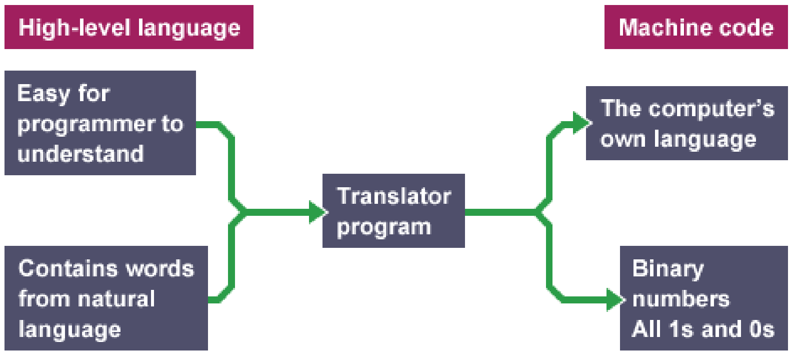
CSC 218 Foundation of Sequential Program
Lecture Note 1 (H/L Languages)

High-level languages

High-level languages allow programmers to write instructions in a language that is easier to understand than low-level languages. Translators are needed to translate programs written in high-level languages into the machine code that a computer understands.

The instructions that tell a computer what to do are written in machine code. Machine code is a series of numbers written in binary. Each number represents a different instruction.

Programmers find machine code difficult to learn, program in and debug. As a result, the majority of programmers write programs in high-level programming languages. These languages are close to natural language - the spoken and written language of humans. For example, Python uses 'print', 'if', 'input' and 'while' statements - all words from the English language - to form instructions. In fact, instructions often look like abbreviated English sentences.


High-level language
Easy for programmer to understand
Contains words from natural language
Translator program
Machine code
The computer's own language
Binary numbers All 1s and 0s

Compare this simple Python program with its comments that are written in English:

while count < 10:

#While the value of count is less than ten

number = int(input("Type in a number"))

#Input a number

total = total + number

#Add the number to the total

count = count + 1

#Add one to the value of count

print("The total is ", total)

#Print out the total

Programmers write in high-level languages because they are easier to understand and are less complex than machine code. They allow the programmer to focus on what needs to be done, rather than on how the computer actually works.
For example, in many high-level languages, to place a message on the screen, a programmer would use the statement 'print'. The programmer might not know how the computer actually generates the message. They just need to know how to use the 'print' statement.

Commonly used high-level languages

Many types of high-level language exist and are in common use today, including:

- Python
- Java
- C++
- C#
- Visual Basic
- JavaScript

Low-level languages

Low-level languages are languages that sit close to the computer's instruction set. An instruction set is the set of instructions that the processor understands.

Two types of low-level language are:

- machine code
- assembly language

Machine code

Machine code is the set of instructions that a CPU understands directly and can act upon. A program written in machine code would consist of only 0s and 1s - binary. This is very difficult to write and debug. Even a very simple program could have thousands of 0s and 1s in it.

Assembly language

Assembly language sits between machine code and high-level language in terms of ease of use. While high-level languages use statements to form instructions, assembly language uses mnemonics - short abbreviations. Each mnemonic directly corresponds with a machine code instruction. Here are some examples of mnemonics:

Mnemonic	Action
LDA	Loads a value from a memory address
STA	Stores a value in a memory address
ADD	Adds the value held in a memory address to the value held in the accumulator
SUB	Subtracts from the accumulator the value held in a memory address
MOV	Moves the contents of one memory address to another

In assembly language, programmers write programs as a series of mnemonics. Mnemonics are much easier to understand and debug than machine code, giving programmers a simpler way of directly controlling a computer.

Writing in mnemonics is easy for programmers because they are usually brief representations of the actual commands. They are quicker to write than binary, and it is easier to spot mistakes.

Little Man Computer (LMC) is a simulation of a very basic processor using Von Neumann architecture. It uses an example of simple assembly language that contains a limited set of mnemonic instructions which can be used to program simple assembly programs. LMC is freely available on the internet for students to use.

