CS1033: Programming Fundamentals

<u>Part C:</u> Computer Systems & Hardware

BSc Engineering - Semester 1 (2020 Intake)
August – December 2021

Department of Computer Science and Engineering

Faculty of Engineering

University of Moratuwa

Sri Lanka

Compiled by:

Chathura De Silva, Sanath Jayasena, Thirasara Ariyarathna, Kalpani Anuradha, Sachini Herath, Madhushi Bandara, Hashini Senaratne, Yasura Vithana, Sandareka Wickramanayake, Gayashan Amarasinghe.

Acknowledgements:

All help and support provided to make this Course Note possible are acknowledged.

Sections of this note were based on:

- "Part C: Computer Systems & Hardware" Course Notes (2015 version) from the Dept of Computer Science & Engineering, compiled by Chathura De Silva and others.

Note: Feedback is very much appreciated for improvement of this document. Please email your feedback at sanath@cse.mrt.ac.lk.

Date of this Part C version: Wednesday, 13 October 2021

Table of Contents

1 - Introduction

Today, computers are everywhere. Apart from computers that you generally see, you can find computers even in unlikely places including automobiles, home appliances, mobile phones, and cameras. Computers are reshaping our lives at home, workplace, school and on the road. Most business and public organizations in Sri Lanka use computers in some way; most of them are networked internally and connected to the Internet. As computers are getting more advanced and capable as time goes by, they will be affecting our lives even more in the future.

Often today we use computers without even knowing. This is because most of today's day-to-day consumer devices such as televisions, audio systems, mobile phones, air-conditioners, washing machines, microwave ovens etc. have computers built-inside them to control their electrical and mechanical functions. When we use these machines, we in fact interact with their internal computers to tell the device what we want. Since these computers are not directly visible as regular "computers" they are also sometimes called "invisible computers" or more technically as "embedded systems".

1.1 What is a Computer?

The Oxford English Dictionary says, "*Computer is an electronic device for analyzing and storing data, making calculations, etc*.". The word computer comes from the word *compute* which means *calculate*. In basic terms we can define a computer as *an electronic device that processes data, converting it into information useful for humans or other systems*. Some important features of a computer are as follows.

- It is a machine.
- It can be "programmed"; that is, it can (understand and) execute a finite set of "instructions".
- It can "process" (perform operations on) "data" according to those "instructions".
- It can execute a "sequence of instructions in a given order" that are "stored" within the machine.
- The actual "order of execution of instructions" in a given sequence of instructions can depend dynamically on data, results of execution of previous instructions or some events.
- It can execute a sequence of instructions "repeatedly", as many times as instructed or forever.
- It can execute instructions very fast (for e.g., billions of instructions per second)

A computer is physically a combination of mechanical, electrical, and electronic components. Computers cannot think as humans or other beings. Computers (i.e., computer hardware) by themselves do not have any intelligence on them or "think" on their own. Their 'intelligence' come from to the features listed above and therefore depends much on the humans who make it work; what a computer can do is limited by the abilities of those humans. However, computers can execute their instructions extremely fast and at pin-point accuracy. Thus, computers generally seem as far ahead of humans due to this high speed and accuracy of computations.

1.2 Uses of Computers

Computers were initially invented and used only for complex calculations by scientists and researchers. However modern computers are being used many different purposes; at home, education, research, business and so on for numerous purposes. We can even categorize computers based on the way they are used.

A student can use the computer to: find information for his/her studies from the Internet or educational compact discs; read electronic-books (e-books); access e-learning resources and activities; play computer games; listen to music and watch movies. They can also: create their own documents, music, movies, and animations; draw pictures and design web pages; communicate, chat, share electronic content with friends and others. Teachers can use computers to make their work more effective.

An employee can use computers to do her day-to-day work such as: preparing reports and letters; printing bills; managing stock; accounting; communicating with co-workers, customers, and suppliers; preparing multimedia presentations. Doctors use computers to keep track of their patients' history, study about diseases and build computerised models of the human body for various simulations. Engineers and scientists use computers to perform complex calculations and simulations; do technical drawings; build computerized models; forecast weather; analyse structures and pictures; send rockets to outer space, and many more. Artists use computers to produce their creative work such as music, movies, 3D animations, pictures, photographs, cartoons, and so on.

In the modern world, computers are also used to control and manage other devices. For instance, most of modern-day household equipment such as washing machines, microwave ovens, televisions, air-conditioners etc use a small computer inside them to control their functions. Such computers are also used in automobiles, airplanes, and trains. It is said that av average modern car has about 50 different computers inside them controlling various components such as the engine, fuel-flow, breaks, safety equipment such as airbags etc. These computers receive inputs from number of sensors (such as temperature sensors, water level sensors, air-flow sensors) and the input provided by the user (i.e., though various buttons and dials in the equipment) and then decides how different parts in the machine (like motors, heating / lighting elements etc) should be operated to provide the desired function expected by the equipment.

Basically, modern computers can do many things faster than humans. If something cannot be done with today's technology, it might be possible in the near future with the advancement of technology as computers are evolving so rapidly.

1.3 Different Types of Computers

The most common type of computers that we see today (i.e., the computers that we see as separate machine, excluding those built inside other devices) are the "Personal Computers" or "PCs" in short. These were first introduced by a company called "International Business Machines" or "IBM" in the early '80s and were intended to be used by a single person. Before the introduction of the PC, computers usually were large machines, expensive occupying larger spaces and used generally owned and used by large organizations. The PC on the other hand was affordable, small, and thus could be owned and used by an individual or a smaller business organization.

The concept of a personal computer became very popular resulting many other companies to manufacture and market similar devices. These in most cases were "compatible" with the IBM PC such that the software and most of the hardware components computers that were originally meant for IBM PC could also be used with them or vice-versa. These were generally referred to as "IBM-compatible" personal computers.

The IBM PC and its compatibles were built around a family of microprocessors referred to as the "8086" manufactured by the INTEL corporation and therefore its structure is commonly referred to as the "x86 architecture". Therefore, also other breads of computers built around different microprocessor families during same time but did not become popular as the PC. One such system that got notable degree of popularity and even exist today is the "Macintosh" invented and marked by the Apple corporation. The software used by these systems in general is not directly compatible with the PC due to the differences in the microprocessor and the hardware architecture used by them.

The original personal computer has evolved a lot into different sub-types of computers in the recent years, and it has become more difficult even to define the term "PC". In general, however, it applies to any computer that is generally intended to be owned and used by an individual. For all other required hardware components and all software (including the operating system), there are several options to build a working PC.

Today the following basic types of **personal computers** can be identified.

- **Desktops**: These are computers designed to sit on (or under) a desk and the type found mostly in offices, computer labs and homes. They are too big to be carried around.

- **Workstations**: These are specialized, desktop computers that have more power than standard desktops and popular among scientists, engineers, and digital artists. These often have large, high-resolution monitors and accelerated graphic-handling abilities making them suitable for advanced architectural or engineering design, simulation, animation, and video editing.

- **Laptops** (or "notebooks"): These look like a notebook, fit inside a briefcase, can be used on our lap, can weigh less than 3kg and have the power of desktops. They can operate on AC power or special batteries. Due to their portability, we can categorize them under *mobile computers*.

- **Tablet computers**: These offer the functionality of a laptop but lighter and can accept input from screen (either via fingers or a special pen called a *stylus* or *digital pen* which is used to tap or write directly on the screen). These are gaining popularity among professionals who need to take lots of notes and deal with hand-written documents.

- **Handheld computers**: These are so small and fit in our hands. A popular type of handheld computer is the *personal digital assistant* (PDA). They are used for special applications such as taking notes, keeping contact information, tracking schedules and tasks, quick email and web browsing, voice recording. They often can be connected to a larger computer and exchange data. Most today's handheld devices let the user input by touching the screen.
- **Smart phones**: These are cellular (mobile) phones that have advanced features and can be considered as a kind of handheld computers. They may allow web and email access, specialized software for personal organizing and special hardware such as cameras and music players.
- **Wearable computers:** The latest trend in computing is wearable computers. Essentially, common computer applications (e-mail, database, multimedia, calendar/scheduler) are integrated into watches, cell phones, visors and even clothing. E.g.: Google glass, Pebble smart watch.

In contrast to the personal computers above, some computers handle the needs of many users at the same time. These are powerful and often used by organizations such as businesses, educational institutions, and government offices. Generally, each user interacts with such a computer through a personal device or computer via a network.

The following types of **computers for organizations** can be identified in today's world.

- **Network servers**: Most organizations have networks based on personal computers that are connected to one or more centralized computers called network servers. These are powerful personal computers with special hardware and software that enable them to function as primary computers in the network. They may be setup in groups called *clusters* or *server farms*.
- **Mainframe computers**: These are used in large organizations such as banks or airlines where many people, even thousands, frequently need to access the same data. Traditionally, each user accesses a mainframe computer through a device called a *terminal*.
- **Minicomputers**: In 1960s, this name was used for computers smaller than mainframes (personal computers were not available then). These have capabilities between those of mainframes and personal computers; now they are also called *midrange computers*.
- **Supercomputers**: These are the most powerful, expensive computers available and physically the largest. They can process huge amounts of data and may contain thousands of processors. They are used for complex computations such as mapping the human genome and forecasting weather. They are used in research centres and very large businesses in some countries in the world. E.g.: IBM Roadrunner, Cray Titan, Tianhe-IA.

In addition to the above types, there are computers not directly used or seen by human users. Instead, they reside in a system such as an automobile or a household appliance and interact with its environment to perform specific tasks like controlling the system operations; humans may indirectly or unknowingly use them. These devices are also referred to as "smart" devices because the computer inside enable them to interact with the environment and users in a more intelligent, adaptive, and efficient way. The software running inside these computers provides multiple advance functionalities, sometimes more than one function on eth same device. For instance, a modern smart TV is not only used for watching television but also as a communication device to access the Internet, access social networks such as Facebook or make video calls with other users. These types of computers are referred to as **embedded computers**.

1.4 Components of a *Computer System*

A computer is a *system*. A complete *computer system* consists of the following essential components.

- **Hardware**: These are the physical (or tangible) elements that can be seen and touched.
- **Software**: These are the programs that allow users to use the computer system and control its activities. Software cannot be seen or touched. A program is an ordered sequence of instructions that the hardware can execute. Software can be further categorized as system software and application software.
- **Data**: These are the individual facts or pieces of information that the computer takes as input for processing or produces as output after processing.

- **User** (or users): Users are also a critical component in a computer system.

We will discuss in detail about hardware in Chapter 3 and about software in Chapter 4.

1.5 Computers and *Information*

Why are computers so important? We can list so many reasons for this. For some examples, recall what we discussed in Section 1.2 *Uses of Computers*. When we take all the benefits we gain from computers and study them closely, we get into a single element: *information*.

Computers are so important because information is so vital to our lives in today's world. Contents in a newspaper, for instance, are only one type of information. It is also more than what we see on television and hear on radio. Mathematical formulas, contents in books, plans for buildings, designs for engines, electrical circuit layouts, cake recipes, pictures, songs, games, addresses, telephone directories: all these are information. They can be stored and processed by computers; they can also be sent to computers or users so far away using networks such as the Internet, provided there is a communication link. The ability to process and transfer information this way is changing the world and bringing people together into a *global village*. The computer and communication technologies that made this possible are together referred to as *information technology* or *IT* (or, sometimes as *information and communication technology* or *ICT*). Computers are therefore at the heart of IT.

Even long time ago, before computers existed, successful organizations knew how important it is to "manage" information to achieve their objectives. They had established policies, procedures, techniques, and resources to collect, store, process and disseminate information. These are collectively known as an *information system*. An information system may not necessarily involve computers; but an efficient and effective information system today requires appropriate use of computers. Many organizations now have information system (IS) departments and IS managers. In some organizations, especially those that identify information as a strategic tool, the information system is managed by a chief information officer (or CIO).

2 - Introduction to Computer Hardware

Hardware is the combination of all physical components that makes up a computer, monitor, printer, etc. A modern computer consists of various hardware components. Some of these components are essential for the proper functionality of a computer while some are optional. These hardware components are governed by various electrical, electronic, and mechanical engineering principles. All these principles combined with engineering excellence produces state of the art hardware components.

This chapter introduces the major hardware components in modern personal computers, their usage, design concepts, operations, capacities, and speeds. Rather than following the standard approach of introducing the processor first, we will start with introducing the motherboard and memory before introducing the processor.

2.1 Major Components of a Personal Computer System

A personal computer system consists of several different parts. Some of these parts can be considered as subsystems within the personal computer system. The following are the common subsystems and parts within a PC (figure 2.2 and 2.3).

- Mother board / System Board (i.e., the main circuit board)
    - Central Processing Unit (CPU / Microprocessor)
    - Memory
        - Random Access Memory (Read/Write memory) (RAM)
        - Read Only Memory (ROM)
    - Graphic / display controller
    - Network / Wi-Fi controllers
    - Audio Interface (Sound card)
    - Universal Serial Bus (USB) controller
- Input / Output sub system (i.e., devices that enable the computer to interact with its users)
    - Display Monitor (VDU)
    - Keyboard
    - Mouse
    - Scanner
    - Printer
    - Microphone / Speakers
    - Joystick / Game controller
- Secondary storage devices (i.e., devices that enable log-term storage of large volumes of data)
    - Solid State Drive (SSD)
    - Hard Disk Drive (HDD)
    - CD-ROM / DVD drive
    - Floppy disk drive
    - Tape drives
- Chassis (i.e., the platform that mounts all other components)
    - Power supply unit
    - Fan / cooling system

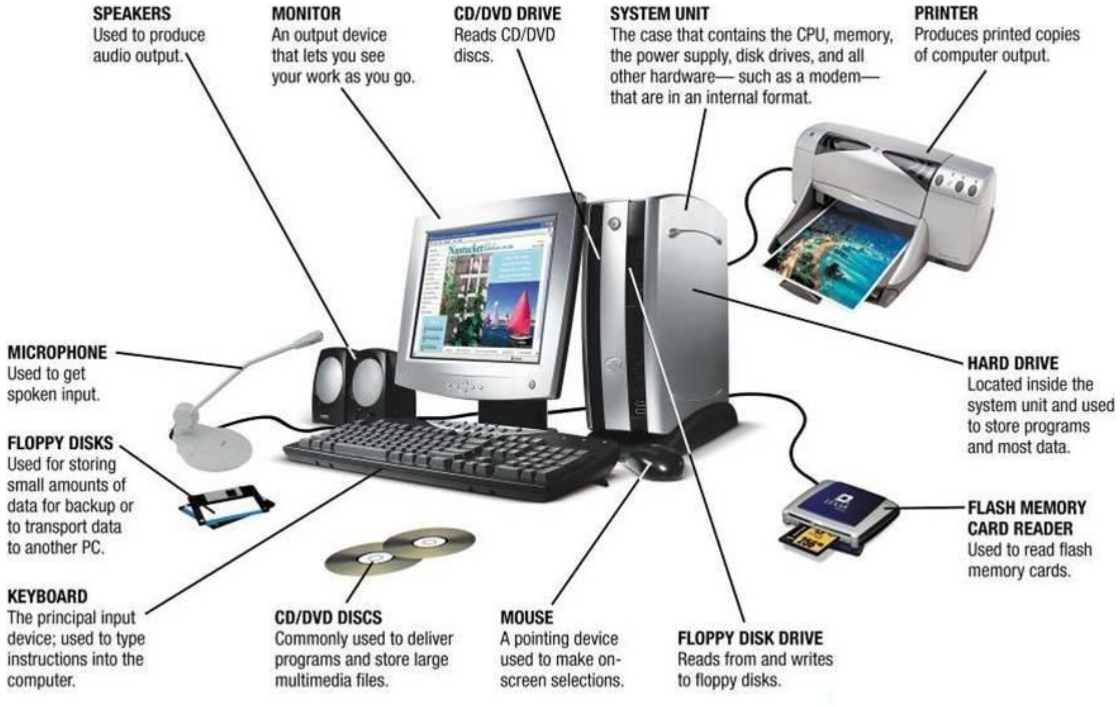

SPEAKERS
Used to produce audio output.
MONITOR
An output device that lets you see your work as you go.
CD/DVD DRIVE
Reads CD/DVD discs.
SYSTEM UNIT
The case that contains the CPU, memory, the power supply, disk drives, and all other hardware— such as a modem— that are in an internal format.
PRINTER
Produces printed copies of computer output.
MICROPHONE
Used to get spoken input.
HARD DRIVE
Located inside the system unit and used to store programs and most data.
FLOPPY DISKS
Used for storing small amounts of data for backup or to transport data to another PC.
FLASH MEMORY CARD READER
Used to read flash memory cards.
KEYBOARD
The principal input device; used to type instructions into the computer.
CD/DVD DISCS
Commonly used to deliver programs and store large multimedia files.
MOUSE
A pointing device used to make on-screen selections.
FLOPPY DISK DRIVE
Reads from and writes to floppy disks.

Figure 2.2 – External view of a PC

2.2 The Traditional View of a Computer System

The architecture of modern computers is somewhat different to traditional ones. However, traditional computers are easier to understand compared to the modern ones. Therefore, first we will look at concepts of a traditional computer.

The traditional view of a computer system can be classified as, *single user computer systems* and *multi-user computer systems*.

2.2.1 The Traditional View of a Single User Computer System

These are said to be single user computer systems because only a single user can use them at a time. Figure 2.4 illustrates different subsystems of a single user computer system.

As shown in figure 2.4 the Central Processing Unit (CPU) communicates with all the other subsystems and manages the overall functionality of the computer system. The CPU is like the processing elements of a human brain. The Video Display Unit (VDU) is responsible for producing the visual output. The VDU, keyboard and printer belongs to the Input Output (IO) subsystem. These are analogues to the sensors (e.g., nose) and actuators (e.g., hands) of a human.

The memory subsystem of the computer is composed of the Random Access Memory (RAM) and the Read Only Memory (ROM). Like the human brain, computers also need a temporary memory system (provided by RAM) and a permanent memory system (provided by ROM).

The RAM keeps all the current data and instructions in memory while ROM keeps all the data and instructions that need to be there for a long-time. RAM loses all the stored data and instructions when the computer is switched off. The capacity of both RAM and ROM is limited but computers demand for more data and instruction storage. As a result, secondary storage was introduced.

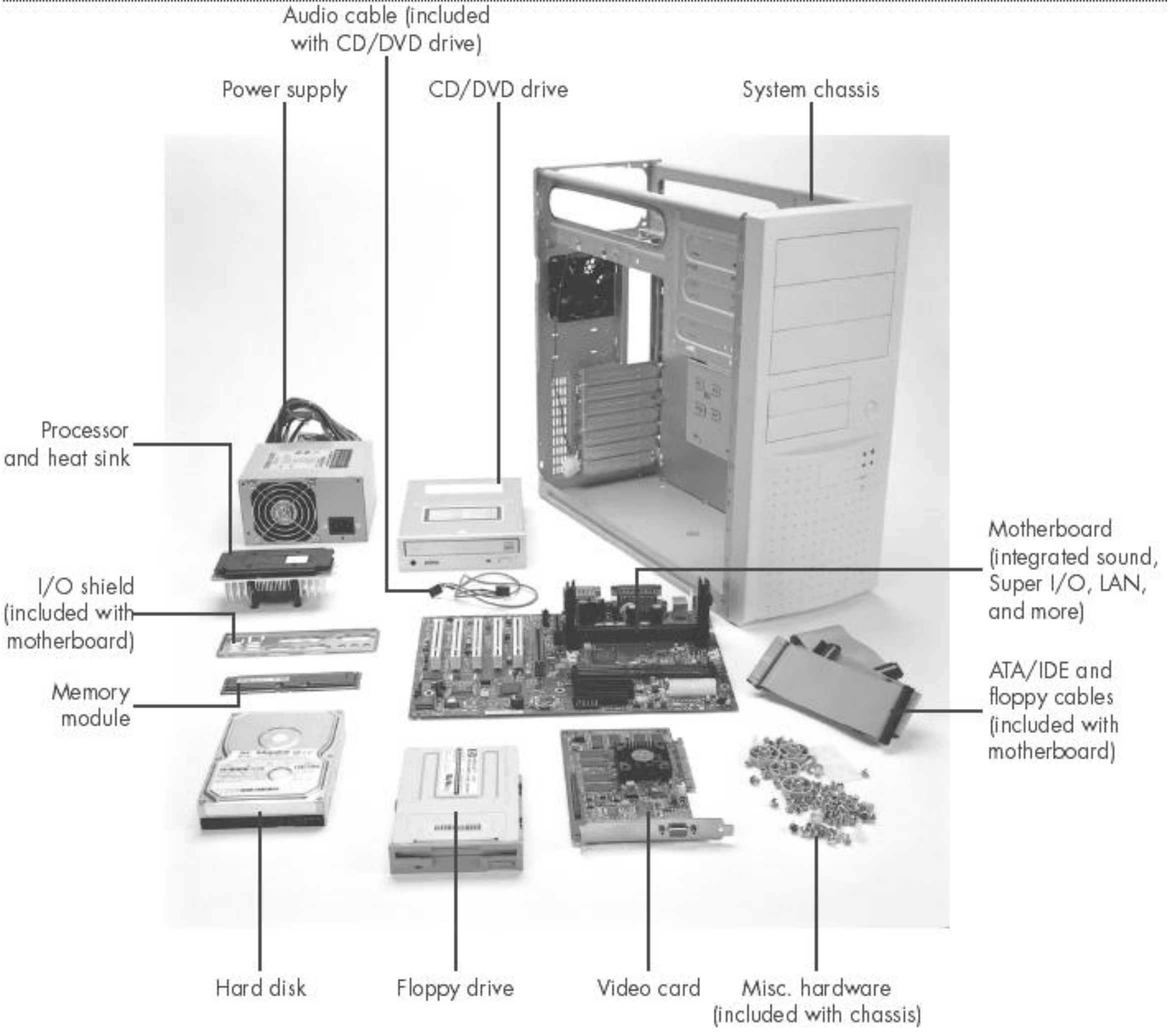

Audio cable (included with CD/DVD drive)
Power supply
CD/DVD drive
System chassis
Processor and heat sink
I/O shield (included with motherboard)
Memory module
Motherboard (integrated sound, Super I/O, LAN, and more)
ATA/IDE and floppy cables (included with motherboard)
Hard disk
Floppy drive
Video card
Misc. hardware (included with chassis)

Figure 2.3 – Parts of a PC

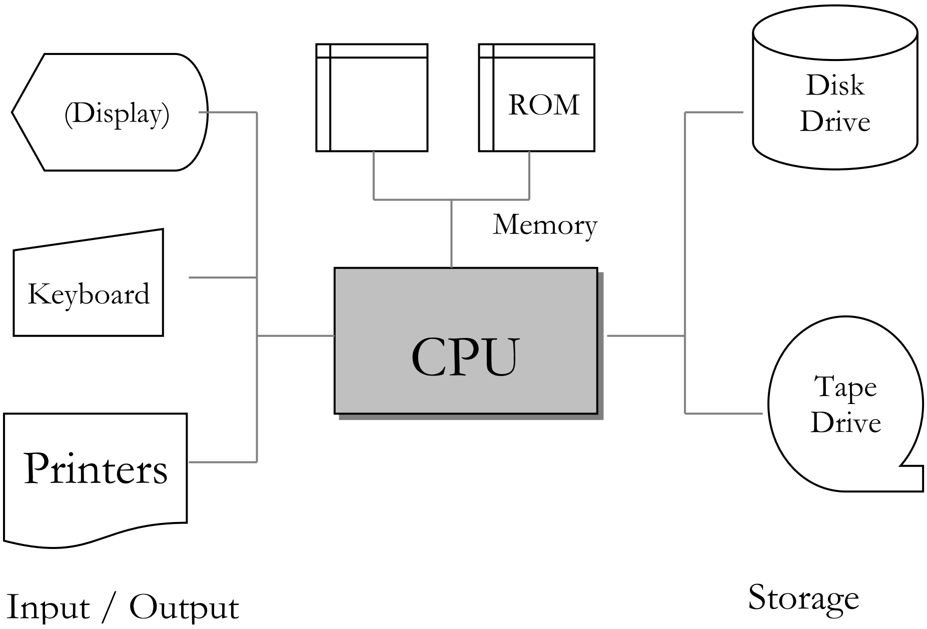


Figure 2.4 – Traditional view of a single user computer system

Secondary storage mainly includes disk drives (solid state drives, hard disks, and floppy disks) and tape drives. Hard disks are a high capacity, fast and permanent storage medium. The concepts behind the hard disk will be discussed later. Tape drives (also called DAT drives) are used store data on magnetic tapes.

