of Accumulator



Logic of Accumulator

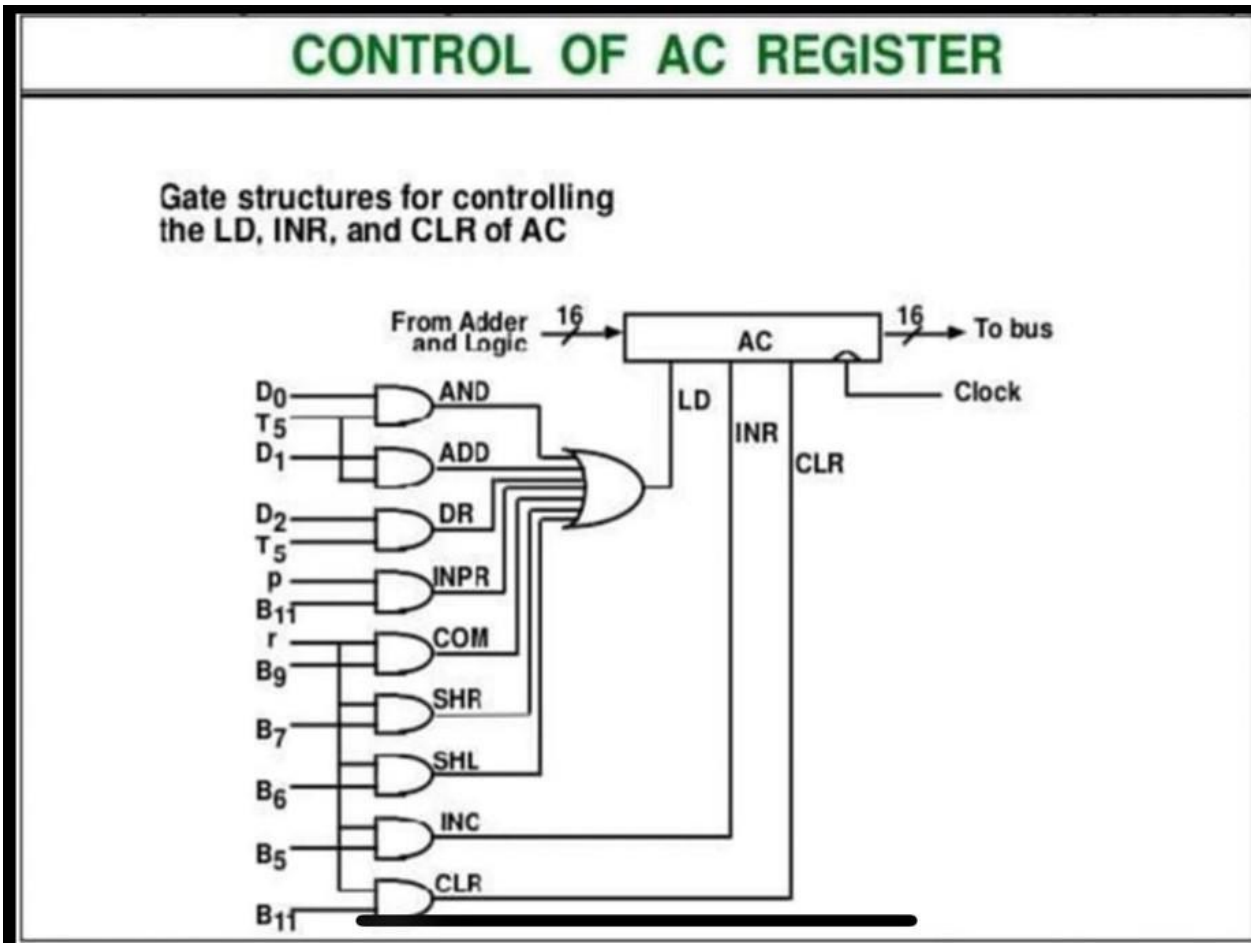


Control of Accumulator Register:

The control function for the clear microoperation is rB11, where r=D-I'T3 and B11=IR(11)

The output of AND gate that generates this control function is connected to the CLR input of the register.

The output of the gate that implements the increment microoperation is connected to the INR input of the register.