Opcodes and operands
Many machine code and assembly instructions contain two parts:

- the opcode - this is the actual instruction
- the operand - this is a value that the instruction uses or manipulates

Consider this set of program instructions:

Assembly language	Opcode	Operand	Instruction
INP	1001	00000000	Input a number
STR 6	0011	00000110	Store it in address 06
LDR A1	0101	10100001	Load data from address A1
ADD #10	0010	00001010	Add the number 10 to the loaded address

Both opcode and operand values are ultimately represented in binary. However, values may also be represented in hexadecimal when programming as this number system can represent larger values in fewer characters, for example, denary 250 is 11111010 in binary, but only FA in hexadecimal. Hexadecimal is therefore easier to read and understand by humans.

Translators

Any program written in a high-level language is known as source code. However, computers cannot understand source code. Before it can be run, source code must first be translated into a form which a computer understands.

A translator is a program that converts source code into machine code. Generally, there are three types of translator:

- compilers
- interpreters
- assemblers

Compilers

A compiler takes the source code as a whole and translates it into machine code all in one go. Once converted, the object code can be run unassisted at any time. This process is called compilation.

Compilers have several advantages:

- Compiled programs run quickly, since they have already been translated.
- A compiled program can be supplied as an executable file. An executable file is a file that is ready to run. Since an executable file cannot be easily modified, programmers prefer to supply executables rather than source code.
- Compilers optimise code. Optimised code can run quicker and take up less memory space.

Compilers also have disadvantages:

- The source code must be re-compiled every time the programmer changes the program.
- Source code compiled on one platform will not run on another - the machine code is specific to the processor's architecture.

Interpreters

An interpreter translates source code into machine code one instruction at a time. It is similar to a human translator translating what a person says into another language, sentence by sentence, as they speak. The resulting machine code is then executed immediately. The process is called interpretation.

Interpreters have several advantages:

- Instructions are executed as soon as they are translated.
- Errors can be quickly spotted - once an error is found, the program stops running and the user is notified at which part of the program the interpretation has failed. This makes interpreters extremely useful when developing programs.

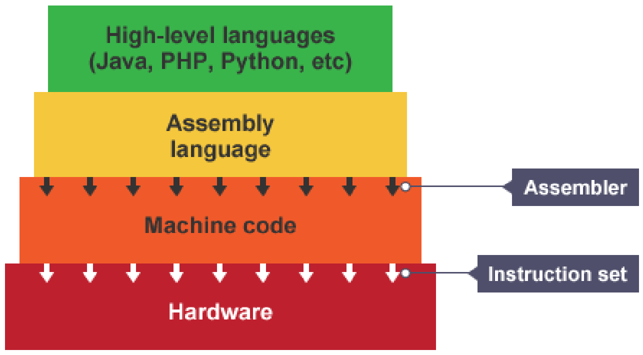
Interpreters also have several disadvantages:

- Interpreted programs run slowly as the processor has to wait for each instruction to be translated before it can be executed.
- Additionally, the program has to be translated every time it is run.
- Interpreters do not produce an executable file that can be distributed. As a result, the source code program has to be supplied, and this could be modified without permission.
- Interpreters do not optimise code - the translated code is executed as it is.

Assemblers

Assemblers are a third type of translator. The purpose of an assembler is to translate assembly language into machine code.

Whereas compilers and interpreters generate many machine code instructions for each high-level instruction, assemblers create one machine code instruction for each assembly instruction.



ADVANTAGES OF HIGH-LEVEL LANGUAGES

The main advantage of high-level languages over low-level languages is that they are easier to read, write, and maintain. Ultimately, programs written in a high-level language must be translated into machine language by a compiler or interpreter.

The first high-level programming languages were designed in the 1950s. Now there are dozens of different languages, including Ada, Algol, BASIC, COBOL, C, C++, FORTRAN, LISP, Pascal, and Prolog.

Computer Architecture

Computer architecture consists of rules and methods or procedures which describe the implementation, functionality of the computer systems. Architecture is built as per the user's needs by taking care of the economic and financial constraints. Earlier architecture is designed on paper built with hardware form.

After it is built-in transistor-transistor logic the architecture is built, tested and formed in the hardware form. We can define computer architecture based on its performance, efficiency, reliability, and cost of the computer system. It deals with software and hardware technology standards. The computer system has the processor, memory, I/O devices and communication channels that connect to it.

