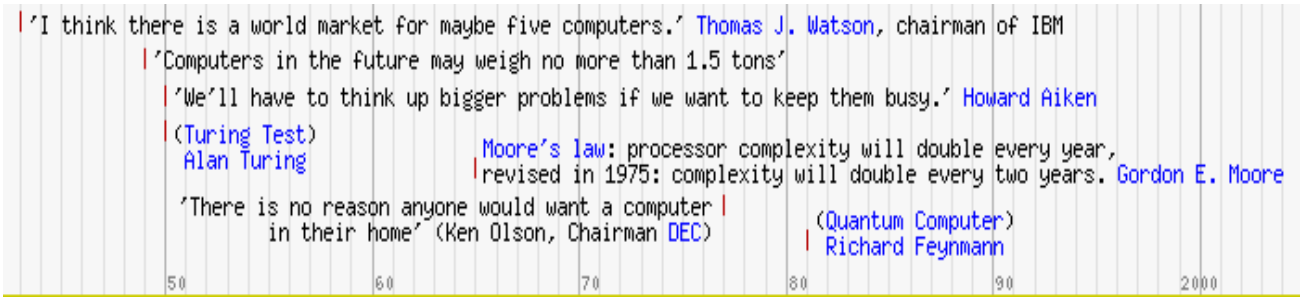
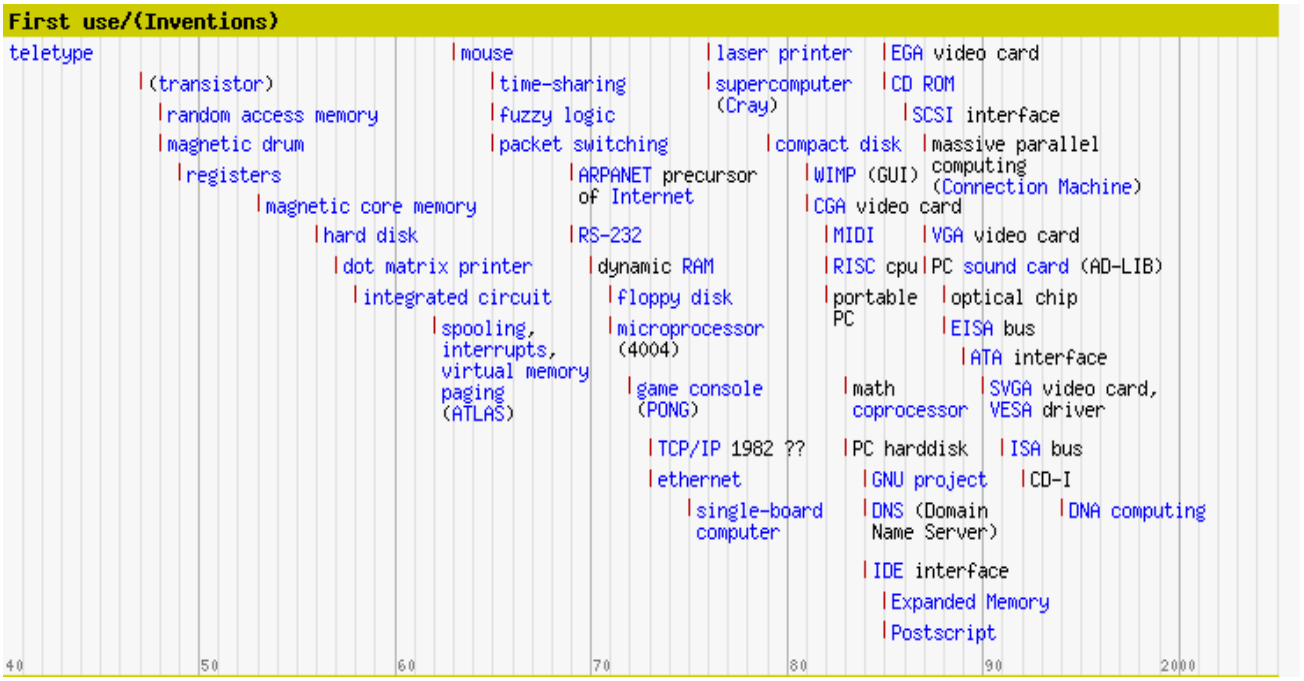
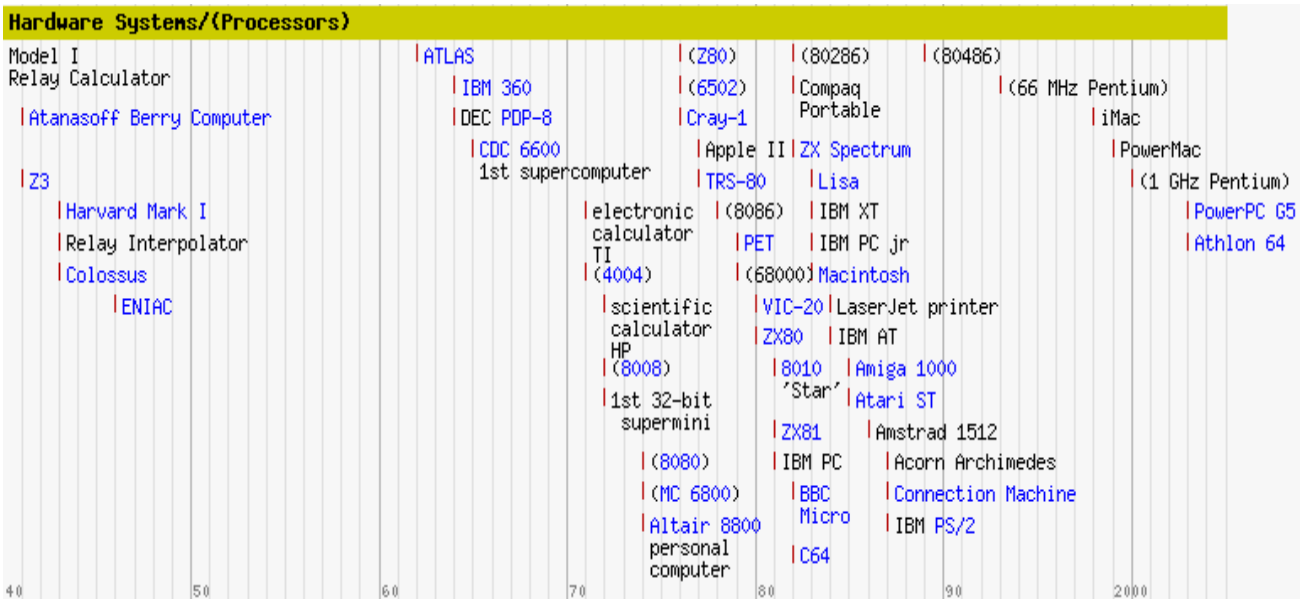
# History of Computing