Here the other seven microoperations are generated in the adder and logic circuit and are loaded into AC at the proper time.

The outputs of the gates for each control function is marked with symbolic name.

These outputs are used in the design of the adder and logic circuit.

Adder and Logic Circuit

The adder and logic circuit can be subdivided into 16 stages, with each stage corresponding to one bit of Accumulator.

The Load (LD) input is connected to the inputs of the AND gates.

The input is labeled li and the output AC(i). when the LD input is enabled, the 16-bit input li for i= 0,1, 2,15 are transferred to AC (0-15).

The AND operation is achieved by ANDing AC(i) with the corresponding bit in the data register DR(i).

Single Accumulator based CPU organization

The computers, present in the early days of computer history, had accumulator-based CPUs. In this type of CPU organization, the accumulator register is used implicitly for processing all instructions of a program and storing the results into the accumulator. The instruction format that is used by this CPU Organization is the **One address field**. Due to this, the CPU is known as **One Address Machine**.

The main points about Single Accumulator based CPU Organization are:

1. In this CPU Organization, the first ALU operand is always stored into the Accumulator and the second operand is present either in Registers or in the Memory.
2. Accumulator is the default address thus after data manipulation the results are stored into the accumulator.
3. One address instruction is used in this type of organization.

The format of instruction is: Opcode + Address

Opcode indicates the type of operation to be performed. Mainly two types of operation are performed in a single accumulator-based CPU organization:

1. Data transfer operation -
In this type of operation, the data is transferred from a source to a destination.

For ex: LOAD X, STORE Y

Here LOAD is a memory read operation that is data is transferred from memory to accumulator and STORE is a memory write operation that is data is transferred from the accumulator to memory.

2. ALU operation -
In this type of operation, arithmetic operations are performed on the data.

For ex: MULT X

where X is the address of the operand. The MULT instruction in this example performs the operation,

AC <-- AC * M[X]

AC is the Accumulator and M[X] is the memory word located at location X.

This type of CPU organization is first used in **PDP-8 processors** and is used for process control and laboratory applications. It has been totally replaced by the introduction of the new general register-based CPU.

Advantages -

- One of the operands is always held by the accumulator register. This results in short instructions and less memory space.
- The instruction cycle takes less time because it saves time in instruction fetching from memory.

Disadvantages -

- When complex expressions are computed, program size increases due to the usage of many short instructions to execute it. Thus, memory size increases.
- As the number of instructions increases for a program, the execution time increases.

7.3 Status Register and Flag

A status register, also known as a flag register or condition code register, is a hardware register used by a processor to store information about the results of recent operations, particularly arithmetic and logic operations. This information is conveyed through a set of flags, each representing a specific condition or result. These flags provide a quick way to assess the outcome of operations and make decisions based on the results.

The flag register is a 16-bit register in the Intel 8086 microprocessor that contains information about the state of the processor after executing an instruction. It is sometimes referred to as the status register because it contains various status flags that reflect the outcome of the last operation executed by the processor.

The flag register is an important component of the 8086 microprocessors because it is used to determine the behavior of many conditional jump and branch instructions. The various flags in the flag register are set or cleared based on the result of arithmetic, logic, and other instructions executed by the processor.

The flag register is divided into various bit fields, with each bit representing a specific flag. Some of the important flags in the flag register include the carry flag (CF), the zero flag (ZF), the sign flag (SF), the overflow flag (OF), the parity flag (PF), and the auxiliary carry flag (AF). These flags are used by the processor to determine the outcome of conditional jump instructions and other branching instructions.

Purpose Register. Depending upon the value of result after any arithmetic and logical operation the flag bits become set (1) or reset (0).