Magnetic tapes were introduced before hard disks, and it was used to store both data and instructions. Magnetic tape is a high capacity and cost-effective data storage solution. However, the data reading/writing speed of a magnetic tape is slower therefore nowadays tapes are used only to back up large amounts of data.

The CPU is the central unit, and it communicates with all the other systems and subsystems. It is also responsible for controlling and managing rest of the system. This is a very simple approach to understand and implement but it does not scale well since everything depends on the CPU.

2.2.2 A Modern Computer System

With the introduction of personal computers and miniaturization, most of the subsystems were included in a box, which is called the 'Machine' (see figure 2.6). Most people refer to this box as the CPU, which is an incorrect term. The CPU is just a single piece of component inside the box.

Modern computer systems contain many additional parts that are not identified with the traditional view of a computer system. These include Video sub-system, Multimedia devices such as Sound cards, speakers, microphone, CD-ROM/DVD-ROM, networking adapters such as Network Interface Cards (NICs) and Modems, and communication ports such as serial and parallel ports.

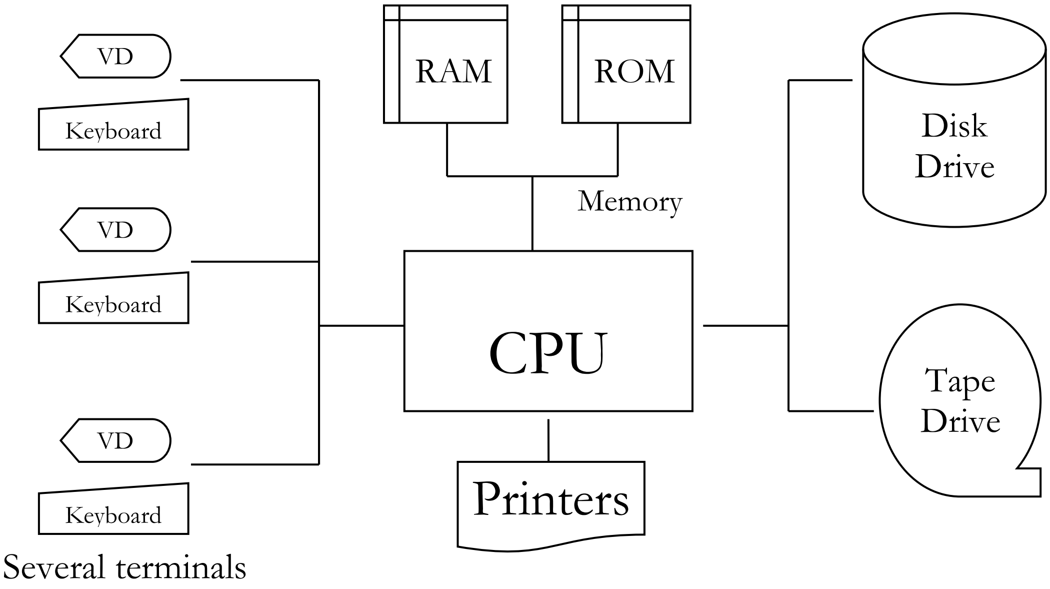


Figure 2.5 – Traditional view of a multiuser computer system

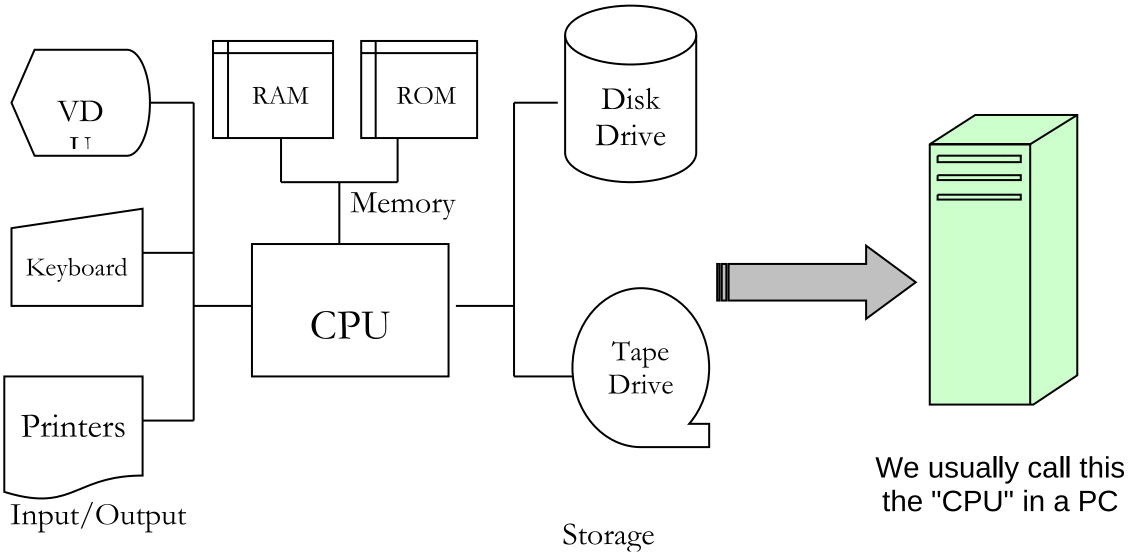


Figure 2.6 – From traditional view to a modern computer system

2.3 The Motherboard

The motherboard is the most important component in a PC, and it is also called the 'main board'. The motherboard is a large circuit board where the processor, memory and other electronic components are attached. Several other ICs called controllers are attached to the motherboard as well. These controllers are responsible for:

- controlling the hard disks and other secondary storage devices
- communicating with input/output devices (keyboard, mouse, printers, modems, etc.)
- supporting operations carried out by the micro-processor (e.g.: Direct Memory Access controller)

The motherboard provides the path through which the processor communicates with memory, disks, expansion cards, keyboard, and other components. Figure 2.7 illustrates the layout of a typical motherboard.

2.4 Memory

As human beings have memory, computers also have some sort of memory. Part of this memory is permanent (non-volatile) and the other portion is temporary (volatile). The volatile memory loses its content whenever power is switched off.

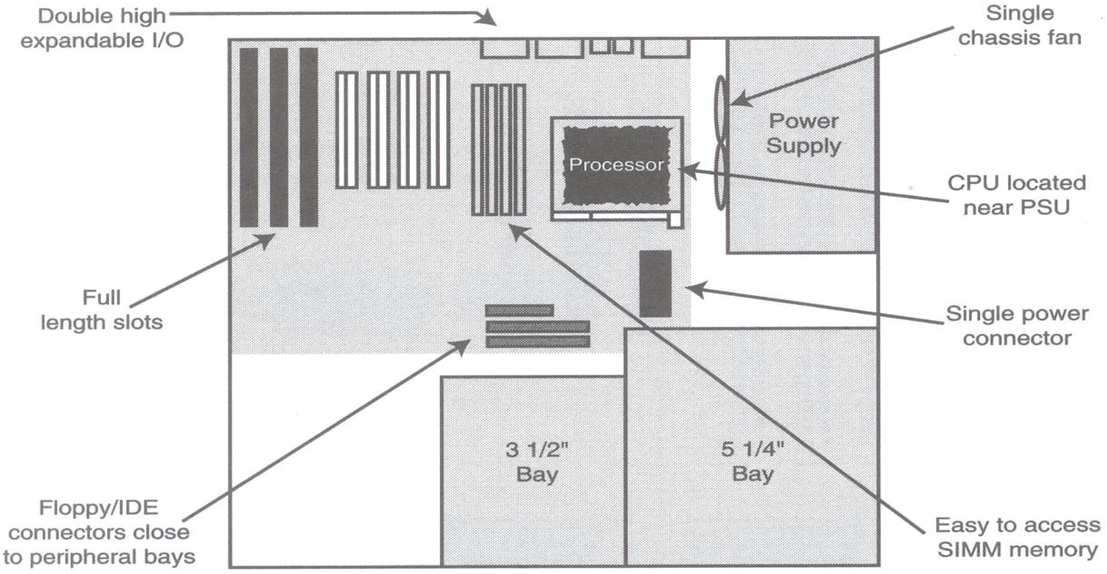


Figure 2.7 – Layout of a typical PC motherboard

Memory is a temporary storage area for programs and data being operated by those programs. Whenever the microprocessor wants to do some processing, it gets both data and instructions from the memory and executes them. After the execution, the results are sent back to the memory. Therefore, the memory is considered as the workspace (or the playground) for the microprocessor.

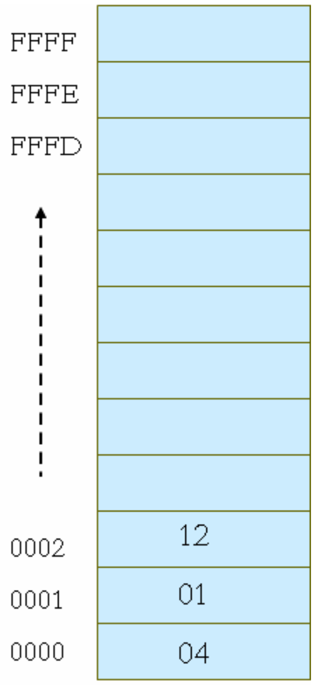


Figure 2.8 – Array of memory locations

Memory consists of an array of consecutive memory locations. Each such location stores a single piece of data (usually a byte). Memory locations are identified by a unique memory address. The first memory location is labelled as memory address zero (0x0000) and rest of the memory locations are labelled one after another. The addresses are normally represented as hexadecimal numbers. In figure 3.8 there are 65536 (2^{16}) memory locations where the first address is 0x0000 and the last address is 0xFFFF. Memory address 0x0000 holds the value 0x04 while memory addresses 0x0001 and 0x0002 hold integers 0x01 and 0x12.

The microprocessor uses memory to store Instructions (programs) and Data (characters and digits). At a time, the microprocessor either reads or writes only one memory location. In the long run it may need to access each memory location within the memory array. However, it is impractical to connect each memory location directly to the microprocessor using a separate set of electrical connections (i.e., wires). In figure 2.8 there are 65536 memory locations, therefore, to connect each memory location 65537 wires are required (assuming a common ground wire). To overcome this issue all the memory locations are connected using shared electrical connections, called the "*Memory Bus*".

2.4.1 Memory Bus

Following is a hypothetical example (figure 2.9) of a bus which runs on a two-way road, and which accommodates only one passenger. Assume there is a passenger '**A**' in 'house ①', who wants to go to 'house ③'. '**A**' can go to the road and get the bus to 'house ③'. When he arrives at 'house ③' he can get off the bus. During this time bus is dedicated only to passenger '**A**' and no one else can travel in the bus. Suppose passenger '**B**' also wants to go the 'house ③'. Then she must wait until the bus is free. When it is free, she can use the bus.

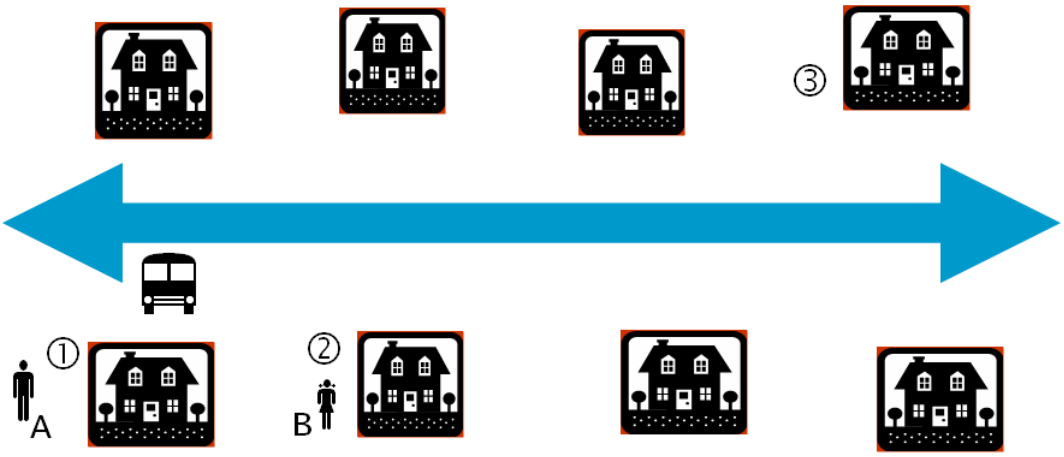


Figure 2.9 – Illustration of a Bus

Now suppose that both '**A**' and '**B**' wants to go to 'house ③' at the same time. In a conventional public transport system '**A**' will always get the bus first since he is at the starting point of the road. However, the bus in a computer is slightly different. There is no specific starting point. Whoever comes first to the road (even just fraction of a second before other passenger) will always get the bus and the other one has to check from time to time whether the bus is free. While the second passenger is waiting if a third passenger comes and tries access the road (just after bus gets free) he/she will get a chance, where the waiting passenger may still need to wait. If the waiting one is unlucky, he/she has to wait forever. This is not a fair deal. Therefore, the microprocessor (or bus controller) decides who is going to access the bus at what time.

The bus inside the computer is designed based on the above algorithm. The road is bidirectional and at a time only one memory location can access the bus. A bus is a set of electrical connections (parallel set of wires or set of copper strips on the motherboard) that connects the memory and the microprocessor. There are 3 types of busses; namely *Address Bus*, *Control Bus* and *Data Bus*. All 3 buses are enclosed within the memory bus (figure 2.10).

- Address Bus – is used by the microprocessor to indicate the memory location (indicated by the memory address) that it is interested in. Address bus goes from CPU to memory.
- Control Bus – is used to send control information such as read request (RD) or write request (WR) to the memory. Control bus goes from CPU to memory.
- Data Bus – used for actual data transmission. This is a bidirectional bus.

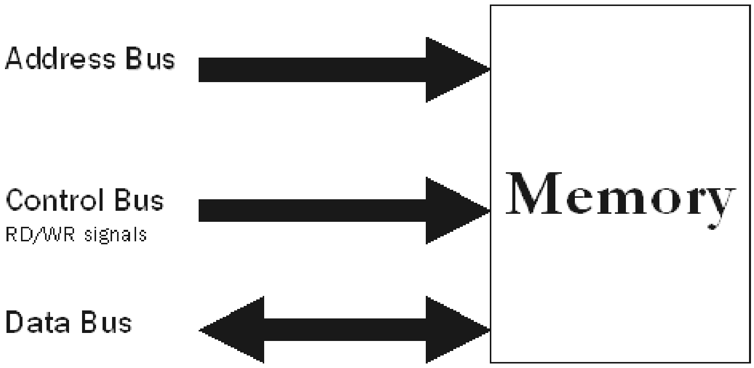


Figure 2.10 – Communicating with the memory

The operations of writing data to the memory and reading from the memory make use of all three buses.

Writing data to the memory

1. First the address of the memory location is placed on the Address Bus
2. Then data is placed on the Data Bus
3. Finally, the CPU activates the Write Signal (WR) in the control bus

Reading data from the memory

1. First the address of the memory location is placed on the Address Bus
2. Then the CPU activates the Read Signal (RD) on the control bus
3. Finally, data is fetched (read) from the Data Bus

2.4.2 Types of Memory

Several different types of memory are used for various purposes within a computer or microprocessor-controlled system. These differences are characterised based on their manufacturing process, volatility, programming methodology and cost.

Read Only Memory – ROM

As its name implies these memories is read only. They are written (developed as pre-wired electronic circuits, referred as *hard coded*) when they are being fabricated as ICs. The ICs that are available in the open market with various melodies (i.e., piece of music) belong to this category. In computers, these memories are used to store initial start-up programs of the computer[1]. Since these memories are hard coded, it is not economical to produce these in small quantities.

Programmable Read Only Memory – PROM

These memories are same as ROMs, but their contents can be written once after the individual ICs are manufactured. The programming process requires special equipment. If a smaller number of ROMs are required for a particular application, programming several PROMs is a more cost effective.

UV Erasable PROM – UVEPROM

[1] Referred as the POST – Power On Self Test. A series of tests run by the computer at power-on to check whether the attached hardware components are working correctly.

UVEPROM is similar to PROM but contents can be written several times. Before writing new content, existing program should be erased by exposing the IC to Ultra Violet (UV) light. For this purpose, there is an opening at the top of the IC and this is normally coved by a sticker, which does not allow UV light to penetrate (figure 2.11). During the content erasing process, this protective cover should be removed before exposing the IC to the UV light. If this protective cover is damaged the content in the IC may get corrupted after a while. Both erasing and programming processes require special equipment.

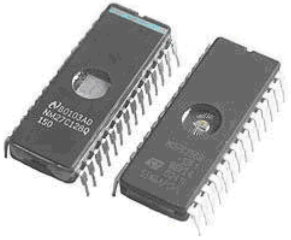


Figure 2.11 – UVEPROM

Electrical Erasable PROM – EEPROM

Same as the UVEPROM, except that the content (the program) of the IC can be erased by applying a special high voltage to some of the pins. The programming process requires the EEPROM to be removed from the application circuit and plugged in to a special device before erasing the content.

In modern televisions, we need this sort of memories to keep track of the current channel, volume level, colour configurations, etc. However, it is not practical to remove the EEPROM from the TV and connect to a special programming circuit. The flash ROM was introduced as a solution.

Flash ROM

Flash ROM is a special type of EEPROM that can be erased or programmed while in the application circuit. Once programmed the contents remains unchanged even after a power failure. Flash ROMs are commonly used in modern PCs, various networking devices such as routers and firewalls and memory pens (also referred as memory sticks or USB pens).

Read Write Memory – RWM

These are traditionally known as the RAM (Random Access Memory). Contents in these memories are erased when power is disconnected. There are two major types of RAMs; the *Static RAM* (SRAM) and *Dynamic RAM* (DRAM). Static RAM is developed using transistors while dynamic RAM is developed using capacitors. Therefore, they reflect the properties of transistors and capacitors (see Table 2.1).

Table 2.1 – Transistors vs. Capacitors

Transistors	Capacitors
Developed using Silicon and other semiconductors	Developed using Silicon and other semiconductors
High speed switching	Slower performance
Will retain state forever (if power is supplied)	Automatically discharges after some time, need refreshing
More reliable	Less reliable
Low density (no of transistors per cm^2)	High density (no of capacitors per cm^2)
High power consumption	Low power consumption
High cost (per bit)	Low cost (per bit)

Static Ram – SRAM

As said earlier static RAMs are built using transistors. Each transistor represents a single unit of data, which is a bit. Arrays of transistors are combined to produce SRAM. Since these are developed using transistors, they inherit all the properties of transistors in Table 2.1.

Dynamic RAM – DRAM

DRAMs are developed using capacitors. Each capacitor represents a single unit of data, and they process all the characteristic of capacitors listed in Table 2.1. Capacitors are cheap but they must be re-charged from time to time (certain DRAMs are needed to be refreshed at 15μs intervals). Due to its lower cost, bulk of the PC memory is made from DRAM.

