1 Introduction

The term computer usually refers to “general purpose” personal computers or workstations. They include all desktop and laptop computers (PC or Mac), workstations (SUN, IBM, HP, SGI, etc.), and large mainframes (IBM, Fujitsu, NEC, Hitachi, NEC, etc.). There are of course many other types of computers, for example special purpose devices employed anywhere from airplanes and space shuttles to industrial robots to household appliances, to GTS systems in newer cars. Finally there are new handheld devices (many smart phones, Palm and similar devices, personal organizers, etc.). In this note we focus on the general purpose machine and review hardware and software for them.

2 Hardware

The physical components of a typical computer such as a common desktop PC include:

2.1 The Central Processing Unit or CPU:

This is where the actual operations of the machine are performed. The modern central processing units contain millions of transistors etched on a tiny silicon chip that can easily fit in one’s palm or sometimes on the tip of one’s thumb. The CPU has circuitry that can

- perform arithmetic operations such addition and multiplication, circuitry that actually do these operations is called “the Arithmetic Logic Unit (ALU)” which in turn may contain other circuits such as “adder” or “multiplier”, or “comparator” (which compares two numbers and decides which one is bigger or if they are equal), left and right “shifters”, etc.,
2.1 The Central Processing Unit or CPU:

- read electronically encoded computer instructions and data, decode them, and channel them to the appropriate circuitry to execute them, the part of the CPU that performs these kinds of operations are called “the Control unit”, which in turn may contain other circuitry such as “decoders”, “multiplexer”, and other logic circuits,

- hold the most immediately needed data and instruction codes in circuits called “registers”,

- Finally wiring, abstractly called “data paths” to connect these components together.

In addition almost all modern computers have extremely fast circuits that can hold somewhere from 256,000 (256K) to about 1,000,000 (1M) “bytes” of data (each byte is roughly equivalent to a single character). Such banks are called “cache”. We will discuss the role of cache below when we examine computer memory.
2.1 The Central Processing Unit or CPU:

Essential components of CPU

There are usually one CPU in each computer, although more expensive and faster computers can have anywhere from two to 64 (or even more) of them.

You may have heard that many CPU manufacturers such as Intel and AMD use “mega Hertz” (MHz) to advertise the speed of their computers. What does it really mean and is it a true measure of speed? To answer this question you need to know the role of the CPU’s “clock”. Each CPU contains a clock which in regular intervals switches from an “up-tick” to a “down-tick”. To understand the role played by the clock let’s look at a simple example. Suppose the CPU needs to add two numbers. Here is what happens:

1. The first number is loaded into a register, say register A.
2. The second number is loaded to a register B,
3. The two numbers are fed to the adder circuit where the sum of the two numbers is calculated
4. The sum is loaded into a register, say register C.

(At this point we don’t concern ourselves how these numbers got into the registers in the first place.) Now to synchronize all these operations the CPU actually allows the adder to add only when the clock is in at an up-tick. When the clock is at a down tick the two numbers are loaded into the adder. A CPU
can perform one or more operations during the up-tick of a clock. If a component of the CPU, say the adder is fast then it can complete its operation in a short up-tick, however a slower CPU may need a longer duration of the up-tick. The number of up-ticks per second is measured in “Hertz”. Thus a 1-GHz (giga Hertz) CPU has a clock that has one billion up-ticks per second. If this CPU can perform one instruction per up-tick, then it can perform about one billion instructions per second. However it is possible to have a CPU with a 900 MHz (Mega Hertz) clock that can perform roughly two operations per up-tick. Then this CPU is faster even though it has a slower clock.

Thus, in evaluating the performance of a CPU, the rule of thumb is that if you are comparing two similar class CPU’s (for instance two Pentium IV chips) then one with faster clock is probably faster. However comparing, say a Power PC chip (manufactured by IBM and Motorola and used in Apple Macintosh computers) and an Intel chip solely based on clock rate is not accurate.

Also you should realize that faster CPU is only one way of measuring overall speed and performance of a computer. There are other factors that you will see shortly.

