Introduction to Computer Programming

Using the Java Programming Language

Introduction to Computer Programming

1. Introduction to Computers and Programming

This course will introduce you to the history of computers and programming languages. You will then learn how to program computers using the Java programming language.

Computers from the Beginning

The first substantial computer was the giant ENIAC machine by John W. Mauchly and J. Presper Eckert at the University of Pennsylvania. ENIAC (Electrical Numerical Integrator and Calculator) used a word of 10 decimal digits instead of binary ones like previous automated calculators/computers. ENIAC was also the first machine to use more than 2,000 vacuum tubes, using nearly 18,000 vacuum tubes. Storage of all those vacuum tubes and the machinery required to keep the cool took up over 167 square meters (1800 square feet) of floor space. Nonetheless, it had punched-card input and output and arithmetically had 1 multiplier, 1 divider-square rooter, and 20 adders employing decimal "ring counters," which served as adders and also as quick-access (0.0002 seconds) read-write register storage.

The executable instructions composing a program were embodied in the separate units of ENIAC, which were plugged together to form a route through the machine for the flow of computations. These connections had to be redone for each different problem, together with presetting function tables and switches. This "wire-your-own" instruction technique was inconvenient, and only with some license could ENIAC be considered programmable; it was, however, efficient in handling the particular programs for which it had been designed. ENIAC is generally acknowledged to be the first successful high-speed electronic digital computer (EDC) and was productively used from 1946 to 1955. A controversy developed in 1971, however, over the patentability of ENIAC's basic digital concepts, the claim being made that another U.S. physicist, John V. Atanasoff, had already used the same ideas in a simpler vacuum-tube device he built in the 1930s while at Iowa State College. In 1973, the court found in favor of the company using Atanasoff claim and Atanasoff received the acclaim he rightly deserved.

In the 1950's two devices would be invented that would improve the computer field and set in motion the beginning of the computer revolution. The first of these two devices was the transistor. Invented in 1947 by William Shockley, John Bardeen, and Walter Brattain of Bell Labs, the transistor was fated to oust the days of vacuum tubes in computers, radios, and other electronics.

The vacuum tube, used up to this time in almost all the computers and calculating machines, had been invented by American physicist Lee De Forest in 1906. The vacuum tube, which is about the size of a human thumb, worked by using large amounts of electricity to heat a filament inside the tube until it was cherry red. One result of heating this filament up was the release of electrons into the tube, which could be controlled by other elements within the tube. De Forest's original device

was a triode, which could control the flow of electrons to a positively charged plate inside the tube. A zero could then be represented by the absence of an electron current to the plate; the presence of a small but detectable current to the plate represented a one.

Vacuum tubes were highly inefficient, required a great deal of space, and needed to be replaced often. Computers of the 1940s and 50s had 18,000 tubes in them and housing all these tubes and cooling the rooms from the heat produced by 18,000 tubes was not cheap. The transistor promised to solve all of these problems and it did so. Transistors, however, had their problems too. The main problem was that transistors, like other electronic components, needed to be soldered together. As a result, the more complex the circuits became, the more complicated and numerous the connections between the individual transistors and the likelihood of faulty wiring increased.

In 1958, this problem too was solved by Jack St. Clair Kilby of Texas Instruments. He manufactured the first integrated circuit or chip. A chip is really a collection of tiny transistors that are connected together when the transistor is manufactured. Thus, the need for soldering together large numbers of transistors was practically nullified; now only connections were needed to other electronic components. In addition to saving space, the speed of the machine was now increased since there was a diminished distance that the electrons had to follow.

The 1960s saw large mainframe computers become much more common in large industries and with the US military and space program. IBM became the unquestioned market leader in selling these large, expensive, error-prone, and very hard to use machines.