SRAMs are more reliable but expensive than DRAMs. They also consume more electrical power compared to DRAMs. These are mostly used as cache memories.

2.4.3 Memory Modules

Memory modules combine set of memory ICs together and present it to the motherboard as a single memory block. Various memory modules such as SIM (Single Inline Memory Module), DIMM (Dual Inline Memory Module) and DDR-DIMM (Double Data Rate-DIMM) are available. Different memory modules have different number of connections, speeds and access times. Figure 2.12 illustrates a DIMM (Dual Inline Memory Module)[2].

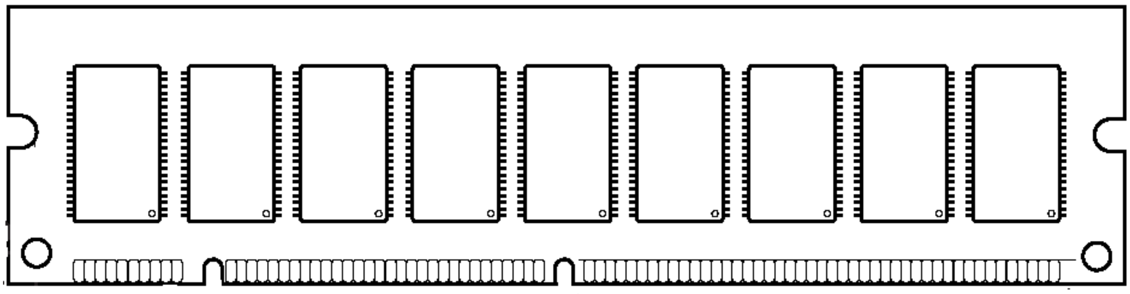


Figure 2.12 – A Dual Inline Memory Module (DIMM)

2.4.4 Memory Characteristics

To compare different types of memories, their characteristics need to be considered. A single type of memory does not process all the preferred characteristics. For example, Static RAM has a very high access speed, but it is also high in cost. Therefore, a mix of different types of memories has to be used to achieve an optimum result. Listed below are some of the important characteristics of memory:

- Access speed – time taken for the CPU to read from or write to memory
- Cycle time – time taken to complete one memory access operation
- Packing Density - memory capacity per unit area
- Power consumption
- Cost - cost per unit of memory capacity

A memory controller is used to control the communication between different types of memory and the microprocessor (see figure 2.13).

2.4.5 Memory Hierarchy

Modern CPUs are much faster than the speed of memory. The memory has to be organised in such a way that its slower speed does not reduce the performance of the overall system. Some memory types such as Static RAM are faster but expensive in contrast, Dynamic RAM is cheaper but slow. The ultimate objective of having a memory hierarchy is to have a memory system with a sufficient capacity and which is as cheap as the cheapest memory type and as fast as the fastest memory type. The main idea is to use a limited capacity of fast but expensive memory types and a larger capacity of slow but cheap memory types. Special methods are used to store the frequently used items in the faster devices and others in slower devices.

[2] DIMM is a 64-bit wide 168-pin memory module used in new PCs.

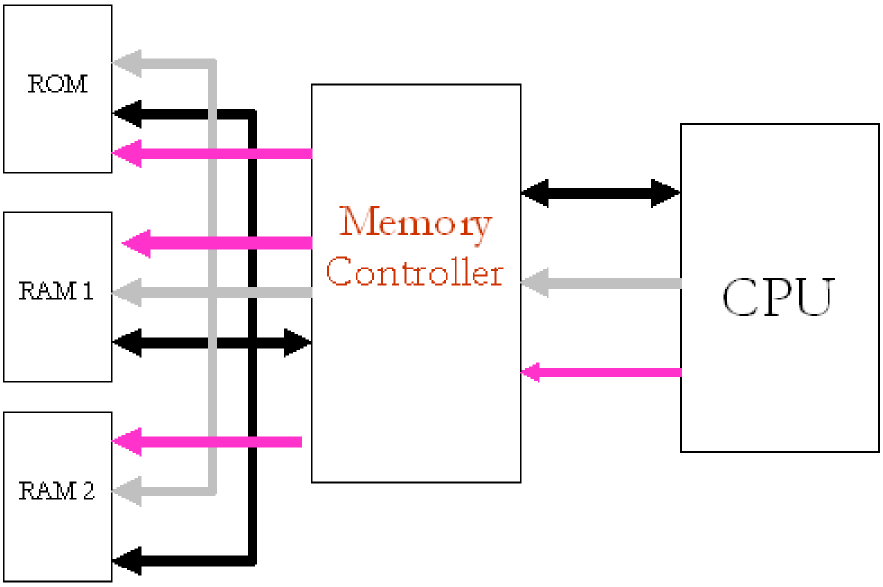


Figure 2.13 – Connecting memory and the microprocessor

Traditional Memory Hierarchy

The traditional memory hierarchy is shown in figure 2.14. The memory access speed gap between secondary storage and registers are filled up by the main memory. Going up the memory hierarchy memory access speed and cost per Megabyte increase. However, the memory capacity (size) increases from top to bottom. In this hierarchy, area of each level is proportional to the memory capacity. Capacities of registers are given in bits (8-bit, 16-bit, 32-bit, 64-bit and 128-bit registers) while capacity of main memory is given in Megabytes (8MB, 16MB, 32MB, 64MB, 128MB, 256MB, etc.). Secondary storage capacities are given in Gigabytes (GB) or sometimes in Terabytes (TB).

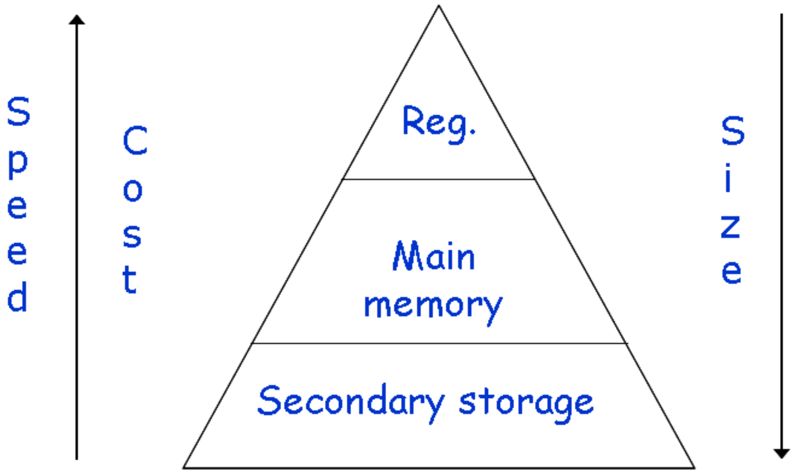


Figure 2.14 – Traditional memory hierarchy

In early days of computing, the speed gap between the memory and the microprocessors was not a big issue. Currently speed of microprocessor increases much faster (speed doubles in every 18 months) compared to the speed of memory access. Therefore, the speed gap is widening day-by-day. Although microprocessors are becoming faster and faster, they have to depend on memory since memory is the workplace for the microprocessor. This will slow down the fast microprocessors resulting in overall performance degradation. To fill up this widening gap another level called the *Cache memory* is added to the memory hierarchy.

Modern Memory Hierarchy

Modern memory hierarchy includes another level called the ‘Cache memory’ in-between registers and main memory (figure 2.15). It is a small amount of memory (capacities are given in Kilobytes; common capacities are 128KB, 256KB and 512KB) which is faster than the main memory but slower than the registers. Cache memory fills up the speed gap between main memory and registers.

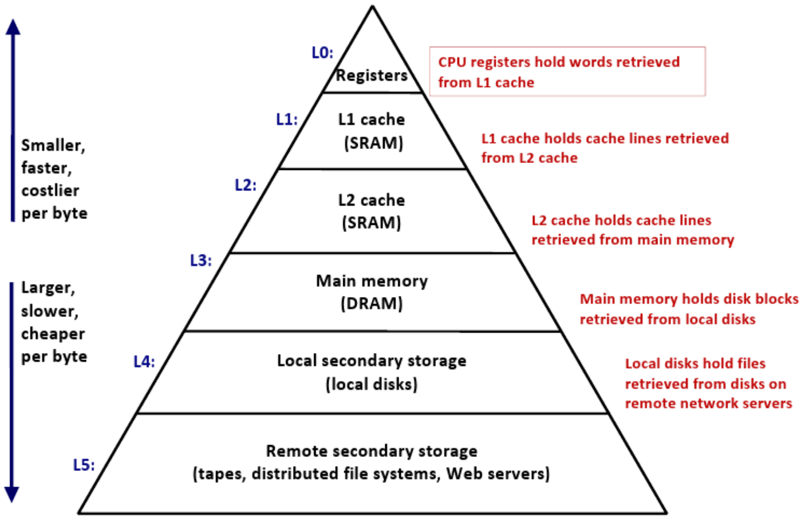

L0:
Registers
CPU registers hold words retrieved from L1 cache
L1:
L1 cache (SRAM)
L1 cache holds cache lines retrieved from L2 cache
L2:
L2 cache (SRAM)
L2 cache holds cache lines retrieved from main memory
L3:
Main memory (DRAM)
Main memory holds disk blocks retrieved from local disks
L4:
Local secondary storage (local disks)
Local disks hold files retrieved from disks on remote network servers
L5:
Remote secondary storage (tapes, distributed file systems, Web servers)
Smaller, faster, costlier per byte
Larger, slower, cheaper per byte

Figure 2.15 – Modern memory hierarchy

Cache is used by the microprocessor to store frequently used instructions and data. Since the cache memory capacity is limited, it cannot hold all the data and instructions that a program needs. Therefore, it needs to swap data and instructions between other levels in the memory hierarchy. Consider a case where you want to execute the Notepad in Microsoft Windows.

Step 1: The Notepad.exe file is stored in the hard disk (i.e., secondary storage)

Step 2: When Notepad is needed to be executed the microprocessor loads it into the memory.

Step 3: Before executing a specific instruction, the microprocessor needs that instruction and associated data to be in the cache memory. Therefore, block of adjacent instructions (not just a single instruction) and related data from the main memory will be sent into the cache memory.

Step 4: Then the microprocessor gets a specific instruction into the registers from the cache memory and execute the instruction.

Step 5: After execution, the results will be sent back into the cache memory.

Step 6: When new instructions (which are not in the cache memory) are needed the microprocessor loads them from the memory into the cache.

Step 7: Repeating "Step 6" several times will fill up the cache memory. This is an issue when new data or instructions are needed to be loaded into cache from the main memory. To accommodate more space in cache memory some of the infrequently used data and instructions are sent back to the main memory (this is called *swapping*).

Step 8: When both steps 6 and 7 repeat after some time the main memory may also fill up. Then some of the data and instructions are copied back to the secondary storage (set of memory blocks are sent) and this process is called the *paging*.

If any of these paged up memory locations are needed again, they are loaded back into main memory by paging some other set of memory blocks into the hard disk.

To further enhance the performance, cache memory is split into several levels of caches (e.g., Level1, Level2 and Level3 in Intel Core i7 processors). L1 is the closest to the registers while L3 is located at the main memory end. The speed of memory accesses decreases from L1 to L3. In some latest microprocessors, L1 cache is divided into two parts called L1-D and L1-I which caches data and instructions respectively.

2.5 Central Processing Unit

CPU is also called the Microprocessor. It is the brain or engine of the computer. The CPU performs arithmetic and logic operations, and it controls the other components of the computer as well. CPU is the most expensive single piece of component in a PC. Over the last 35 years, microprocessors have gone through tremendous enhancements and evolved by several generations. The first microprocessor was introduced in the 1970s and currently we are in the 7th generation (refer table 2.2).

Table 2.2 – Generations of microprocessors

Generation	Microprocessor(s)
1st generation	Intel 8080/8086/8088
2nd generation	Intel 80286
3rd generation	Intel 80386 (DX/SX)
4th generation	Intel 80486 (SX/DX/DX2/DX4)
5th generation	Intel Pentium, AMD K5
6th generation	Intel Pentium Pro, AMD K6
7th generation	Intel Pentium IV/IV HT/IV D, AMD Athlon/Duran/Opteron
8th generation	Intel Core i3/i5/i7

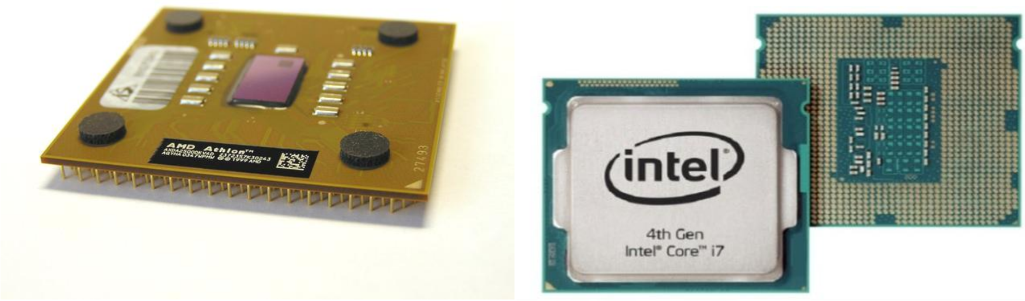


Figure 2.16 – External view of microprocessors

2.5.1 Components of a CPU

As shown in figure 2.17 three major components can be found inside the microprocessor: the Arithmetic and Logic Unit (ALU), the Control unit and a set of Registers.

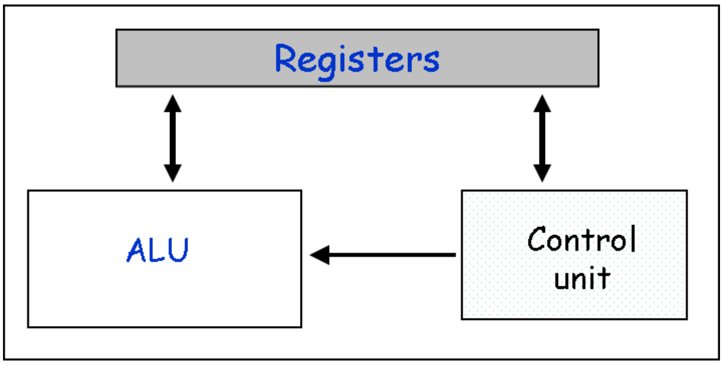


Figure 2.17 – Components of a CPU

Arithmetic and Logic Unit – ALU

This is the data processing unit of the CPU. The arithmetic unit can perform arithmetic operations while the logic unit performs logical operations.

Control Unit

This unit controls the operation of the CPU and rest of the machine.

Registers

Registers are type of memory located inside the CPU that can hold a unit of data. This data is useful for both data processing and control functionalities. Registers hold data before they are processed and after processing the results are sent back to registers. Note the bidirectional arrows that connect the registers with the ALU and control unit in figure 2.18. Since registers are located inside the CPU they can be accessed faster by the ALU and control unit. Several types of CPU registers exist:

- Program Counter (PC)
- Instruction Register (IR)
- Accumulator (A)
- Flag register (F)
- General Purpose Registers (GPR)

Program Counter – PC

The Program Counter (PC) is used to keep track of memory address of the *next instruction to be executed*. It will always point to the next instruction to be executed. When an instruction is fetched[3], always the instruction pointed by the PC is loaded into the CPU. Once the instruction is fetched, the program counter is automatically updated so that it will points to the next instruction (it will always be incremented by 1). The PC can also be incremented or decremented by the programmer.

Instruction Register – IR

Once an instruction is fetched into the CPU it is stored in the IR for execution. The IR is located closely to the control unit, which decodes the instruction. The control unit decode (understand) the instruction and ask the respective circuit to handle the instruction accordingly. Keeping IR closer to the control unit make this process must faster[4].

Accumulator – A

Results of arithmetic and logical operations always go to the accumulator. Accumulator is connected directly to the output of the ALU. Accumulator is normally indicated by symbol ‘A’.

Flag Register – F

Flag Register stores the status of the last operation carried out by the ALU. After an arithmetic or logic operation there can be various states such as; overflow, division by zero, final result is a zero, positive or negative result, results of comparisons, etc. In certain parts of a program before preceding to the next instruction this state information is checked to determine the next operation. Such state information is indicated by setting up various bits in the flag register.

General Purpose Registers – GPRs

These registers are said to be general purpose because they can be used for various tasks. They are used to store intermediate results of the ALU operations. A limited number of GPRs are available inside a CPU and programmers have to use those registers in an effective manner to run even a very complex program. They are normally labelled as register ‘B’, ‘C’, ‘D’, etc (number of GPRs may vary depending on the CPU).

Identify different components and their locations insider the CPU using the diagram given in figure 2.18. Three (3) buses (data, control, and address) run all around the CPU. Both control and address buses go out of the CPU while data bus is bidirectional. Also, note that IR is located near the control unit. Flag registers are set by the main ALU but they are accessible only to the control unit.

[3] Fetching is the process of loading an instruction into the CPU. This stage is called the fetch cycle.

[4] Inside a CPU having more than 100 million transistors in less than a square inch several micro meters (μm) is a long distance.

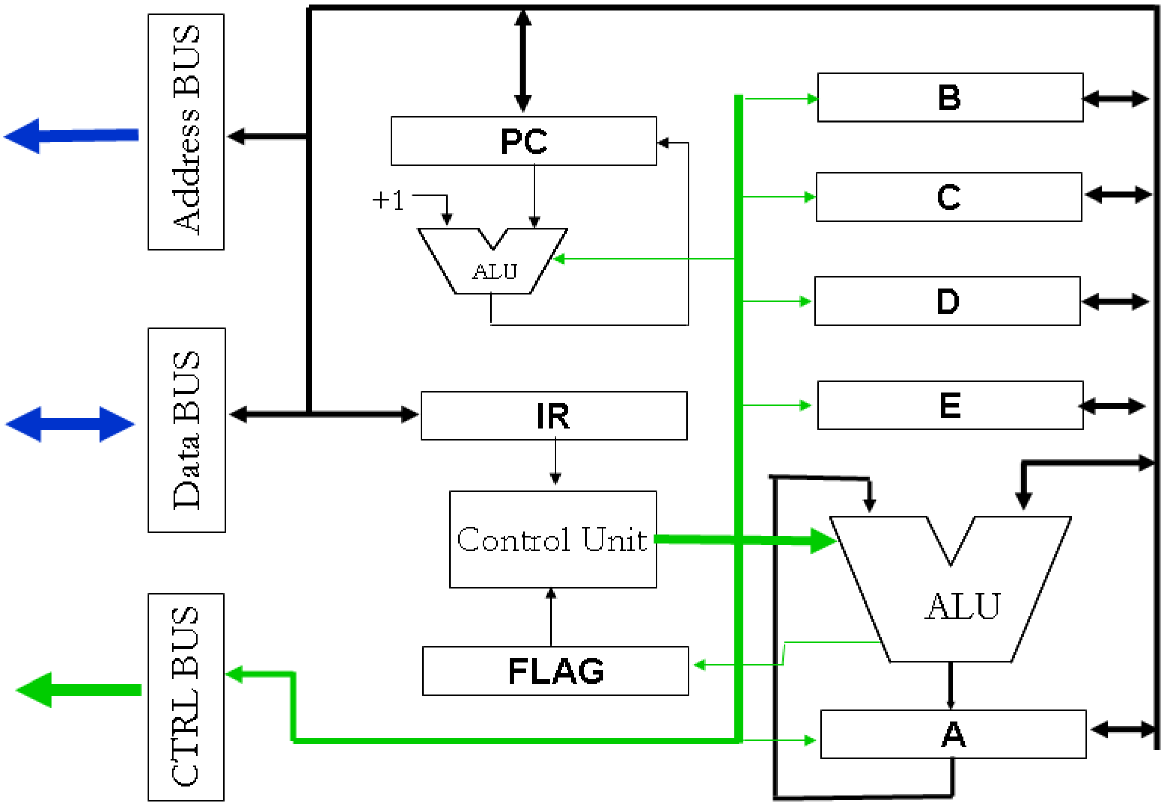


Figure 2.18 – Internal Structure of the CPU

Note the secondary ALU after the PC. This ALU increment the PC after fetching a new instruction. Accumulator ('A') is just after the main ALU and all the results of the ALU operations are automatically sent to the accumulator. The main ALU accepts a maximum of 2 inputs (*operands*). However, one input should come through the accumulator.

2.5.2 Execution of a Program

Consider the following program segment, which is written in Assembly language[5].

```
100: Load A,10          ; A ← 10
101: Load B,15          ; B ← 15
102: Add A,B            ; A ← A+B
103: STORE A,[20]       ; [20] ←A
```

The above program stores two values (0x10 and 0x15) in registers A and B and add them together. The final answer is stored in memory address 0x20. Let us go through the code line by line. An Assembly language statement can be divided into several sections.

<Identifier>	Operation	<Operand(s)>	<;Comment>
100:	Load	A, 10	; A ← 10

Identifier – is an optional element, which allows pointing to a line in the code or memory location.
Operation – indicate the specific assembly instruction (i.e. the action to be performed).
Operand – provides data for the operation to act upon (for certain instructions this is optional).
Comment – can be used by the programmer to keep notes within the program (optional).

Consider the first line of code in the above program. `100: Load A,10 ; A ← 10`

Here, the `100:` says store the first instruction at memory address 0x100. `Load A,10` says load the value 0x10 to the accumulator. In this case `Load` is the operation while `A` and `0x10` are the operands. The comment `; A ← 10` indicates move value 0x10 to register A. The direction of data flow is indicated by the direction of the arrow.