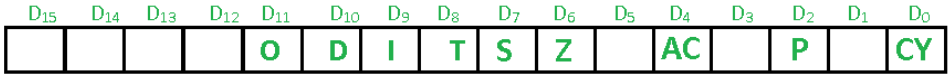


Figure - Format of flag register

There are total 9 flags in 8086 and the flag register is divided into two types: **(a) Status Flags -** There are 6 flag registers in 8086 microprocessor which become set (1) or reset (0) depending upon condition after either 8-bit or 16-bit operation. These flags are conditional/status flags. 5 of these flags are same as in case of 8085 microprocessor and their working is also same as in 8085 microprocessors. The sixth one is the overflow flag. The 6 status flags are:

1. **Sign Flag (S)**
2. **Zero Flag (Z)**
3. **Auxiliary Carry Flag (AC)**
4. **Parity Flag (P)**
5. **Carry Flag (CY)** These first five flags are defined here
6. **Overflow Flag (O) -** This flag will be set (1) if the result of a signed operation is too large to fit in the number of bits available to represent it, otherwise reset (0). After any operation, if D[6] generates any carry and passes to D[7] OR if D[6] does not generates carry but D[7] generates, overflow flag becomes set, i.e., 1. If D[6] and D[7] both generate carry or both do not generate any carry, then overflow flag becomes reset, i.e., 0. **Example:** On adding bytes 100 + 50 (result is not in range -128...127), so overflow flag will set.

MOV AL, 50 (50 is 01010000 which is positive)

MOV BL, 32 (32 is 00110010 which is positive)

ADD AL, BL (82 is 10000010 which is negative)

1. Overflow flag became set as we added 2 +ve numbers and we got a -ve number.

(b) Control Flags - The control flags enable or disable certain operations of the microprocessor. There are 3 control flags in 8086 microprocessor and these are:

1. **Directional Flag (D) -** This flag is specifically used in string instructions. If directional flag is set (1), then access the string data from higher memory location towards lower memory location. If directional flag is reset (0), then access the string data from lower memory location towards higher memory location.

2. **Interrupt Flag (I) -** This flag is for interrupts. If interrupt flag is set (1), the microprocessor will recognize interrupt requests from the peripherals. If interrupt flag is reset (0), the microprocessor will not recognize any interrupt requests and will ignore them.

3. **Trap Flag (T) -** This flag is used for on-chip debugging. Setting trap flag puts the microprocessor into single step mode for debugging. In single stepping, the microprocessor executes a instruction and enters into single step ISR. If trap flag is set (1), the CPU automatically generates an internal interrupt after each instruction, allowing a program to be inspected as it executes instruction by instruction. If trap flag is reset (0), no function is performed.

Uses of Flag register in 8086 microprocessors:

The flag register in the 8086 microprocessor has several important uses, including:

1. Conditional branching: The flags in the flag register can be used to control conditional branching in assembly language programming. Conditional jump instructions allow a program to take different paths based on the state of the flags in the flag register.

2. Arithmetic and logic operations: The flag register is used to store the results of arithmetic and logic operations. The flags in the flag register provide information about the outcome of these operations, such as whether a result is negative or zero, or whether there was an overflow or carry.

3. Error detection and handling: The flag register can be used to detect errors and exceptions, such as overflow or divide-by-zero errors. This allows programs to handle these errors gracefully and to take appropriate corrective action.

4. Debugging: The flag register provides a convenient way to access important information about the status of the processor after executing an instruction. This information can be used to debug programs and to optimize performance.

5. Optimization: The flag register can be used to optimize the performance of assembly language programs by avoiding unnecessary instructions or reducing the number of conditional jumps required.

Advantages:

The flag register in the 8086 microprocessor provides several advantages, including:

1. Efficient conditional branching: The flag register enables efficient conditional branching in assembly language programming. Programmers can use conditional jump instructions to make decisions based on the state of the flags in the flag register, allowing for more efficient and optimized code.

2. Improved arithmetic and logic operations: The flag register is used to store the results of arithmetic and logic operations, allowing for more complex calculations to be performed efficiently. The various flags in the flag register provide information about the outcome of these operations, such as whether a result is negative or zero, or whether there was an overflow or carry.

3. Easy access to processor status information: The flag register provides a convenient way to access important information about the status of the processor after executing an instruction. This information can be used to debug programs and to optimize performance.

4. Improved error handling: The flag register can be used to detect errors and exceptions, such as overflow or divide-by-zero errors. This allows programs to handle these errors gracefully and to take appropriate corrective action.

