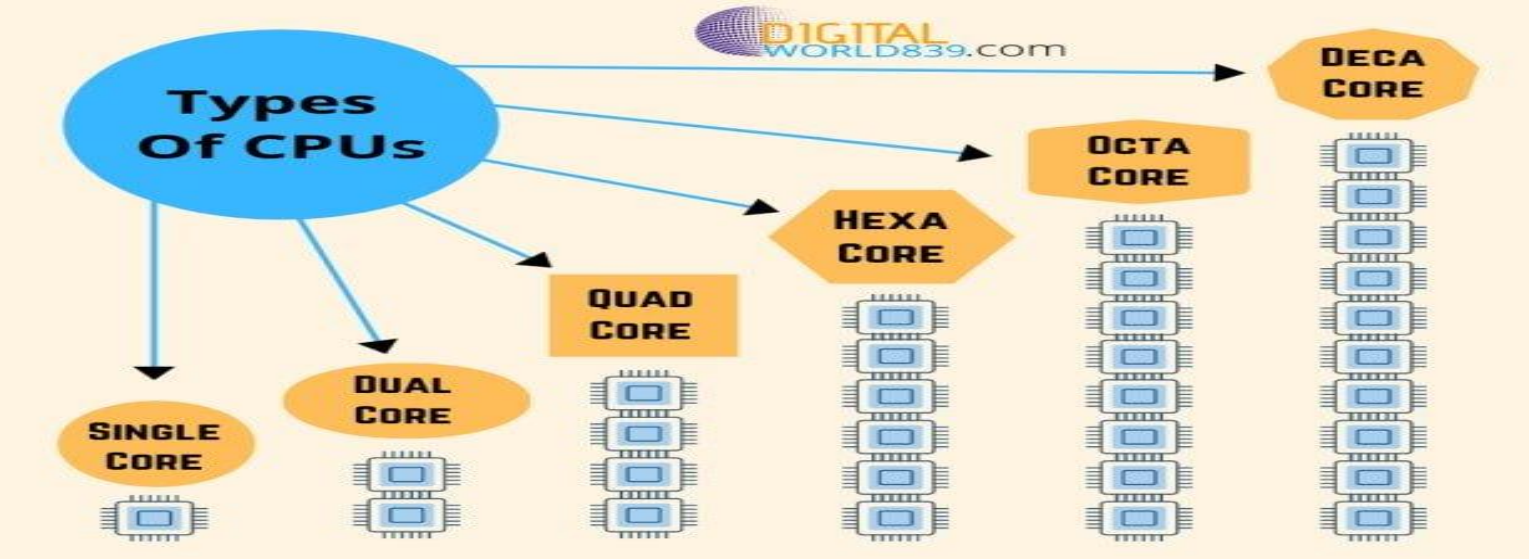
Types Of CPU

the Central Processing Unit (CPU) is a vital component being the **brain of a computer device** that handles all the instructions and arithmetic or logical calculations.



1. **Single Core CPU**

The single-core CPU is the oldest type available and it was introduced in the early history of computing this was the only one that could be used in computers. It was possible to carry out certain tasks, but it was a bit slow.

Single-core CPUs can perform only one operation at a single point in time, that's why they are **not a very good choice** at multitasking. This meant, at the time, a notable decrease in the performance whenever the user executes more than one application in them.

Although only one operation could be started at a time, another could be activated before the first was finished, but with each new operation, the equipment was slower. Also, the clock-speed of these processors was very low. And the performance, power, and speed of any CPU depend on its clock-speed.

Characteristics of a single-core CPU are very few compared to the latest cores for system operations. However, this was the beginning of a great revolution in computing.

Intel 400 is invented in the year 1971 4740 KHz and 640 bytes of RAM.

2. **Dual Core CPU**

A dual-Core CPU is actually a single CPU, but it is comprised of two cores on the chip that's why this CPU functions like two CPUs.The clock speed of this processor also almost doubles make it possible to handle multitasking more comfortably and much better than processors that have only one core.Dual-core CPU can execute more than one operation at the same time and at a higher speed. This means that only one operation could be started at a time, the second operation would be started a little before the first one was over. In this way, performance capacity and speed reduce a bit.The dual-core CPU processor has to switch back and forth to different sets of data streams if more operation is to be done.In order to use the full potential of Dual Core, it is necessary to have a special code written in the operating system called **Simultaneous Multi-Thread Technology** (SMT) so as to run programs more efficiently. Dual-core CPUs are faster than single-core CPUs, but not as fast as the four-core CPUs that have replaced them.Pentium D in 2005 It has 2.80 GHz and2 MB of Cache.