Types of Computer Architecture

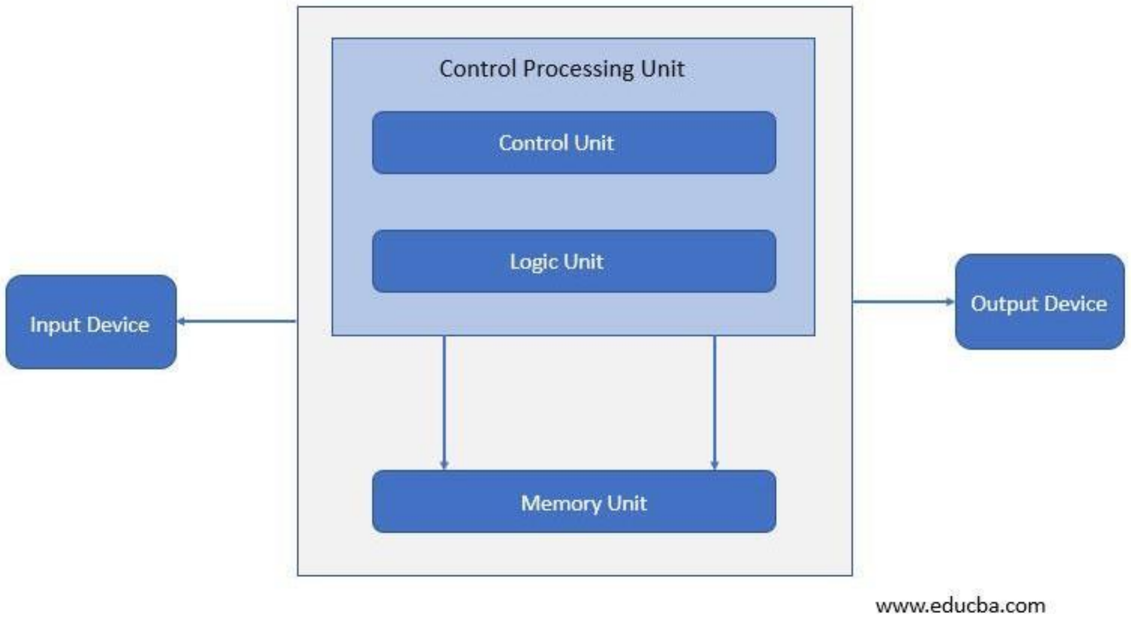
Given below are the types of Computer Architecture:

1. Von-Neumann Architecture

This architecture is proposed by john von-neumann. Now a day's computer we are using are based on von-neumann architecture. It is based on some concepts.

The memory we have a single read/write memory available for read and write instructions and data. When we talk about memory, it is nothing but the single location which is used for reading and writing instructions for the data and instructions are also present in it. Data and instructions are stored in a single read/write memory within the computer system.

Each memory has multiple locations and each location has a unique address. We can address the contents of memory by its location irrespective of what type of data and instructions are present in the memory, because of which we can read or write any data and instructions. Execution always occurs in a sequential manner unless the change is required. For example, suppose we are executing an instruction from line 1 to line 10 but now we required to execute line 50 instead of line 11 then we jump to instruction 50 and execute it.



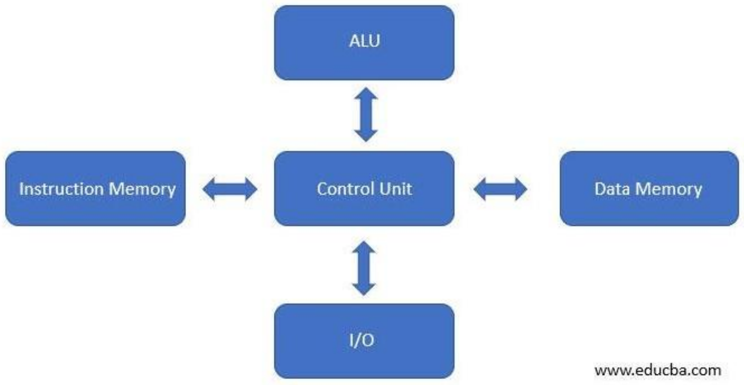
There is a bus (address bus/data bus/control bus) used for the instruction and data code execution. Input device takes data or instruction and the Central processing unit (CPU) performs one operation at a time, either fetching data or instruction in/out of the memory. Once the operation is done it is sent to the output device. Control and logic units for processing operations are within the central processing unit.