A veritable explosion of personal computers occurred in the early 1970s, starting with Steve Jobs and Steve Wozniak exhibiting the first Apple II at the First West Coast Computer Faire in San Francisco. The Apple II boasted built-in BASIC programming language, color graphics, and a 4100 character memory for only $1298. Programs and data could be stored on an everyday audio-cassette recorder. Before the end of the fair, Wozniak and Jobs had secured 300 orders for the Apple II and from there Apple just took off.

Also introduced in 1977 was the TRS-80. This was a home computer manufactured by Tandy Radio Shack. In its second incarnation, the TRS-80 Model II, came complete with a 64,000 character memory and a disk drive to store programs and data on. At this time, only Apple and TRS had machines with disk drives. With the introduction of the disk drive, personal computer applications took off as a floppy disk was a most convenient publishing medium for distribution of software.

IBM, which up to this time had been producing mainframes and minicomputers for medium to large-sized businesses, decided that it had to get into the act and started working on the Acorn, which would later be called the IBM PC. The PC was the first

computer designed for the home market, which would feature modular design so that pieces could easily be added to the architecture. Most of the components, surprisingly, came from outside of IBM, since building it with IBM parts would have cost too much for the home computer market. When it was introduced, the PC came with a 16,000-character memory, keyboard from an IBM electric typewriter, and a connection for tape cassette player for $1265.

By 1984, Apple and IBM had come out with new models. Apple released the first generation Macintosh, which was the first computer to come with a graphical user interface(GUI) and a mouse. The GUI made the machine much more attractive to home computer users because it was easy to use. Sales of the Macintosh soared like nothing ever seen before. IBM was hot on Apple's tail and released the 286-AT, which with applications like Lotus 1-2-3, a spreadsheet, and Microsoft Word, quickly became the favorite of business concerns.

Now people have their own personal graphics workstations and powerful home computers in addition to smart phones that are virtually handheld computers. The average computer a person might have is more powerful by several orders of magnitude than a machine like ENIAC. The computer revolution has been the fastest growing technology in man's history.

Programming Languages

There are computers that work differently, but virtually all computers understand only binary code. That is, a series of 0s and 1s. Computers use this approach because it correlates well with electronic switching: off = 0, and on = 1. Computers are actually dumb.

At the beginning, in order to program a computer you had to speak its language. That is, you had to work with 0s and 1s. Because this is obviously a nightmare for most human beings, computer scientists have tried hard to abstract this process to make it easier for people to make programs.

Machine code

Machine code or machine language is a set of instructions executed directly the computer's CPU.

Each instruction performs a very specific and low-level task. Every program directly executed by a CPU is made up of a series of such instructions.

Numerical machine code (i.e. not assembly code) may be regarded as the lowest-level representation of a compiled and/or assembled computer program or as a primitive and hardware-dependent programming language.

While it is possible to write programs directly in numerical machine code, it is tedious and error prone to manage individual bits (1's or 0's) and calculate numerical addresses and constants manually. It is therefore rarely done today, except for situations that require extreme optimization or debugging.

Assembly language

Assembly language is not machine code, but almost. It's directly correlated to the underlying architecture, so there is no abstraction.

To turn assembly language into machine code, an assembler is used. The assembler is the first software tool ever invented.

Assemblers were available in the 50s, since they don't require much code analysis: most of the times they simply take the instructions and translate them directly into their corresponding executable value. As a programmer, you have to think like the underlying machine or architecture.

Assembly language is still used on electronic equipment with limited resources, and before high-level languages and libraries get loaded (ex., hardware firmware).

In the 70s and 80s assembly language was fairly common. For instance, all game console used assembly language, as the amount of memory available could be as

little as a few kilobytes. Lots of code in the original 1984 Macintosh was written in assembly language to save resources.

Fortran

By this time, a lot of research had been done in high-level programming languages (meaning anything more abstract than assembly).

Fortran was originally developed by IBM in the 50s.

At the time, languages were created to be specialized to solve a specific set of problems: Fortran was intended for scientific processing, and it became the dominant language within this field right away, and enjoyed an enormous amount of success during the following 50 years.

