Top of Form

**GENERAL MACHINE STRUCTURE**

All the conventional modern computers are based upon the concept of *stored program computer*, the model that was proposed by John von Neumann.

The components of a general machine are as follows:

***Instruction interpreter:*** A group of electronic circuits performs the intent of instruction of fetched from memory.

***Location counter:*** LC otherwise called as *program counter PC* or *instruction counter IC*, is a hardware memory device which denotes the location of the current instruction being executed.

***Instruction register:*** A copy of the content of the LC is stored in IR.

***Working register:*** are the memory devices that serve as “scratch pad” for the instruction interpreter.

***General register:*** are used by programmers as storage locations and for special functions.

***Memory address register (MAR):*** contains the address of the memory location that is to read from or stored into.

***Memory buffer register (MBR):*** contain a copy of the content of the memory location whose address is stored in MAR. The primary interface between the memory and the CPU is through *memory buffer register*.

***Memory controller:*** is a hardware device whose work is to transfer the content of the MBR to the core memory location whose address is stored in MAR.

***I/O channels:*** may be thought of as separate computers which interprets special instructions for inputting and outputting information from the memory.

Let us take a command **ADD 2,176.**

This instruction has three parts first the opcode i.e. ADD, second is the number of the register that contain the first operator, third is the memory location address that contain the second operand.

* At first, the address from the IC is copied to the MAR.
* Then the instruction is fetched to the MBR.
* The instruction is then transferred to the IR.

***Memory:*** Basic unit, size and addressing scheme.

***Registers*** Number of registers, and size, functions, interrelation of each register.

***Data:*** Types of data and their storing scheme.

***Instruction:*** Classes of instructions, allowable operations and their storing scheme.

***Special Features:*** Additional features like interrupt and protections.

**Registers**

There are a total of **16 *general purpose registers*** of 32 bits each. In addition there are ***4 floating point register*** of 64 bits each. It also has a 64 **bits *program status word (PSW)*** that contains the value of the location counter, protection information and interrupt status.

The general purpose registers are basically used in arithmetic and logical operations as *base registers* and helps in address formation. The general purpose registers also used as scratch pads for the programmers. Let us take an instruction A 1,901(2,15).

A(opcode)1(operand in register 1),901(offset) (2(index register),15(base register))

This is how the memory locations are addressed in case of 360 and 370. The use of base registers in addressing is twofold.

First it helps the loader in the process of relocation (changing the content of base register causes the code to be relocated to the specified location).

Secondly it decreases the size of instruction as follows: since the memory of 360 is of 224 hence a total of 24 bits are required to specify a particular location of memory. This increases the size of instruction as opcode takes 8 bits and the registers require 4 bits each and the address requires a 24 bits hence the size of the instruction is 8+4+4+24=40 bits (without base register)

All the conventional modern computers are based upon the concept of *stored program computer*, the model that was proposed by John von Neumann.

**What Does Address Space Mean?**

An address space is a range of valid addresses in memory that are available for a program or process. That is, it is the memory that a program or process can access. The memory can be either physical or virtual and is used for executing instructions and storing data.

On a computer, each process and device is allocated an address space, which holds a certain portion of the processor's address space. The processor's address space is typically restricted to the width of its registers and address bus.

Address space is often classified as either flat, where the addresses are represented as incrementally increasing integers that start at zero, or segmented, where the addresses are portrayed as independent segments augmented by offsets. In some systems, the address space may be modified from one format to the other via a process generally known as thunking.

The size of an address space can be made larger than that of physical memory by using a memory management technique called virtual memory. A virtual memory, also known as a page file, is actually a physical file on disk that acts like an additional RAM or RAM module. Thus, an address space consists of both physical memory and virtual memory.

**Interface**

An **interface** may refer to any of the following:

1. When referring to software, an **interface** is a program that allows a user to interact computers in person or over a network. An interface may also refer to controls used in a program that allow the user to interact with the program. One of the best examples of an interface is a GUI (Graphical User Interface). This type of interface is what you are using now to navigate your computer and how you got to this page.