2.2 Main Memory

The main memory is is electronic circuitry capable of holding information (as long as the computer is on), and releasing it when needed. The least expensive modern computers nowadays have at least 64 megabytes of memory and one giga byte computers are becoming mainstream. The capacity and speed of main memory has a direct effect on the speed of a computer. Often one can significantly improve the performance by adding more memory to a computer, a relatively inexpensive and easy procedure. On the other hand replacing the CPU by a faster one is usually much more expensive and not as straightforward, and the improvement in performance may not turn out to be noticeable.
2.3 Cache

These are circuits that can hold information and release them at a later time, just like the main memory. However, they can send and receive data to and from CPU several times faster than a typical main memory unit can. The cache however, has much less capacity than main memory; as we said earlier, cache circuits can hold anywhere from 256K to 1M bytes. Usually there are one or two levels of cache in a computer. Level one (L1) cache is usually on the same chip as the CPU and it is extremely fast. Level 2 may be on a separate chip that resides on the motherboard or may be on the same chip as the CPU. It is somewhat slower than L1 cache, but still quite fast. Faster and larger cache has a strong influence on the speed of a computer. In the Intel Pentium processors, two chips may have the same clock speed, but one may have on chip (L1) cache and the other may not (or may have a much smaller cache). The chip with larger cache has a noticeable advantage in performance. For example Intel Pentium Celeron processors have smaller cache than their corresponding Pentium II or III, or IV processors. Thus a Celeron with approximately 800MHz clock is slower than the same generation Pentium III at about 800MHz.

2.4 Hard Disk and Other Secondary Storage Devices

In addition to main memory a computer almost always needs one or more devices that store information. There are two reasons for this. First, even one giga byte is not enough nowadays to store all the information a computer needs. Second, the information stored in the main memory is volatile: When the computer is
turned off all the information in it is lost. Therefore, we need other storage devices where the information is stored indefinitely until it is explicitly wiped out, devices that keep this information even if the computer is shut down.

The most important secondary storage device is the hard disk which is usually a magnetic device (as opposed to solid state circuitry) and can hold large amounts of information. The hard disk contains one or several circular disks that can turn about a spindle through its center. A reader head is attached to a handle going access the circular disk. To fetch data at particular location the disk spins and the head move on top of the location and then the information, stored magnetically, is collected. These days hard disks with a capacity of sixty, eighty, even 200 giga bytes are available at prices in the neighborhood of $100. Hard disks are quite fast, but they are still much slower than the main memory.

Other types of secondary storage devices exist which are usually meant to be portable, that is they can be removed quickly from one machine and put into another. CD-Rom and DVD-Rom devices that are based on storing information and retrieving it using laser beams are prominent examples of removable storage devices; CD-Roms are older technologies that can hold up to 700 mega bytes of
information, DVD-Roms are newer technologies and can hold up to 4 giga bytes of information. Other removable storage devices include compact flash devices, smart media and similar small devices that are based on solid state circuitry, but can keep the information stored in them even if power shuts off. The speed of information exchange to and from these devices is pretty fast, but not as fast as the main memory.

2.5 Input/Output devices and peripherals

A computer with only CPU, memory and hard drive is hardly useful unless it has means to receive information to process and means to communicate the results of its process back. The input/output devices are used to communicate information to and from computers. Keyboards and mice are the most typical input devices that can receive information directly from the person working on the computer, but so are all the external secondary storage devices. The display monitor is the most typical output device that can display information processed by the computer directly to the person working on the computer. In addition to these devices one can add electronic pens, tablets, joysticks, microphones and
digital cameras as other typical input devices. Loud speakers and printers are other typical output devices.

To process sound, as input and output, computers are equipped with sound cards. They transform input sound (coming say, out of a microphone) and turn it into a stream of bytes that can be processed by a computer or stored in primary or secondary storage devices. And they transform a stream of bytes back to analog signal that can be sent to loud speakers and played as sound.

