

# Registers



**EXPLANATION**

# What is register??



- Processor operations mostly involve processing data. This data can be stored in memory and accessed from thereon. However, reading data from and storing data into memory slows down the processor, as it involves complicated processes of sending the data request across the control bus and into the memory storage unit and getting the data through the same channel.



- To speed up the processor operations, the processor includes some internal memory storage locations, called **registers**.
- The registers store data elements for processing without having to access the memory. A limited number of registers are built into the processor chip.

# Processor Registers??



- There are ten 32-bit and six 16-bit processor registers in IA-32 architecture. The registers are grouped into three categories –
- General registers,
- Control registers, and
- Segment registers.

The general registers are further divided into the following groups –

- Data registers,
- Pointer registers, and
- Index registers.

# Data Registers

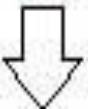


- Four 32-bit data registers are used for arithmetic, logical, and other operations. These 32-bit registers can be used in three ways –
- As complete 32-bit data registers: EAX, EBX, ECX, EDX.
- Lower halves of the 32-bit registers can be used as four 16-bit data registers: AX, BX, CX and DX.
- Lower and higher halves of the above-mentioned four 16-bit registers can be used as eight 8-bit data registers: AH, AL, BH, BL, CH, CL, DH, and DL.



32-bit registers

16-bit registers

	31	16	15	8	7	0	
 EAX					AH	AL	AX Accumulator
EBX					BH	BL	BX Base
ECX					CH	CL	CX Counter
EDX					DH	DL	DX Data



- Some of these data registers have specific use in arithmetical operations.
- **AX is the primary accumulator**; it is used in input/output and most arithmetic instructions. For example, in multiplication operation, one operand is stored in EAX or AX or AL register according to the size of the operand.
- **BX is known as the base register**, as it could be used in indexed addressing.
- **CX is known as the count register**, as the ECX, CX registers store the loop count in iterative operations.
- **DX is known as the data register**. It is also used in input/output operations. It is also used with AX register along with DX for multiply and divide operations involving large values.
-

# Working of Registers:



- When we provide the system with input, that input is stored in registers, and when the system returns results after processing, those results are also drawn from the registers. so that the CPU can use them to process the data that the user provides.
- Registers are performed based on three operations:
- **Fetch:** The Fetch Operation is used to retrieve user-provided instructions that have been stored in the main memory. Registers are used to fetch these instructions.
- **Decode:** The Decode Operation is used to interpret the Instructions, which means that the CPU will determine which Operation has to be carried out on the Instructions after the Instructions have been decoded.
- **Execute:** The CPU manages the Execute Operation. The results that the CPU generates are then stored in the memory before being presented on the user screen.
- **Types of Registers:**
  - Status and control registers.
  - General-purpose data registers.
  - Special purpose register.





- **Status and Control Register:**
- Status and Control registers report and allow modification of the state of the processor and of the program being executed.
- **General-Purpose Data Registers:**
- General purpose registers are extra registers that are present in the CPU and are utilized anytime data or a memory location is required. These registers are used for storing operands and pointers. These are mainly used for holding the following:
  - Operands for logical and arithmetic operations
  - Operands for address calculation
  - Memory pointers
- There are 3 types of General-purpose data registers they are:



- **Data registers:** Data registers consists of four 32-bit data registers, which are used for arithmetic, logical and other operations. Data registers are again classified into 4 types they are:
- **AX:** This is known as the accumulator register. Its 16 bits are split into two 8-bit registers, AH and AL, allowing it to execute 8-bit instructions as well. In 8086 microprocessors, it is used in the arithmetic, logic, and data transfer instructions. One of the numbers involved in manipulation and division must be in AX or AL.
- **BX:** This is called a Base register. It has 16 bits and is split into two registers with 8 bits each, BH and BL. An address register is the BX register. It typically includes a data pointer for indirect addressing that is based, based indexed, or register-based.
- **CX:** This is known as the Count register. Its 16 bits are split into two 8-bit registers, CH and CL, allowing it to execute 8-bit instructions as well. This acts as a counter for loops. It facilitates the development of program loops. Shift/rotate instructions and string manipulation both allow the use of the count register as a counter.
- **DX:** This is known as the Data register. Its 16 bits are split into two 8-bit registers, DH and DL so that it can execute 8-bit instructions as well. In I/O operations, the data register can be used as a port number. It is also applied to division and multiplication.