Dis-advantages:

There are not many disadvantages of the flag register in the 8086 microprocessors, but some potential drawbacks include:

1. Limited number of flags: The 8086-flag register has a limited number of flags, which can make it difficult to handle complex calculations or to detect certain types of errors or exceptions.

2. Limited precision: The flags in the flag register are often limited in precision, which can lead to inaccuracies or errors in certain types of calculations or operations.

3. Difficulty in understanding and using: The flag register can be difficult for beginners to understand and use effectively, which can lead to mistakes or errors in programming.

4. Overhead in execution time: Accessing the flag register can add overhead to program execution time, which can impact performance in certain types of applications.

7.4 Central Processing Unit (CPU)

The **Central Processing Unit (CPU)** is like the brain of a computer. It's the part that does most of the thinking, calculating, and decision-making to make your computer work. Whether you're playing a game, typing a school assignment, or watching a video, the CPU is busy handling all the instructions to get the job done.

The CPU is usually placed in a special slot called a **socket** on the computer's **motherboard**, which is like the main circuit board that connects all the parts of a computer. The CPU handles tasks like:

- Doing math calculations (like adding or multiplying numbers).
- Running apps or games.
- Helping the keyboard, mouse, and screen work together.
- Storing and retrieving information during tasks.

Without a CPU, a computer wouldn't know what to do.

Why CPU is Important in Computing

The CPU is super important because it handles every task your computer does. Without it, your computer would just be a fancy box! A fast CPU means your games run smoothly, your apps open quickly, and your homework gets done faster.

History of CPU

The story of the CPU started long ago and has some exciting milestones that changed how computers work. Here's a simple timeline for students:

- **1823**: A scientist named Baron Jons Jakob Berzelius discovered **silicon**, a material still used to make CPUs today.
- **1947**: Scientists John Bardeen, Walter Brattain, and William Shockley invented the **transistor**, a tiny switch that helped make modern CPUs possible.
- **1958**: Engineers Jack Kilby and Robert Noyce created the **integrated circuit**, which combined many transistors into a single chip.
- **1971**: Intel released the **Intel 4004**, the first-ever microprocessor (a CPU on a single chip), starting the era of personal computers.
- **1979**: Motorola introduced the **Motorola 68000**, a powerful CPU used in early computers and gaming consoles.
- **1999**: Intel launched the **Celeron** processors, making computers faster and more affordable.
- **2005**: AMD introduced the first **dual-core processor**, allowing CPUs to handle multiple tasks at once.
- **2009**: Intel released the **Core i5**, a four-core processor that made computers even faster.
- **2017-2018**: Intel introduced the **Core i9**, one of the most powerful CPUs for desktops and laptops.

Each step made CPUs smaller, faster, and more powerful, helping computers do more amazing things!

Components of CPU

The components of a CPU include the ALU (Arithmetic Logic Unit), CU (Control Unit), registers, cache, and clock.