A sound card with jacks for microphone, and speakers

To process visual data, that is both still pictures and moving pictures, computers must be equipped with video cards. Like sound cards, these cards can transform a stream of analog signal representing a picture or a sequence of pictures into bytes that can be stored and processed in a computer, and also can transform such stream of bytes back into analog signal that can be turned into pictures in the display monitor. Since the amount of information in a picture or movie is extremely large, video cards these days have their own dedicated memory that can be as much as 256 mega bytes. The speed of video cards often is very important in applications that are heavily dependent on visual data such as engineering applications of computer aided design (CAD), some computer games, video editing and computer animation.
A graphics card with 256 Mega bytes of dedicated memory

Another set of input/output devices are those that allow computers to communicate to each other. They involve Ethernet cards, modems, wireless cards, infrared and bluetooth devices, and of course all the infrastructure that carry information between computers in great distances (telephone lines, cable lines, satellites, and dedicated Ethernet wiring.) This set of devices are used for e-mail, web browsing, uploading and downloading information to and from other computers. Again the speed of these devices are significant in the overall performance of computers. 56K modems can communicate over existing telephone lines to other computers with at most 56,000 bits (not bytes!) per second (bps); This is quite slow by todays standards, but fairly adequate for e-mail and web browsing involving mainly text. DSL (using telephone lines with a different technology), cable (using cable TV lines) and satellite communications are much faster (they are sometimes dubbed “broad band”). They are fast for web browsing and downloading large files. However, they are still quite slow for downloading movies and files in excess of 100 mega bytes. Wireless device are nowadays about half as slow as Ethernet cards (which is still quite fast) and liberate laptops and desktops from being tied to wires to stay online.
2.6 Input Output ports

External input and output devices such as mice, keyboards, scanners, digital cameras, printers, etc. need special “ports” that act as plugs for them to communicate to other components. There are many kinds of ports available on computers, and I list some more common ones below.

- **Serial and Parallel ports**: These are older and slower ports. The serial port is used to connect mice, external modems, external interface to older handheld devices, older tablets and similar devices. Parallel ports are used primarily for printers, but can also be used to connect to some older devices, such as old zip drives. Both of these devices are considered outdated today and they are found in modern computers only not to render older devices useless. They are however on their way out. For Mice and keyboards, there are also PS/2 ports, which are quite common today.

- **USB and Fire wire or IEEE 1394**: These are essentially replacements for parallel and serial ports. USB ports are faster interfaces with additional advantage that one can in a daisy chain like way connect several USB devices together and connect one to the USB port of the computer. Then all the devices are detected by the computer. Thus a single USB port can connect potentially to several devices. Firewire is a similar technology. Firewire (or IEEE 1394) devices are faster than older USB devices (called USB 1). But now there is USB2, which is newer and faster but still compatible with USB 1 devices. Firewire also has a new version (sometimes called Firewire 800 for its 800 Megabytes per second speed). Devices with USB and Firewire are rapidly replacing serial and parallel port devices. These devices have made it possible that massive amounts of data are passed from external devices to computer at acceptable speed. For instance it is unthinkable to connect a digital camcorder via a serial or parallel port. However, they are connected through USB 2 or Firewire ports with very good performance.
2.7 The Motherboard

In the picture above

- **COM ports** are for connecting serial bus devices, such as older external modems and mice,

- **LAN Port** is for connecting to the Internet through cable or DSL modems or the Local Area Network (LAN), that is the private network of computer or school which may or may not be connected to the Internet.

**2.7 The Motherboard**

All the devices above need a platform to be mounted on, and more importantly to communicate to each other. The “motherboard” fulfills this task. It is a board where other devices are mounted directly, or attached through ports, and communicate with each other through wirings provided on it. The motherboard has interfaces, where the CPU, the main memory, external (L2) caches, video and sound cards, and internal modems and Ethernet cards are mounted or attached by specialized wirings. These and other plugs (such as USB, serial and parallel ports as wells as plugs for CD/DVD-Rom and hard drives can be
connected. It is through motherboard that these devices communicate to the CPU and the main memory. The motherboard contains a very small special purpose computer on it with special programming that directs the way physical components can communicate with each other. This special purpose computer is programmed by BIOS (Basic Input Output System) and this program may be changed and upgraded.