## 1. Predictions / Concepts




## 2. History of Hardware



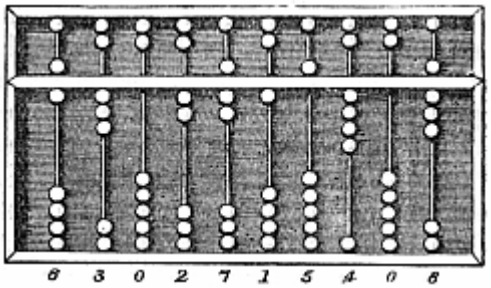
Computing hardware has been an essential component of the process of calculation and data storage since it became necessary for data to be processed and shared. The first known computing hardware was a recording device; the Phoenicians stored clay shapes representing items, such as livestock and grains, in containers. These were used by merchants, accountants, and government officials of the time.

Devices to aid computation have evolved over time. Even today, an experienced abacus user using a device designed thousands of years ago can often complete basic calculations more quickly than an unskilled person using an electronic calculator — though for more sophisticated calculations, computers out-perform even the most skilled human.

This article presents the major developments in the history of computing hardware and attempts to put them in context. For a detailed timeline of events, see the computing timeline article. The history of computing article is a related overview and treats methods intended for pen and paper, with or without the aid of tables.

## 2.1. Earliest devices for facilitating human calculation

Humanity has used devices to aid in computation for millennia. A more arithmetic-oriented machine the abacus is one of the earliest machines of this type was the Roman abacus.



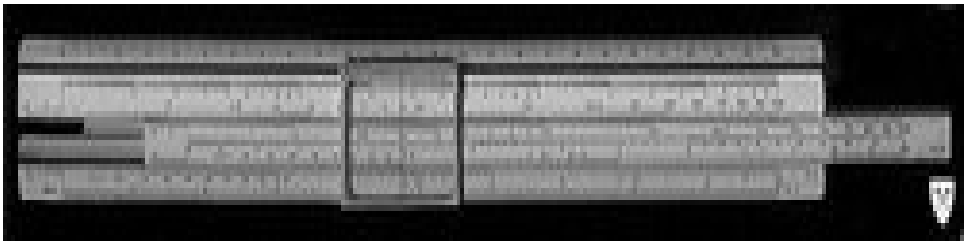
**Babylonians and others, frustrated with counting on their fingers, invented the Abacus.**

## 2.2. First mechanical calculators

**Gears are at the heart of mechanical devices like the Curta calculator.**

In 1623 Wilhelm Schickard built the first mechanical calculator and thus became the father of the computing era. Since his machine used techniques such as cogs and gears first developed for clocks, it was also called a 'calculating clock'. It was put to practical use by his friend Johannes Kepler, who revolutionized astronomy.

John Napier noted that multiplication and division of numbers can be performed by addition and subtraction, respectively, of logarithms of those numbers. Since these real numbers can be represented as distances or intervals on a line, the slide rule allowed multiplication and division operations to be carried significantly faster than was previously possible. Slide rules were used by generations of engineers and other mathematically inclined professional workers, until the invention of the pocket calculator.



**The slide rule, a basic mechanical calculator, facilitates multiplication and division.**

## 2.3. Punched card technology 1801–

In 1801, Joseph-Marie Jacquard developed a loom in which the pattern being woven was controlled by punched cards. The series of cards could be changed without changing the mechanical design of the

loom. This was a landmark point in programmability. Herman Hollerith invented a tabulating machine using punch cards in the 1880s.

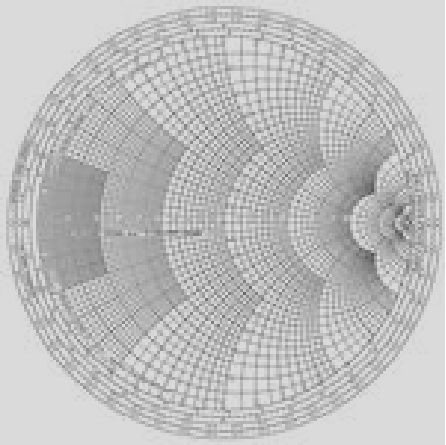
## 2.4. First designs of programmable machines 1835–1900s

The defining feature of a "universal computer" is programmability, which allows the computer to emulate any other calculating machine by changing a stored sequence of instructions. In 1835 Charles Babbage described his analytical engine. It was the plan of a general-purpose programmable computer, employing punch cards for input and a steam engine for power. One crucial invention was to use gears for the function served by the beads of an abacus. In a real sense, computers all contain automatic abaci (technically called the ALU or floating-point unit).