3. Quad Core CPU

The term Quad indicates the 4 cores present in the processor characterized by taking into account speed and power consumption, they also have great capabilities for multitasking in comparison to the single and dual-core CPUs.

It has **four cores working simultaneously**, capable of performing any task for which it has been intended, at high speed, and above all, performing several tasks at the same time. It is the immediate antecedent of what would later be the 8 or10 core CPUs.

The Quad-Core CPU divides the workload into four cores. This does not mean that it will multiple the speed four times for a single application. In fact, it will only accelerate multitasking work that means executing multiple tasks at the same time efficiently.

When you have to do a lot of work on a computer, like for Video Editing, Gaming, Designing, etc. then you use these types of CPUs with executing the SMT code will boost the speed more.

3 GHz and 2 MB of Cache. Athlon II X4 in 2009

4. Hexa Core CPU

The Hexa Core CPU has six cores on the chip to execute the task and transmit all data more rapidly as compared to quad-core and dual-core processors. Therefore, its working speed, clock speed, performance will be faster and will also be more suitable for multitasking.

The first Hexa core CPU was launched in Intel core i7 processor in 2010.

Intel core i7-980X 3.60 GHz and

12 MB of Cache. In 2010

5. Octa-Core CPU

The octa core CPU as its name suggests another multi-core processor are consists of 8 cores, which takes the performance and speed of multitasking to a different level than the above CPU processors for any computer device.Numerous octa-core CPU processors comprise a **dual set of quad-core processors** that separates various activities between the different sorts that's why it is also known as a "dual quad-core processor".Commonly, these types of CPUs are utilized to execute advanced tasks on computers and other devices.

Intel Core i7-5960X 3.50 GHz and 20 MB of Cache. Year **2014**.

. Deca Core CPU

The dual-core is built with two cores, four cores are built-in quad-core, six cores are built-in hexa-core, eight cores are built-in octa-core, where Deca core comes with 10 cores where the Deca processors are developed with eight autonomous cores to execute a task will be **much efficient and rapid** than all the core processors until now.

It is the most powerful and successful CPU than the above CPU processors for any computer device. Now you would have a clear idea that on increasing the number of the core of the processors, the performance and its capacity to perform multitasking to increments.

Xeon Silver 4114T in 2017 3 GHz and 13.75 MB of Cache

The above listed **different types of CPUs** are the significant processors in the computer world for increasing the speed and performance of our computing devices from mobile phones to laptops and desktops. And now the article on different types of central processing units ends.

What is a programming language?

A programming language defines a set of instructions that are compiled together to perform a specific task by the CPU (Central Processing Unit). The programming language mainly refers to high-level languages such as C, C++, Pascal, Ada, COBOL, etc.

Each programming language contains a unique set of keywords and syntax, which are used to create a set of instructions. Thousands of programming languages have been developed till now, but each language has its specific purpose. These languages vary in the level of abstraction they provide from the hardware. Some programming languages provide less or no abstraction while some provide higher abstraction. Based on the levels of abstraction, they can be classified into two categories:

Low-level language

The low-level language is a programming language that provides no abstraction from the hardware, and it is represented in 0 or 1 forms, which are the machine instructions. The languages that come under this category are the Machine level language and Assembly language.

Machine-level language

The machine-level language is a language that consists of a set of instructions that are in the binary form 0 or 1. As we know that computers can understand only machine instructions, which are in binary digits, i.e., 0 and 1, so the instructions given to the computer can be only in binary codes. Creating a program in a machine-level language is a very difficult task as it is not easy for the programmers to write the program in machine instructions. It is error-prone as it is not easy to understand, and its maintenance is also very high. A machine-level language is not portable as each computer has its machine instructions, so if we write a program in one computer will no longer be valid in another computer.

The different processor architectures use different machine codes, for example, a PowerPC processor contains RISC architecture, which requires different code than intel x86 processor, which has a CISC architecture.

Assembly Language

The assembly language contains some human-readable commands such as mov, add, sub, etc. The problems which we were facing in machine-level language are reduced to some extent by using an extended form of machine-level language known as assembly language. Since assembly language instructions are written in English words like mov, add, sub, so it is easier to write and understand.

As we know that computers can only understand the machine-level instructions, so we require a translator that converts the assembly code into machine code. The translator used for translating the code is known as an assembler.

The assembly language code is not portable because the data is stored in computer registers, and the computer has to know the different sets of registers.