2. Harvard Architecture

Harvard architecture is used when data and code is present in different memory blocks. A separate memory block is needed for data and instruction. Data can be accessed by one memory location and instruction can be accessed by a different location. It has data storage entirely contained within the central processing unit (CPU). A single set of clock cycles is required. The pipeline is possible. It is complex to design. CPU can read and write instructions and process data access. Harvard architecture has different access codes and data address spaces that is, the instruction address zero is not the same as data address zero. Instruction address zero identifies 24-byte value and data address zero identifies 8-byte value which is not the part of the 24-byte value.

Modified harvard architecture is like a harvard architecture machine and it has a common address space for the separate data and instruction cache. It has digital signal processors that will execute small or highly audio or video algorithms and it is reproducible. Microcontrollers have a small number of programs and data memory and it speeds up the processing by executing parallel instructions and data access.

We can observe in the below image, there are separate data and instruction memory that is a bus available to perform operations. It is contained entirely within the Central processing unit. It can perform Input/output operation simultaneously and it has a separate arithmetic and logic unit.



3. Instruction Set Architecture

To make up the architecture, instruction set architecture is needed because it has a set of instructions that the processor understands. It has two instruction set one is RISC (reduced instruction set computer) and the second is CISC (complex instruction set computer).

Reduced instruction set computer architecture was realized in the 90's by IBM. Instruction has multiple address modes, but programs do not use all of them that is the reason multiple address modes were reduced. This helps the compiler to easily write the instructions, performed is increased.

Complex instruction set architecture is the root of compilers because earlier compilers were not there to write programs, to ease programming instructions are added. The best performance is obtained by using simple instruction from ISA.

4. Microarchitecture

Microarchitecture is known as computer organizations and it is the way when instruction set architecture is a built-in processor. Instruction set architecture is implemented with various microarchitecture and it varies because of changing technology.

Microarchitecture performs in a certain way. It reads the instruction and decodes it, will find parallel data to process the instruction and then will process the instruction and output will be generated.

It is used in microprocessors, microcontrollers. Some architectures overlap multiple instructions while executing but this does not happen in microarchitecture. Execution units like arithmetic logic units, floating-point units, load units, etc are needed and it performs the operation of the processor. There are microarchitecture decisions within the system such as size, latency, and connectivity of the memories.

5. System Design

The name defines itself, the design will satisfy user requirements such as architecture, module, interfaces and data for a system and it is connected to product development. It is the process of taking marketing information and creating product design to be manufacture. Modular systems are made by standardizing hardware and software.

The Main Types of Semiconductor Chips

The types of chips produced by semiconductor companies can be categorized in two ways. Usually, chips are categorized in terms of their functionality. However, they are sometimes divided into types according to the integrated circuits (ICs) used.

When looked at according to functionality, the four main categories of semiconductors are memory chips, microprocessors, standard chips and complex systems-on-a-chip (SoCs). When organized by types of integrated circuitry, the three types of chips are digital, analog and mixed.

Memory Chips

From the perspective of functionality, semiconductor memory chips store data and programs on computers and data storage devices. Random-access memory (RAM) chips provide temporary workspaces, whereas flash memory chips hold information permanently unless erased. Read-only memory (ROM) and programmable read-only memory (PROM) chips cannot be modified. In contrast, erasable programmable read-only memory (EPROM) and electrically erasable read-only memory (EEPROM) chips can be changed.

Microprocessors

Microprocessors contain one or more central processing units (CPUs). Computer servers, personal computers (PCs), tablets and smartphones may each have multiple CPUs. The 32- and 64-bit microprocessors in PCs and servers are based on x86, POWER, and SPARC chip architectures. On the other hand, mobile devices typically use an ARM chip architecture. Less powerful 8-, 16- and 24-bit microprocessors turn up in products such as toys and vehicles.

Graphic Processing Units (GPUs)

Technically a type of microprocessor, Graphics Processing Unit (GPU) is capable of rendering graphics for display on an electronic device. The GPU was introduced to the wider market in 1999 and is best known for its use in providing the smooth graphics that consumers expect in modern videos and games. Before the arrival of GPUs in the late 1990s, graphic rendering was handled by the Central Processing Unit (CPU). When used in conjunction with a CPU, a GPU can increase computer performance by taking on some computationally-intensive functions, such as rendering, from the CPU. This accelerates how quickly applications can process since the GPU can perform many calculations simultaneously. This shift also allowed for the development of more advanced and resource-intensive software and activities such as cryptocurrency mining.