It is still one of the most popular languages in the area of high-performance computing and is the language used for programs that benchmark and rank the world's fastest supercomputers.

Fortran established the convention of using the asterisk for multiplication, which is still used today in virtually all languages.

COBOL

COBOL (COmmon Business-Oriented Language) was designed for business use. It was an attempt to make programming languages more similar to English, so that both programmers and management could read it.

Among its designers was Grace Hopper (the lady who discovered "the bug") and who had invented the English-like data processing language FLOW-MATIC, the perfect candidate to help create a common business language that looked similar to English.

BASIC

BASIC (Beginner's All-purpose Symbolic Instruction Code) was designed in 1964 by John G. Kemeny and Thomas E. Kurtz at Dartmouth College in New Hampshire.

BASIC was developed specifically for timesharing. It was a very stripped-down version of Fortran, to make it easier to program.

It came with a clever way of editing programs using line numbers, for both programming and other operations like GOTO line jumping.

Versions of BASIC were extremely common on microcomputers in the mid-70s and 80s, which usually shipped with BASIC directly in the machine's firmware, allowing small business owners, professionals, hobbyists, and consultants to develop custom software on computers they could afford.

BASIC spawned different languages, including Visual BASIC, the most popular programming language in the world for a long time, which Microsoft created from Microsoft BASIC.

ALGOL 60

ALGOL 60 (ALGOrithmic Language 1960) is a committee-driven, very good and influential language that came out in 1960.

It never got popular but it introduced a lot of important concepts, including getting rid of GOTO.

Jumping around from line to line in languages like BASIC made it hard to follow the flow of the program, and made writing programs error-prone.

ALGOL 60 introduced structure programming and blocks: it used BEGIN and END (because curly braces weren't available), and it's thanks to ALGOL 60 if we have blocks now instead of GOTO.

ALGOL also wanted to be less specialized, good for both scientific and business processing.

Pascal

Pascal was designed in 1968–1969 and published in 1970 by Niklaus Wirth and was inspired by ALGOL.

It was extremely popular, and although originally designed as a teaching tool, lots of people used it for general programming for a long time.

However, it wasn't modular enough and had some design challenges that made programming hard.

B

B was developed at Bell Labs in 1969. It was inspired by Fortran and BCPL.

B was essentially the BCPL system stripped of any component Thompson felt he could do without in order to make it fit within the memory capacity of the minicomputers of the time.

http://en.wikipedia.org/wiki/B_(programming_language)

B introducted the += operator (although spelled =+), and the incre/decrement operators (++ and –)

C

C was born from taking B and adding some good ideas from Pascal. It was developed by Dennis Ritchie between 1969 and 1973 at Bell Labs (again).

C is probably the most important language out there. It was unbelievably successful (and it still is).

In addition, many languages are based on C, including:

C++ (1976)
Objective-C (1986)
Perl (1988)
Java (1991)
Python (1991)
JavaScript (1995)
PHP (1995)
C# (1999)
Go (2007)
…and many others

Simula

Norwegian simulation languages Simula I and Simula 67 are—syntactically—a fairly faithful superset of ALGOL 60.

Its contribution was enormous. Simula took ALGOL 60 and added objects to it, and is therefore considered the first object-oriented programming language.

Simula 67 (released in 1967) introduced objects, classes, inheritance and subclasses, as well as virtual methods, coroutines, and discrete event simulation.

Smalltalk

Simula had a huge influence on Alan Kay—who went to Xerox PARC and in 1972 started working on Smalltalk.

Smalltalk was originally designed for kids. It was thoroughly tested in the real world and revised many times. It was finally published 8 years after its first release, after several generations.

Smalltalk was a great language, and the first truly modern object-oriented programming language.

Because of its hard-to-grasp syntax, it never became popular. However, it influenced virtually all modern programming languages. Objective-C, C++, Java, C#, Eiffel, and Ruby are basically C combined with Smalltalk.