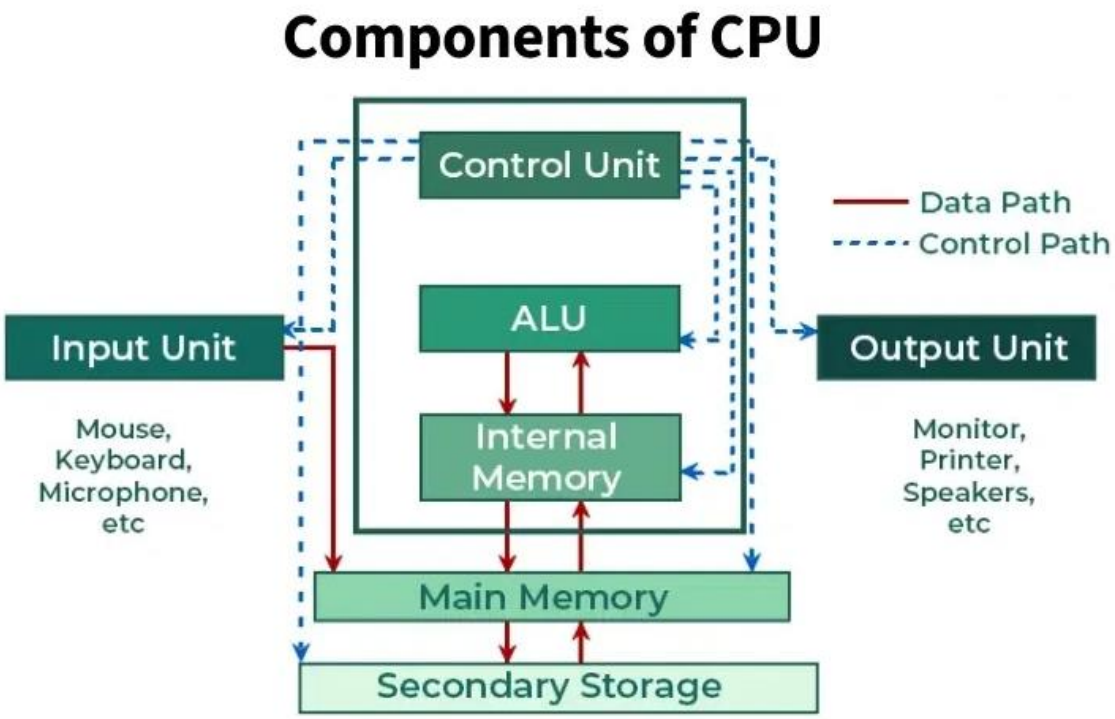


Fig: Components of CPU

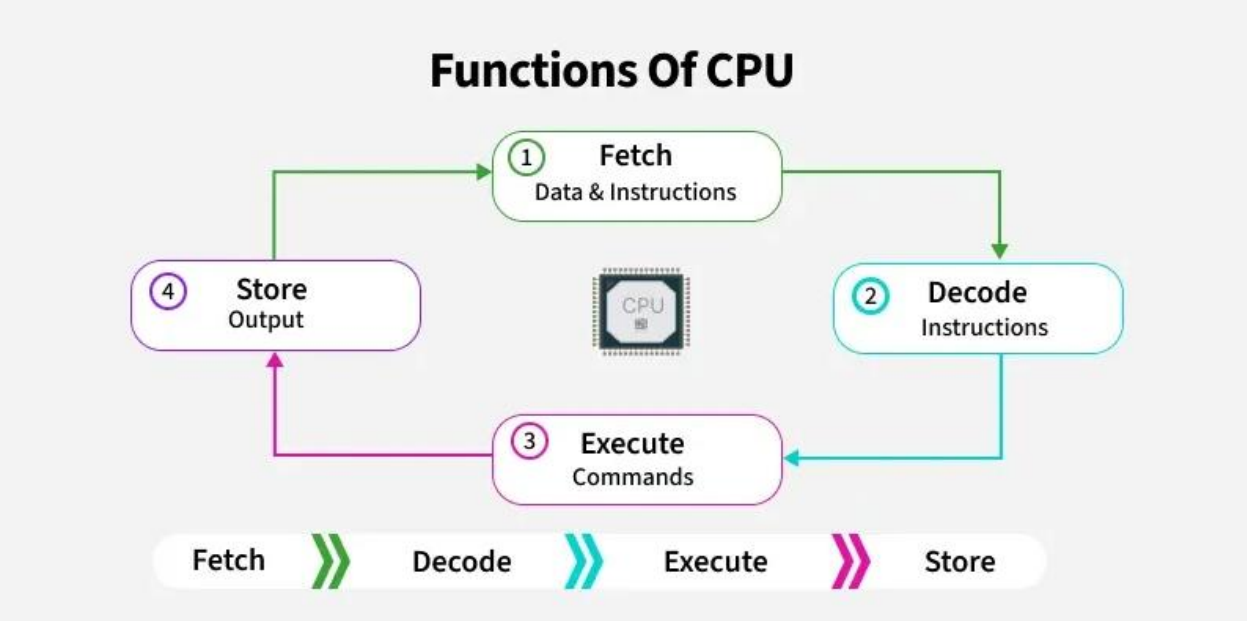
The red lines show how data moves between the parts, while the blue lines show how the CPU sends control signals to manage everything.