The assembly code is not faster than machine code because the assembly language comes above the machine language in the hierarchy, so it means that assembly language has some abstraction from the hardware while machine language has zero abstraction.

Machine-level language vs Assembly language:-

The machine-level language comes at the lowest level in the hierarchy, so it has zero abstraction level from the hardware.

The assembly language comes above the machine language means that it has less abstraction level from the hardware.

It cannot be easily understood by humans.

It is easy to read, write, and maintain.

The machine-level language is written in binary digits, i.e., 0 and 1.

The assembly language is written in simple English language, so it is easily understandable by the users.

It does not require any translator as the machine code is directly executed by the computer.

In assembly language, the assembler is used to convert the assembly code into machine code.

It is a first-generation programming language.

It is a second-generation programming language.

High-Level Language

The high-level language is a programming language that allows a programmer to write the programs which are independent of a particular type of computer. The high-level languages are considered as high-level because they are closer to human languages than machine-level languages.

When writing a program in a high-level language, then the whole attention needs to be paid to the logic of the problem.

A compiler is required to translate a high-level language into a low-level language.

Advantages of a high-level language

- The high-level language is easy to read, write, and maintain as it is written in English like words.
- The high-level languages are designed to overcome the limitation of low-level language, i.e., portability. The high-level language is portable; i.e., these languages are machine-independent.

Low-level language Versus High-level language

It is a machine-friendly language, i.e., the computer understands the machine language, which is represented in 0 or 1.

It is a user-friendly language as this language is written in simple English words, which can be easily understood by humans.

The low-level language takes more time to execute.

It executes at a faster pace.

It requires the assembler to convert the assembly code into machine code.

It requires the compiler to convert the high-level language instructions into machine code.

The machine code cannot run on all machines, so it is not a portable language.

The high-level code can run all the platforms, so it is a portable language.

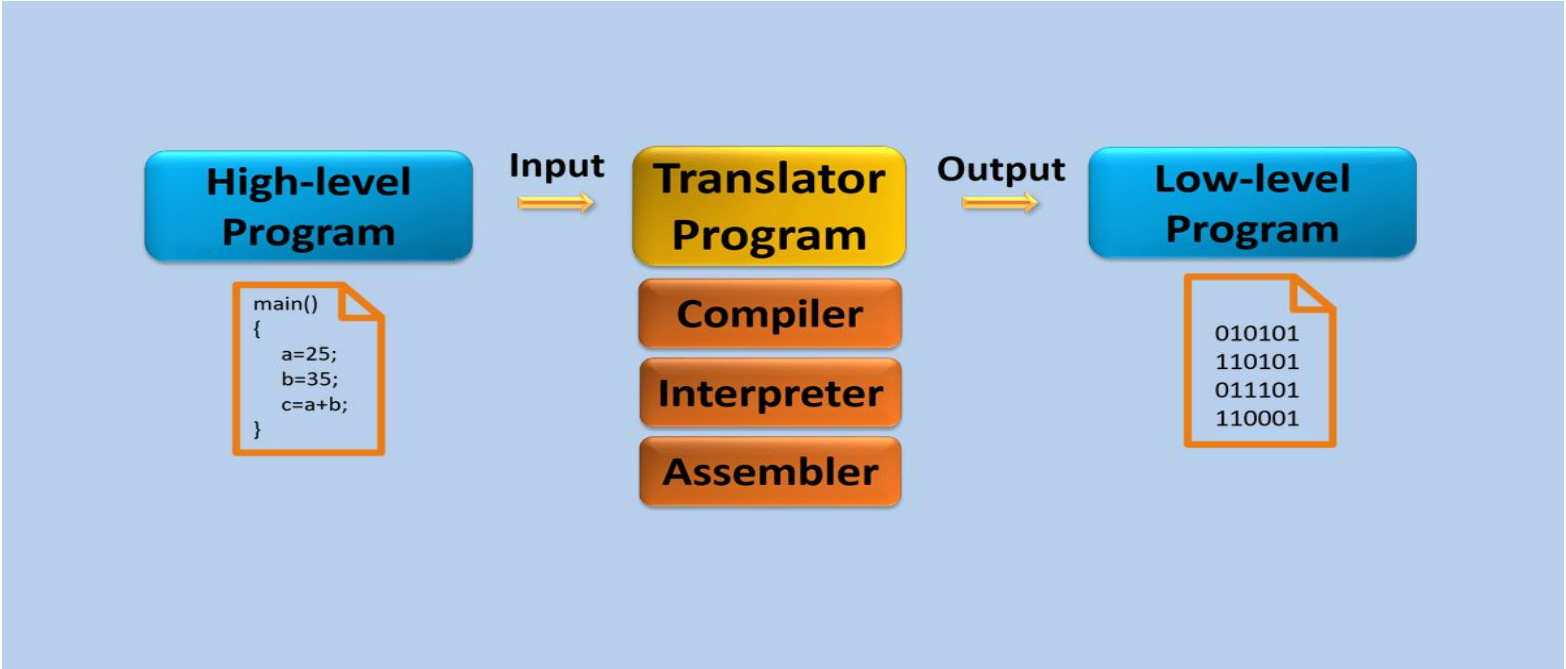
It is memory efficient.