Self

Another language influenced by Smalltalk—developed at Xerox PARC as well—was Self.

Self was designed for performance. It took Smalltalk, and removed classes to make it faster.

Instead of using classes, is used prototypes: Self allowed objects to inherit directly from other objects, without passing by classes.

Scheme

Scheme is based on the actor model. The actor model originated in 1973, and was a radical concept when it was conceived. It's implemented in LISP, an artificial intelligence language created at MIT in 1958.

It includes tail recursion, lexical closures, and other brilliant stuff.

Java

Java was developed by Sun Microsystems in 1996. Details of how Java was developed and why it became so popular is detailed later in this chapter.

JavaScript

JavaScript has become an extremely important language. It's ubiquitous in web-capable devices, a pillar of the Web Platform and the base for most web and mobile applications.

JavaScript is based on Java, Scheme, and Self.

It was originally developed at Netscape, which wanted to implement something similar to Apple's HyperCard—an application program and programming tool for Apple Macintosh and Apple IIGS that made it easy to build apps—and add it to the browser.

Its author Brendan Eich wanted to base it on Scheme, but Netscape's management thought that people wouldn't like its syntax and asked him to make it more similar to Java.

JavaScript borrows Java's syntax (just because it had to), Scheme's function model, and the prototypical nature of Self.

Netscape implemented and released JavaScript in 2 weeks!!!

Operating Systems

An operating system is the most important software that runs on a computer. It manages the computer's memory and processes, as well as all of its software and hardware. It also allows you to communicate with the computer without knowing how to speak the computer's language. Without an operating system, a computer is useless.

Your computer's operating system (OS) manages all of the software and hardware on the computer. Most of the time, there are several different computer programs running at the same time, and they all need to access your computer's central processing unit (CPU), memory, and storage. The operating system coordinates all of this to make sure each program gets what it needs.

The 1940's - First Generations

The earliest electronic digital computers had no operating systems. Machines of the time were so primitive that programs were often entered one bit at time on rows of mechanical switches (plug boards). Programming languages were unknown (not even assembly languages). Operating systems were unheard of .

The 1950's - Second Generation

By the early 1950's, the routine had improved somewhat with the introduction of punch cards. The General Motors Research Laboratories implemented the first operating systems in early 1950's for their IBM 701. The system of the 50's generally ran one job at a time. These were called single-stream batch processing systems because programs and data were submitted in groups or batches.

The 1960's - Third Generation

The systems of the 1960's were also batch processing systems, but they were able to take better advantage of the computer's resources by running several jobs at once. So operating systems designers developed the concept of multiprogramming in which several jobs are in main memory at once; a processor is switched from job to job as needed to keep several jobs advancing while keeping the peripheral devices in use.

For example, on the system with no multiprogramming, when the current job paused to wait for other I/O operation to complete, the CPU simply sat idle until the I/O finished. The solution for this problem that evolved was to partition memory into several pieces, with a different job in each partition. While one job was waiting for I/O to complete, another job could be using the CPU.

Another major feature in third-generation operating system was the technique called spooling (simultaneous peripheral operations on line). In spooling, a high-speed device like a disk interposed between a running program and a low-speed device involved with the program in input/output. Instead of writing directly to a printer, for example, outputs are written to the disk. Programs can run to

completion faster, and other programs can be initiated sooner when the printer becomes available, the outputs may be printed.

Note that spooling technique is much like thread being spun to a spool so that it may be later be unwound as needed.

Another feature present in this generation was time-sharing technique, a variant of multiprogramming technique, in which each user has an on-line (i.e., directly connected) terminal. Because the user is present and interacting with the computer, the computer system must respond quickly to user requests, otherwise user productivity could suffer. Timesharing systems were developed to multiprogram large number of simultaneous interactive users.