2. When referring to hardware, an **interface** is a physical device, port, or connection that interacts with the computer or other hardware device. For example, IDE and SATA are disk drive interfaces for computerHard Drives and ATAPI is an early interface for CD-ROM drives.

**Examples of drive interfaces**

The following list is a list of different internal and external interfaces that connect a drive to a computer.

ATA

ATAPI

eSATA

FireWire

IDE

Parallel Port

SATA

SCSI

USB

**Computer Languages**

•The computer language is defined as code or syntax which is used to write programs or any specific applications. It is used to communicate with computers.

•

•Computer languages can be broadly classified into

   3 major categories :

   1. assembly language

   2. machine language

    3. High-level language.

**1. Machine Language**

The machine language is sometimes referred to as machine code or object code which is a set of binary digits 0 and 1. These binary digits are understood and read by a computer system and interpreted easily. It is considered a native language as it can be directly understood by a central processing unit (CPU). The machine language is not so easy to understand, as the language uses the binary system in which the commands are written in 1 and 0 form which is not easy to interpret. There is only one language that is understood by computer which is machine language. The operating system of the computer system is used to identify the exact machine language used for that particular system.

The operating system defines how the program should write so that it can be converted to machine language and the system takes appropriate action. The computer programs and scripts can also be written in other programming languages like C, C++, and JAVA. However, these languages cannot be directly understood by a computer system so there is a need for a program that can convert these computer programs to machine language. The compiler is used to convert the programs to machine language which can be easily understood by computer systems. The compiler generates the binary file and executable file.

Example of machine language for the text “Hello World”:-

01001000 0110101 01101100 01101100 01101111 00100000 01010111 01101111 01110010 01101100 01100100

**2. Assembly Language**

The assembly language is considered a low-level language for microprocessors and many other programmable devices. The assembly language is also considered a second-generation language. The first generation language is machine language. The assembly language is mostly famous for writing an operating system and also in writing different desktop applications. The operations carried out by programmers using assembly language are memory management, registry access, and clock cycle operations. The drawback of assembly language is the code cannot be reused and the language is not so easy to understand. The assembly language is considered a group of other languages. It is used to implement the symbolic representation of machine code which is used to program CPU architecture. The other name of assembly language is assembly code. For any processor, the most used programming language is assembly language.

In assembly language, the programmer does the operation which can be directly executed on a central processing unit (CPU). The language has certain drawbacks as it does not contain any variables or functions in programs and also the program is not portable on different processors. The assembly language uses the same structure and commands which machine language does use but it uses names in place of numbers. The operations performed using the assembly language are very fast. The operations are much faster when it is compared to high-level language.

**3. High-Level Language**

The development of high-level language was done when the programmers face the issue of writing programs as the older language has portability issues which means the code written in one machine cannot be transferred to other machines. This led to the development of high-level language. The high-level language is easy to understand and the code can be written easily as the programs written are user-friendly in a high-level language. The other advantage of code written in a high-level language is the code is independent of a computer system which means the code can be transferred to other machines. The high-level of language uses the concept of abstraction and also focuses on programming language rather than focusing on computer hardware components like register utilization or memory utilization.

The development of higher-level language is done for a programmer to write a human-readable program that can be easily understood by any user. The syntax used and the programming style can be easily understood by humans if it is compared to low-level language. The only requirement in a high-level language is the need for a compiler. As the program written in a high-level language is not directly understood by the computer system. Before the execution of high-level programs, it needs to be converted to machine-level language. Examples of high-level languages are C++, C, JAVA, FORTRAN, Pascal, Perl, Ruby, and Visual Basic.