It is less memory efficient.

Debugging and maintenance are not easier in a low-level language.

Debugging and maintenance are easier in a high-level language.

Translator:-


High-level Program
main()
{
a=25;
b=35;
c=a+b;
}
Input
Translator Program
Compiler
Interpreter
Assembler
Output
Low-level Program
010101
110101
011101
110001

Compiler:

A compiler is a type of translator program that Scans the entire program written in a high-level language and translates it as a whole into machine code.. That is, the source program into an object program.

The compiler completes the task of translation in two steps –

In the first step, the compiler reads the source program and converts into the objects programs. In this step, if there is a syntax error in the source program, the compiler gives the user an error message to correct it. This error message is called Compiled Time Diagnostic Error Message. Once the program is compiled, there is no need to compile it later.

Functions of the Compiler:

1. Link with the corresponding sub-routine.

2. Convert source program statements into object programs or machine language.

3. If there is any syntax error in the program, make a list of errors during translation.

4. Identify the range of main memory.

A compiler is likely to perform many or all of the following operations: pre-processing, lexical analysis, parsing, semantic analysis, conversion of input programs to an intermediate representation, code optimization and code generation.

Advantages of Compiler:

1. The compiler translates the entire program together, the overall execution time is comparatively faster.

2. The program converted through the compiler is completely converted into machine language.

3. Generates output program which can be run independently from the original program.

Disadvantages of Compiler:

1. The compiler displays all the errors of the program at once, so it takes more time to correct the program.

2. Program debugging and testing is slower.

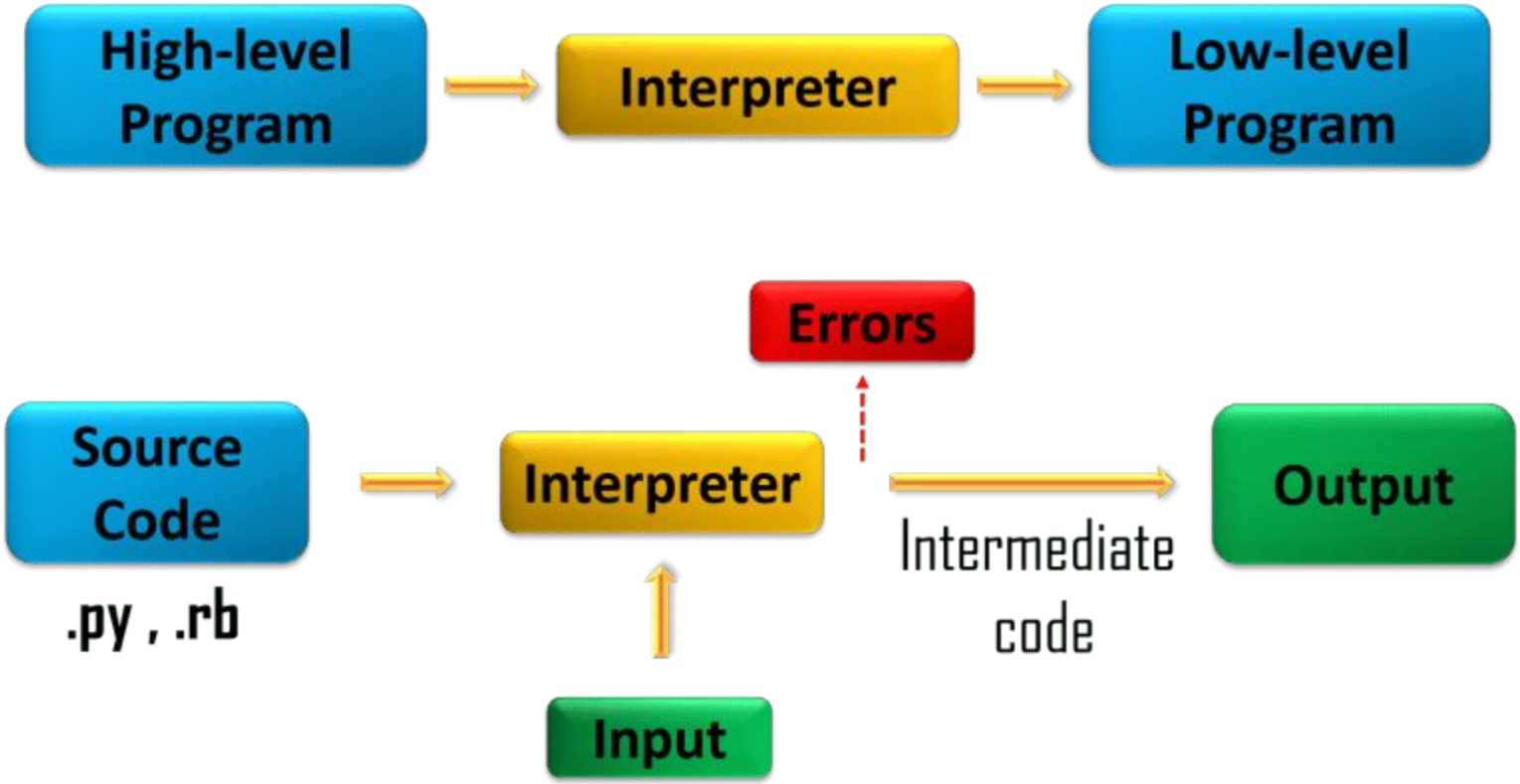
3. As the compiler is a large program, it requires more space in memory to store.