Standard Chips (Commodity ICs)

Standard chips, also known as commodity ICs, are simple chips used for performing repetitive processing routines. Produced in large batches, these chips are generally used in single-purpose appliances such as barcode scanners. Characterized by razor-thin margins, the commodity IC market is dominated by large Asian semiconductor makers. If an IC is made for a specific purpose, it is called an ASIC, or application-specific integrated chips.

The SoC, the newest type of chip, is the most welcoming to new manufacturers. In the SoC, all of the electronic components needed for an entire system are built into a single chip. The capabilities of a SoC are more extensive than those of a microcontroller chip, which generally combines the CPU with RAM, ROM, and input/output (I/O). In a smartphone, the SoC might also integrate graphics, camera, and audio and video processing. Adding a management chip and a radio chip results in a three-chip solution.

Taking the other approach to categorizing chips, most computer processors currently use digital circuits. These circuits usually combine transistors and logic gates. Sometimes, microcontrollers are added. Digital circuits use digital, discrete signals that are generally based on a binary scheme. Two different voltages are assigned, each representing a different logical value.

Analog Chips

Analog chips have been mostly, but not entirely, replaced by digital chips. Power supply chips are usually analog chips. Analog chips are still required for wideband signals, and they are still used as sensors. In analog chips, voltage and current vary continuously at specified points in the circuit. An analog chip typically includes a transistor along with passive elements such as an inductor, capacitors, and resistors. Analog chips are more prone to noise, or small variations in voltage, which can cause errors.

Mixed Circuit Semiconductors

Mixed circuit semiconductors are typically digital chips with added technology for working with both analog and digital circuits. A microcontroller might include an analog-to-digital converter (ADC) for connecting to an analog chip, such as a temperature sensor, for example. A digital-to-analog converter (DAC), conversely, can allow a microcontroller to produce analog voltages for making sounds through analog devices.

Computer organization

Computer organization helps optimize performance-based products. For example, software engineers need to know the processing power of processors. They may need to optimize software in order to gain the most performance for the lowest price. This can require quite a detailed analysis of the computer's organization. For example, in a SD card, the designers might need to arrange the card so that the most data can be processed in the fastest possible way.

Computer organization also helps plan the selection of a processor for a particular project. Multimedia projects may need very rapid data access, while virtual machines may need fast interrupts. Sometimes certain tasks need additional components as well. For example, a computer capable of running a virtual machine needs virtual memory hardware so that the memory of different virtual computers can be kept separated. Computer organization and features also affect power consumption and processor cost.

Implementation

Once an instruction set and micro-architecture have been designed, a practical machine must be developed. This design process is called the *implementation*. Implementation is usually not considered architectural design, but rather hardware design engineering. Implementation can be further broken down into several steps:

- **Logic implementation** designs the circuits required at a logic-gate level.
- **Circuit implementation** does transistor-level designs of basic elements (e.g., gates, multiplexers, latches) as well as of some larger blocks (ALUs, caches etc.) that may be implemented at the logic-gate level, or even at the physical level if the design calls for it.
- **Physical implementation** draws physical circuits. The different circuit components are placed in a chip floorplan or on a board and the wires connecting them are created.
- **Design validation** tests the computer as a whole to see if it works in all situations and all timings. Once the design validation process starts, the design at the logic level are tested using logic emulators. However, this is usually too slow to run a realistic test. So, after making corrections based on the first test, prototypes are constructed using Field-Programmable Gate-Arrays (FPGAs). Most hobby projects stop at this stage. The final step is to test prototype integrated circuits, which may require several redesigns.

Cache memory vs Registers

Fastest memory is cache memory.
Registers are temporary memory units that store data and are located in the processor, instead of in RAM, so data can be accessed and stored faster. Cache memory is extremely fast memory that is built into a computer's central processing unit.

Cache memory is a small-sized type of volatile computer memory that provides high-speed data access to a processor and stores frequently used computer programs, applications and data. It is the fastest memory in a computer, and is typically integrated onto the motherboard and directly embedded in the processor or main random access memory