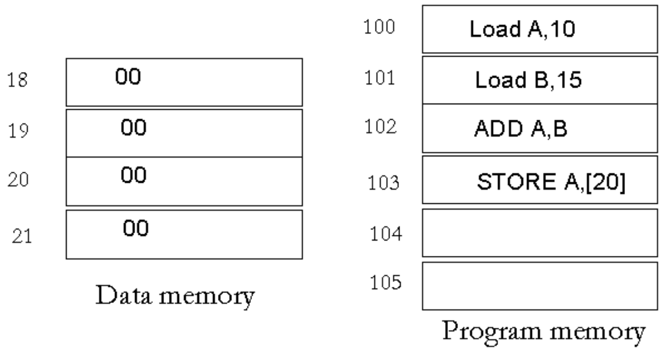
[5] Assembly language is the lowest level of programming that a programmer can do.

Figure 2.19 indicates how these instructions are stored inside the memory. Normally, memory is divided into two portions called the *data* memory and *program* (or instruction) memory. Since the program has not executed the memory location pointed by memory address 0x20 is zero.

Let us observe how various registers are changed when we run the above program. Each instruction in the above given program is executed in a separate instruction cycle. To follow the execution of an instruction cycle, the values of the registers have to be observed in each step of the instruction cycle.

Figure 2.20 indicates status of various registers just before starting the first fetch cycle. Since the program has not executed yet, the PC will point to the next memory address where the instruction to be executed. Then in the next stage the CPU gets the first instruction from memory address 0x100 and stores it in IR register (this is the end of first fetch cycle).

Figure 2.21 indicates status of various registers just after the first fetch cycle. Now the Assembly instruction `Load A,10` is loaded into the instruction register. Meanwhile PC automatically gets incremented and starts pointing to the memory address of the next instruction to be executed (0x101).



18	00
19	00
20	00
21	00

Data memory

100	Load A,10
101	Load B,15
102	ADD A,B
103	STORE A,[20]
104	
105	

Program memory

Figure 2.19 – Content of memory when the program is loaded

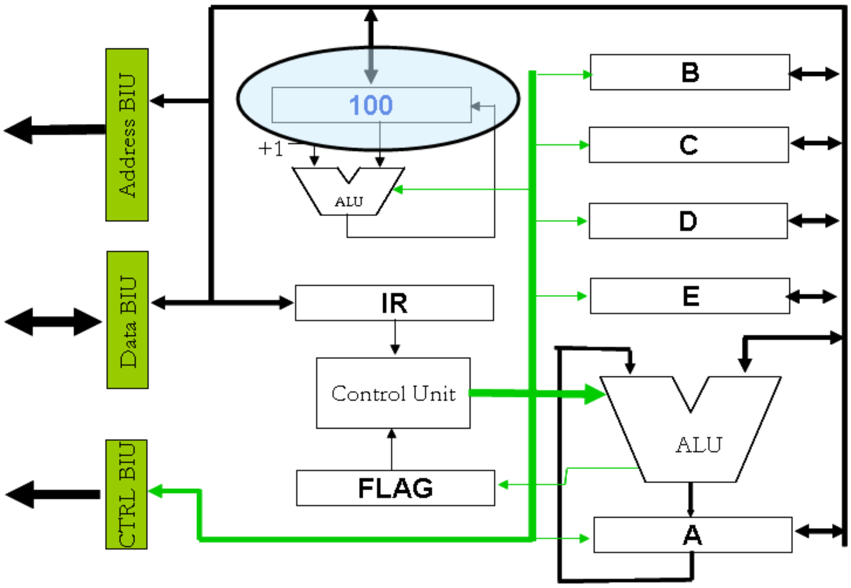


Figure 2.20 – Before execution of the 1st fetch cycle

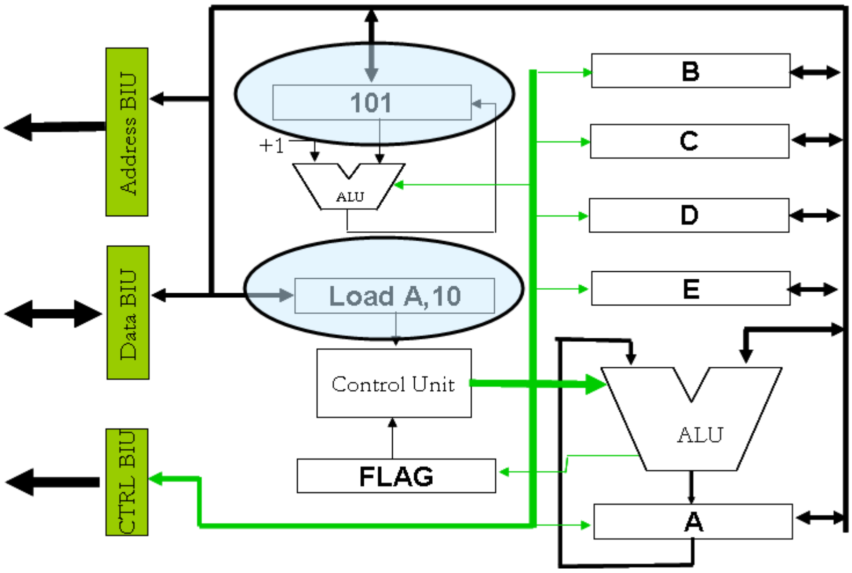


Figure 2.21 – After the 1st fetch cycle

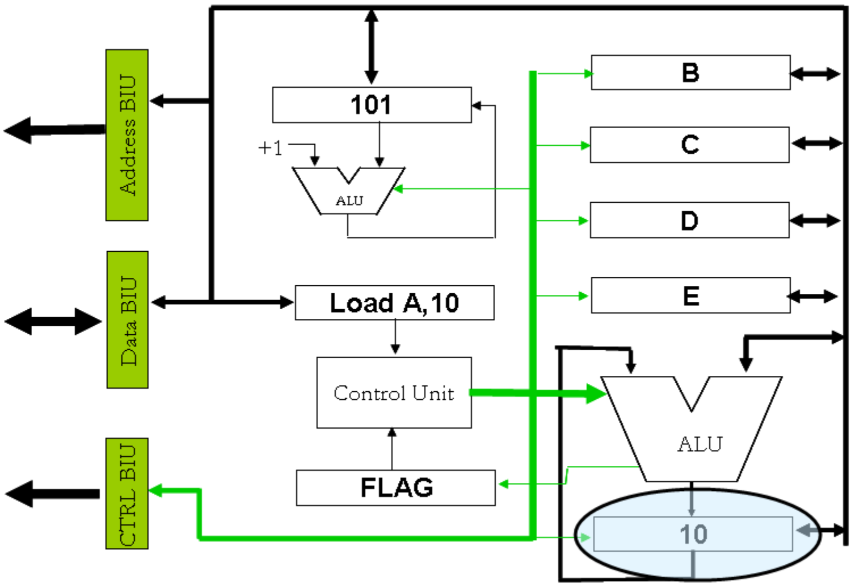


Figure 2.22 – After the 1st instruction cycle

Then the control unit decodes the instruction and understand it as a Load operation. Then after the first instruction cycle (also referred as the *execution cycle*) it will set the accumulator (register A) as 0x10 (figure 2.22). Now the CPU has fully executed the first Assembly language instruction (`Load A,10`).

Next the second assembly instruction (`Load B,15`) which is stored in memory address pointed by PC (0x101) is loaded into the CPU. Figure 2.23 indicates the status of registers after the second fetch cycle. At the end of the second fetch cycle the PC is automatically incremented and starts pointing to the next memory address (0x102). Now IR will hold the second Assembly instruction. During the second instruction cycle the control unit identify this as another Load instruction and at the end of the cycle integer 0x15 is stored in general purpose register B (figure 2.24).

Up to now, two instructions have been executed. Then after the third fetch cycle the next instruction pointed by PC (which is in memory address 0x102) will be loaded into the IR (see figure 2.25). At end of this stage, the PC will point to the next memory location, which is 0x103.

Then at the third instruction cycle, the control unit understands this is an addition operation, so it calls the addition circuit to add the two operands, which are stored in registers A and B. After the execution of the third instruction cycle the result of the add operation (0x10+0x15=0x25) will be saved in the accumulator (A) (see figure 2.26).

In the fourth fetch cycle instruction pointed by the PC will be loaded into the IR. Then at the end of the fourth instruction cycle the value stored in the accumulator will be saved in the memory location pointed by memory address 0x20.

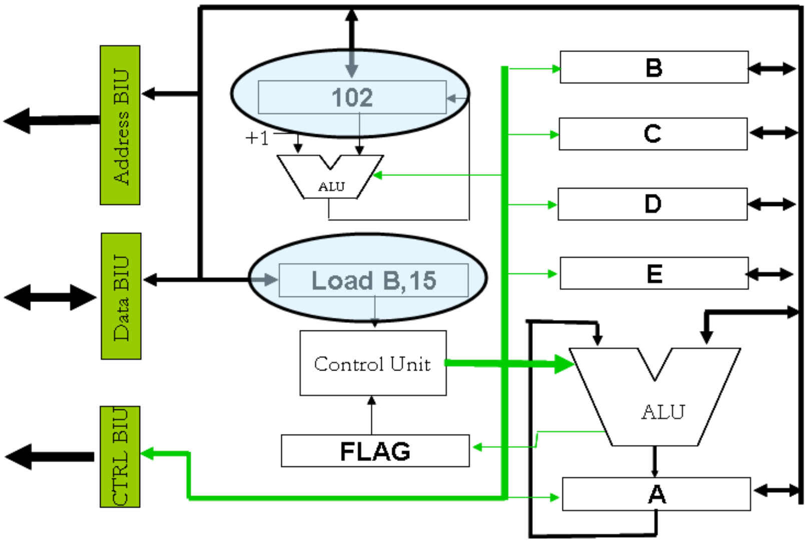


Figure 2.23 – After the 2nd fetch cycle

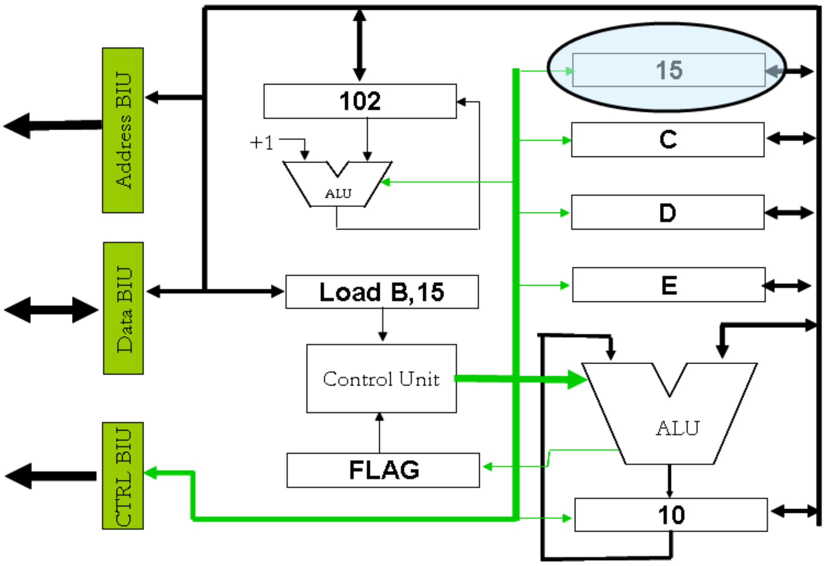


Figure 2.24 – After the 2nd instruction cycle

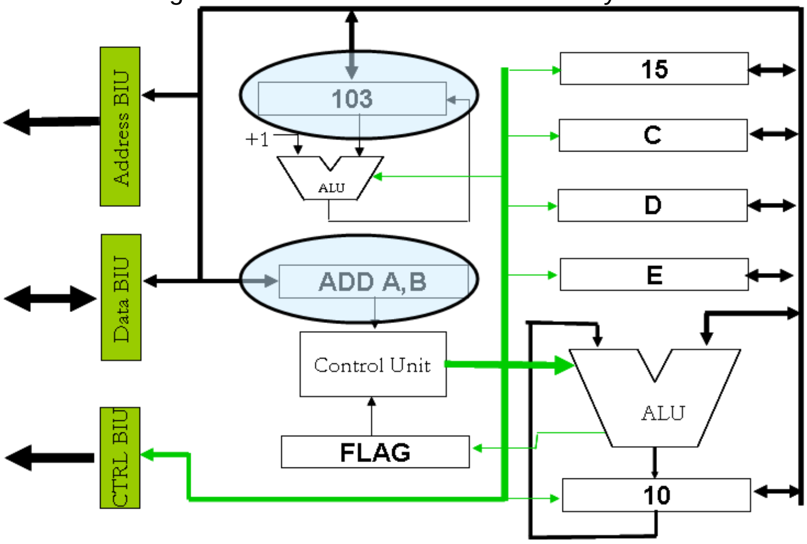


Figure 2.25 – After the 3rd fetch cycle

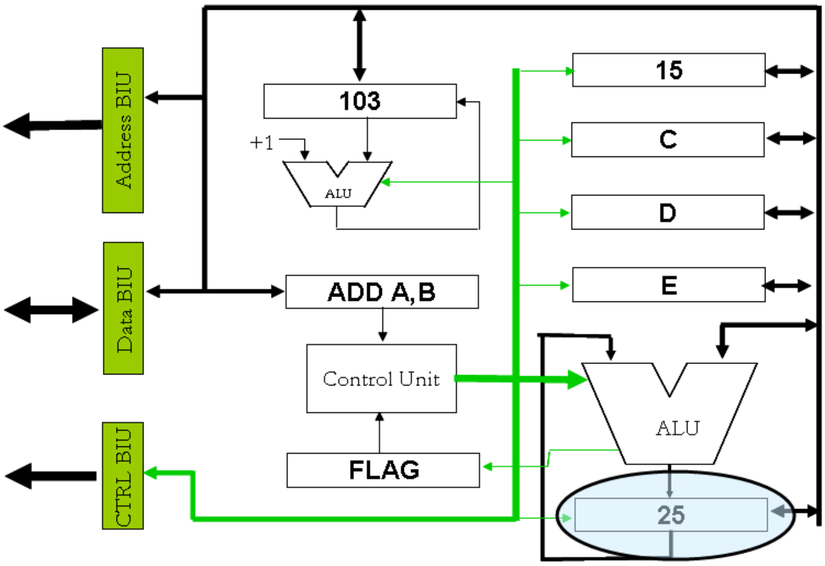


Figure 2.26 – After the 3rd instruction cycle

2.5.3 *Enhancing CPU Performance*

With the advancement of electronics microprocessors became smaller and faster. Faster CPUs were developed by having faster transistors. When the industry demands for faster and faster CPUs manufacturers has to use faster and faster transistors. However, over the last several years speed of transistors has been saturated. Therefore, having faster transistors is no longer the solution to achieve faster CPUs. Then manufacturers came up with several optimisations to improve the performance of CPUs. Some approaches are:

- Instruction pre-fetching
- Instruction Pipelining
- Hyper Threading (HT)
- Multicore processors

Instruction Pre-fetching

In section 2.7.2 when we studied how a program is executed, we realised there are 2 cycles called the; fetch cycle and the instruction cycle (also referred as execution cycle). When an instruction is executed, it first goes through the fetch cycle then through the execution cycle. Then only next instruction is fetched into the instruction register. During each cycle, either the ALU or the fetching circuit (mainly the IR and PC registers) is not utilized.

In figure 2.28 instructions are first executed in the conventional way (upper half of the figure) and later they are executed using pre-fetching (lower half of the figure). In conventional execution, fetch cycle of the 2nd instruction needs to wait until the execution cycle of the 1st instruction is fully complete. Similarly, 3rd instruction needs to wait until the 2nd instruction is fully complete.

In the second approach when one instruction is in the execution stage, the next instruction is fetched into the CPU. The fetch cycle of the 2nd instruction starts while the 1st instruction is in the execution cycle. However, the execution cycle of the 2nd instruction will not start until the execution of the 1st instruction is complete. Similarly, 3rd instruction starts its fetch cycle while 2nd instruction is in its execution cycle. Also note that the 3rd instruction will not start its execution cycle until the 2nd instruction is fully executed.

At the end of the execution total time took to execute 3 instructions using the instruction pre-fetching is lesser than the conventional approach. Therefore, by using instruction pre-fetching the CPU can do more work within a given time (i.e., this approach improves overall CPU performance).

Instruction Pipelining

The term pipelining is used because this process is derived from an industrial assembly line where output of one-step is fed to the next step as an input. In an assembly line of a car production company, multiple cars are assembled at the same time. This is achieved by dividing the car assembling process into multiple sub-stages and carrying out each substage at the same time. Someone observing at the end of the assembly line will realise that multiple cars are being manufactured within one hour, whereas to assemble a single car, it may take several hours. If the production line can be divided into multiple stages, more cars can be in the production line. This approach increases the number of cars assembled per day, but do not reduce the time spent on a single car.

The same concept is used inside the CPU. Pipelining divides the instruction cycle into a series of sub-operations and a separate segment of the CPU is dedicated to one sub-operation. Since there are many stages, multiple instructions can be in the instruction cycle at the same time. Each of these instructions should be in a different stage or a sub-operation. To achieve pipelining more electronic circuits are needed inside the CPU core.

This is an extension of the idea of instruction pre-fetching. This approach will increase the throughput of the microprocessor[6]. Pipelining will not speedup a single instruction, it will speed up a set of instructions.

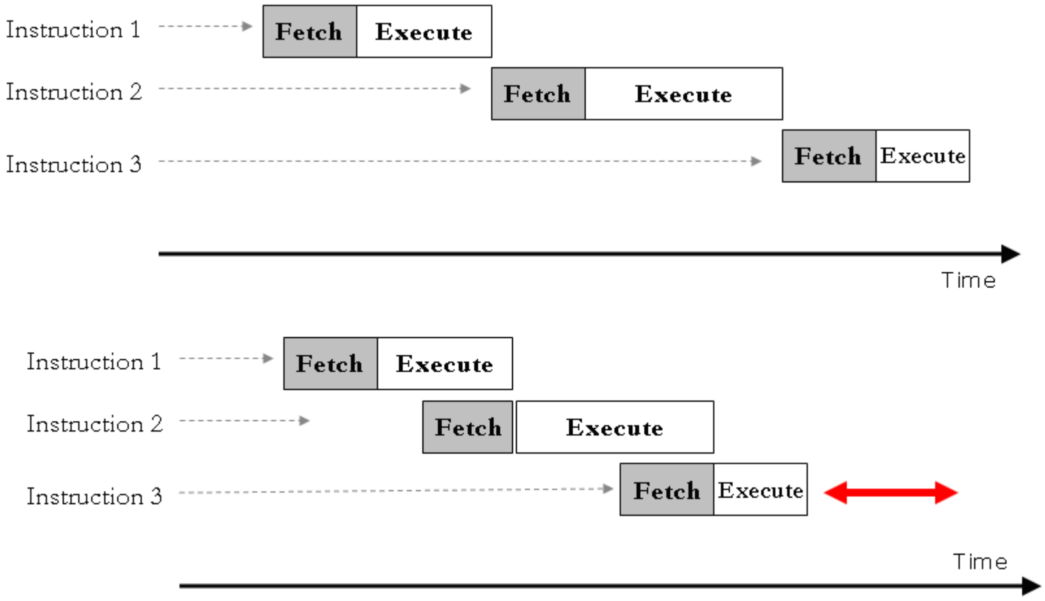


Figure 2.27 – Instruction pre-fetching

Suppose there are 2 CPUs A and B. A has 5 stages (i.e., number of sub-operations) pipeline while B has a 9-stage pipeline. Since A has 5 stages 5 instructions can be in the instruction cycle at the same time. In CPU B, 9 instructions can be in the instruction cycle. Therefore, CPU B executes more instructions compared to CPU A in a given time interval (i.e. CPU B has a higher throughput so it has better performance than A).

Hyper Threading

Hyper Threading (HT) is another approach that improves the performance of a CPU. HT was first introduced by Intel with their Pentium IV 2.8GHz processors. It allows two different resources of the CPU to be used at the same time. For an example when one thread[7] (instruction) makes use of the integer unit of the ALU another thread (instruction) can makes use of the floating-point unit. However, the two threads cannot use the same resource at the same time; therefore, HT does not always allow two threads (instructions) to execute at the same time (i.e., if two threads require the same CPU resource then they have to execute one after the other). When hyper threading is possible the operating systems will feel that it is running on top of a two CPU computer.

Hyper Threading is achieved by having a mix of shared, replicated, and partitioned chip resources, such as registers, arithmetic units and cache memory. However, to make use of HT the computer should satisfy following four conditions:

a) The CPU should support HT technology
b) HT technology enabled chipset

[6] Throughput is the number of outputs per unit time

[7] Thread is an operating system construct that actually executes inside the CPU. In simple terms it can be considered as the execution part of a program.

c) HT technology enabled BIOS
d) HT technology enabled/optimized operating system

Multicore processors

To overcome the limitations of not being able to improve the performance of uniprocessors due to various reasons such as high heat dissipation and transistor density issues, processor manufacturers developed multi-core processors. The idea of multi-core was introduced with the IBM Power4 microprocessor consisting of two cores; however, AMD is the one who actually brought it to the consumer market. As its name implies it combines multiple independent microprocessors and their respective caches onto a single Silicon chip (figure 2.28). Since it is two (in dual cores) different microprocessors not just a set of replicated components, performance gained by dual core is higher than HT.

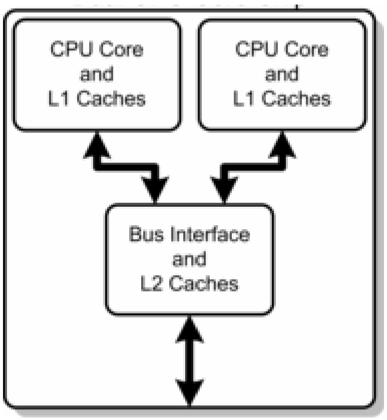


Figure 2.28 – Components of a Dual Core chip

2.5.4 Improving Overall System Performance

Section 2.7.3 introduced some of the approaches that can be used to improve the performance of the CPU. What really matters is how we can improve the performance of the overall computer rather than just the CPU. Having a faster CPU will help the overall performance, but it is not the only factor that will decide the performance of the entire machine. Consider 2 computers: ‘A’ and ‘B’ with 1GHz and 2GHz CPUs. Other than the CPU, both ‘A’ and ‘B’ have the same hardware configuration. Computer ‘B’ is faster, but it does not mean that it is twice as fast as computer ‘A’.