Interpreter:

An interpreter, like a compiler, is a kind of translator program that reads one line of a program written in a high-level language and converts it into mechanical language and executes it, displaying instant results.

The difference with the compiler is that the compiler converts the entire source program into an object program and provides the latest results but the interpreter converts the source program into a line-by-line object program and displays instant results.

Interpreter translates line by line and reports the error once it encountered during the translation process. It directly executes the operations specified in the source program when the input is given by the user. It gives better error diagnostics than a compiler.


High-level Program
Interpreter
Low-level Program
Errors
Source Code
.py , .rb
Interpreter
Intermediate code
Output
Input

Functions of the Interpreter:

1. Link with the corresponding sub-routine.

2. Convert source program statements into object programs or machine language.

3. If there is any syntax error in the program, make a list of errors during translation.

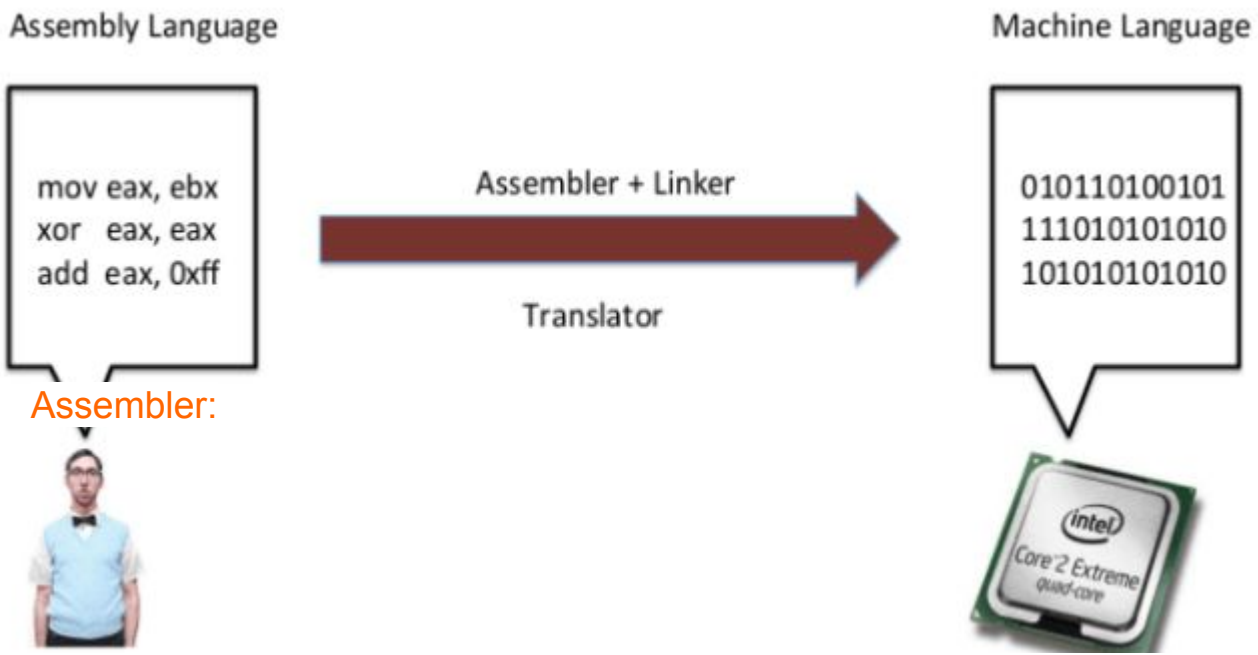
4. Identify the range of main memory.