## 2.5. More limited types of mechanical gear computing 1800s–1900s

By the 1900s earlier mechanical calculators, cash registers, accounting machines, and so on were redesigned to use electric motors, with gear position as the representation for the state of a variable. People were computers, as a job title, and used calculators to evaluate expressions. During the Manhattan project, future Nobel laureate Richard Feynman was the supervisor of the roomful of human computers, many of them women mathematicians, who understood the differential equations which were being solved for the war effort. Even the renowned Stanislaw Marcin Ulam was pressed into service to translate the mathematics into computable approximations for the hydrogen bomb, after the war

## 2.6. Analog computers, pre-1940



Before World War II, mechanical and electrical analog computers were considered the 'state of the art', and many thought they were the future of computing. Analog computers use continuously varying amounts of physical quantities, such as voltages or currents, or the rotational speed of shafts, to represent the quantities being processed. An ingenious example of such a machine was the Water integrator built in 1936.

**Nomograms, like this Smith chart serve as analog computers for specific classes of problems.**

## 2.7. First generation of electrical digital computers 1940s

The era of modern computing began with a flurry of development before and during World War II, as electronic circuits, relays, capacitors and vacuum tubes replaced mechanical equivalents and digital calculations replaced analog calculations. The computers designed and constructed then have sometimes been called 'first generation' computers. First generation computers were usually built by hand using circuits containing relays or vacuum valves (tubes), and often used punched cards or punched paper tape for input and as the main (non-volatile) storage medium. Temporary, or working storage, was provided by acoustic delay lines (which use the propagation time of sound in a medium such as wire to store data) or by Williams tubes (which use the ability of a television picture tube to store and retrieve data).

**Electronic computers became possible with the advent of the vacuum tube.**

By 1954, magnetic core memory was rapidly displacing most other forms of temporary storage, and dominated the field through the mid-1970s.

In this era, a number of different machines were produced with steadily advancing capabilities. At the beginning of this period, nothing remotely resembling a modern computer existed, except in the long-lost plans of Charles Babbage and the mathematical musings of Alan Turing and others. At the end of the era, devices like the EDSAC had been built, and are universally agreed to be universal digital computers. Defining a single point in the series as the "first computer" misses many subtleties.

Alan Turing's 1936 paper has proved enormously influential in computing and computer science, in two ways. Its main purpose was an elegant proof that there were problems (namely the halting problem) that could not be solved by a mechanical process (a computer). In doing so, however, Turing provided a definition of what a universal computer is: a construct called the Turing machine, a purely theoretical device invented to formalize the notion of algorithm execution, replacing Kurt Gödel's more cumbersome universal language based on arithmetics. Modern computers are Turing-complete (i.e., equivalent algorithm execution capability to a universal Turing machine), except for their finite memory. This limited type of Turing completeness is sometimes viewed as a threshold capability separating general-purpose computers from their special-purpose predecessors.

However, as will be seen, *theoretical* Turing-completeness is a long way from a practical universal computing device. To be a practical general-purpose computer, there must be some convenient way to input new programs into the computer, such as punched tape.

For full versatility, the Von Neumann architecture uses the same memory both to store programs and data; virtually all contemporary computers use this architecture (or some variant). Finally, while it is theoretically possible to implement a full computer entirely mechanically (as Babbage's design showed), electronics made possible the speed and later the miniaturization that characterises modern computers.

There were three, parallel streams of computer development in the World War II era, and two were either largely ignored or were deliberately kept secret. The first was the German work of Konrad Zuse. The second was the secret development of the Colossus computer in the UK. Neither of these had much influence on the various computing projects in the United States. After the war British and American computing researchers cooperated on some of the most important steps towards a practical computing device.

## 2.8. American developments

In 1937, Claude Shannon produced his master's thesis at MIT that implemented Boolean algebra using electronic relays and switches for the first time in history. Entitled *A Symbolic Analysis of Relay and Switching Circuits*, Shannon's thesis essentially founded practical digital circuit design.

In November of 1937, George Stibitz, then working at Bell Labs, completed a relay-based computer he dubbed the "Model K" (for "kitchen", where he had assembled it), which calculated using binary addition. Bell Labs thus authorized a full research program in late 1938 with Stibitz at the helm. Their Complex Number Calculator, completed January 8, 1940, was able to calculate complex numbers. In a demonstration to the American Mathematical Society conference at Dartmouth College on September 11, 1940,

Stibbitz was able to send the Complex Number Calculator remote commands over telephone lines by a teletype.

It was the first computing machine ever used remotely over a phone line. Some participants of the conference who witnessed the demonstration were John Von Neumann, John Mauchly, and Norbert Wiener, who wrote about it in his memoirs.

In 1938 John Vincent Atanasoff and Clifford E. Berry of Iowa State University developed the Atanasoff Berry Computer (ABC), a special purpose computer for solving systems of linear equations, and which employed capacitors fixed in a mechanically rotating drum, for memory. The ABC machine was not programmable, though it was a computer in the modern sense in several other respects.