The overall performance of a computer can be improved by:

- using a high-performance CPU
- having an effective memory hierarchy
- using buses with different speeds
- using CPU support chips

Performance is mostly degraded due to slowness in memory. Therefore, CPU performance can be highly improved by having an effective memory hierarchy. The cache memory plays a major role in improving performance of the modern CPU.

Some CPUs such as Intel Pentiums use different speeds for its internal operations and for communication with memory. It uses a high-speed bus for its internal communication while a slower bus is used to communicate with main memory. This is analogous to having a conventional road and a superhighway. Vehicles that can go faster can use the highway while slower ones have to use the conventional road. In such an arrangement, a faster vehicle is not slowed because of a slower one. In modern motherboards Front Side Bus (FSB) is used to communicate with the slower memory. Usually, the speed of FSB ranges from 133MHz to 833MHz.

2.5.5 CPU Support Chips

Von Neumann's Architecture

Von Neumann, a consultant who worked in the ENIAC project, published a paper in 1945, which described all the parts of a stored-program computer:

- A memory, containing both data and instructions

- A calculating unit, capable of performing both arithmetic and logical operations on the data
- A control unit, which could interpret an instruction retrieved from the memory and select alternative courses of action based on the results of previous operations

The computer structure resulting from these criteria is popularly known as a Von Neumann Machine[8]. The Von-Neumann Architecture is a CPU centric system where:

- Each operation is carried out only by the CPU
- Every movement of data must be made via the CPU
- Memory is the only "Direct Access" storage device for the CPU
- Only a single operation is carried out by the CPU at any time

Von Neumann's architecture is a simple and implementable proposal. However, for some operations such as disk access, giving attention to peripheral devices and for periodic or time critical operations this is not the best mechanism.

If the CPU needs to access a file, it should be first loaded into the memory. Therefore, the CPU has to instruct the hard disk that it needs a file. Then the file is divided in blocks and each block is sent from the hard disk into the CPU. Then the CPU has to store it in the memory. Then only it can access the file from the memory. In this case, all the communication is done through the CPU, so it has to stop all other work and help the communication. The CPU time is precious thus, it should not be wasted unnecessarily. Instead of this approach, if another controller can get the file from the hard disk and put it in memory on behalf of the CPU it will save valuable CPU time. Then the CPU can access the file from memory. CPU support chips are used to carry out such tasks on behalf of the CPU.

CPU support chips are used to improve the overall performance of a system. In this approach, the CPU is like the manager of an organization while CPU support chips are like supervisors under him. Manager will give orders to supervisors, and they will ask the workers (in a computer, workers are components such as hard disk, floppy disk, keyboard, communication ports, etc.) to carry out the actual work.

If this approach is compared with the conventional Von Neumann's architecture, it is like having an organization with a single employee, where that person is the manager as well as the only worker. If a file is required, the manager has to go through all the files and find out the correct one. This approach wastes manager's time that could have been used for something more useful. In an organization with a well-established management hierarchy, the manager can request a clerk to find the file on behalf of him/herself. When the file is available, the manager can do his/her work and when it is finished, the clerk can put the file back again. These approaches allow the manager to concentrate on matters that are more important. Some of the CPU support chips are:

- Direct Memory Access (DMA) controllers
- Interrupt Controllers
- Real-Time Clock (RTC)
- Other devices
    - Disk controllers
    - Communication controllers
    - Display controllers

Direct Memory Access (DMA) Controller

The DMA controller provides a way of bypassing the CPU when transferring data between memory and Input/Output (IO) devices. This is in contrast to Von Neumann's architecture where everything has to go through the CPU (figure 2.29-A). The DMA controller resides between the CPU and memory. When the CPU needs something to be in memory, it informs the DMA controller. Then the DMA controller accesses the resource on behalf of the CPU and loads it into the memory (figure 2.29-B).

[8] Virtually all the computers produced since were Von Neumann machines.

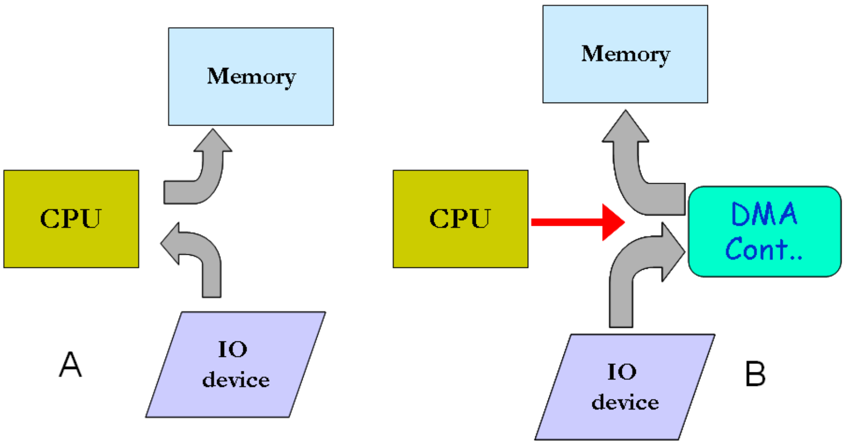


Figure 2.29 – Communication with and without DMA controller

Disk Controllers

Some of the common disk controllers are Floppy Disk Controller (FDC) and ATA Controller (ATAC) for hard disks. When reading/writing to/from a disk the CPU will create a special memory area (called a *buffer*) containing the sector address and the data to be written or read. Then the CPU informs the FDC (or ATAC) about the location of the buffer. The disk controller then transfers the content of the buffer directly from memory to the disk sector.

Real Time Clocks - RTC

The RTC is used to keep track of time of the day. It is usually backed-up by an extra power source (generally by a lithium battery). Additionally, it is used to store some of the configuration information such as CMOS[9] setup memory.

2.6 Display Controllers

Display controllers are used to generate images and text that you see on the displaying device on behalf of the CPU. Video controllers are used display the image that you see on monitor. Display controllers are either available as a separate expansion card or integrated into the motherboard (figure 2.30). Display commands are given by the CPU, and they are carried out by the display controller. Display controller generates the actual image in its memory; called the *Refresh Buffer*, with 1's and 0's (figure 2.31). Then it is passed through to the video controller to generate the actual image.

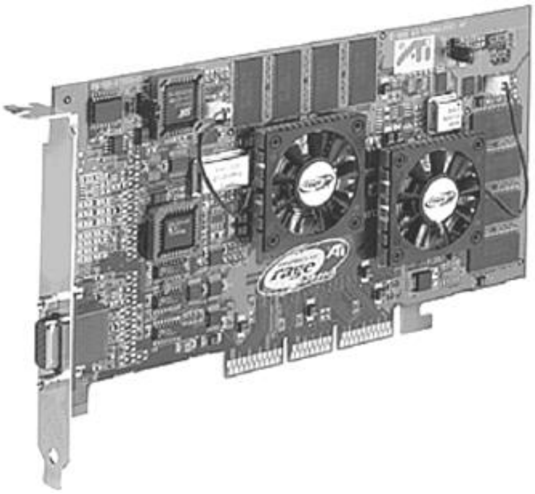
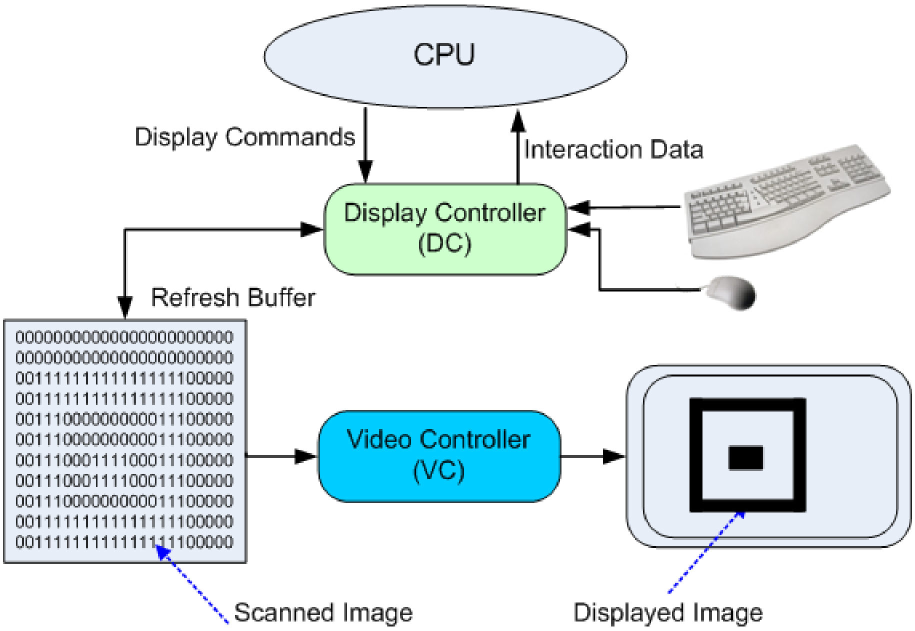


Figure 2.30 – A video card

[9] CMOS (Complementary Metal-Oxide Semiconductor) memory is used to store system configuration data.



```
00000000000000000000000
00000000000000000000000
00111111111111111100000
00111111111111111100000
00111000000000011100000
00111000000000011100000
00111000111100011100000
00111000111100011100000
00111000000000011100000
00111111111111111100000
00111111111111111100000
```

Figure 2.31 – Display and Video controllers

There are several video standards:

- VGA – Video Graphics Array. Graphics are supported at a minimum resolution of 320x240 pixels in 256 colours and for 640x480 in 16 colours.
- SVGA – Super VGA. Supports resolution up to 800x600.
- VESA SVGA - Video Electronic Standards Association SVGA. Was developed to standardise SVGA. Also includes a video standard for connecting high-speed adaptors directly to the processor bus.

Video cards are classified based on their video processor and video memory. High-speed video processors can render (i.e. colour) 2D and 3D images much faster (suitable for 3D animations and gaming). High-capacity video memory produces better quality images without distortion or flicker.

2.7 Secondary Storage

Secondary storage devices are needed in addition to the volatile memory as a permanent and high-capacity storage solution. Cost per a Megabyte of secondary storage is lower than the main memory. Different secondary storage devices such as hard disks, floppy disks, CD-ROMs, DVD-ROMs and ZIP disks are used in personal computer systems.

These storage solutions can be divided into two broader classes namely, *Magnetic Storage* and *Optical storage*. A combination of magnetic and optical technology called *floptical* technology is also available.

2.7.1 Hard Disk Drive

Hard disks are one form of magnetic storage. Hard disk is the main secondary storage device used in a computer. Its operation is identical to a conventional radio cassette tape. However, hard disk uses a disk coated with magnetic medium rather than a plastic tape. Hard disks contain a rigid disk-shaped *platter*, which is constructed using glass or Aluminium. A magnetic medium is applied on this platter and read/write heads are used to read and write data to and from each platter. Heads are connected to the head arm, and it is controlled by the head actuator (figure 2.32).

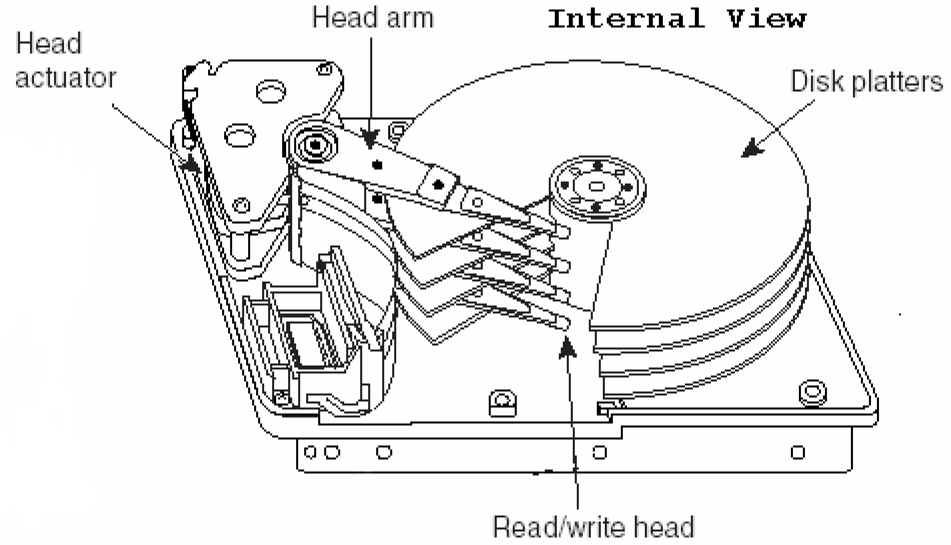


Figure 2.32 – Internal view of a hard disk

Compared to most other secondary storage devices hard disk can store large capacity of data and currently have capacities in tens of Gigabytes. It also has a much higher data transfer speed. Higher capacity is achieved by having multiple platters in the same hard disk and storing data on both sides of the platter.

Hard disks are categorized based on their capacity, controller, and platter rotation speed. Capacities range from 500GB to about 8TB. There are several hard disk controllers such as IDE – Integrated Device Electronics, SCSI – Small Computer System Interface and Serial ATA (SATA) – Serial AT Attachment Interface. Common rotation speeds are 3600, 5400 and 7200 RPM. Platters are kept in a dust free environment inside the hard disk casing. When the disk is spinning the read/write heads move very close to the surfaces of the platters at a considerable speed. Therefore, even a tiny dust particle trying to pass this gap could damage the head, platter or both.

A *track* is a single ring of data on one side of a platter (figure 2.33). A disk track is too large to manage data effectively as a single storage unit. Some disk tracks can store more than 100 KB of data which is very ineffective when managing small files. Therefore, tracks are divided into several fixed size divisions called *sectors*. These sectors represent arc shaped pieces of the track (figure 2.33). In a typical hard disk, there are more than 900 sectors in a single track and typical size of a sector is 512 bytes. The set of tracks on a disk that are on each side of all the platters in a stack and are at the same distance from the centre of the disk is called a *cylinder* (figure 2.33).

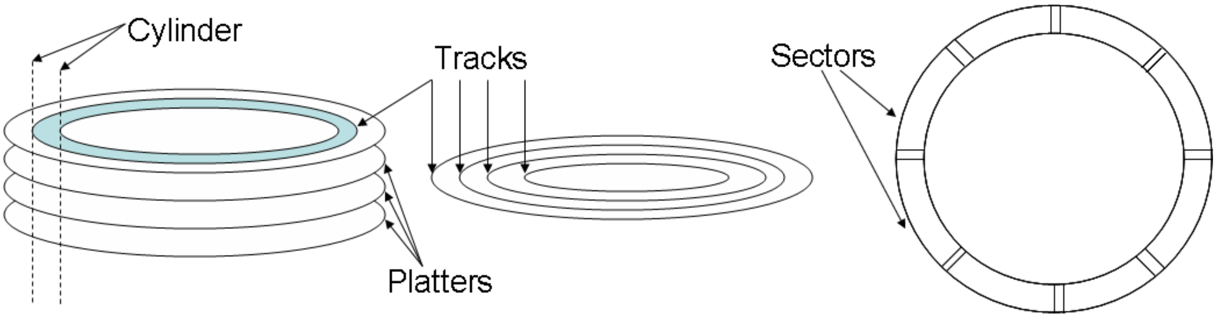


Figure 2.33 – Tracks, sectors and cylinders

Most commodity hard disks have only a single platter therefore both sides of the platter are used to store data to gain higher capacity. Other hard disks use a stack of platters. In such cases top and bottom most surfaces of the platters are not used for data storage. For an example consider a hard disk with 4 platters (figure 2.34). It uses only 6 surfaces to store data out of the 8 available surfaces therefore such a hard disk needs only 6 read/write heads.

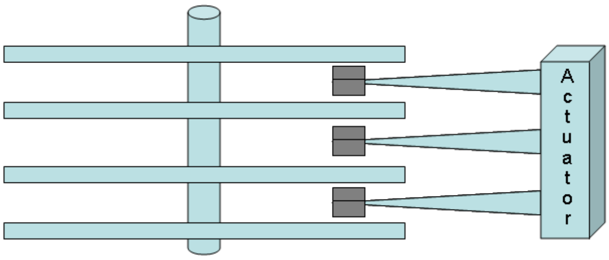


Figure 2.34 – A four platter hard disk drive

2.7.2 *Floppy Disk Drive*

Floppy disk is a removable disk which has a flexible magnetic medium that is enclosed in a semi rigid or rigid plastic case. The principle behind the floppy disk is identical to the hard disk. In later 1960's IBM developed the first floppy disk drive. Early floppies were having a diameter of 8" and its capacity was limited to 300KB. Later 5¼" *minifloppy* was introduced which was having capacity of either 720KB or 1.2MB. However currently we use the 3½" High Density (HD) floppy disk which comes in a rigid plastic case (not really as flexible as earlier floppies) and having a capacity of 1.44MB.

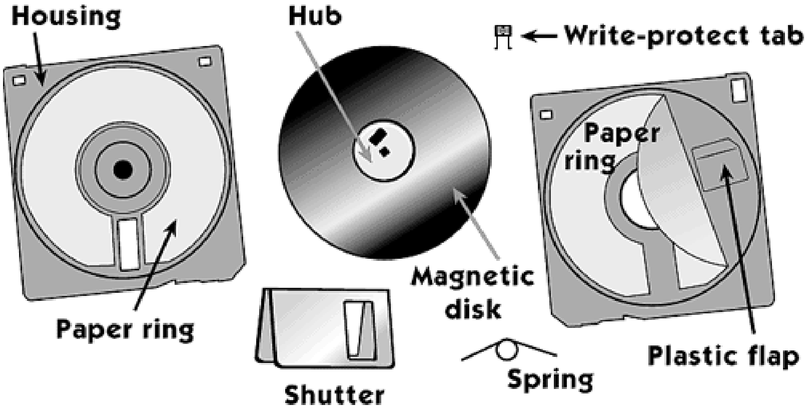


Figure 2.35 – Parts of a floppy disk

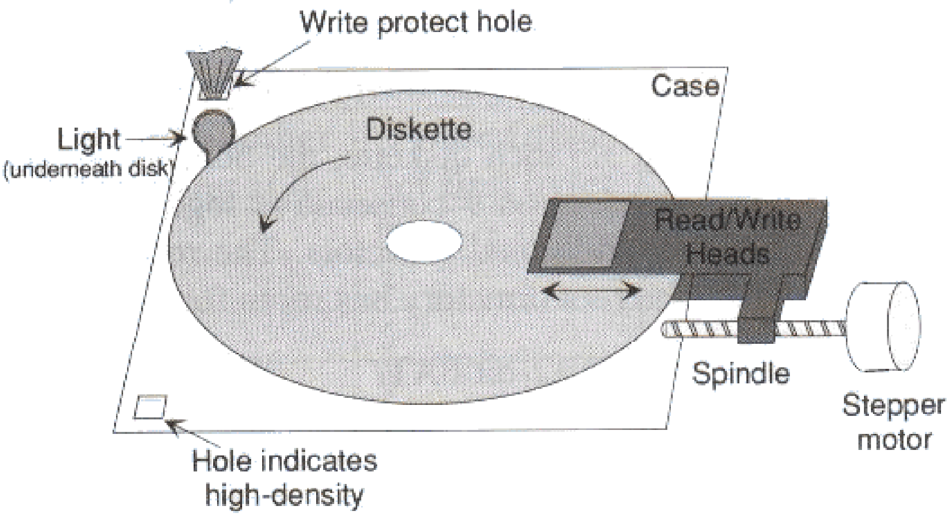


Figure 2.36 – Functionality of a floppy disk

Figure 2.35 shows the parts of a 3½" floppy disk. The magnetic coated, semi rigid, plastic disk is kept in a plastic housing and its two surfaces are covered by two paper rings (for protection). The disk is mounted on a *hub* and a rectangular shaped *cut-out* is used by the disk driver to firmly grab the disk while rotating. The read/write head access the disk through a small opening called the *flap*. The *spring*-loaded *shutter* is used to cover up the flap so that prevents any damages to the disk by dust particles. The shutter will open up only when the disk is inside the disk driver and when it is ejected the spring will automatically close the shutter. A *write protect tab* is used to prevent the disk been overwritten and it either opens or closes the *write protect* *hole* (figure 2.36). If the write protect hole is closed the disk cannot be overwritten (then light cannot penetrate through the hole) and if the hole is open (light can penetrate through the hole) it can be overwritten. The head actuator mechanism is slightly different to the mechanism in a hard disk. In a hard disk the head arm moves laterally from centre of the disk towards to the edge while in a floppy disk the movement is horizontal (figure 2.36). The read/write head is mounted on a *spindle* and the spindle is controlled by a stepper motor. The *high density hole* is only available in High Density floppy disks which have a higher data density.

2.7.3 Optical Storage

Optical storage devices make use of light instead of magnetism. There are different forms of optical storage such as; CD-ROM (Compact Disk – Read Only Memory), CD-R (CD – Recordable), CD-RW (CD-Rewritable), DVD (Digital Versatile/Video Disk), DVD-R, DVD-RW, etc. Most of these storage mechanisms use tiny visible light beams or laser.

A CD is made of polycarbonate wafer, 120mm in diameter and 1.2mm thick, with a 15mm hole in the centre. This wafer base is stamped or moulded with a single physical track in a spiral configuration starting from the inside of the disk and spiralling outwards. If you examine the spiral track under a microscope, you will see that along the track are raised bumps, called *pits*, and flat areas between the pits called *lands* (figure 2.37).