Advantages of Interpreter:

1. The biggest advantage of the interpreter is that it is user friendly.
2. It is easy to use to correct and change program errors.
3. The smaller the interpreter, the less space it takes up in memory.
4. It is commonly used in small computers.

Disadvantages of Interpreter:

1. Since the interpreter translates the program line-by-line, it takes more time than the compiler to translate.
2. Programs converted through interpreters are not completely converted into machine or object programs.
3. Translation is required every time the program is executed.
4. Do not generate output program. So they evaluate the source program at every time during execution.



Assembler is a translator program which is used to translate the assembly language code into machine language code.

Functions of assembler:

1. Convert mnemonic codes into equivalent machine code.
2. Convert symbolic operands or address to machine addresses.
3. Keep all instructions and data in main memory.
4. If there is any error in program, notify with error message.
5. Informing the program control after correcting all the errors in the program etc.

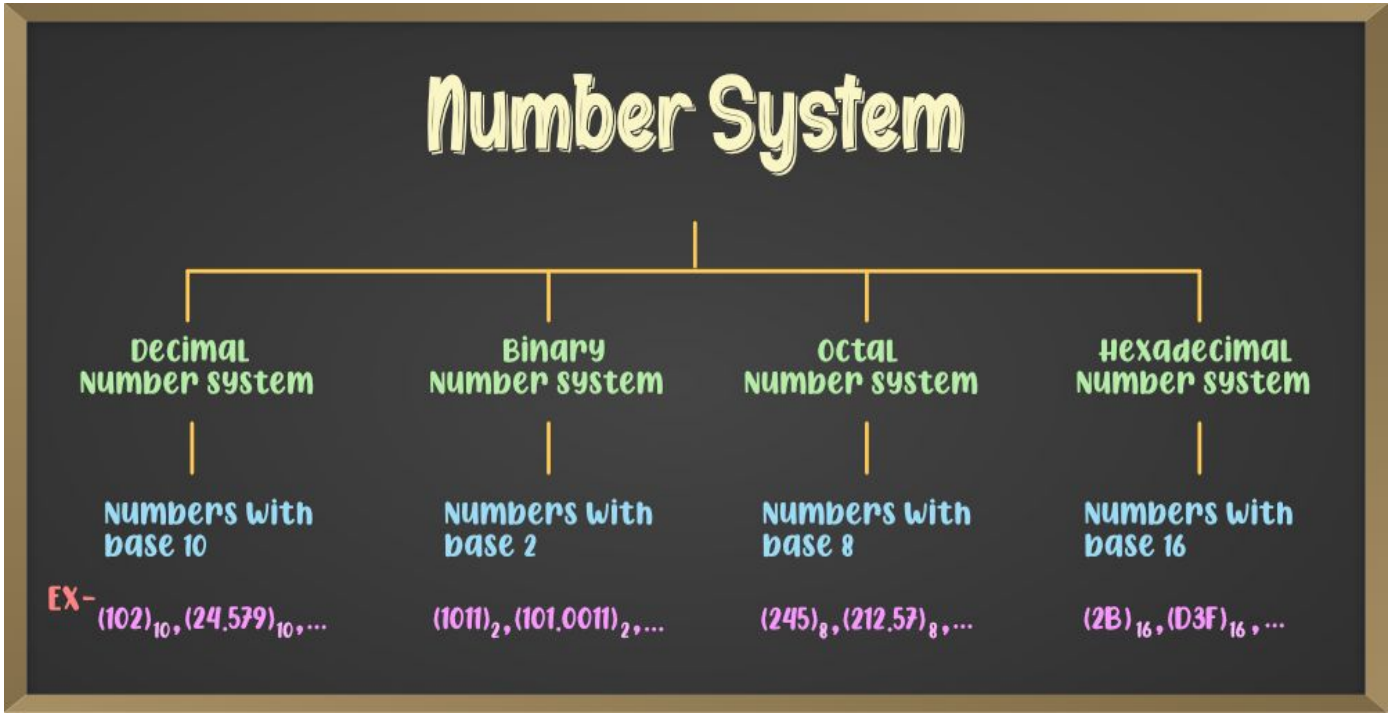
The difference between an interpreter and a compiler is given below:

Compiler	Interpreter
Scans the entire program and translates it as a whole into machine code.	Translates program one statement at a time.
It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.	It takes less amount of time to analyze the source code but the overall execution time is slower.
Generates intermediate object code which further requires linking, hence requires more memory.	No intermediate object code is generated, hence are memory efficient.
Compiler displays all errors and warning at the compilation time. Therefore, you can't run the program without fixing errors.	The interpreter reads a single statement and shows the error if any. You must correct the error to interpret next line.
It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.	Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.
Store machine language as machine code on the disk.	Not saving machine code at all.
Generates output program which can be run independently from the original program.	Do not generate output program. So they evaluate the source program at every time during execution.
Programming language like C, C++ use compilers.	Programming language like Python, Ruby use interpreters.

Types of Number Systems

Based on the base value and the number of allowed digits, number systems are of many types. The four common types of Number System are:

1. **Decimal Number System**
2. **Binary Number System**
3. **Octal Number System**
4. **Hexadecimal Number System**


Number System
Decimal Number system
Binary Number system
Octal Number system
Hexadecimal Number system
Numbers with base 10
Numbers with base 2
Numbers with base 8
Numbers with base 16
EX- $(102)_{10}, (24.579)_{10}, \ldots$
$(1011)_2, (101.0011)_2, \ldots$
$(245)_8, (212.57)_8, \ldots$
$(2B)_{16}, (D3F)_{16}, \ldots$

1. Decimal Number System

Number system with base value 10 is termed as Decimal number system. It uses 10 digits i.e. 0-9 for the creation of numbers. Here, each digit in the number is at a specific place with place value a product of different powers of 10. Here, the place value is termed from right to left as first place value called units, second to the left as Tens, so on Hundreds, Thousands, etc. Here, units has the place value as 10_0, tens has the place value as 10_1, hundreds as 10_2, thousands as 10_3, and so on.

For example: 10285 has place values as

$(1 \times 10_4) + (0 \times 10_3) + (2 \times 10_2) + (8 \times 10_1) + (5 \times 10_0)$

$1 \times 10000 + 0 \times 1000 + 2 \times 100 + 8 \times 10 + 5 \times 1$

$10000 + 0 + 200 + 80 + 5$

10285

2. Binary Number System

Number System with base value 2 is termed as Binary number system. It uses 2 digits i.e. 0 and 1 for the creation of numbers. The numbers formed using these two digits are termed as Binary Numbers. Binary number system is very useful in electronic devices and computer systems because it can be easily performed using just two states ON and OFF i.e. 0 and 1.

Decimal Numbers 0-9 are represented in binary as: 0, 1, 10, 11, 100, 101, 110, 111, 1000, and 1001

Examples:

14 can be written as 1110

19 can be written as 10011

50 can be written as 110010

3. Octal Number System

Octal Number System is one in which the base value is 8. It uses 8 digits i.e. 0-7 for creation of Octal Numbers. Octal Numbers can be converted to Decimal value by multiplying each digit with the place value and then adding the result. Here the place values are 8_0, 8_1, and 8_2. Octal Numbers are useful for the representation of UTF8 Numbers.

Example:

$(135)_{10}$ can be written as $(207)_8$

$(215)_{10}$ can be written as $(327)_8$

4. Hexadecimal Number System

Number System with base value 16 is termed as Hexadecimal Number System. It uses 16 digits for the creation of its numbers. Digits from 0-9 are taken like the digits in the decimal number system but the digits from 10-15 are represented as A-F i.e. 10 is represented as A, 11 as B, 12 as C, 13 as D, 14 as E, and 15 as F. Hexadecimal Numbers are useful for handling memory address locations.