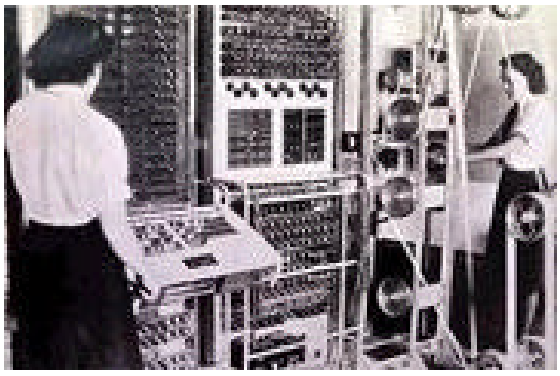
In 1939, development began at IBM's Endicott laboratories on the Harvard Mark I. Known officially as the Automatic Sequence Controlled Calculator, the Mark I was a general purpose electro-mechanical computer built with IBM financing and with assistance from some IBM personnel under the direction of Harvard mathematician Howard Aiken. Its design was influenced by the Analytical Engine.

It was a decimal machine which used storage wheels and rotary switches in addition to electromagnetic relays. It was programmable by punched paper tape, and contained several calculators working in parallel. Later models contained several paper tape readers and the machine could switch between readers based on a condition. Nevertheless, this does not quite make the machine Turing-complete. The Mark I was moved to Harvard University to begin operation in May 1944.

**ENIAC performed ballistics trajectory calculations with 160kW of power.**

The US-built ENIAC (Electronic Numerical Integrator and Computer), often called the first electronic general-purpose computer, publicly validated the use of electronics for large-scale computing. This was crucial for the development of modern computing, initially because of the enormous speed advantage, but ultimately because of the potential for miniaturization. Built under the direction of John Mauchly and J. Presper Eckert, it was 1,000 times faster than its contemporaries. ENIAC's development and construction lasted from 1941 to full operation at the end of 1945

## 2.9. Colossus



**Colossus was used to break German ciphers during World War II.**

During World War II, the British at Bletchley Park achieved a number of successes at breaking encrypted German military communications. The German encryption machine, Enigma, was attacked with the help of electro-mechanical machines called *bombes*. The bombe, designed by Alan Turing and Gordon Welchman, ruled out possible Enigma settings by performing chains of logical deductions implemented electrically. Most possibilities led to a contradiction, and the few remaining could be tested by hand.

The Germans also developed a series of teleprinter encryption systems, quite different from Enigma. The Lorenz SZ 40/42 machine was used for high-level Army communications, termed "Tunny" by the British. The first intercepts of Lorenz messages begain in 1941. As part of an attack on Tunny, Professor Max Newman and his colleagues helped specify the Colossus. The Mk I Colossus was built in 11 months by Tommy Flowers and his colleagues at the Post Office Research Station at Dollis Hill in London and then shipped to Bletchley Park.

Colossus was the first totally *electronic* computing device. The Colossus used a large number of valuves (vacuum tubes). It had paper-tape input and was capable of being configured to perform a variety of boolean logical operations on its data, but it was not Turing-complete. Nine Mk II Colossi were built (The Mk I was converted to a Mk II making ten machines in total). Details of their existence, design, and use were kept secret well into the 1970s.

## 2.10. Konrad Zuse's Z-Series

Working in isolation in Nazi Germany, Konrad Zuse started construction in 1936 of his first Z-series calculators featuring memory and (initially limited) programmability. Zuse's purely mechanical, but already binary Z1, finished in 1938, never worked reliably due to problems with the precision of parts.

## 2.11. Postwar von Neumann machines -- the first generation

The first working von Neumann machine was the Manchester "Baby" or Small-Scale Experimental Machine, built at the University of Manchester in 1948; it was followed in 1949 by the Manchester Mark I computer which functioned as a complete system using the Williams tube for memory, and also introduced index registers. The other contender for the title "first digital stored program computer" was EDSAC, designed and constructed at the University of Cambridge. Operational less than one year after the Manchester "Baby", it was capable of tackling real problems.

EDSAC was actually inspired by plans for EDVAC (Electronic Discrete Variable Automatic Computer), the successor of ENIAC; these plans were already in place by the time the ENIAC was successfully operational. Unlike the ENIAC, which used parallel processing, EDVAC used a single processing unit. This design was simpler and was the first to be implemented in each succeeding wave of miniaturization, and increased reliability. Some view Manchester Mark I / EDSAC / EDVAC as the "Eves" from which nearly all current computers derive their architecture.

The first universal programmable computer in continental Europe was created by a team of scientists under direction of Sergei Alekseyevich Lebedev from Kiev Institute of Electrotechnology, Soviet Union (now Ukraine). The computer MESM (*? ? ? ?* , *Small Electronic Calculating Machine*) became operational in 1950. It had about 6,000 vacuum tubes and consumed 25 kW of power. It could perform approximately 3,000 operations per second. Another early machine was CSIRAC, an Australian design that ran its first test program in 1949.

**UNIVAC I, above, the first commercial electronic computer, achieved 1900 operations per second in a smaller and more efficient package than ENIAC.**

In June 1951, the UNIVAC I (Universal Automatic Computer) was delivered to the U.S. Census Bureau. Although manufactured by Remington Rand, the machine often was mistakenly referred to as the "IBM UNIVAC". Remington Rand eventually sold 46 machines at more than $1 million each. UNIVAC was the first 'mass produced' computer; all predecessors had been 'one-off' units. It used 5,200 vacuum tubes and consumed 125 kW of power. It used a mercury delay line capable of storing 1,000 words of 11 decimal digits

plus sign (72-bit words) for memory. Unlike earlier machines it did not use a punch card system but a metal tape input.