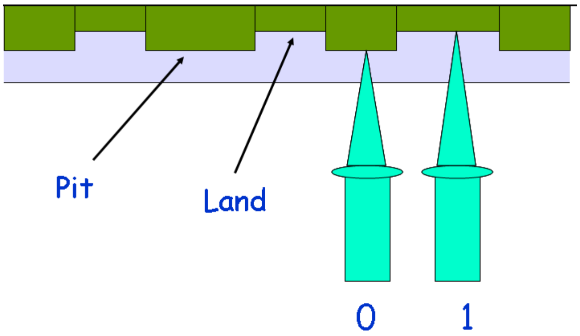


Figure 2.37 Geometry of a compact disk

The light beam used to read the disk would pass through the clear plastic, so that the stamped surface is coated with a reflective layer of aluminium to make it reflective. Then the aluminium is coated with a thin protective layer of acrylic lacquer and finally a label or printing is added. Data recorded on the CD is read based on the reflection of light from pits and lands. However, if the aluminium layer is damaged the light passes through the CD and will not reflect resulting data loss. Therefore, special care must be given to protect both sides of the CD.

The pit's height above a land is especially critical as it relates to the wavelength (λ) of the light beam used to read the disk. The pit height is exactly ¼λ above the land. Therefore, a light beam striking the land travels ½λ (¼λ + ¼ λ) further than a light beam striking the top of the pit. This means a light beam reflected from a pit is ½λ out-of-phase with the rest of the light being reflected from the disk. The out-of-phase waves cancel each other out. Therefore, a light beam hitting a pit will not return back to the light sensor and it will appear as dark spot. A dark light spot represents logical '0' where a visible light spot represents logical '1'.

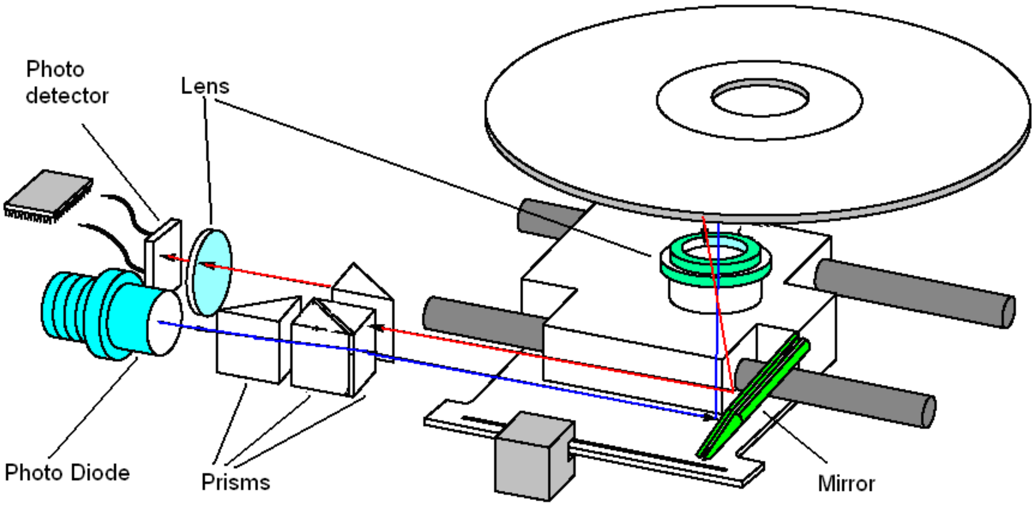


Figure 2.38 – Components of a CD-ROM drive

Figure 2.38 shows various components of a CD-ROM drive. The light beam is generated by a light source (photo diode) and it is directed through several prisms. Then the reflecting mirror rotates the light beam by 90°. The servomotor positions the beam onto the correct track on the CD by moving the reflective mirror. Then the reflected light from the surface of the CD is sent out through a focusing lens to the same mirror. Mirror rotates the beam by 90° and it is sent through a beam splitter (set of prisms). The beam splitter directs the returning light towards another focusing lens. Then the directed light is detected by a photo detector (phototransistor) and it will invert the light into set of electrical impulses. These electrical impulses will indicate whether each bit is a '1' or a '0'.

2.7.4 Flash Storage

Flash storage is a widely used non-volatile computer storage medium. Flash storage is built using EEPROMs (Electrically Erasable Programmable Read Only Memory). Flash storage is categorized into two main divisions as NAND Flash and NOR Flash based on their operational characteristics.

Flash memory stores information in an array of memory cells made from floating gate transistors. In the traditional flash drives, each cell stores a single bit of data. However, newer flash devices known as multi-level cell devices can store more than one bit per cell by choosing between multiple levels of electrical charge.

Due to its high capacity and high transfer speeds compared to other forms of portable media, flash storage devices have become a popular portable storage option and is widely being used as thumb drives, memory cards etc. Due to advantages like low latency, low noise, low power consumption and high reliability, computer vendors have started using flash memory technology as a replacement for hard disk drives (Solid State Devices) and Random Access Memory (RAM) modules.

With the development of more and more sophisticated embedded devices that need storage, flash memory will prove to be more and more valuable and useful.

Figure 2.39 – A thumb drive

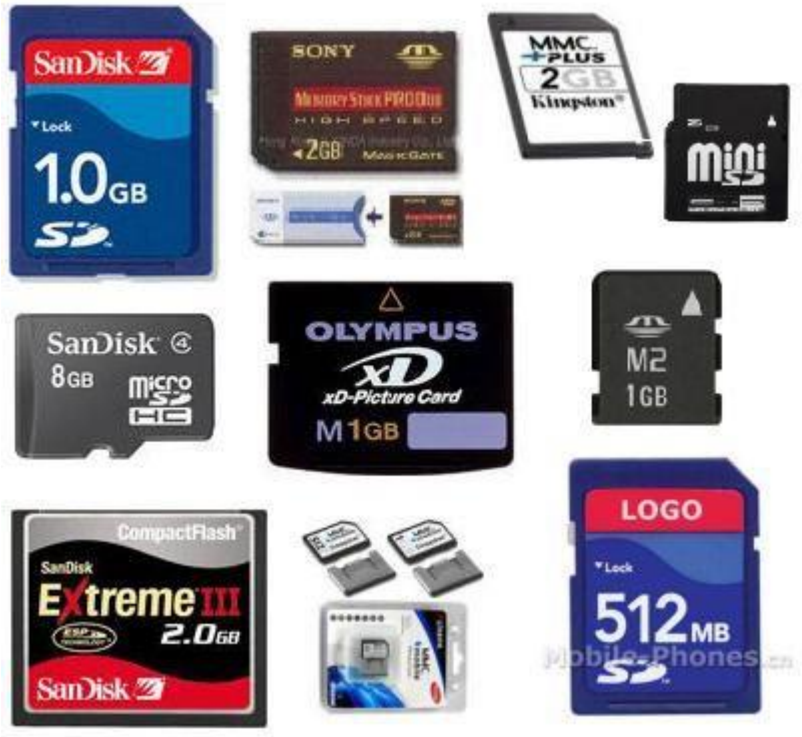


Figure 2.40 – Memory Cards

3 - Introduction to Computer Software

Software cannot be seen or touched, unlike hardware. Without software, however, no computer is usable. *Software* is *the programs that allow users to use a computer system and control its activities*. A *program* is *an ordered sequence of instructions that the hardware can execute*. Software can be categorized as *system software* and *application software*.

3.1 System Software

System software can be described generally as software that satisfies any one of the following:

- manages the computer hardware.
- can be used to maintain the computer so that it runs efficiently.
- helps to do tasks easily and quickly.
- helps to create new software.
- may not be targeted for end-users[10].

Some system software takes control of the computer when it starts and then plays the central role in controlling everything happens after that. We can identify the following basic types of system software.

An *operating system* is the best example for system software. An operating system manages the hardware resources and provides a friendly interface to the users by hiding the complexity of the hardware. Microsoft Windows, Linux, and Mac OS are a few examples. Operating systems are discussed in detail in Section 3.3.

Among other software, categorizing some as either application software or system software can be a bit difficult. One possible, perhaps not perfect, approach is to categorize based on the end-user, as stated above, so that software (other than an operating system) not designed for direct use by an end-user is system software. In this context, *utility software* and *software development tools* can also be categorized as system software.

Utility software performs very specific tasks, usually related to managing computer system resources. They make computer systems easier to use or more efficient. An operating system may contain several utilities within it. Utilities differ from application software mostly in terms of size, complexity, and function. Some utilities provide low-level (i.e., closer to hardware) functionalities to complement an operating system. Note that there is no standard definition for utility software. Functionalities of some utility software products may be provided by some operating systems but not by others. Many utility software are available as individual products from software providers (not by the operating system provider). Specialized software for disk management and partitioning, network administration, data compression, diagnostic and troubleshooting, text editing, text processing and file handling are some examples. Widely used utility software today can be incorporated into operating systems in the future.

We can also group software development tools under system software. These are specialized software required to create new software and include the language translators such as compilers and assemblers and tools like linkers and debuggers. While some operating systems like Linux provide such software as a part of the package, others do not.

In some embedded systems, the application software and the system software may be indistinguishable to the user, as in the case of software used to control a DVD player or a microwave oven.

System software embedded in a hardware device (i.e., stored on a non-volatile storage), is usually termed *firmware*. The BIOS in personal computers and controlling software in DVD players are examples for firmware.

Systems programming (or system programming) is to produce system software and requires a greater degree of hardware awareness than programming application software.

[10] The *end-user* uses the software after it has been fully developed and marketed. They require a bug -free and finished product. The term usually implies an individual with a relatively low level of computer expertise.

3.2 Application Software

These software employ the capabilities of a computer directly to a task that a user wishes to perform. They are programs designed for an end-user, such as word processors, database systems, spreadsheet programs, media players, image processing, scientific computation and so on (the list goes on…). Basically, all software that are not system software can be called application software, or sometimes, simply *applications*.

In contrast to utilities discussed above under system software, most applications are large programs that perform a variety of functions not directly related to managing computer resources. Figuratively speaking, application software sits on top of systems software because it is unable to run without the operating system and system utilities, as shown in Figure 3.1.

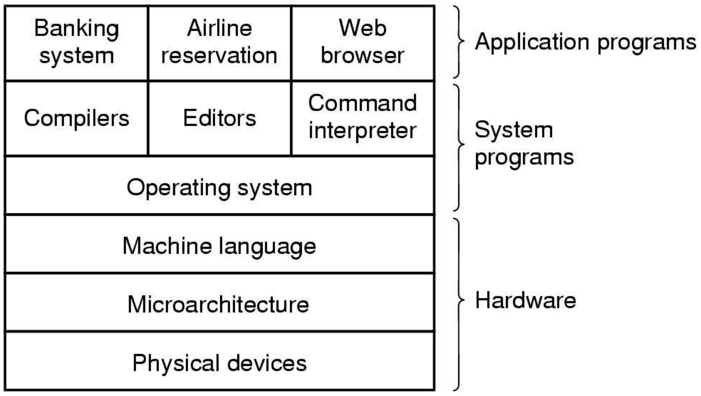


Figure 3.1 – System software and application software

3.3 Operating Systems (OS)

Different computers have certain degree of differences in hardware used by them. In the early days all the programs were developed from the beginning to run on a specific computer suited for the nature of hardware available. If the same program to be used with a different computer (i.e., hardware), it was required to be redone from the beginning with modifications to suit the new hardware environment. Also, these programs had to control the CPU scheduling, memory management, storage management, input/output and so on in addition to its expected functionality. With the advancement of technology, programs became more complex and developing software from the beginning to every specific hardware platform became more difficult.

A layer of software called the operating system (OS) was introduced to overcome this issue. The OS acts as a *virtual machine* on top of the hardware, as seen in Figure 3.2. The virtual machine provided a common interface (i.e., a view) of the hardware to software programmes regardless of the specific features of its underlying hardware. Thus, the program needed only to be compatible with this virtual machine interface.

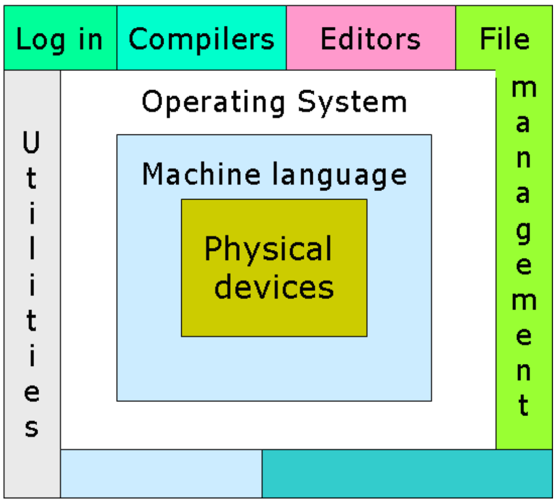


Figure 3.2 – OS as a virtual machine

Writing programs for the virtual machine was much easier than writing programs for real machine. The OS provides to programs a set of services which are not tightly coupled to specific hardware components. The

OS manages all the devices and provides users and programs with a simple interface to deal with the hardware. Application software also use the services provided by the OS.

An operating system provides two major services: as an *extended machine* and as a *resource manager*.

3.3.1 Operating System as an Extended Machine

The OS hides all the messy details about the underlying hardware and how to deal with them. It presents users (including application software) with a *virtual machine* that is easier to use (Figure 3.2). It offers a uniform way of doing something. Although the approach in saving a file in a floppy disk, hard disk or magnetic tape is physically different, the OS users do not need to be aware of the details. For them, saying "save this file in that place" to the OS is enough. All the matters relating to where exactly to save and at what speed to write and so on are handled by the OS. When a user wants to open a file, he/she can just say "open the file at that place" regardless of knowing how to read from a floppy, hard disk or a CD-ROM. It is up to the OS to do all the hard work and open the file.

3.3.2 Operating System as a Resource Manager

The OS is responsible for managing resources in a computer. It is its responsibility to manage them in the most effective manner while ensuring user satisfaction. The OS will decide which program runs at which time, how much of memory to be allocated for the program, where to save a file so that the disk space is optimally utilised, how to deal with concurrent users, how to enforce security, and so forth.

Generally, programs such as the "shell" (e.g., DOS prompt), editors (e.g., Notepad, WordPad, vi), compilers, file management systems (e.g., Windows Explorer) and many other tools that are bundled with an OS are not part of the actual OS. As described previously, they are utility software, and they use the services of the core of the OS.

The core of an OS is the *kernel.* Kernel loads before any of the user programs and remains in the memory. It is responsible for managing CPU, memory, disk, processes[11], etc.

3.4 History of Operating Systems

First generation (1945-1955) "operating systems" cannot be considered as real operating systems. Those days programs were written by changing the wire in a *plugboard*[12]. When a different program is to be executed the plugboard needs to be rewired.

The second generation (1955-1965) systems were mainly batch systems. Figure 3.3 illustrates steps in a batch system. Users prepared "jobs" and submitted them to the operator.

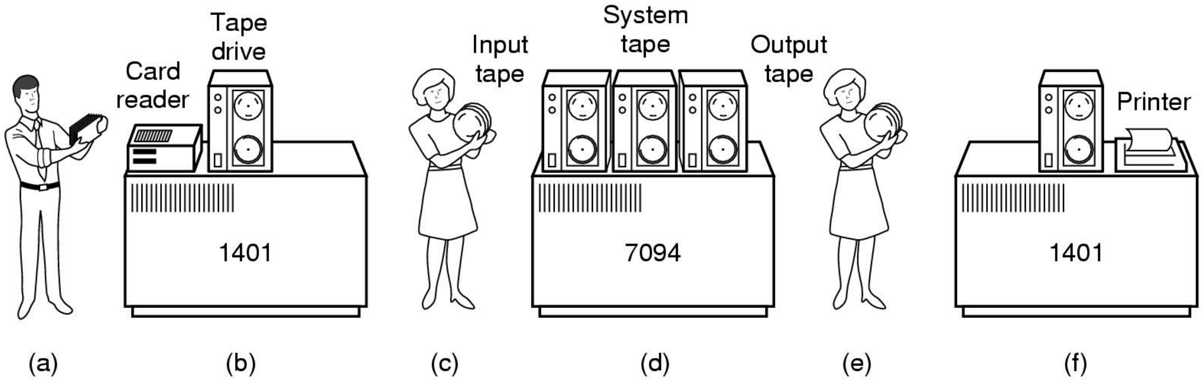


Figure 3.3 – Batch systems

Normally jobs are usually submitted in the form of punch cards (Figure 3.3 step *a*). It takes a lot of time to read a punch card therefore jobs are transferred from punch cards into a magnetic tape (Figure 3.3 step *b*). To further reduce the overhead multiple jobs with similar needs are batched together. Then the tape is inserted

[11] Process is an instance of a running program. A process is more than the program code, it also includes the state of the current activities represented by CPU registers, variables, memory addresses, function arguments, etc.

[12] Plugboard is a large circuit board where different vacuum tubes are connected to form a circuit. The tip of the wire includes a plug, therefore instead of soldering those wires they were directly plugged into the plugboard.

into the main computer (Figure 3.3 step *c*) and it does the processing. When the outputs of the jobs are ready (Figure 3.3 step *e*) it is sent back to the user mostly in a form of a hardcopy (Figure 3.3 step *f*).

Some of the jobs take a long time, possibly a day or even more. Most of these jobs are highly I/O intensive (input-output intensive, that is requiring a lot of data read/written from/to the storage media) hence the CPU is mostly idle. A typical computer has large number of resources, but a single job rarely uses all of these resources. While a computer is worth several million dollars (those days), resources with lower utilization (e.g., an idling CPU) was a major issue. Therefore, to make sure CPU has enough work and to enhance the resource utilization *multiprogramming* was introduced, which led to the third generation (1965-1980) of OSs.

In multiprogramming the OS accesses a number of jobs from a magnetic disk rather than from a magnetic tape. When the computer is running the OS keeps multiple jobs in the memory. The CPU executes one of the jobs in the memory and it continues executing the selected one, until it gets blocked due to some I/O activity. During this time, to maximize the CPU utilization the OS switches to one of the remaining jobs in the memory and executes the newly selected one. When the current job gets blocked due to some I/O activity the CPU switches to yet another job in the memory. Eventually the 1st job will finish its I/O activity and gets the CPU back. This switching will continue until all the jobs are finished. Due to this approach CPU idle time is minimized and resource utilization is improved.

Later a variation of multiprogramming called *time-sharing systems* (also referred as *multitasking*) was introduced. In multiprogramming there is no user interaction. The user submits his/her job to the operator and then operator has the control; even the operator has a very little control while the batch is being executed. In time sharing systems, the user provides instructions directly from the keyboard to the computer and some of the results will be shown through the terminal at the same time.

These interactive I/O runs at "human speed" but for a computer it is slow. A user would be happy if he/she can type about 7 characters per second, but a computer can execute thousands of instructions within that time. Therefore, the CPU can execute another program by the time a user types the next character. During this small period of time the CPU executes multiple jobs by switching among them at a rapid speed. Each user is given a particular timeslot to access the CPU. When the timeslot gets expired the CPU will switch to another user's job. After some time, the first user will get the CPU for another timeslot.

Consider the example given in Figure 3.4 where three users having three unequal jobs: A, B and C[13].

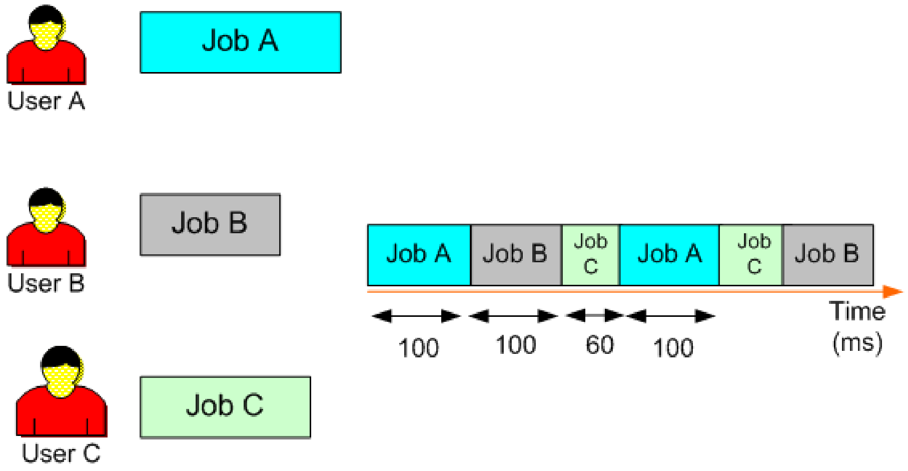


Figure 3.4 – A time-sharing system with 3 users

Suppose each user is given a time slot worth 100ms of CPU time. The CPU executes user A's job for 100ms then it will switch to user B. Then B's job is executed for another 100ms and the CPU switches to job C. Suppose that C's job requires some I/O activity during its timeslot, after 60ms. Then the CPU stops executing job C and again switches back to job A. This switching among jobs will continue until either a job expires its timeslot or gets blocked due to some I/O activity.

[13] The length of the job indicates the required CPU time.

Time-sharing system allows users to interact with their programs while it is being executed. Timesharing improves resources utilization and CPU is never idle. None of the individual users will realise that there are other users in the system, and everyone feels that they own the computer.

Currently we are in the fourth generation (1980 - present) of OSs which is referred to as personal computer OSs. Sometimes these OSs are also referred to as *desktop systems*. The jobs handled by these systems are neither CPU intensive nor I/O intensive and they make use of commodity hardware components. These systems are more interested on maximising user convenience and responsiveness. Anyway, they also incorporate many of the features available in larger OSs. Some of the features such as security and reliability which were not initially incorporated were added later.