The motherboard and and ports and interfaces where components are mounted

In the picture above

- *Socket 478 Connector* is a special kind of socket where the an AMD CPU is mounted,
- *AGP slot* is where the graphics card is mounted,
- *PCI slots* are for mounting devices such as sound cards and modems and Ethernet card,
- *IDE ports* are for hard drive, DVD/CD-Rom drives,
- *DDR SDRAM slots* are where main memory is mounted.
3 Programming computers

As was stated earlier the CPU receives instructions and executes them. A “*program*” is a sequence of instructions that are executed by the CPU in order to accomplish some task. The task could be anything that you do with a computer; it could be as elementary as a simple arithmetic operation like adding or multiplying numbers, or it could be putting a list of names in alphabetical order, it could be transforming a picture by scaling or rotating it, or it could be receiving and sending e-mail. Let’s look at the process of executing a program a bit more closely.

3.1 The alphabet of language of machines: bits, bytes and words

The first thing we need to realize is that the instructions and data that the CPU needs to process is in a specific language called “*machine language*”. The words in this language use an alphabet that is made of “*bytes*”. At the lowest level a computer can detect a 0 or a 1, called a “*bit*”, usually encoded in the
form of low or high voltage in a circuit. A byte is made of 7 or 8 bits. Thus one byte can have 128 (for seven bit) or 256 (for 8 bit) distinct basic sequences of 0’s and 1’s, or an alphabet. Roughly speaking a byte can encode a single letter in Latin alphabet, which includes 52 lower case and capital letters, 10 numerical digits, and a score of other symbols from punctuation symbols to key strokes corresponding to carriage return and bell sound. Traditionally the ASCII character set is a table to lookup the exact binary encoding of characters. Thus, when you punch in, say the letter ‘e’ on the keyboard, a sequence of 0’s and 1’s corresponding to the binary encoding of letter ‘e’ is produced by the keyboard and is sent to the computer to be processed. These days however a newer character set is slowly replacing ASCII called Unicode which uses 16 bits and can therefore encode 65,536 distinct characters. This is good enough to encode all characters from all known human languages (including in addition to Latin, Greek, Russian, Arabic, Hebrew, Tamil and other Indian dialects, Chinese, and Japanese.) Java uses primarily the Unicode encoding.

3.2 How is a program executed?

A program is encoding of a sequences of instructions (such as add, multiply, load, write, etc.) and data upon which the operations should be applied. Typically this sequence is stored on a secondary storage such as the hard disk or CD-ROM. When the computer is ready to execute them, the instructions are loaded into the main memory and from there they are sent to the CPU. If there is no cache instructions and data are sent to the CPU one at a time. But if there is cache, the instructions are sent in blocks to the cache and from there to the CPU. The second way is faster as the communication between cache and CPU is faster than the communication between main memory and the CPU. The CPU reads instructions one at time, loads the appropriate data and launches the right circuitry to perform the operation. It then sends the results back to the main memory via the cache. For Example If we have two numbers called A and B and want to find their sum, it is assumed that A and B, the binary encoding of our numbers are loaded into memory along with the encoding of the operation ADD. The CPU first loads A to one of its data registers. Then it loads B to another data register. Next it loads the encoding of the instruction (ADD here) to an instruction register. Next it invokes the adder circuit to add the numbers A and B. Finally the result is then passed back to the main memory. So this simple operation actually requires 4 operations:

Load from memory location A
Load from memory location B
ADD A and B
WRITE To memory location C
If the CPU can perform one operation per cycle this will take 4 clock cycles. But modern CPU’s may perform these process in one cycle.