Examples:

$(255)_{10}$ can be written as $(FF)_{16}$

$(1096)_{10}$ can be written as $(448)_{16}$

$(4090)_{10}$ can be written as $(FFA)_{16}$

Binary to Decimal Conversion

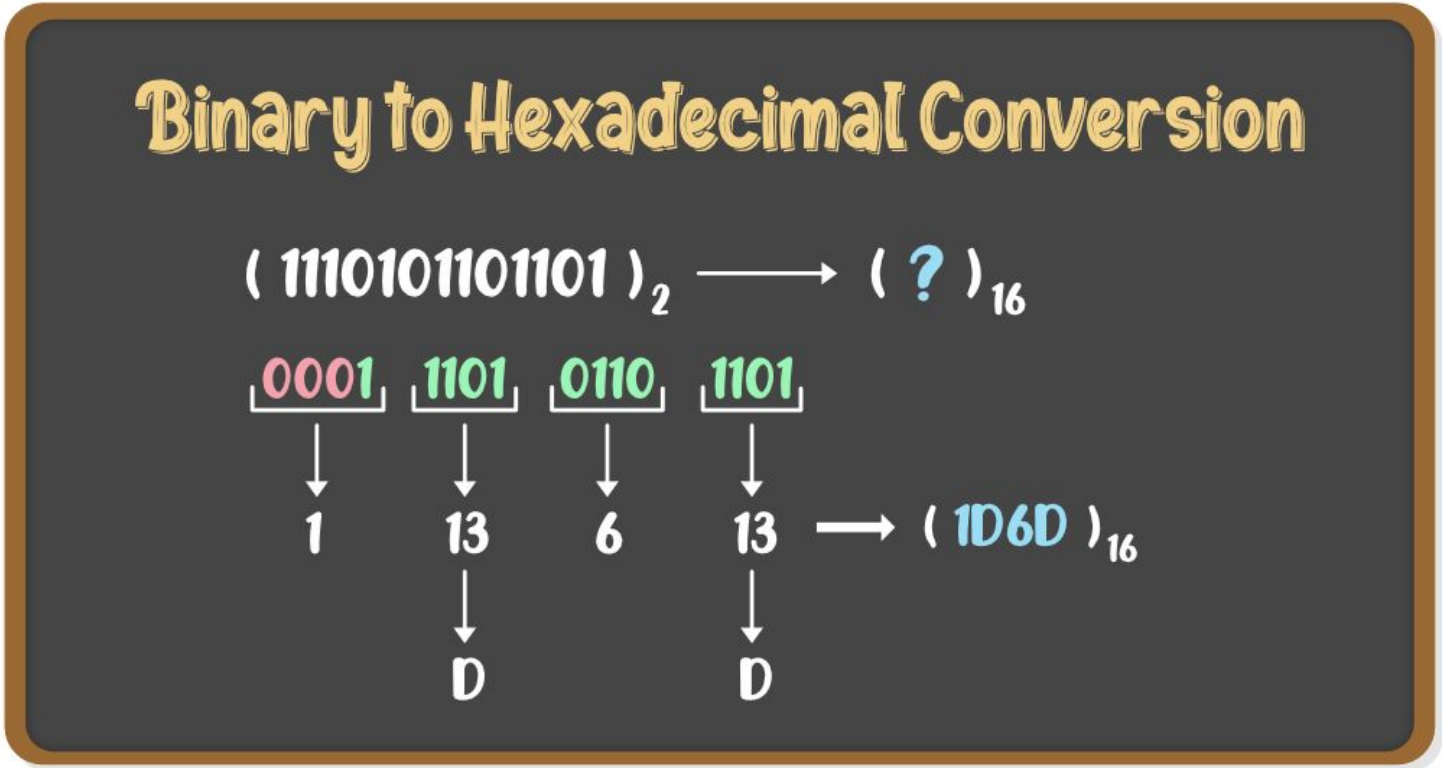
$$(11101011)_2 \longrightarrow (?)_{10}$$

$$1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$128 + 64 + 32 + 0 + 8 + 0 + 2 + 1$$

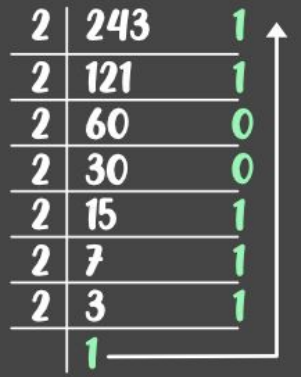
$$(235)_{10}$$

Binary to Hexadecimal Conversion



Decimal to Binary Conversion

$(243)_{10} \longrightarrow (?)_2$



2	243	1
2	121	1
2	60	0
2	30	0
2	15	1
2	7	1
2	3	1
	1	

$\longrightarrow (11110011)_2$

Decimal to Octal Conversion

$(243)_{10} \longrightarrow (?)_8$

$\longrightarrow (363)_8$