**Submitted by**

Name: MOHAMMAD TALHA

Roll no: BITF19M525

**Submitted to**

SIR HASSAN KHAN

**Programming Languages**

**Abstract of Home Work**

In order to make better understanding with programming languages and to have detailed knowledge of all categories of languages. To know about history and working model of each category with examples.

**Machine Languages**

**Introduction:**

Machine language is the lowest-level [programming language](https://www.webopedia.com/TERM/P/programming_language.html) (except for computers that utilize programmable [microcode](https://www.webopedia.com/TERM/M/microcode.html)). Machine languages are the only [languages](https://www.webopedia.com/TERM/L/language.html) directly understood by [computers](https://www.webopedia.com/TERM/C/computer.html).[1] Machine language is the actual language in which a program is stored in memory in a binary or binary-coded format

**History:**

The first computer programming language was created in 1883, when a woman named Ada Lovelace worked with Charles Babbage on his very early mechanical computer, the Analytical Engine. While Babbage was concerned with simply computing numbers, Lovelace saw that the numbers the computer worked with could represent something other than just amounts of things. She wrote an algorithm for the Analytical Engine that was the first of its kind. Because of her contribution, Lovelace is credited with creating the first computer programming language. As different needs have arisen and new devices have been created, many more languages have followed. [2]

**Working Model:**

Each CPU has its own specific machine language. The processor reads and handles instructions, which tell the CPU to perform a simple task. Instructions are comprised of a certain number of [bits](https://whatis.techtarget.com/definition/bit-binary-digit). If instructions for a particular processor are 8 bits, for example, the first 4 bits part (the opcode) tells the computer what to do and the second 4 bits (the operand) tells the computer what data to use. [4]

**Examples:**

Below is an example of machine language (binary) for the text "Hello World".

01001000 01100101 01101100 01101100 01101111 00100000 01010111 01101111 01110010 01101100 01100100. [3]

**Conclusion:**

While easily understood by computers, machine languages are almost impossible for humans to use because they consist entirely of numbers. [Programmers](https://www.webopedia.com/TERM/P/programmer.html), therefore, use either a high-level programming language or an [assembly language](https://www.webopedia.com/TERM/A/assembly_language.html).

**Assembly Languages**

**Introduction**:

. An assembly language contains the same [instructions](https://www.webopedia.com/TERM/I/instruction.html) as a machine language, but the instructions and [variables](https://www.webopedia.com/TERM/V/variable.html) have [names](https://www.webopedia.com/TERM/N/name.html)(mnemonics) instead of being just numbers.

**History:**

The history of assembly languages is closely mingled with that of the stored-program computer. When the Electronic Delay Storage Automatic Calculator (EDSAC) was incorporated with an assembler, ‘initial orders’, which used one letter mnemonics in 1949. Stan Poley wrote the Symbolic Optimal Assembly Program or SOAP assembly language for the IBM 650 computer in 1955.

Assembly languages started being used widely as they relieved the programmers from tedious tasks such as remembering numeric codes. Their use, however, was reduced substantially by the 1980’s due to the introduction of high-level languages. [6]

**Working Model:**

Assembly language programs are translated into machine language by a program called an [assembler](https://www.webopedia.com/TERM/A/assembler.html). When you pass these assembly instructions through an assembler, they are translated into machine code they represent, which is what the CPU and its various co-processors interpret and execute (it's generally taken down into smaller units by the CPU, called micro-ops). [7]

**Examples:**

The following assembly language can be used to add the numbers 3 and 4:

Mov eax, 3 (loads 3 into the register "eax")  
mov ebx, 4 (loads 4 into the register "ebx")  
add eax, ebx, ecx (adds "eax" and "ebx" and stores the result (7) in "ecx") [5]

**Conclusion:**

Each assembly language corresponds to only one computer – that is, there is a high degree of specificity in case of assembly languages. This makes assembly languages quite unlike most of the high-level languages as they cannot be used on a variety of computers whereas high level languages are mostly portable. [6]

**Compiled Languages**

**Introduction:**

A compiled program is not human readable, but instead is in an architecture-specific machine language. A **compiled** **language** is one where the program, once compiled, is expressed in the instructions of the target machine; this machine code is undecipherable by humans. [10]

**History:**

Autocode was a generic term for a family of early computer programming languages. The first was developed by Alick Glennie for the Mark 1 computer at the University of Manchester in the U.K in 1952. Some consider autocode to be the first compiled computer programming language. [2]

**Working Model:**

Creating a compiled program requires several steps. First, the programmer, using a development tool or even a simple text editor, writes the source code in a chosen computer language. If the program is complex, pieces of it may be spread across several files. The programmer then compiles the program, sorting and linking the modules and translating it all into machine code that the computer understands. [8]

**Examples:**

RUST, ForTran, COBOL, C, C++, Erlang and Haskell etc. are some compiled programming languages. [10]

Following is an example of C++, which will print hello world:

#include <iostream>

using namespace std;

int main()

{

cout << ”Hello World”;

return 0;

}

**Conclusion:**

Because different kinds of computers do not speak each others' machine languages, a compiled program will only work on the platform it was designed for. For example, a program written for HP-UX normally will not work on a Mac OS computer or a computer running Solaris. Despite this drawback, compiled programs are faster than those that must be run through an interpreter. Also, it is often possible to recompile the program so that it will run on different platforms. [8]

**Interpreted Languages**

**Introduction:**

An interpreted language is a type of programming language for which most of its implementations execute instructions directly and freely, without previously compiling a program into machine-language instructions. In an interpreted program, the source code typically is the program (often known as scripts). [8]

**History:**

In the early days of computing, language design was heavily influenced by the decision to use compiling or interpreting as a mode of execution. For example, [Smalltalk](https://en.wikipedia.org/wiki/Smalltalk) (1980), which was designed to be interpreted at run-time, allows generic objects to dynamically interact with each other.

Initially, interpreted languages were compiled line-by-line; that is, each line was compiled as it was about to be executed, and if a loop or subroutine caused certain lines to be executed multiple times, they would be recompiled every time. This has become much less common. [9]

**Working Model:**

Programs of this type require an interpreter, which parses the commands in the program and then executes them. Some interpreters, read and then immediately execute each command, while others, analyze the entire script before sending the corresponding machine language instructions. [8]

**Examples**:

PHP, Perl, JavaScript, Ruby, and Python etc. [10]

Following is an example of Python, which will print hello world:

print('Hello, world!')

**Conclusion:**

The advantage of a script is that it is very portable. Any computer that has the appropriate interpreter installed may run the program more or less unchanged. This is a disadvantage as well, because the program will not run at all if the interpreter is not available. In general, interpreted programs are slower than compiled programs, but are easier to debug and revise. [8]

**REFRENCES**

[1] <https://www.webopedia.com/TERM/M/machine_language.html>

[2] <https://www.onlinecollegeplan.com/computer-programming-languages/>

[3] <https://www.computerhope.com/jargon/m/machlang.htm>

[4] <https://whatis.techtarget.com/definition/machine-code-machine-language>

[5] <https://techterms.com/definition/assembly_language>

[6] <https://www.cleverism.com/skills-and-tools/assembly-language/>

[7] <https://stackoverflow.com/questions/6463938/how-do-assembly-languages-work>

[8] <https://kb.iu.edu/d/agsz>

[9] <https://en.wikipedia.org/wiki/Interpreted_language>

[10] <https://thecodeboss.dev/2015/07/programming-concepts-compiled-and-interpreted-languages/>