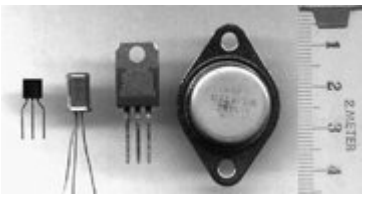
In November 1951, the J. Lyons company began weekly operation of a bakery valuations job on the LEO (Lyons Electronic Office). This was the first business application to go live on a stored program computer.

In 1953, IBM introduced the IBM 701 Electronic Data Processing Machine, the first in its successful 700/7000 series and its first mainframe computer. The first implemented high-level general purpose programming language, Fortran, was also being developed at IBM around this time. (Konrad Zuse's 1945 design of the high-level language Plankalkül was not implemented at that time.

In 1956, IBM sold its first magnetic disk system, RAMAC (Random Access Method of Accounting and Control). It used 50 24-inch metal disks, with 100 tracks per side. It could store 5 megabytes of data and cost $10,000 per megabyte. (As of 2005, disk storage costs less than $1 per gigabyte).

## 2.12. Second generation -- late 1950s and early 1960s

The next major step in the history of computing was the invention of the transistor in 1947. This replaced the fragile and power hungry valves with a much smaller and more reliable component. Transistorised computers are normally referred to as 'Second Generation' and dominated the late 1950s and early 1960s. By using transistors and printed circuits a significant decrease in size and power consumption was achieved, along with an increase in reliability

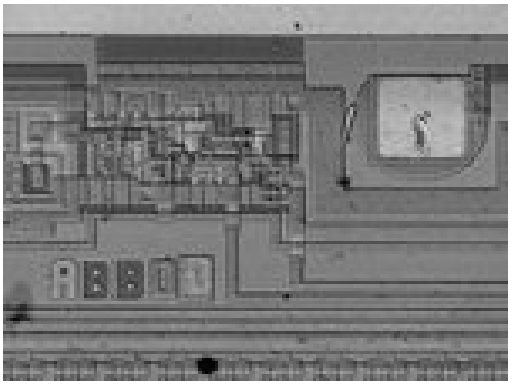


**Transistors, above, revolutionized computers as smaller and more efficient replacements for vacuum tubes.**

Second generation computers were still expensive and were primarily used by universities, governments, and large corporations.

In 1959 IBM shipped the transistor-based IBM 7090 mainframe and medium scale IBM 1401. The latter was designed around punch card input and proved a popular general purpose computer. Some 12,000 were shipped, making it the most successful machine in computer history at the time. It used a magnetic core memory of 4000 characters (later expanded to 16,000 characters).

In 1964 IBM announced the S/360 series, which was the first family of computers that could run the same software at different combinations of speed, capacity and price. It also pioneered the commercial use of microprograms, and an extended instruction set designed for processing many types of data, not just arithmetic. In addition, it unified IBM's product line, which prior to that time had included both a "commercial" product line and a separate "scientific" line. The software provided with System/360 also included major advances, including commercially available multi-programming, new programming languages, and independence of programs from input/output devices



**The microscopic integrated circuit, above, combined many hundreds of transistors into one unit for fabrication.**

## 2.13. Third generation and beyond, post-1964

The explosion in the use of computers began with 'Third Generation' computers. These relied on Jack St. Clair Kilby's and Robert Noyce's independent invention of the integrated circuit (or microchip), which later led to Ted Hoff's invention of the microprocessor, at Intel.

The microprocessor led to the development of the

microcomputer, small, low-cost computers that could be owned by individuals and small businesses. Microcomputers, the first of which appeared in the 1970s, became ubiquitous in the 1980s and beyond. Computing has evolved with microcomputer architectures, with features added from their larger brethren, now dominant in most market segments.

## 2.14. Fourth Generation (1972-1984)

The next generation of computer systems saw the use of large scale integration (LSI - 1000 devices per chip) and very large scale integration (VLSI - 100,000 devices per chip) in the construction of computing elements. At this scale entire processors will fit onto a single chip, and for simple systems the entire computer (processor, main memory, and I/O controllers) can fit on one chip. Gate delays dropped to about 1ns per gate. Semiconductor memories replaced core memories as the main memory in most systems; until this time the use of semiconductor memory in most systems was limited to registers and cache.

During this period, high speed vector processors, such as the CRAY 1, CRAY X-MP and CYBER 205 dominated the high performance computing scene.

Computers with large main memory, such as the CRAY 2, began to emerge. A variety of parallel architectures began to appear; however, during this period the parallel computing efforts were of a mostly experimental nature and most computational science was carried out on vector processors. Microcomputers and workstations were introduced and saw wide use as alternatives to time-shared mainframe computers.

## 2.15. Fifth Generation (1984-1990)

The development of the next generation of computer systems is characterized mainly by the acceptance of parallel processing. Until this time parallelism was limited to pipelining and vector processing, or at most to a few processors sharing jobs.

The fifth generation saw the introduction of machines with hundreds of processors that could all be working on different parts of a single program. The scale of integration in semiconductors continued at an incredible pace - by 1990 it was possible to build chips with a million components - and semiconductor memories became standard on all computers. Other new developments were the widespread use of computer networks and the increasing use of single-user workstations.