4 Software

4.1 Programming Languages, Assembly and Machine Languages

In the early days of computers back in the 1950’s engineers who wanted to program a computer had to master the language of the machine. They literally needed to know the 0-1 formulation of various instruction and they composed programs by writing these instructions in a sequence and fed them to the computer. Later they made a slight improvement by adding easy to remember words for instructions. Thus a programmer needed only to remember “ADD” rather than the binary encoding of addition to create programs. In late fifties a revolutionary idea was thought of which was the invention of “high level programming languages”. The idea was to use mathematical symbols such as ‘+’, ‘-’, ‘=’ and the like and write operations using algebraic symbols. In addition for other operations such as “repeat the following instruction so many times” or “jump to line x”, they created certain easy to remember words. People started to use this type of language which was more intuitive and suitable for humans than machines. The first such language was FORTRAN (short for FORMula TRANslation), followed quickly by Algol-60 in Europe. An avalanche of other programming languages followed suit. The most popular were Algol-68, PASCAL (1971), C (1969) and COBOL (1967). One big advantage of these languages was that the program written for them was somewhat independent of the machine on which they were run. This freed programmers from worrying about the instructions available in their particular machine, and instead let them focus on designing programs.

To see the difference, let’s return to our adding two number example. You want to add the contents of two memory cells–let us call them x and y–and put
the result in another location, say z. A typical assembly language program may look something like this:

LOAD X
LOAD Y
ADD
SAVE Z

In a high level language this same operation is written simply as

\[ z = x + y; \]

### 4.2 Compilers and interpreters

Of course the question is how is a program written in a high level language such as FORTRAN, PASCAL or C is made to run on a computer that can understand only its own set of arcane instructions in 0’s and 1’s? The answer is “compilers”. A compiler is a program that reads a set of characters representing presumably a meaningful program in a high level language such as C and translates it to the language of the particular machine. Thus, compliers are specific to the programming language and to the machine language they are written for.

A compiler takes a program written in a high level language such as C. It then attempts to translate it into the language of the machine it is running on, say a PC. If the program is correct in syntactic form (that is follows all the grammatical rules of the language) the compiler produces an “executable file”. If you open this file inside an editor chances are you will only see a set of incomprehensible characters. These are actually ASCII encoding of sequences of 0’s and 1’s that encode instructions to CPU. In other words they are the text of the translated program in the machine language. You should realize that compilers are generally not capable of logical (or semantic) errors in a program; they can only detect syntactic or grammatical errors.

![Life cycle of a program from compilation to execution](image)
4.3 **Interpreters and script languages**

Interpreters are another form of translators that are simpler than compilers. Compilers translate a program once and thereafter the translated program can run as many times as you wish. An interpreter generally reads the high level language program one line at a time, translates it, and then runs it, and then goes to the next line and repeats the same process. Thus, in a sense each time you wish to run the program you must also translate it. Interpreters are usually used for very high level languages known as “script” languages. These are usually special purpose languages designed to be useful for a particular or a series of related tasks. Here are some examples:

- **Perl, Python** Very good for text processing, they are used for searching and extracting specific text from documents such as web pages and XML documents. Perl and Python are also used by systems programmer to achieve tasks (such as filtering certain users, being on the look out for intruders, programming the machine to back up its disk on certain times regularly, and other tasks) quickly.

- **SQL** A language specifically designed to handle database operations

- **Visual Basic** A language adopted mainly by Microsoft for writing scripts for Word, Excel and other Office documents,

- **PHP** A language for web servers that processes request from users of web browsers,

- **Javascript** (not to be confused with Java) A language for running programs on the web browsers,

- **Matlab, Mathematica, Maple** Languages for processing mathematical operations in linear algebra, calculus, plotting curves and surfaces, statistics and (in case of Mathematica and Maple) symbolic computation.

