Difference Between Machine Language and Assembly Language

www.differencebetween.com

Key Difference - Machine Language vs Assembly Language

Programming languages allow humans to create instructions for a computer to perform tasks. There are three categories of programming languages such as High-level programming languages, Assembly language, and Machine language. High-level programming languages are easier for humans to understand. Language recognized by a computer is known as machine language. Assembly language is the language between high-level languages and machine language. The **key difference** between machine language and assembly language is that, **machine language executes directly by a computer and assembly language requires an assembler to convert to machine code or object code to execute by the CPU.**

What is Machine Language?

Humans can understand High-level programming languages. It is not necessary to have a deep understanding of the internal CPU, to program using high-level languages. They follow a syntax similar to the English language. Java, C, C++, Python are some high-level programming languages. A computer recognizes machine language but does not understand high-level languages. Therefore, those programs should be converted to computer understandable machine language. This translation is done using a compiler or an interpreter.

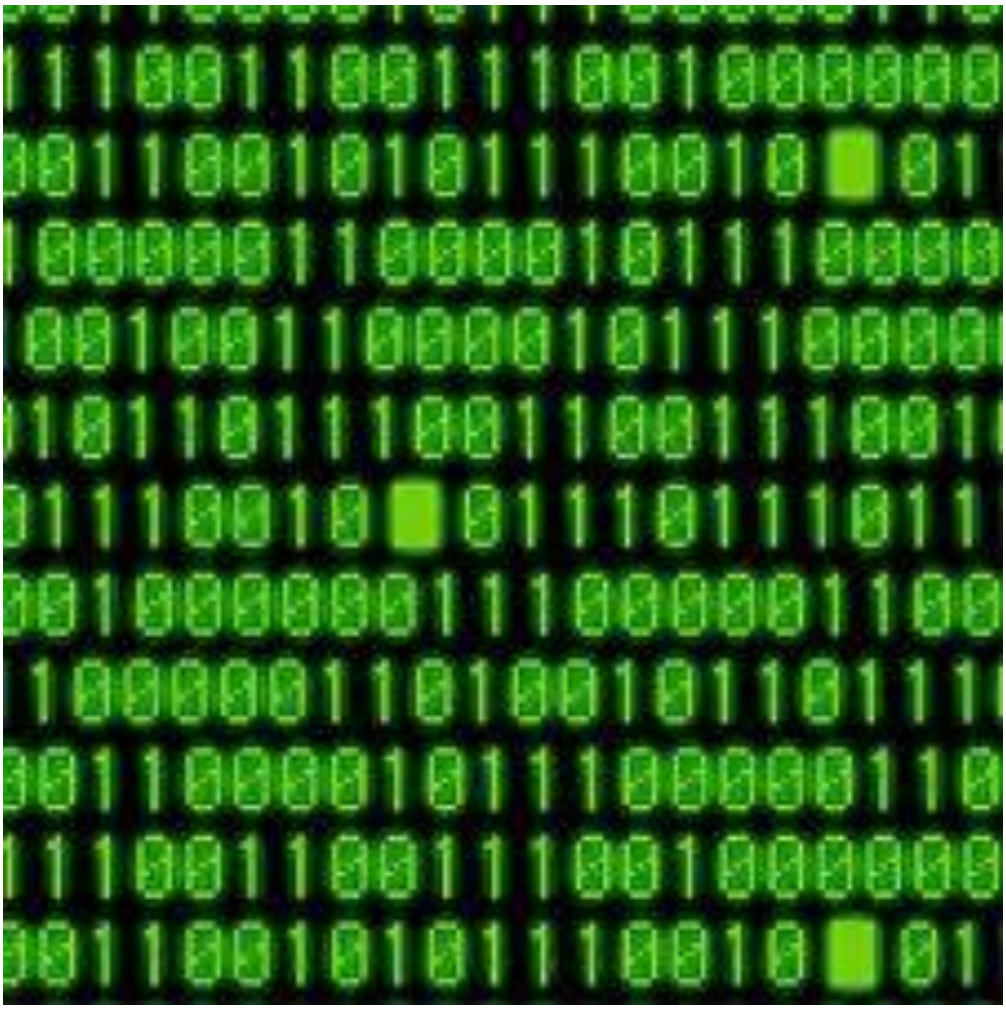

Figure 01: Machine Language of Zeros and Ones.

A machine language consists of binary digits which are zeros and once. A computer is a digital electronic device, so it uses binary for operations. One indicates the true state / on state while zero indicates the false state / off state. The way of converting a program from high-level language to machine language depends on the CPU.

What is Assembly Language?