3.5 Functions of an Operating System

Functionality of an OS slightly varies based on the type. The following is a list of common tasks carried out by a typical OS.

- **Memory management**: The OS is responsible for allocating memory for programs. The objective is to utilize limited memory in an efficient manner. It is also responsible for sending data and instruction from memory to the hard disk (i.e., swapping) when memory is filled up. The OS is also responsible to send the data back to the memory from disk when required for processing.
- **Spooling print jobs**: All the print jobs will be collected onto the hard disk and later they are scheduled by the OS for printing.
- **Configuring devices**: OS allows easy access to devices, their installation and configuration.
- **Monitoring system performance**: A modern OS can display how system resources such as CPU time and memory are utilized. This information can be used to identify whether resources are underutilized or over utilized.
- **Administering security**: The OS supports multiple concurrent users while making sure each user uses the system without interference by others. It also enforces authentication[14] and authorization[15].
- **Managing storage media and files**: The OS will make sure the secondary storage devices are optimally used. It will do all the hard work of saving and retrieving files to and from disks, timestamping, setting various file attributes such as read-only, hidden and so on.

3.6 Popular Operating Systems

There are three main classes of operating systems and many variations of those: Microsoft OSs, UNIX and mainframe OSs from IBM. Microsoft Disk Operating System (MS-DOS) later evolved into Windows while UNIX led to different variants such as Linux, Sun Solaris, FreeBSD, AIX, HP-UX and so on. IBM operating systems were mainly targeted towards mainframes and some of the well-known ones are OS/360, OS/390 and OS/400. MacOS for Apple Macintosh computers is another popular OS.

DOS was the first disk-based OS developed for IBM-PC by Microsoft. The objective was to keep the OS, application programs and all user files on a disk and managing them through set of commands called DOS commands. There are two well-known variants of DOS called PC-DOS (Personal Computer DOS) and MS-DOS (Microsoft-DOS). PC-DOS was developed and sold with IBM PCs while MS-DOS was sold in open market. DOS was simple to use and learn therefore Microsoft was able to win a large market share.

Inspired by the user interface of Apple Lisa, Microsoft decided to give DOS a Graphical User Interface (GUI). Microsoft released their first version of GUI based OS called Windows 1.0 in 1985. However, only Windows 3.1 was commercially successful. Early versions of Windows were just an application running on top of DOS; behind the scenes, GUI was issuing DOS commands. With the success of Windows 3.1, Windows 95 was introduced. Then with the introduction of Windows 98 Microsoft was able to escape from DOS legacy and built a complete GUI-based OS. It also supported multitasking (i.e., allowing users to have several applications running at the same time).

[14] Proving that a user is who he/she claims to be. This is mostly achieved by having a user name and a password.

[15] Based on privileges a user has, making sure each user can do (or access) only authorized tasks (or resources).

MS-Windows is used by many "home" users who do not have much technical background. It supports most of the hardware and software components. Recent versions of Windows such as Windows 7 and Windows 8 incorporate more user friendliness, better performance, efficient use of resources, security, and stability than before. However, compared to some other OSs Windows is not so stable, does not use resources in an effective way and have security issues. Yet due to its user-friendliness Windows is the most widely used commercial OS.

UNIX was developed in 1969 at Bell Labs by Ken Thompson and Dennis Ritchie. UNIX is still being used with many variants and versions. It was developed as a time-sharing system for minicomputers and was initially mostly used by universities. UNIX is so stable and highly reputed for its security, reliability, robustness, and performance.

It was developed by engineers for engineers. Therefore, it was harder for an average person to effectively use UNIX or any of its variants. As a result, UNIX was not so popular among the average users. However, modern variants with GUIs are much better in terms of user friendliness.

Linux was developed by Linus Torvalds in 1991. Linux is a UNIX-like OS developed originally for home PCs. However today it runs on many platforms including Intel, PowerPC, Macintosh, and Sun Sparc. It was developed with the intension of making it simple so that anyone can understand and improve. In contrast to all other OSs, *Linux is totally free, and the source code of the OS is* "*open*". Since its source code is freely available many people around the world had studied and improved it. Therefore, Linux is a complete OS which is stable, reliable, and efficient compared to most other OSs. It also supports excellent networking facilities.

Compared with MS-Windows, Linux requires less disk space, memory, and processing power. Both OSs support multitasking and multiprocessing; however, Linux seems better in terms of handling multiple users. Linux is free while MS-Windows costs about Rs. 10,000-20,000.

4 - Programming an Embedded System

4.1 Introduction to Embedded Systems

We discussed about hardware and software in personal computers and laptops in previous chapters. Yet today there are computers embedded into all sorts of everyday items from common house-hold appliances to automobiles and airplanes. These are usually called embedded computers or embedded systems, and account for more than 90% of all the world's manufactured processors.

An embedded system is a special-purpose computer which is completely encapsulated by the device it controls. A typical embedded system has a single specialized function (such as controlling a room temperature). Users of these equipment usually do not directly see or think that there is a computer embedded within the systems during their day-to-day use. The computer sits behind the equipment and controls its various components based on the input provided by the user as well as the inputs obtained from number of sensors attached to the system. The way this interaction occurs is usually implemented by the software that run inside the embedded computer. Due to the flexibility and advanced processing capabilities provided by the software, manufacturers of these equipment increasingly prefer to use embedded systems rather than their traditional hardwired controllers.

The main differences between the computers we know, and the embedded systems are that the embedded systems are used over a very long period of time and generally they cannot be programmed or maintained by the end user. When designing an embedded system there are design constraints such as limited memory, requirement of low cost, strict performance guarantee, fail-safe operations, low power consumption, reliability and guaranteed real-time behaviour. Most of the above constraints are not applicable to personal computers we daily use. Since the embedded systems are dedicated to a specific task, design engineers can optimize it, reducing the size and cost of the product.

These embedded systems often use simple executives (Operating System kernels) or real-time operating systems, support for real-time scheduling and no hard drives. Many embedded systems also interact with their physical environment using a variety of sensors and/or actuators.

Some examples of embedded systems include ATMs, cell phones, printers, thermostats, calculators, and video game consoles. Handheld computers or PDAs are also considered embedded devices because of the nature of their hardware design, even though they are more expandable in software terms. This line of definition continues to blur as devices expand.

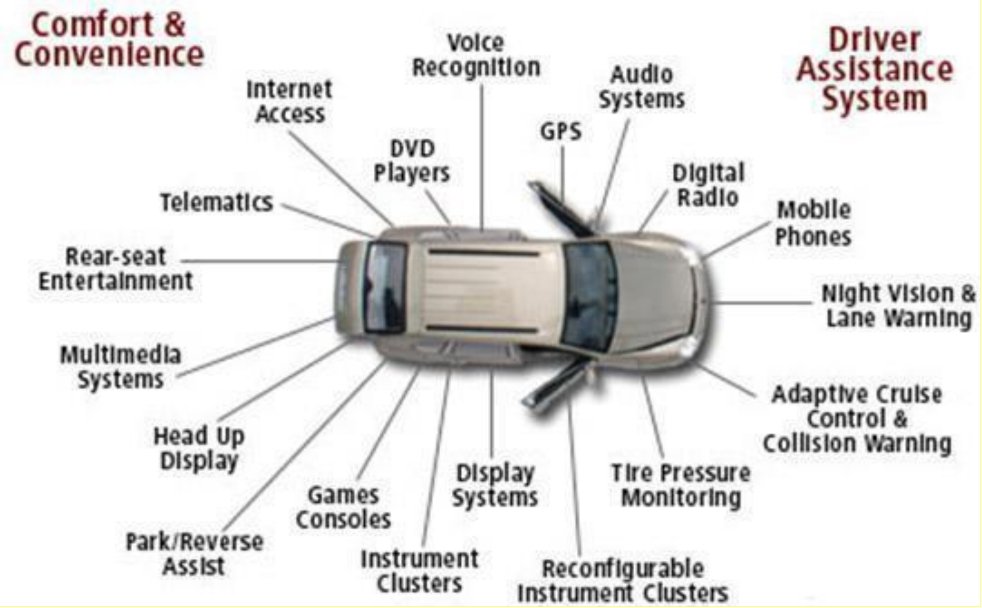


Figure 4.1 – Use of embedded computing in a new generation automobile

The controlling unit or the processing unit of an embedded system is typically a microcontroller. A microcontroller can be considered as a very simple and a small-scale computer in a single chip. It has a program memory, a processing unit and a random access memory (registers) which maps to hard disk, CPU and RAM in a typical personal computer respectively. Hence microcontrollers can be used in lightweight embedded applications.

Yet programming and designing circuits with microcontrollers are quite complex and not friendly for beginners. We have to write microcontroller programs separately and insert it into the device using a separate tool- a programmer. So, it is hard to debug and develop programs.

4.2 Introduction to Arduino

Arduino is an open-source physical computing platform based on a simple general purpose microcontroller board, and a development environment for writing software for the board. Arduino is a platform that can be used to develop and implement embedded systems.

Arduino can be used to develop interactive devices, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can communicate with software running on a computer (e.g., Flash, Processing, MaxMSP). The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free as well. So, it suits well for beginner in embedded programming.

Programming an embedded platform like the Arduino is somewhat different from programming a PC. Unlike the PC, Arduino does not have a keyboard, or a display screen directly attached to it. Moreover, the small amount of memory and storage available is insufficient to hist a usable program development environment such as an IDE and a compiler or an interpreter. Hence Arduino programs are usually written and compiled in a different computer (usually a PC) and later downloaded into the Arduino board via the USB cable. This process (i.e., compiling in a different computer and later downloading) is commonly referred to as “cross-compiling”).

Learning to program the Arduino environment is easy since it contains number of tools which hide messy details of microcontroller programming and wrap it up in an easy-to-use package. The advantage of this is that while simplifying the process of working with microcontrollers, it offers some advantage for teachers, students, and interested amateurs over other systems. It comes with a lot of hardware facilities ready to use.

In the lab 10 and 11 you will learn the basics of Arduino, an open-source development platform built around AVR microcontrollers. You will follow these two labs based on Lakduino UNO (Arduino UNO compatible) which uses ATmega328P. A development board (here Lakduino UNO) is basically the microcontroller plus additional circuitry like power units, protection circuits and communication units. In the labs we refer the Lakduino UNO board provided when it is mentioned ‘Arduino’. Please note that when using different Arduino boards, the designs and the schematics would be slightly different.

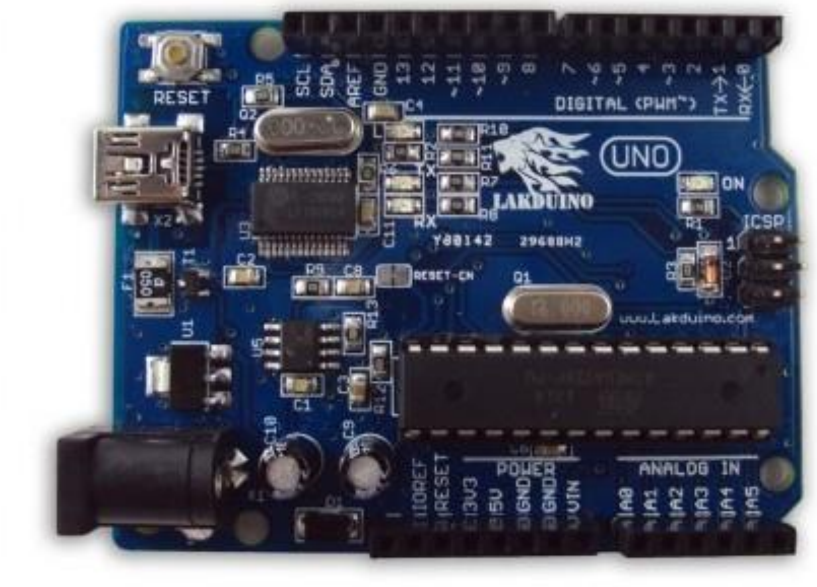


Figure 4.2 – Lakduino UNO

Arduino is a ready to go platform which you can use to simply plug into the computer, upload a program and run it without any additional circuit development. The Arduino IDE (Integrated Development Environment) is the software tool used to write programs and upload to the Arduino in this process.

4.3 Arduino Hardware

The Arduino board is where the code you write is executed. The board can only control and respond to electricity, so specific components are attached to the board to enable it to interact with the real world. These components can be sensors, which convert some aspect of the physical world to electricity so that the board can sense it, or actuators, which get electricity from the board and convert it into something that changes the world. Examples of sensors include switches, accelerometers, and ultrasound distance sensors. Examples for actuators are the things like lights and LEDs, speakers, motors, and displays.

The most popular boards contain a USB connector that is used to provide power and connectivity for uploading your software onto the board. Figure 4.3 shows the basic components of an Arduino Uno.

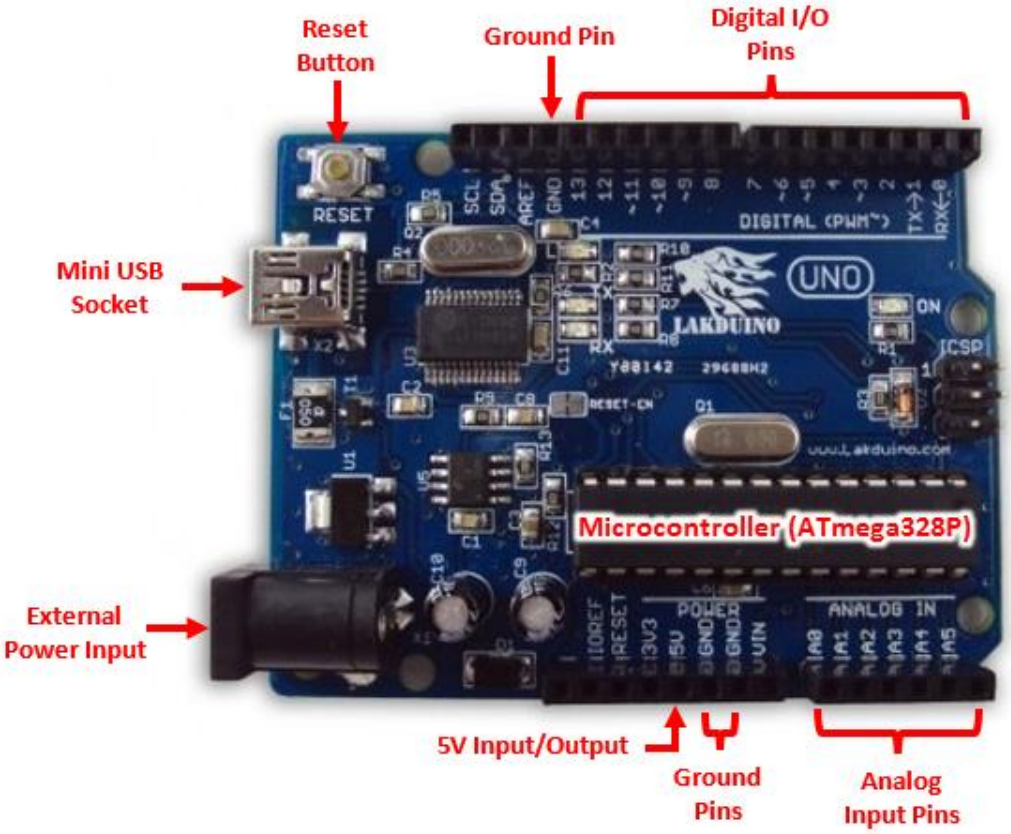


Figure 4.3 – Basic components of an Arduino board

The "Microcontroller" is the most important part of this board. It stores and executes the programs you write. All the other components are used as supporting units for this microcontroller's functions. The "Mini USB Socket" is used to connect the board to a computer using a standard USB mini cable. This USB connection can be used to power the board. This is a useful feature in the Arduino board since for simple applications, you don't need to worry about any external power supplies. The "External Power Input" socket can be used to power the Arduino board with a 6-20V supply. The "5V Input/Output" pin can be used either to power the board using a 5V supply when there is no other mode of power supplying is used or to get a 5V supply to provide to any other device when some other power supplying mode is in use on the board. The "Reset Button" can be used to force the board to re-start executing the uploaded program from the very beginning. The "Digital I/O Pins" can be used to provide digital (0 or 5V) outputs and to read digital inputs by configuring them in the program. You can see that certain pins (3, 5, 6, 9, 10 & 11) are marked with a tilde (~). These pins can be used as PWM (Pule-Width Modulation) outputs. In PWM configuration, you can use these pins to effectively output voltage values in the range 0-5V. The "Analog Input Pins" can be used either as digital I/O pins or as input pins which can read voltage values in the range 0 – 5V by configuring them in the program. The "Ground Pins" can be used as ground connections for external circuitry. This is a very basic introduction to the pins and other components of the Arduino Uno board which will be needed in this module.

4.4 Arduino Software

Software programs, called sketches, are created on a computer using the Arduino integrated development environment (IDE). The IDE enables you to write and edit code and convert this code into instructions that Arduino hardware understands. The IDE also transfers those instructions to the Arduino board (a process called uploading).

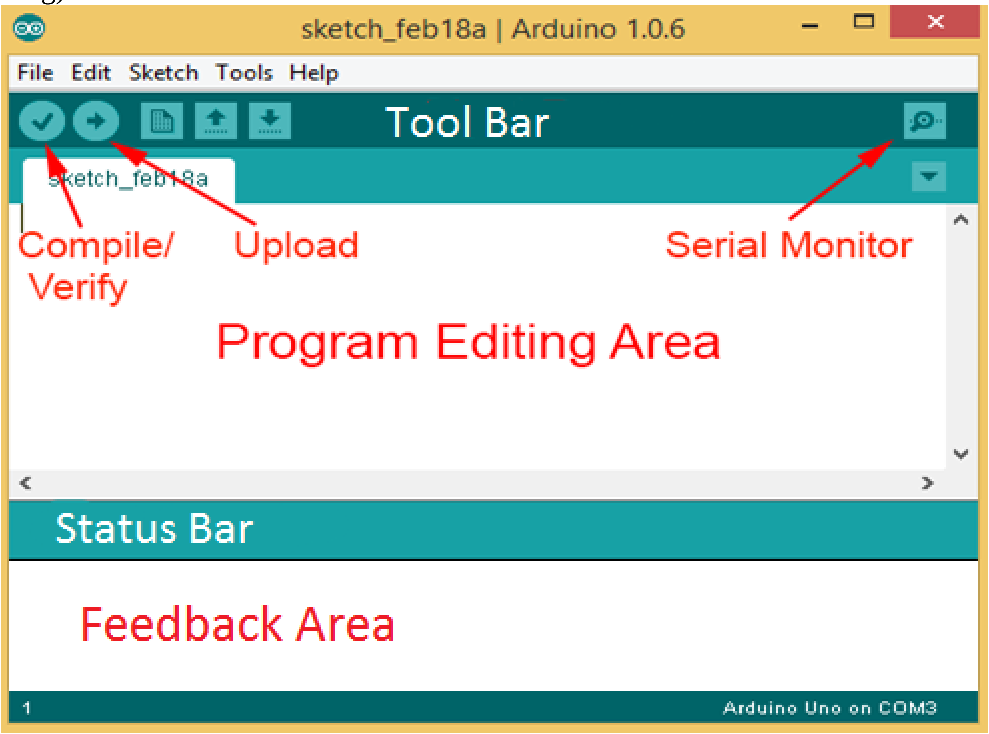


Figure 4.4 – Arduino IDE's interface

Figure 4.4 shows the interface components of the Arduino IDE. The "Program Editing Area" is where you should write your Arduino program in. The "Compile/Verify" button compiles the program written in the "Program Editing Area" and checks for any syntax errors. The "Upload" button is used to compile and upload the program to the Arduino board. The "Serial Monitor" button will open another window which can be used in serial communication with the Arduino board. The "Status Bar" and the "Feedback Area" show the state and any compiler/upload errors.

4.5 Brief Introduction to Electronic Components

Resistor

Various electronic circuit components, such as the Arduino's LED, require only a small amount of current to function- usually around 10mA. When LED receives excess current, it converts the excess to heart, which results in burning the LED. To reduce the flow of the current we can add a resistor between the voltage source and the component, which will convert certain amount of current (proportional to the resistor value) into heat energy, reducing the load of the current on the component.

Reading resistance values:
As resistor is very small, the resistance values cannot be printed to resistor itself. To indicate the resistor value of the resistors a series of color-coded bands are used. They should be read from left to right as follows:

First band: First digit of the resistance
Second band: Second digit of the resistance
Third band: The multiplier (if resistor has four bands) or the third digit (if the resistor has five bands)
Fourth band: Represents the multiplier for the five-band resistor.
Fifth band: Shows the tolerance (accuracy of resistance values)

The table 4.1 lists the colors of resistors and their corresponding values.

Color	**Ohms**
Black	0
Brown	1
Red	2
Orange	3
Yellow	4
Green	5
Blue	6
Violet	7
Gray	8
White	9
Gold	±5%
Silver	±10%

Table 4.1 – Colors of the resistor bands and their corresponding values

The example resistor diagram in the figure 4.5 can be read as 47,000 **Ω**. The yellow, violate, and orange resistance bands are read as 4,7 and 3 respectively. The orange band represents the multiplier 3 – which indicates 10^3. The 4th band indicates that the resistor value can vary from 10% of its marked values.

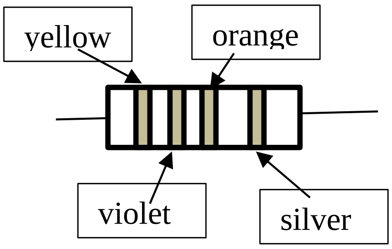


Figure 4.5 – Example resistor diagram

Light-Emitting Diode (LED)

The LED is very common and largely used component that converts electrical current into light. There are LEDs of various shapes, sizes, and colours.