Fourth Generation

With the development of LSI (Large Scale Integration) circuits, chips containing thousands of transistors on a square centimeter of silicon, the age of the personal computer dawned. In terms of architecture, personal computers (initially called microcomputers) were not all that different from minicomputers of the PDP-11 class, but in terms of price they certainly were different. Where the minicomputer made it possible for a department in a company or university to have its own computer, the microprocessor chip made it possible for a single individual to have his or her own personal computer.

In 1974, when Intel came out with the 8080, the first general-purpose 8-bit CPU, it wanted an operating system for the 8080, in part to be able to test it. Intel asked one of its consultants, Gary Kildall, to write one. Kildall and a friend first built a controller for the newly-released Shugart Associates 8-inch floppy disk and hooked the floppy disk up to the 8080, thus producing the first microcomputer with a disk. Kildall then wrote a disk-based operating system called CP/M (Control Program for Microcomputers) for it. Since Intel did not think that disk-based microcomputers had much of a future, when Kildall asked for the rights to CP/M, Intel granted his request. Kildall then formed a company, Digital Research, to further develop and sell CP/M.

In 1977, Digital Research rewrote CP/M to make it suitable for running on the many microcomputers using the 8080, Zilog Z80, and other CPU chips. Many application programs were written to run on CP/M, allowing it to completely dominate the world of microcomputing for about 5 years.

In the early 1980s, IBM designed the IBM PC and looked around for software to run on it. People from IBM contacted Bill Gates to license his BASIC interpreter. They also asked him if he knew of an operating system to run on the PC. Gates suggested that IBM contact Digital Research, then the world's dominant operating systems company. Making what was surely the worst business decision in recorded history, Kildall refused to meet with IBM, sending a subordinate instead. To make matters worse, his lawyer even refused to sign IBM's nondisclosure agreement covering the

not-yet-announced PC. Consequently, IBM went back to Gates asking if he could provide them with an operating system.

When IBM came back, Gates realized that a local computer manufacturer, Seattle Computer Products, had a suitable operating system, DOS (Disk Operating System). He approached them and asked to buy it (allegedly for $50,000), which they readily accepted. Gates then offered IBM a DOS/BASIC package, which IBM accepted. IBM wanted certain modifications, so Gates hired the person who wrote DOS, Tim Paterson, as an employee of Gates' fledgling company, Microsoft, to make them. The revised system was renamed MS-DOS (MicroSoft Disk Operating System) and quickly came to dominate the IBM PC market. A key factor here was Gates' (in retrospect, extremely wise) decision to sell MS-DOS to computer companies for bundling with their hardware, compared to Kildall's attempt to sell CP/M to end users one at a time (at least initially).

By the time the IBM PC/AT came out in 1983 with the Intel 80286 CPU, MS-DOS was firmly entrenched and CP/M was on its last legs. MS-DOS was later widely used on the 80386 and 80486. Although the initial version of MS-DOS was fairly primitive, subsequent versions included more advanced features, including many taken from UNIX. (Microsoft was well aware of UNIX, even selling a microcomputer version of it called XENIX during the company's early years.) CP/M, MS-DOS, and other operating systems for early microcomputers were all based on users typing in commands from the keyboard. That eventually changed due to research done by Doug Engelbart at Stanford Research Institute in the 1960s. Engelbart invented the GUI (Graphical User Interface), pronounced ''gooey,'' complete with windows, icons, menus, and mouse. These ideas were adopted by researchers at Xerox PARC and incorporated into machines they built.

One day, Steve Jobs, who co-invented the Apple computer in his garage, visited PARC, saw a GUI, and instantly realized its potential value, something Xerox management famously did not (Smith and Alexander, 1988). Jobs then embarked on building an Apple with a GUI. This project led to the Lisa, which was too expensive and failed commercially. Jobs' second attempt, the Apple Macintosh, was a huge success, not only because it was much cheaper than the Lisa, but also because it was user friendly, meaning that it was intended for users who not only knew nothing about computers but furthermore had absolutely no intention whatsoever of learning. This led up to the release of Mac OS X in 2012, followed by OS X in 2016. Version 10 of the Mac OS is when it became based on UNIX. Prior to that it was based on Mac OS.