Assembly language is the intermediate language between high-level programing languages and machine language. It is one level above machine language. Assembly language is easier to understand than machine language but harder than high-level programming languages. This language is also known as a low-level language because it is close to the hardware level. In order to write effective programs using Assembly, the programmer should have a good understanding of the computer architecture and the register structure. A special compiler known as an assembler is used to convert assembly language instructions to machine code or object code.

Assembly language statements have four sections. They are a label, mnemonic, operand, comment. Label and comments are optional. Mnemonic is the instruction to execute and operands are parameters for the command. Assembly language also supports macros. A macro can be defined as a set of instructions with a name. It can be used elsewhere in the program.

Some examples of Assembly language statements are as follows.

MOV SUM,50 - This instruction, copies the value 50 to the variable SUM.

ADD VALUE1,20 - This is to add 20 to the VALUE1 variable

ADD AH, BH - This instruction is to copy the content in AH register to BH register.

INC COUNT - This is to increment the variable COUNT by one.

AND VALUE1,100 - This is to perform AND operation on variable VALUE1 and 100.

MOV AL,20 - This is to copy value 20 to AL register

```
100
101
102
103
104
105
106
107 00000030 B9FFFFFFFF
108
109
110 00000035 41
111 00000036 803C0800
112
113
114 0000003A 75F9
115
116
117
118
119
120
121
122 0000003C C3
```

```
;-----------------------------------------------------------
; zstr_count:
; Counts a zero-terminated ASCII string to determine its size
; in:   eax = start address of the zero terminated string
; out:  ecx = count = the length of the string

zstr_count:                 ; Entry point
    mov  ecx, -1            ; Init the loop counter, pre-decrement
                            ;  to compensate for the increment
.loop:
    inc  ecx                ; Add 1 to the loop counter
    cmp  byte [eax + ecx], 0  ; Compare the value at the string's
                            ;  [starting memory address Plus the
                            ;  loop offset], to zero
    jne  .loop              ; If the memory value is not zero,
                            ;  then jump to the label called '.loop',
                            ;  otherwise continue to the next line
.done:
                            ; We don't do a final increment,
                            ;  because even though the count is base 1,
                            ;  we do not include the zero terminator in the
                            ;  string's length
    ret                     ; Return to the calling program
```

Figure 02: A Program written using Assembly Language

Set of Assembly statements is an Assembly program. It can be seen that the assembly language is easier than machine language. It has a syntax similar to the English language. Assembly language has around thirty instructions. The required memory and execution time is minimum comparing to high-level languages.

In real-time systems, there can be events which require CPU action immediately. These events are special subroutines called Interrupt service routine (ISR). Assembly language is useful for programming ISR.

What is the Similarity Between Machine Language and Assembly language?

- Both machine language and assembly language are related to the hardware level.

What is the Difference Between Machine Language and Assembly language?

Machine Language vs Assembly Language	
Machine language is the lowest level programming language where the instructions execute directly by the CPU.	Assembly language is a low-level programming language which requires an assembler to convert to machine code/object code.
Comprehensibility	
Machine language is comprehensible only to the computers.	Assembly language is comprehensible to humans.
Syntax	
A machine language consists of binary digits.	Assembly language follows a syntax similar to the English language.
Dependency	
Machine language varies depending on the platform.	Assembly language consists of a standard set of instructions.
Applications	
Machine language is machine code.	Assembly language is using for microprocessor-based, real-time systems.

Summary - Machine Language vs Assembly Language

The difference between machine language and assembly language is that machine language is directly executed by a computer and assembly language is a low-level programming language which requires an assembler to convert to object code or machine code. Assembly language is one step ahead of machine language. Assembly language is an ideal language to program microcontroller based systems. This language also gives a good understanding of how the CPU is working and about the internal components of the computer.

Reference:

1.The point, Tutorials. “Assembly Introduction.” Tutorials Point, 15 Aug. 2017. Available here
2.The point, Tutorials. “Assembly Basic Syntax.” Tutorials Point, 15 Aug. 2017. Available here
3.“What is Machine Language?” Computer Hope, 11 Oct. 2017. Available here

Image Courtesy:

1.'Machine language'By Turkei89 - Own work, (CC BY-SA 3.0) via Commons Wikimedia
2.'Zstr count x86 assembly' By OldCodger2, (Public Domain) via Commons Wikimedia

How to Cite this Article?

APA: Difference Between Machine Language and Assembly Language.(2017 December 12). Retrieved (date), from http://differencebetween.com/difference-between-machine-language-and-vs-assembly-language /

MLA: "Difference Between Machine Language and Assembly Language" Difference Between.Com. 12 December 2017. Web.

Chicago: “Difference Between Machine Language and Assembly Language.” Difference Between.Com. http://differencebetween.com/difference-between-machine-language-and-vs-assembly-language / accessed (accessed [date]).