- **Control Unit (CU):** It controls the CPU's operations by reading and following instructions. It also manages the flow of data inside the CPU.
- **ALU (Arithmetic Logic Unit):** It does all the math and logic calculations, like addition, subtraction, and comparisons (such as checking if two numbers are equal).
- **Input Unit:** This part gets data from devices like a keyboard, mouse, or microphone, and sends it to the CPU to be processed.
- **Output Unit:** After the CPU processes the data, the output unit sends it to devices like a monitor, printer, or speakers so the user can see or hear the result.
- **Internal Memory:** This store temporary data and instructions that the CPU needs while it's working, like the registers and cache memory.

- **Main Memory:** It holds the data and instructions the CPU is currently using, often in RAM (Random Access Memory).
- **Secondary Storage:** This is where data and programs are stored when they're not being used right away, like on hard drives or SSDs.

Functions of the CPU

The CPU's main job is to process instructions from programs. It does this through a process called the **Fetch-Decode-Execute-Store cycle**:



This cycle happens billions of times a second, letting the CPU handle tons of tasks super-fast

- **Fetch:** the first CPU gets the instruction. That means binary numbers that are passed from RAM to CPU.
- **Decode:** When the instruction is entered into the CPU, it needs to decode the instructions. with the help of ALU (Arithmetic Logic Unit), the process of decoding begins.
- **Execute:** After the decode step the instructions are ready to execute.
- **Store:** After the execute step the instructions are ready to store in the memory.

Types of CPUs

CPUs come in different types, depending on how many **cores** they have. A core is like a mini-CPU inside the main CPU, and more cores mean the CPU can do more tasks at once. Here are the main types:

- **Single-Core CPU**: The oldest type, used in the 1970s. It can only handle one task at a time, so it's slow for modern apps like games or web browsers.
- **Dual-Core CPU**: Has two cores, so it can handle two tasks at once. It's faster and better for multitasking, like listening to music while doing homework.
- **Quad-Core CPU**: Has four cores, making it great for heavy tasks like video editing or playing modern games. It's very fast and common in today's computers.

Why is the CPU Called the Brain of the Computer?

The CPU earns its nickname as the "brain" because it's responsible for thinking through and executing every task in a computer. Just like your brain processes information to make decisions, the CPU processes instructions to make your computer do what you want. Without a CPU, a computer would just be a lifeless box of parts.

How Does the CPU Make Computers Faster?

Modern CPUs are designed to be super-efficient. Here are a few ways they speed things up:

- **Multiple Cores**: Many CPUs have multiple cores, which are like mini-CPUs that can work on different tasks at the same time. It's like having several chefs in the kitchen instead of one.
- **Faster Clocks**: The clock speed (measured in GHz, like 3.5 GHz) determines how many instructions the CPU can handle per second.
- **Bigger Cache**: More cache means the CPU can store more data close by, reducing wait times.
- **Pipelining**: This lets the CPU start working on the next instruction before finishing the current one, like a factory line.

Advantages of CPUs

- **Versatile**: CPUs can handle all kinds of tasks, from simple math to running complex games.
- **Fast**: Modern CPUs process billions of instructions per second.
- **Multi-tasking**: Multi-core CPUs let you run many programs at once, like watching a video while chatting with friends.
- **Compatible**: CPUs work with tons of software, so you can use the same CPU for different apps.

Disadvantages of CPUs

- **Heat**: CPUs get hot when working hard, so computers need fans or cooling systems to stay safe.
- **Power Use**: Powerful CPUs use a lot of electricity, which can raise power bills.
- **Cost**: High-performance CPUs, like Intel Core i9, can be expensive.
- **Not Perfect for All Tasks**: For tasks like graphics or video editing, specialized chips like **GPUs (Graphics Processing Units)** are better than CPUs.

Modern Applications

CPUs are everywhere, not just in computers:

CPU in Personal Computers: In your laptop or desktop, the CPU runs your games, apps, and homework programs, making sure everything works smoothly.

Role in Mobile Devices: Your phone or tablet has a CPU too! It's smaller and uses less power but still handles calls, apps, and videos.

Use in Servers and Data Centers: In big data centers, CPUs power websites like YouTube and Google, processing millions of requests every second.