4.4 **Operating systems**

Programs in a computer use various resources of the computer. A program may write data on the hard disk or memory, generate pictures to be displayed on the monitor, or send signals to other computers through the Ethernet card. But how does a program determine whether the part of the memory or the hard disk it wants to write to is available. How do programs make sure they do not accidentally overwrite on the partial work of other programs. The answer is a program called “the operating system”. The operating system is a program that manages the resources of a computer. When a program needs memory or wants to write to a particular section of the hard disk, or wants to send a message to the Internet, it asks the operating system to provide the resources for it. The operating system in turn makes sure that there is no conflict among various programs. It makes sure that a program does not tie up the CPU or use excessive amount of memory.
In early times, during the sixties and seventies, mainframe computers such as IBM 360 or minicomputers such DEC PDP-11 were dominant. These computers allowed several programs run simultaneously, and sometimes also allowed several users to be logged in to the system at a time. As a result they required sophisticated operating systems that made sure each user or each program gets a fair amount of the resources of the computer and also policed the behavior of errant programs. A program would not be allowed to tie up resources such as the CPU and the main memory. Neither would it be allowed to write over parts of the memory and hard disk that belongs to other users or programs. The Unix operating system was probably the most famous one of these more sophisticated operating systems. In addition Unix was written in the high level language C\(^1\) and thus it was easier to use in several types of computers. Other operating systems were usually custom-made for a single machine, and they were usually written in the assembly language of that machine.

Later when PC industry came along, since these computers were too small to run several programs, and were assumed to be used by a single person at a time, more primitive operating systems such as DOS, CP/M and original MacOS were designed. These operating systems were not as sophisticated as the ones designed for mainframes. For instance Mac-OS and DOS would crash when a single program made an error. The Macs and belatedly the PC’s, however were responsible for another revolution on operating systems, and that was the use of graphical user interface (GUI) which worked with the mouse and used icons, buttons, pop-up menus, drag and drop, and other technologies. The results were the Macintosh operating system designed by Apple Computer for Macintosh computers introduced in 1984, and the Windows line of operating systems built on DOS by Microsoft in 1990. Unix and other mainframe and minicomputer operating systems also incorporated GUI’s (the X window system for example for Unix was developed at MIT and has been adopted by most computers running Unix). Today, Windows line on PC’s has improved to accommodate multiple programs running at the same time. Thus, you may be editing a document using a word processor, while listening to music CD playing on your CD-ROM drive, and your mail program may be fetching and updating your mailbox behind the scenes, and the anti-virus software may be running in the background checking the hard drive for viruses, all this simultaneously. As for the Mac-OS from version X (as in the Roman numeral for 10) they have switched to a version of Unix called Darwin, that runs underneath and the special Mac-OS user interface called Aqua which runs on top. Thus Mac-OS has resolved its multi-processing problems by switching to Unix.

In addition in the PC world (and to some extend other computers including Macintosh PC’s) a new operating systems has burst into the scene. The story begins in early 1990’s when a Finnish student called Linus Torvalds wrote a simple implementation of the Unix operating system designed to run on Intel 386 processor. The remarkable thing about this operating system was that

\(^1\)C was designed specifically for writing the code for the Unix operating system, but then became very popular for general purpose computing in the 80’s.
Torvalds\textsuperscript{2} published the source code of his implementation on the Internet and invited other programs to contribute and improve it. It turned out that many many programmers, in the hundreds, took the challenge and wrote drivers and other embellishments, ported the well-known Unix programs such as the X-window system to Torvalds operating system. The “Linux” operating system was born. Today this system is available for free to download from the Internet, but can also be purchased commercially from companies such as Red Hat and Suse, which package implementations of the Linux operating system along with massive amounts of software, mostly freely available on the Internet.

\textsuperscript{2}The practice of publishing the source code and freeing access to it was originally developed by Richard Stallman of the MIT AI Lab. Stallman has founded the Free Software Foundation and has developed the GNU Public License or GPL which allows programmers to publish their source code, and allows other programmers to adopt and improve the code, but does not allow them to make the derivative code proprietary and \textit{closed source}. Torvalds published his implementation of Unix under the GPL. Thus all Linux companies are allowed to improve it and contribute to it, but are not allowed to restrict access to the improved product.