- **Pointer registers:** The pointer registers consist of 16-bit left sections (SP, and BP) and 32-bit ESP and EBP registers.
- **SP:** This is known as a Stack pointer used to point the program stack. For accessing the stack segment, it works with SS. It has a 16-bit size. It designates the item at the top of the stack. The stack pointer will be (FFFE)H if the stack is empty. The stack segment is relative to its offset address.
- **BP:** This is known as the Base pointer used to point data in the stack segments. We can utilize BP to access data in the other segments, unlike SP. It has a 16-bit size. It mostly serves as a way to access parameters given via the stack. The stack segment is relative to its offset address.



- **Index registers:** The 16-bit rightmost bits of the 32-bit ESI and EDI index registers. SI and DI are sometimes employed in addition and sometimes in subtraction as well as for indexed addressing.
- **SI:** This source index register is used to identify memory addresses in the data segment that DS is addressing. Therefore, it is simple to access successive memory locations when we increment the contents of SI. It has a 16-bit size. Relative to the data segment, it has an offset.
- **DI:** The function of this destination index register is identical to that of SI. String operations are a subclass of instructions that employ DI to access the memory addresses specified by ES. It is generally used as a Destination index for string operations.

# Special Purpose Registers:



- To store machine state data and change state configuration, special purpose registers are employed. In other words, it is also defined as the CPU has a number of registers that are used to carry out instruction execution these registers are called special purpose registers. Special purpose registers are of 8 types they are cs, ds, ss, es, fs, and gs registers come under segment registers. These registers hold up to six segment selectors.
- **CS (Code Segment register):** A 16-bit register called a code segment (CS) holds the address of a 64 KB section together with CPU instructions. All accesses to instructions referred to by an instruction pointer (IP) register are made by the CPU using the CS segment. Direct changes to CS registration are not possible. When using the far jump, far call, and far return instructions, the CS register is automatically updated.
- **DS (Data Segment register):** A 64KB segment of program data is addressed using a 16-bit register called the data segment. The processor by default believes that the data segment contains all information referred to by the general registers (AX, BX, CX, and DX) and index registers (SI, DI). POP and LDS commands can be used to directly alter the DS register.



- **SS (Stack Segment register):** A 16-bit register called a stack segment holds the address of a 64KB segment with a software stack. The CPU by default believes that the stack segment contains all information referred to by the stack pointer (SP) and base pointer (BP) registers. POP instruction allows for direct modification of the SS register.
- **ES (Extra Segment register):** A 16-bit register called extra segment holds the address of a 64KB segment, typically holding program data. In string manipulation instructions, the CPU defaults to assuming that the DI register refers to the ES segment. POP and LES commands can be used to directly update the ES register.
- **FS (File Segment register):** FS registers don't have a purpose that is predetermined by the CPU; instead, the OS that runs them gives them a purpose. On Windows processes, FS is used to point to the thread information block (TIB).



- **GS (Graphics Segment register):** The GS register is used in Windows 64-bit to point to operating system-defined structures. OS kernels frequently use GS to access thread-specific memory. The GS register is employed by Windows to control thread-specific memory. In order to access CPU-specific memory, the Linux kernel employs GS. A pointer to a thread local storage, or TLS, is frequently used as GS.
- **IP (Instruction Pointer register):** The registers CS and IP are used by the 8086 to access instructions. The segment number of the following instruction is stored in the CS register, while the offset is stored in the IP register. Every time an instruction is executed, IP is modified to point to the upcoming instruction. The IP cannot be directly modified by an instruction, unlike other registers; an instruction may not have the IP as its operand.



- **Flag register:** The status register for an x86 CPU houses its current state, and it is called the FLAGS register. The flag bits' size and significance vary depending on the architecture. It often includes information about current CPU operation limitations as well as the outcome of mathematical operations. Some of these limitations might forbid the execution of a particular class of “privileged” instructions and stop some interrupts from triggering. Other status flags may override memory mapping and specify the response the CPU should have in the event of an arithmetic overrun.