**Introduction to Computer Process**

Graphical user interface, application

Description automatically generated

**1. CPU: Central Processing Unit** (Intel Core i9, etc.)

Contains transistors, which are on-off. This represents binary system. On = 1, Off = 0. The value at one of the switches is called a bit.

A picture containing text

Description automatically generatedA picture containing table

Description automatically generated

These bits come in combinations, 8 (a byte), 16, 32, or 64. Different combinations of bit can be used to represent different data, through ASCII values (everything we type in keyboard).

- 8 bits or 16, 32, or 64 bits is the size in which the processor can handle data. E.g., for a 16-bit processor, a binary number of up to 216 can be represented

**🡪 There are 8-bit, 16-bit, 32-bit or 64-bit processors.**

**2. Computer Programming**

- Computer programs consist of a sequence of instructions. All instruction is stored in memory as 1’s and 0’s, this is called Machine Code.

So how do the computer generate these Machine Codes?

Diagram

Description automatically generated

* User writes a program in C language (high-level language).
* **Pre-processor** adds in all preprocessed libraries and functions required.
* The **C compiler** compiles the program and translates it to **assembly program** (low-level language).
* An **assembler** then translates the **assembly program** into **machine code** (object).
* A **linker tool** is used to link all the parts of the program together for execution (executable machine code).
* A **loader** loads all of them into memory and then the program is executed.

We write the program in C language, then the compiler & assembler translates it to an assembly program (that the computer can understand). The computer then executes these 1’s and 0’s at the memory.

\*Preprocessor: A tool that produces input for compilers and is a part of compiler. It deals with macro-processing, file inclusion (#include, #define)

\*Assembler: An assembler translates assembly language programs into machine code. The output of an assembler is called an object file, which contains a combination of machine instructions as well as the data required to place these instructions in memory.

\*Linker: Linker is a computer program that links and merges various object files together in order to make an executable file. All these files might have been compiled by separate assemblers. The major task of a linker is to search and locate referenced module/routines in a program and to determine the memory location where these codes will be loaded, making the program instruction to have absolute references.

\*Loader: Loader is a part of operating system and is responsible for loading executable files into memory and execute them. It calculates the size of a program (instructions and data) and creates memory space for it. It initializes various registers to initiate execution.

**3. Memory (RAM)**

- Memory holds bits of data. Memory is organized in groups of 8, 16, 32, or 64.

- Every location has a **memory address** that the CPU and read and write to. Each memory address contains 1 byte (8 bits, or a string of 0s and 1s), or sometimes multiples of 8. The address of the first of the bytes that make up multiple memory location is used as the address for this larger memory location.

**Graphical user interface, diagram, application

Description automatically generated**

**4. Operating System Kernels** (MacOS, Windows, Linux)

Control all hardware resources via device drivers.

We can communicate with the Kernel using Shell, which is a program that exposes OS to the user.

Diagram

Description automatically generated

So, as we put an input to the computer, and receives an output, we are actually working on Shells that uses the **Command Line Interface (CLI)**

**Graphical user interface, text

Description automatically generated**

We can also use CLI and **Secure Shell Protocol (SSH)** to connect with remote computer, or Mainframe.

**5. Programming Language**

The programming languages are tools that uses the abstraction principle to make computers do the work for human.

*Abstraction principle means the program hides implementation. We only care about input/output without caring about how the work is implemented. Functionality, on the other hand, is public.*

**+) Interpreted language (Python, Java)**

That means there’s a program called interpreter that will execute each lines of code one by one.

**+) Compiled language (C/C++)**

Use a compiler to convert the entire program into machine code before the CPU attempts to execute it 🡪 .exe files (executable files) that can be run by the OS without any extra discrepancies.