When Microsoft decided to build a successor to MS-DOS, it was strongly influenced by the success of the Macintosh. It produced a GUI-based system called Windows, which originally ran on top of MS-DOS (i.e., it was more like a shell than a true operating system). For about 10 years, from 1985 to 1995, Windows was just a graphical environment on top of MS-DOS. However, starting in 1995 a freestanding version of Windows, Windows 95, was released that incorporated many operating

system features into it, using the underlying MS-DOS system only for booting and running old MS-DOS programs. In 1998, a slightly modified version of this system, called Windows 98 was released. Nevertheless, both Windows 95 and Windows 98 still contain a large amount of 16-bit Intel assembly language.

Another Microsoft operating system is Windows NT (NT stands for New Technology), which is compatible with Windows 95 at a certain level, but a complete rewrite from scratch internally. It is a full 32-bit system. The lead designer for Windows NT was David Cutler, who was also one of the designers of the VAX VMS operating system, so some ideas from VMS are present in NT. Microsoft expected that the first version of NT would kill off MS-DOS and all other versions of Windows since it was a vastly superior system, but it fizzled. Only with Windows NT 4.0 did it finally catch on in a big way, especially on corporate networks. Version 5 of Windows NT was renamed Windows 2000 in early 1999. It was intended to be the successor to both Windows 98 and Windows NT 4.0. That did not quite work out either, so Microsoft came out with yet another version of Windows 98 called Windows Me (Millennium edition). This was followed by Windows XP, Windows Vista, Windows 7, Windows 8, and Windows 10 in 2015.

The other major contender in the personal computer world is UNIX (and its various derivatives). UNIX is strongest on workstations and other high-end computers, such as network servers. It is especially popular on machines powered by high-performance RISC chips. On Pentium-based computers, Linux is becoming a popular alternative to Windows for students and increasingly many corporate users. (As an aside, throughout this book we will use the term ''Pentium'' to mean the Pentium I, II, III, and 4.) Although many UNIX users, especially experienced programmers, prefer a command-based interface to a GUI, nearly all UNIX systems support a windowing system called the X Windows system produced at M.I.T. This system handles the basic window management, allowing users to create, delete, move, and resize windows using a mouse. Often a complete GUI, such as Motif, is available to run on top of the X Windows system giving UNIX a look and feel something like the Macintosh or Microsoft Windows, for those UNIX users who want such a thing.

The History of Java

Java, having been developed in 1991, is a relatively new programming language. At that time, James Gosling from Sun Microsystems and his team began designing the first version of Java aimed at programming home appliances which are controlled by a wide variety of computer processors.

Gosling's new language needed to be accessible by a variety of computer processors. In 1994, he realized that such a language would be ideal for use with web browsers and Java's connection to the internet began. In 1995, Netscape Incorporated released its latest version of the Netscape browser which was capable of running Java programs.

Why is it called Java**?** It is customary for the creator of a programming language to name the language anything he/she chooses. The original name of this language was Oak, until it was discovered that a programming language already existed that was named Oak. As the story goes, after many hours of trying to come up with a new name, the development team went out for coffee and the name Java was born.

While Java is viewed as a programming language to design applications for the Internet, it is in reality a general all-purpose language that can be used independent of the Internet.

Useful Acronyms & Vocabulary

You will need to be familiar with the following acronyms associated with computer programming. You will see references to these acronyms throughout this course.

ALU - Arithmetic Logic Unit - the part of the processor which performs =, +, -, <, >, ,>, etc. operations and comparisons.

CRT - Cathode Ray Tube - an older electrical device found in monitors for displaying images by exciting phosphor dots with a scanned electron beam. CRT is often used to mean monitor.