Prior to 1985 large scale parallel processing was viewed as a research goal, but two systems introduced around this time are typical of the first commercial products to be based on parallel processing. The Sequent Balance 8000 connected up to 20 processors to a single shared memory module (but each processor had its own local cache). The machine was designed to compete with the DEC VAX-780 as a general purpose Unix system, with each processor working on a different user's job. However Sequent provided a library of subroutines that would allow programmers to write programs that would use more than one processor, and the machine was widely used to explore parallel algorithms and programming techniques.
The Intel iPSC-1, nicknamed ``the hypercube'', took a different approach. Instead of using one memory module, Intel connected each processor to its own memory and used a network interface to connect processors. This distributed memory architecture meant memory was no longer a bottleneck and large systems (using more processors) could be built. The largest iPSC-1 had 128 processors.
Toward the end of this period a third type of parallel processor was introduced to the market. In this style of machine, known as a data-parallel or SIMD, there are several thousand very simple processors. All processors work under the direction of a single control unit; i.e. if the control unit

says ``add a to b" then all processors find their local copy of a and add it to their local copy of b. Machines in this class include the Connection Machine from Thinking Machines, Inc., and the MP-1 from MasPar, Inc. Scientific computing in this period was still dominated by vector processing.

Most manufacturers of vector processors introduced parallel models, but there were very few (two to eight) processors in this parallel machines. In the area of computer networking, both wide area network (WAN) and local area network (LAN) technology developed at a rapid pace, stimulating a transition from the traditional mainframe computing environment toward a distributed computing environment in which each user has their own workstation for relatively simple tasks (editing and compiling programs, reading mail) but sharing large, expensive resources such as file servers and supercomputers. RISC technology (a style of internal organization of the CPU) and plummeting costs for RAM brought tremendous gains in computational power of relatively low cost workstations and servers. This period also saw a marked increase in both the quality and quantity of scientific visualization.

## 2.16. Sixth Generation (1990 - )

Transitions between generations in computer technology are hard to define, especially as they are taking place. Some changes, such as the switch from vacuum tubes to transistors, are immediately apparent as fundamental changes, but others are clear only in retrospect.

Many of the developments in computer systems since 1990 reflect gradual improvements over established systems, and thus it is hard to claim they represent a transition to a new ``generation", but other developments will prove to be significant changes. In this section we offer some assessments about recent developments and current trends that we think will have a significant impact on computational science.

This generation is beginning with many gains in parallel computing, both in the hardware area and in improved understanding of how to develop algorithms to exploit diverse, massively parallel architectures.

Parallel systems now compete with vector processors in terms of total computing power and most expect parallel systems to dominate the future. Combinations of parallel/vector architectures are well established, and one corporation (Fujitsu) has announced plans to build a system with over 200 of its high end vector processors.

Manufacturers have set themselves the goal of achieving teraflops ( 10 arithmetic operations per second) performance by the middle of the decade, and it is clear this will be obtained only by a system with a thousand processors or more. Workstation technology has continued to improve, with processor designs now using a combination of RISC, pipelining, and parallel processing. As a result it is now possible to purchase a desktop workstation for about $30,000 that has the same overall computing power (100 megaflops) as fourth generation supercomputers.

This development has sparked an interest in heterogeneous computing: a program started on one workstation can find idle workstations elsewhere in the local network to run parallel subtasks. One of the most dramatic changes in the sixth generation will be the explosive growth of wide area networking. Network bandwidth has expanded tremendously in the last few years and will continue to improve for the next several years. T1 transmission rates are now standard for regional networks, and the national ``backbone" that interconnects regional networks uses T3.

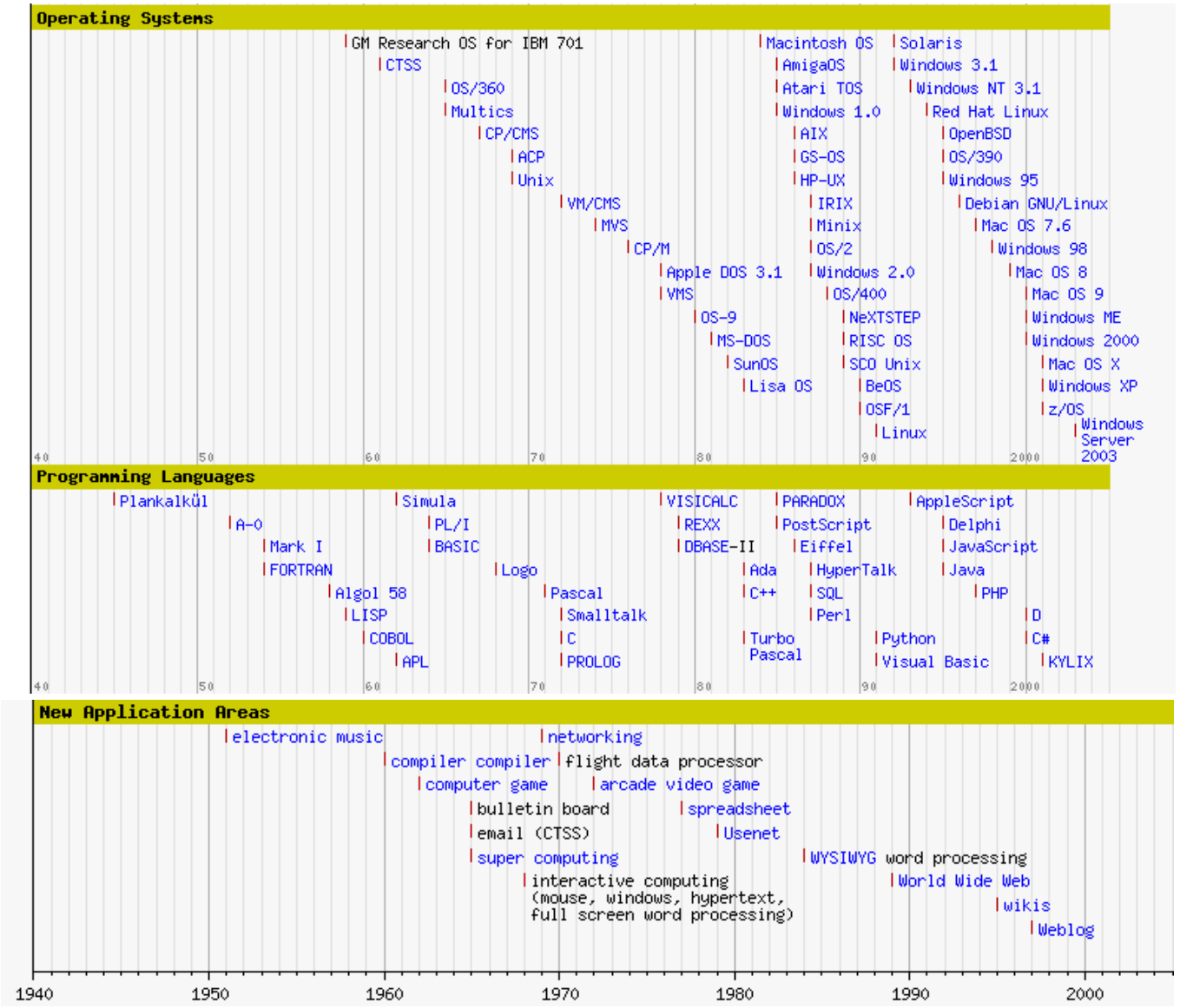
Networking technology is becoming more widespread than its original strong base in universities and government laboratories as it is rapidly finding application in K-12 education, community networks