C0mputer languages open an array of career opportunities. Here are types of computer languages list for you:

* Java
* VHDL
* S-Lang
* Oz
* ML
* Nim
* C++
* C
* Python
* PHP
* JavaScript
* SQL
* Swift
* C#
* A+
* APL
* Chapel
* Simula
* Julia
* Xojo
* Dart
* Lasso
* Fortran 90
* Rust
* Lucid
* Max
* HTML/CSS

**Java**

As indicated by Tobie Programming Community Index, Java has consistently featured at the top of the sought-after computer languages in the world since its creation in the mid-90s. A significant number of the world’s best organizations use Java to manufacture their office applications and backend web frameworks. If you know Java, odds are you won’t be edgy for work! Various variables make Java so famous and the most prominent ones are Portability and Versatility.

**Portability**: Thanks to the stage freethinker Java Virtual Machine (JVM), Java can run on almost every framework. Other than being a popular computer language, it is additionally the most well-known Android language and several android applications run in Java.

**Versatility**: In the words of James Governor, “when web organizations grow up, they become Java shops”. Java is known for its adaptability as a top priority, which is the reason it is so well known among corporate endeavors and scaling new businesses (Twitter moved from Ruby to Java for various purposes). Since Java is a statically-composed computer language, it’s quicker and simpler to keep up with fewer bugs.

**C Programming Language**

C is one of the most seasoned and mainstream computer languages because of its convenience and appropriation by tech giants like Microsoft, Apple, Linux, and Oracle. C is likewise a sought-after language for installed frameworks in vehicles, hardware, and different gadgets. Almost every digital device that we use, from our mobile phones to morning timers, incorporated —if not legitimately programmed in—the C language. C additionally has highlights that make it entirely equipped for working frameworks and implanted frameworks. Numerous calculations composed and shared online are processed in C. It’s the “general language” of programming dialects. C side projects like C++ and C# are likewise among the best five most mainstream dialects for aspiring programmers and data scientists.

**Python**

The prominence of Python has risen consistently in recent years, at long last breaking into the consistent five computer languages on the Tobie Index. It is because Python is a significant language and is probably the most energizing advancements today. **AI, Big Data Analytics** and Robotics all depend intensely on Python (Robotics likewise depends on C for its utilization in frameworks programming).

**JavaScript**

With the popularity of internet browsers, JavaScript is another important mention to add in our list of best computer languages as it featured number 1 on GitHub as the most in-demand programming languages. It has stood its ground against more current dialects and will keep on assuming a job on the web. JavaScript permits engineers to add impacts to site pages. It regularly works nearby HTML, yet it is increasingly utilized for web applications to be constructed altogether in JavaScript.

**Computer Languages In-Demand**

The list of computer languages is never-ending. From time to time the demand for computer languages also fluctuates. Some languages were in demand years ago but as technology changes the demand also changes. Here is the list of computer languages in demand all around the world:

1. Ruby
2. JavaScript
3. Python
4. PHP
5. Java
6. C#
7. Objective-C & Swift

**Ruby**

Ruby is a powerfully composed computer language considered as highly adaptable and extraordinary however hard to keep up at scale. As a Ruby application develops, the dynamic idea of the language clouds the wellspring of code blunders and gobbles up registering assets. Ruby is one of the most well-known dialects among tech companies.

Numerous Silicon Valley unicorns have been based on Ruby, including Airbnb, Twitch, GitHub, and Twitter. Firstly, Ruby has an inconceivably basic delightful sentence structure that permits a designer to accomplish more with less code. Secondly, because of Ruby on Rails, getting a web application fully operational takes less time than in different structures.

**References**

1. [kishoreinvssut.yolasite.com/resources/System\_programming](http://kishoreinvssut.yolasite.com/resources/System_programming/02.Machine%20structure_machine%20language_assembly%20language.pdf)
2. [Machine structure\_machine language\_assembly language.pdf](http://kishoreinvssut.yolasite.com/resources/System_programming/02.Machine%20structure_machine%20language_assembly%20language.pdf)
3. <http://tinf2.vub.ac.be/~dvermeir/courses/compilers/yacc.pdf>
4. <https://www3.diism.unisi.it/~maggini/Teaching/TEL/slides%20EN/04%20-%20YACC.pdf>

Bottom of Form