CPU - Central Processing Unit - the heart (brains) of the computer system. It is comprised of the control unit, the arithmetic logic unit, and temporary storage (RAM).

DOS - Disk Operating System - was the first widely-installed operating system for personal computers developed for IBM by Microsoft. It is software used in most computer systems to manage storage devices as well as data of any kind, including files. It is referred to as a "disk" operating system because the storage devices are made of rotating platters.

IDE - Integrated Development Environment - a system where you can control the editing and compiling from one program.

LAN - Local Area Network - a set of computers connected in order to share programs and storage space. "Local" implies that the network is contained within a relatively small space, such as a classroom, an office, one section of the building, or one building.

OOP - Object Oriented Programming - the use of small, reusable components to construct large software systems.

LCD – Liquid Crystal Display – an electrical device used as newer monitors for displaying images. It consists of a layer of liquid crystal sandwiched between two transparent electrodes. The display is changed by using electrical pulses.

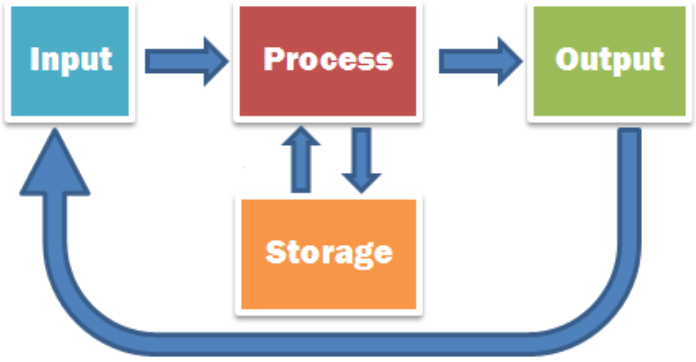
OS - Operating System - the program that manages all the other programs in a computer. Some popular operating systems include MS-DOS, Windows 98/2000/NT/XP/Vista/7/8/10, MacOS, Unix, and Linux.

RAM - Random Access Memory - temporary memory lost when the computer is turned off.

ROM - Read Only Memory - hardwired memory which cannot be changed. Contains the system directions.

Main Functions of a Computer System

There are four main equipment functions of a computer system: Input, Processing, Storage and Output.



Input: the transferring of information into a computer system - for our purposes this will be accomplished when you, the programmer, type source code at the keyboard or open a previously typed program.

Processing: the manipulation and control of information within the computer system. Such manipulations are handled by the Control Unit, the Arithmetic Logic Unit and Temporary Storage.

Storage: the means by which information can be "permanently" saved (until such time as you wish to delete it). This usually occurs on a hard drive, a diskette or a CD.

Output: the displaying of information - for our purposes this will be accomplished when your programs display information on the monitor.

Number Bases

One of the skills which is useful to programmers is an understanding of the relationship between number bases. When you understand how numbers are represented in base two (**Binary**), base eight (**Octal**), and base sixteen (**Hexadecimal**), you will better understand references which will be made later in this course.

The number system that we all know and love is the **Decimal** number system, base 10. In our number system we utilize the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. All numbers are formed by combining these digits.

Other number bases work in the same manner only with different digits. The **Binary** number system, base 2, uses only the digits 0 and 1. The **Octal** number system, base 8, uses the digits 0, 1, 2, 3, 4, 5, 6, and 7. The **Hexadecimal** number system, base 16, uses the digits and letters 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

Notice how, in each of these number systems, the number of digits contained within the system is the name of the system. Base 2 has 2 digits, Base 10 has 10 digits, and so on. Just be careful to remember that each system starts counting with 0.

Each system uses its digits to form all of the numbers contained within the system. In the number system that we know and love, base 10, we are familiar with the terms ones, tens, hundreds, thousands, etc. used to describe the position a digit holds within a number.

$10^0 = 1$ (ones)
$10^1 = 10$ (tens)
$10^2 = 100$ (hundreds)
$10^3 = 1000$ (thousands)

These terms are actually words that describe **powers of 10.** The positions within a number in base 10 are formed by systematically raising the number 10 to various powers.