and private industry. A little over a decade after the warning voiced in the Lax report, the future of a strong computational science infrastructure is bright.

The federal commitment to high performance computing has been further strengthened with the passage of two particularly significant pieces of legislation: the High Performance Computing Act of 1991, which established the High Performance Computing and Communication Program (HPCCP) and Sen. Gore's Information Infrastructure and Technology Act of 1992, which addresses a broad spectrum of issues ranging from high performance computing to expanded network access and the necessity to make leading edge technologies available to educators from kindergarten through graduate school.

In bringing this encapsulated survey of the development of a computational science infrastructure up to date, we observe that the President's FY 1993 budget contains $2.1 billion for mathematics, science, technology and science literacy educational programs, a 43% increase over FY 90 figures

# 3. History of Software



## 3.1. Introduction

A simple question: "What is software?" A very simple answer is: Hardware you can touch, software you can't. But that is too simple indeed.

When talking about software we talk about programming and programming languages and about producing and selling the products made by programming (languages) which are called “Software”.

## 3.2. How It All Started

It shouldn't be a big surprise that the creation or software also went in large but distinguishable steps. Compared with hardware there were fewer developments that went parallel or overlapping. In rare cases developments were reinvented sometimes because the development or invention was not published, even prohibited to be made public (war, secrecy acts etc.) or became known at the same time and after (legal) discussions the "other" party won the honors.

The earliest practical form of programming was probably done by Jaquard (1804, France). He designed a loom that performed predefined tasks through feeding punched cards into a reading contraption.

This picture shows the manufacturing of punched cards for looms

The technology of punched cards will later be adapted by (IBM's) Recording and Tabulating Company to process data
A problem needed to be solved thus a machine was built. (Pascal, Babbage, Scheultz & Son) And it required some sort of instruction; a sequence was designed or written and transferred to either cards or mechanical aids such as wires, gears, shafts actuators etc. This was the birth of programming.
First Ada Lovelace, was writing a rudimentary program (1843) for the Analytical Machine, designed by Charles Babbage in 1827, but the machine never came into operation.

Then there was George Boole (1815-1864), a British mathematician, who proved the relation between mathematics and logic with his algebra of logic (BOOLEAN algebra or binary logic) in 1847.This meant a breakthrough for mathematics. Boole was the first to prove that logic is part of mathematics and not of philosophy.

### 3.2.1. A big step in thinking

But it took **one hundred years** before this algebra of logic was put to work for computing. It took Claude Shannon (1916-2001) who wrote a thesis (*A Mathematical Theory of Communication* in the *Bell System Technical Journal* -1948) on how binary logic could be used in computing to complete the software concept of modern computing. Now things were in place to start off with the information age.

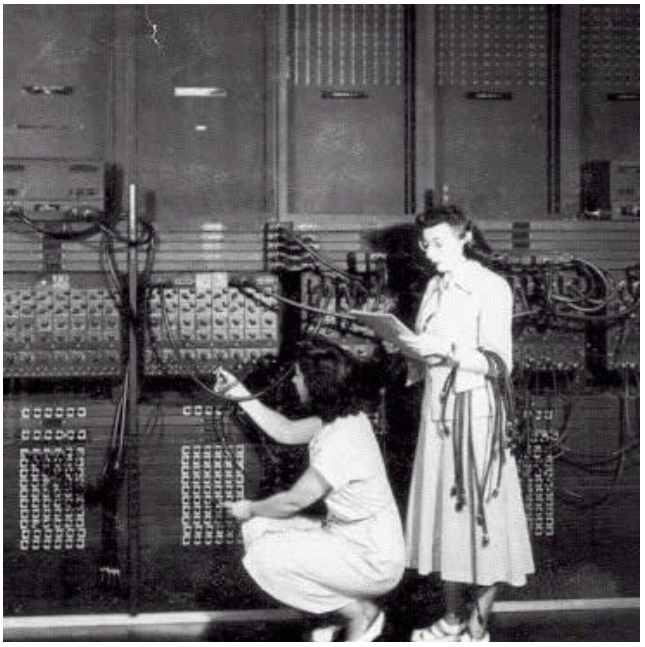
### 3.2.2. Enter the Information Age

In the beginning of the so called "Information Age" computers were programmed by "programming" direct instructions into it. This was done by setting switches or making connections to different logical units by wires (circuitry).