As the figure 4.6 indicates the LEDs are polarized. The current should enter via anode and leave via cathode. Hence special care should be taken when connecting LEDs to the circuit. A large current to the opposite direction through an LED will burn the component. The anode and the cathode can be identified by the length of the legs as shown in the figure 4.6.

When developing a circuit using LEDs, you need to consider operating voltage and current. For example, common red LEDs require around 1.7 V voltage and 5 to 20 mA current. But this is bit problematic because the Arduino board outputs 5V and a high current. Therefore, appropriate a resistor should be connected with the LED serially to limit the current. The resistor values should be selected by calculating the appropriate value using Ohm's law.

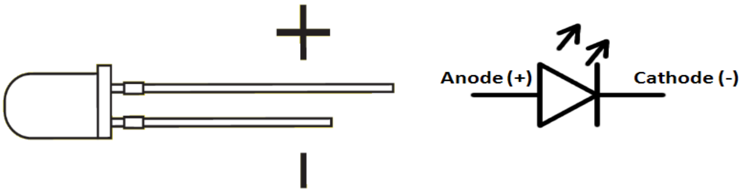


Figure 4.6 – Current flow though LED. The design of LED (left) and the schematic diagram (right)

Solderless Breadboard

The circuits you build need to be changed frequently, to improve or fix the errors. So, a base is needed to hold the components of the circuit together and build upon. The solderless breadboard (commonly known as breadboard) is useful for this purpose. It is a plastic base with rows of electrically connected sockets.

To use a breadboard, we should know how the sockets are connected. They vary by the board. If we take the configuration of the breadboard shown in figure 4.7 the selected five-hole columns are internally connected, but not horizontally. The selected rows also connect internally.

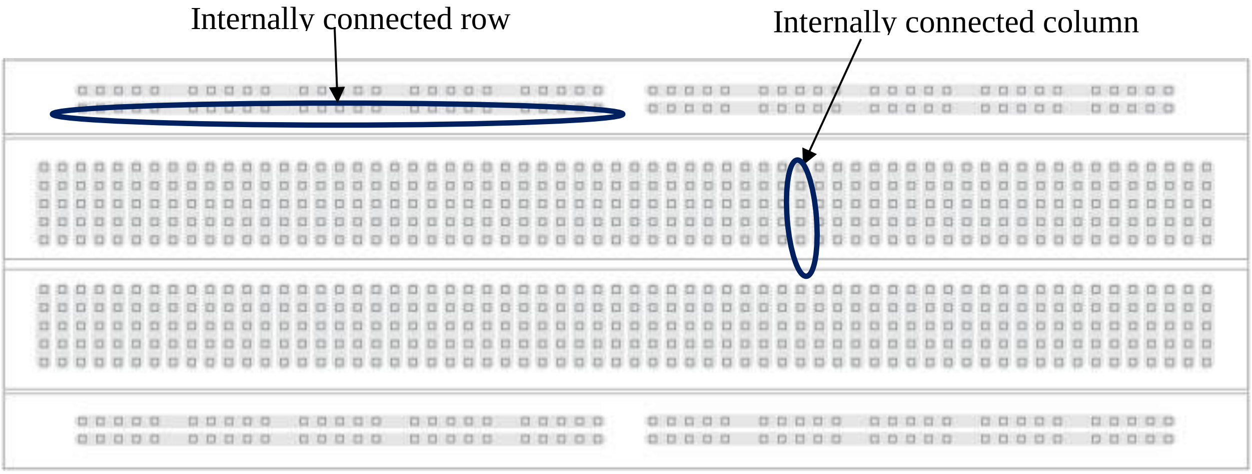


Figure 4.7 – Connections of the Solderless Breadboard

4.6 Installing the Integrated Development Environment (IDE)

You want to install the Arduino development environment on your computer to develop software for Arduino. The Arduino software for Windows, Mac, and Linux can be downloaded from http://arduino.cc/en/Main/Software. Follow the instructions provided in the http://arduino.cc/en/Guide/HomePage for installation details.

4.7 Developing Embedded System Applications Using Arduino

By developing software that can manipulate hardware components and properly integrating them, you can develop various useful embedded systems. The lab 10 and 11 provide guidelines on how to develop a basic Arduino program. Using the resources and tutorials provide in the reference section you can we can learn and develop further applications.

When starting to develop an embedded systems application there are basic preparatory steps you can use in order to get expected result effectively. The basic steps you can follow are:

1. **Determine objective of your application**: how it should behave.
2. **Write your algorithm**: Write a pseudo code or flow chart, which contains the set of instructions that describe how you are going to achieve the objective.
3. **Select your hardware**: Select the hardware components you need and decide how it should be connected to the Arduino. Draw a schematic diagram.
4. **Write the sketch program**: Develop the algorithm into software program using Arduino IDE.
5. **Connect the hardware components**: Connect the hardware, circuitry, and other items to Arduino board.
6. **Test and debug**: Your program may not give intended output first time. Identify errors and their causes. They may be in the hardware, the sketch or algorithm. Fix them.

Reference for this section:

1. Arduino web site: http://www.arduino.cc/
2. Arduino Cookbook: https://www.safaribooksonline.com/library/view/arduino-cookbook/9781449399368/ch01.html
3. Arduino uno data sheet (http://docs-europe.electrocomponents.com/webdocs/0e8b/0900766b80e8ba21.pdf)

5 - Current Trends in Computing

5.1 Cloud Computing

Have you used Facebook, Google Search, Dropbox or a web-based email such as Gmail, Yahoo! or Hot mail? If so, you have experienced cloud computing. These applications require that, you log into an account remotely. The software and storage for your account is on the service provider's systems which are collectively referred to as the "Cloud". Facebook, Google Search Engine, Skype, Evernote, etc. use cloud technology due to its enormous benefits.

Cloud computing is a subscription-based service which provides a pool of configurable computer resources: storage, networks, servers, applications, and services, over the internet. A company can subscribe for a "cloud service" provided by a service provider and use the resources provided by that service provider for the operation of their business. For an example, imagine a company that wants to have an additional 100 computers only for a week to process a large set of data for one of their urgent projects. Instead of buying 100 computers this company can subscribe to a cloud service and use computers at service provider's end through the internet during the week. Then they only have to pay for the week they used those computers. This approach would save a lot of money for the company.

Characteristics of cloud computing:

- **On demand service:** The users of a cloud service can change the service according to their requirements, usually through a web portal without human intervention. A pricing model will determine the charges for used services.
- **Broad network access**: Cloud computing resources can be accessed using mobile phones, tablets, laptops, or personal computers over the network.
- **Rapid Elasticity:** Cloud resources can be provisioned or released at any time based on application demand, enabling the application to have exact amount of resources that it needs.
- **Pool of Computing Resources:** Service providers serve multiple consumers with a pool computing resources by dynamically allocating and de-allocating resources.
- **Measured Service:** Cloud resources usage is monitored and measured by both service user and service provider and billed appropriately. Therefore, businesses have to pay only for what they use.

While cloud computing can bring many advantages to a business, it also can bring some disadvantages. Possible downtime, security and privacy concerns are some of them.

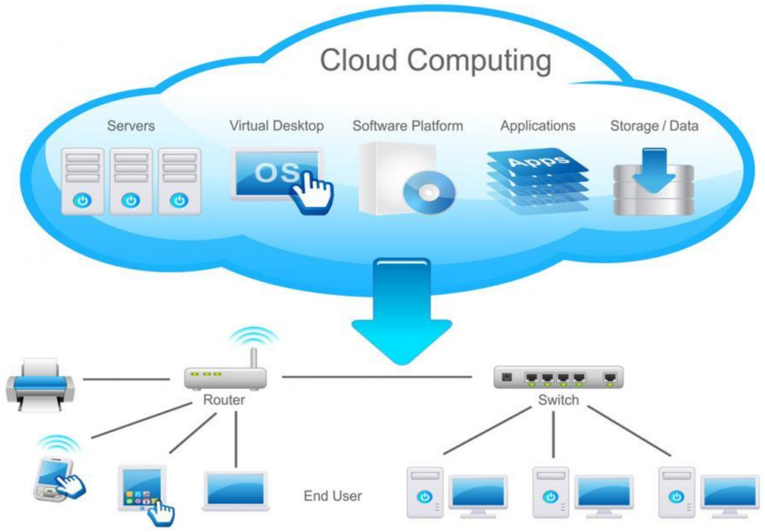


Figure 5.1 – Cloud

5.2 Augmented Reality

As name suggests Augmented Reality (AR) augments / supplements user’s physical, real-world environment with computer generated sensory inputs such as video, sound, graphics, or location (E.g.: GPS) data. Therefore, AR enhances one’s perspective of reality with additionally provided interactive and digitally manipulable information about the environment. Users can visualize information about the environment and its objects which are laid on top the real world (see Figure 5.2).

Augmented Reality is applicable for many fields including architecture, art, commerce, education, gaming, medical, etc.

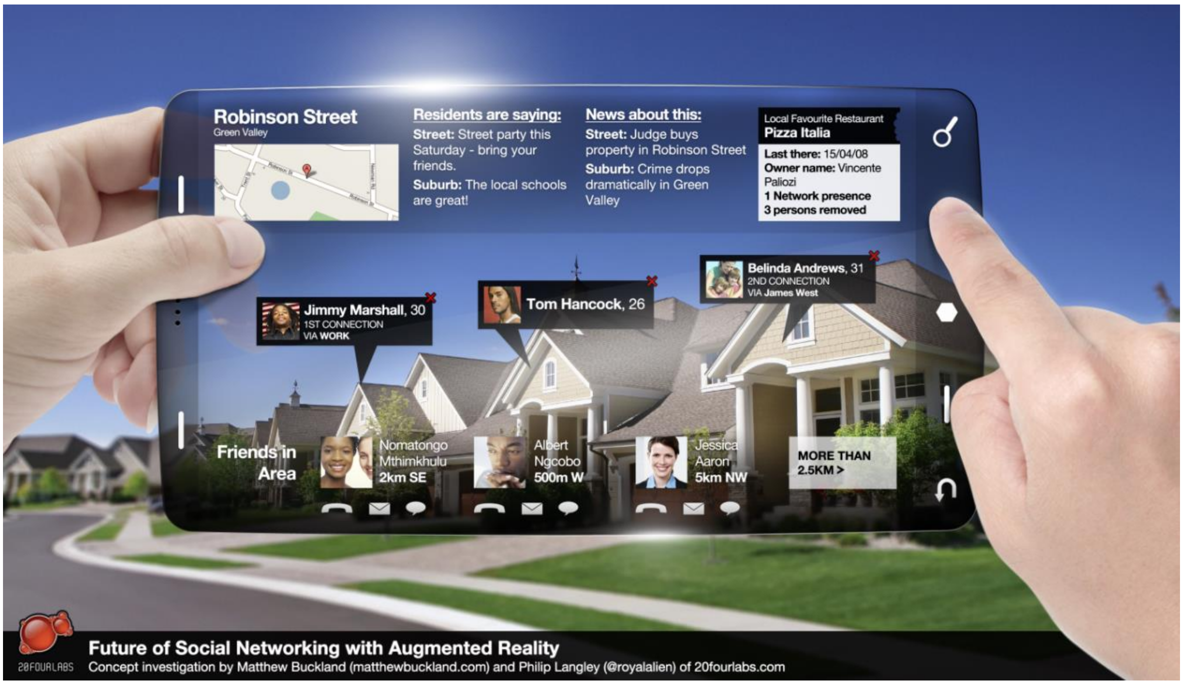

Robinson Street
Green Valley
Residents are saying:
Street: Street party this Saturday - bring your friends.
Suburb: The local schools are great!
News about this:
Street: Judge buys property in Robinson Street
Suburb: Crime drops dramatically in Green Valley
Local Favourite Restaurant
Pizza Italia
Last there: 15/04/08
Owner name: Vincente Paliozi
1 Network presence
3 persons removed
Jimmy Marshall, 30
1ST CONNECTION
VIA WORK
Tom Hancock, 26
Belinda Andrews, 31
2ND CONNECTION
VIA James West
Friends in Area
Nomatongo Mthimkhulu
2km SE
Albert Ngcobo
500m W
Jessica Aaron
5km NW
MORE THAN 2.5KM >
20FOURLABS
Future of Social Networking with Augmented Reality
Concept investigation by Matthew Buckland (matthewbuckland.com) and Philip Langley (@royalalien) of 20fourlabs.com

Figure 5.2 – Augmented Reality Example

5.3 Ubiquitous Computing

Ubiquitous computing (also referred to as pervasive computing) is the concept of enabling computing everywhere at any time. Basically, this refers to the use of computers in everyday life. Ubiquitous computing integrates computers in any device from clothing to tools to vehicles to homes to human body. This enables people to interact with computers in a more natural way. Examples for use of ubiquitous computing are smart phones, PDA (Personal Digital Assist), Smart Glasses, digital watches, and interactive white boards.

Characteristics of ubiquitous devices:

- Very tiny in physical size
- Being integrated into any shape of device and displacing services of desktop systems
- Augment the original use-value of devices they are embedded
- Communicating through increasing interconnected networks

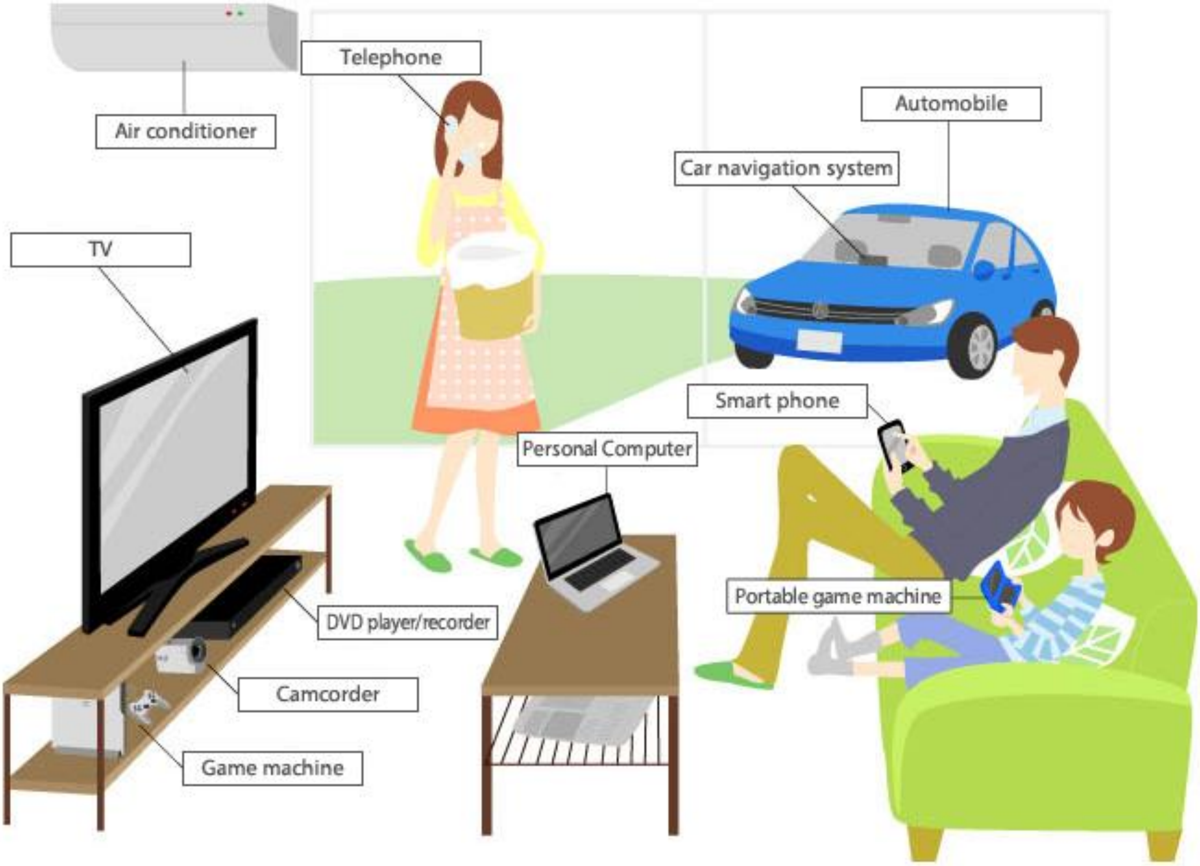


Figure 5.3 – Examples of Ubiquitous Computing Usage

6 - Future Trends and Challenges of Computers

6.1 Challenges in Embedded Computing

The field of embedded system research is rich with potential because it combines two factors. First, the system designer usually has control over both the hardware design and the software design, unlike general-purpose computing. Second, embedded systems are built upon a wide range of disciplines, including computer architecture (processor architecture and micro architecture, memory system design), compiler, scheduler/operating system, and real-time systems. Combining these two factors means that barriers between these fields can be broken down, enabling synergy between multiple fields, and resulting in optimizations which are greater than the sum of their parts.

One challenge with embedded systems is delivering predictably good performance. Many embedded systems (e.g., anti-lock brakes in a car) have real-time requirements; if computations are not completed before a deadline, the system will fail, possibly injuring the user. Unfortunately, many of the performance enhancing features which make personal computers so fast also make it difficult to predict their performance accurately. Such features include pipelined and out-of-order instruction execution in the processor, and caches in the memory system. Hence the challenge for real-time system researchers is to develop approaches to design fast systems with easily predicted performance, or to measure existing complex but fast systems more accurately.

6.2 GPU-Accelerated Computing

GPU-accelerated computing is the use of a graphics processing unit (GPU) together with a CPU to accelerate scientific, engineering, and enterprise applications. GPU-accelerated computing offers unprecedented application performance by offloading compute-intensive portions of the application to the GPU, while the remainder of the code still runs on the CPU. From a user's perspective, applications simply run significantly faster.

Figure 6.1 – Nvidia TESLA GPU

A simple way to understand the difference between a CPU and GPU is to compare how they process tasks. A CPU consists of a few cores optimized for sequential serial processing while a GPU consists of thousands of smaller, more efficient cores designed for handling multiple tasks simultaneously.

6.3 Reconfigurable Computing

Reconfigurable computing is a computer architecture combining some of the flexibility of software with the high performance of hardware by processing with very flexible high speed computing fabrics like field-programmable gate arrays (FPGAs). The principal difference when compared to using ordinary microprocessors is the ability to make substantial changes to the data path itself in addition to the control flow. On the other hand, the main difference with custom hardware, i.e., application-specific integrated circuits (ASICs) is the possibility to adapt the hardware during runtime by "loading" a new circuit on the reconfigurable fabric.

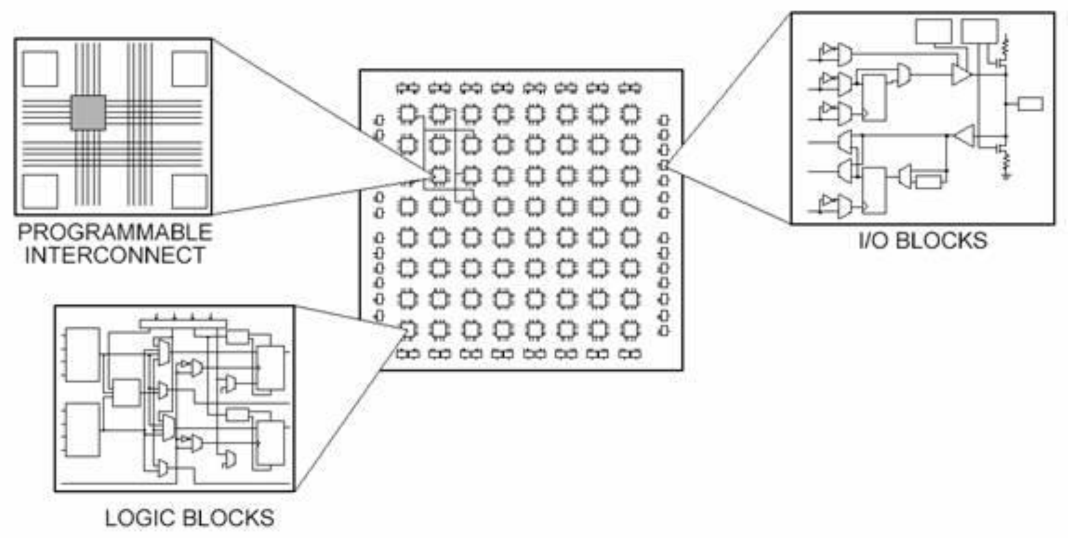


Figure 6.2 – Different Parts of an FPGA

6.4 Quantum Computing

A quantum computer (also known as a quantum supercomputer) is a computation device that makes direct use of quantum-mechanical phenomena, such as superposition and entanglement, to perform operations on data. Quantum computers are different from digital computers based on transistors. Whereas digital computers require data to be encoded into binary digits (bits), each of which is always in one of two definite states (0 or 1), quantum computation uses qubits (quantum bits), which can be a *superposition* of states. A theoretical model is the quantum Turing machine, also known as the universal quantum computer.

As of 2020 quantum computing is still in its infancy but experiments have been carried out in which quantum computational operations were executed on a small number of qubits.

Both practical and theoretical research continues, and many national governments and military funding agencies support quantum computing research to develop quantum computers for both civilian and national security purposes, such as cryptanalysis. Large-scale quantum computers will be able to solve certain problems much more quickly than any classical computer using the best currently known algorithms.

References for this section:

- http://www.ece.ncsu.edu/research/cas/ecs
- http://www.nvidia.com/object/what-is-gpu-computing.html
- http://en.wikipedia.org/wiki/Reconfigurable_computing
- https://www.ibm.com/quantum-computing/learn/what-is-quantum-computing/
- https://research.google/teams/applied-science/quantum/

----------------- END OF PART C ------------------