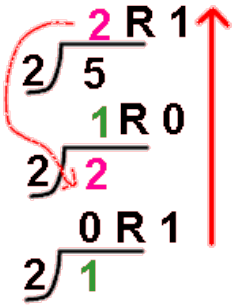
The same process is used to create numbers in other number systems as well. The only difference is that instead of raising 10 to a power, the new base number is used. For example, when working in base 8, you are working with powers of 8 (8^0, 8^1, 8^2, 8^3, ...). Base 2 works with powers of 2, base 16 works with powers of 16, and so on.

In our study of computer programming, most of our references will be made to the binary number system. The simplicity of only dealing with two digits (0 or 1) is very appealing to a computer system. The 0 and 1 can represent current being turned OFF or ON. Notice how switches on computer hardware often display a 0 or a 1 to represent off or on. A "bit" is the smallest unit of data in a computer. It contains a binary value of 0 or 1.

Converting From Base 10

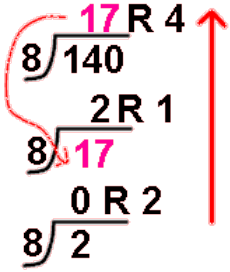
Let's see if we can convert some numbers from one base to another. There are many ways to work with number base conversions. We will be discussing only one method. Please feel free to use any conversion system with which you are comfortable.

Example 1: Convert 5_{10} (read 5 base 10) into base 2.



The Process:
Divide the "desired" base (in this case base 2) INTO the number you are trying to convert.
Write the quotient (the answer) with a remainder like you did in elementary school.
Repeat this division process using the whole number from the previous quotient (the number in front of the remainder).
Continue repeating this division until the number in front of the remainder is only zero.
The answer is the remainders read from the bottom up.
5_{10} = 101_2 (a binary conversion)

Example 2: Convert 140_{10} to base 8.



The process is the same as in example 1. The answer is:
$140_{10} = 214_8$
(an octal conversion)

Converting To Base 10

The computer is representing a number from the keyboard by a pattern of ON and OFF signals represented by 1011. What number was typed at the keyboard to produce this pattern?

Let's see how we can quickly convert numbers back to base 10 so that they will have more meaning to us humans.

Example 1: Convert 235_8 into base 10.

The Process:
Above each of the digits in your number, list the power of the base that the digit represents. See the example on the left. It is now a simple process of multiplication and addition to determine your base 10 number. In this example you have

$5 \times 8^0 = 5$
$3 \times 8^1 = 24$
$2 \times 8^2 = 128$

Now simply add these values together.
$5 + 24 + 128 = 157$
Answer: $235_8 = 157_{10}$

Example 2: Convert 1011_2 to base 10.
The process is the same as in example 1.

$1 \times 2^0 = 1$
$1 \times 2^1 = 2$
$0 \times 2^2 = 0$
$1 \times 2^3 = 8$

Now simply add these values together.
$1 + 2 + 0 + 8 = 11$
Answer: $1011_2 = 11_{10}$

Example 3: Convert $1C4_{16}$ to base 10.

$4 \times 16^0 = 4$
$C \times 16^1 = 12 \times 16^1 = 192$
$1 \times 16^2 = 256$

$4 + 192 + 256 = 452$
Answer: $1C4_{16} = 452_{10}$

Converting Via Base 10

This method will allow you to convert from any base to any base! We just use base 10 as the intermediary step.

When converting from any number base to any other number base, and neither of the bases is base 10, you must go via (through) base 10.

Example: Convert 24_8 into base 2.
First Convert 24_8 into base 10.
$4 \times 8^0 = 4$
$2 \times 8^1 = 16$
$4 + 16 = 20$

So, $24_8 = 20_{10}$

And now convert that answer into base 2.

$20_{10} = 10100_2$

Therefore: $24_8 = 10100_2$