Programming like this was nothing else but rewiring these huge machines in order to use all the options, possibilities and calculations. Reprogramming always meant rewiring.

In that way calculations needed days of preparations, handling thousands of wires, resetting switches, plugs etc. (in the most extreme case that is). And the programmed calculation itself just took a few minutes. If the "programming" of wiring did not have a wrong connection, the word bug was not in used yet, for programming errors.

The coding panels very much looked like that a few telephone switchboards hustled together, and in fact many parts actually came from switch boards.



**Two women wiring the right side of the ENIAC with a new program (US Army photo, from archives of the ARL Technical library, courtesy of Mike Muuss)**

With the invention of vacuum tubes, along with many other inventions, much of the rewiring belonged to the past. The tubes replaced the slow machines based on relays.

When Shannon reinvented or better rediscovered the binary calculus in 1948 and indicated how that could be used for computing a revolution started. The race was on!

First programming was done by typing in 1's or 0's that were stored on different information carriers. Like paper tapes, punched hole cards, hydrogen delay lines (sound or electric pulse) and later magnetic drums and much later magnetic and optical discs.

By storing these 0's and 1's on a carrier (first used by Karl Suze's X1 in 1938) it was possible to have the computer read the data on any later time. But mis typing a single zero or one meant a disaster because all coding (instructions) should absolutely be on the right place and in the right order in memory. This technology was called absolute addressing.

**An example:**

1010010101110101011

If this is a sequence for switches it means switch one on, switch two off etc. etc.

In the early 50's programmers started to let the machines do a part of the job. This was called automatic coding and made live a lot easier for the early programmers.

The next development was to combine groups of instruction into so called words and abbreviations were thought up called: opcodes (Hopper 1948)

## 3.2.3. Machine Language

Opcode works like a shorthand and represents as said a group of machine instructions. The opcode is translated by another program into zero's and one's, something a machine could translate into instructions. But the relation is still one to one: one code to one single instruction. However very basically this is already a programming language. It was called: assembly language.

**An example:**

| Label | Opcode | Register |
|---|---|---|
| **CALC:** | **STO** | **R1, HELP0** |
| | **STO** | **R2, HELP2** |
| | **LD** | **R3, HELP1** |
| | **ADD** | **R3, HELP2** |
| | **LD** | **R4, HELP1** |
| | **SUB** | **R4, HELP2** |
| | **RSR** | **SP, 0** |
| **HELP1:** | **DS** | **2** |
| **HELP2:** | **DS** | **2** |

**This piece of assembly code calculates the difference between two numbers.**

## 3.2.4. Subroutines

Soon after developing machine languages and the first crude programming languages began to appear the danger of inextricable and thus unreadable coding became apparent. Later this messy programming was called: "spaghetti code".

One important step in unraveling or preventing spaghetti code was the development of subroutines. And it needed Maurice Wilkes, when realizing that "a good part of the remainder of his life was going to be spent in finding errors in ... programs", to develop the concept of subroutines in programs to create reusable modules. Together with Stanley Gill and David Wheeler he produced the first textbook on "The Preparation of Programs for an Electronic Digital Computer". The formalized concept of software development (not named so for another decade) had its beginning in 1951.

### Fortran

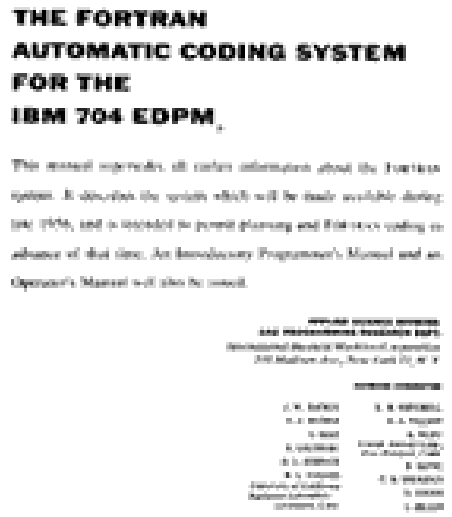
The next big step in programming began when an IBM team under John W. Backus created FORTRAN - FORmula TRANslator 1952. It could only be used on their own machine, the: IBM 704. But later versions for other machines, and platforms were sold soon after. Until long past 1960 different CPU's required an other kind instruction set to add a number, thus for each different machine a different compiler was needed. Typically the manual came years later in 1957!

**Programming language**


Programmer's
Reference Manual
October 15, 1956

FORTRAN soon was called a programming language. This was because some characteristics of a language are met:

- It must have a vocabulary - list of words
- It must have a grammar - or syntax
- It must be unique, both in words and in context


THE FORTRAN
AUTOMATIC CODING SYSTEM
FOR THE
IBM 704 EDPM

All the above criteria were easily met for this - strictly defined- set of computer instructions.
An example:

**Enter "C"**

C came into being in the years 1969-1973 and was developed by Dennis Richey and David Kerningham both working at the Bell laboratories. And the magic word was portability. Parallel at the development of computer languages, Operating Systems (OS) were developed. These were needed because to create programs and having to write all machine specific instructions over and over again was a "waste of time" job. So by introducing the OS 'principle' almost all input and output tasks were taken over by the OS.
Such as:

- writing data to and from memory,
- saving data on disks,
- writing data to screens and printers,
- starting tapes,
- refreshing memory,
- scheduling specific tasks
- etcetera, etcetera.

As the common computer languages had trouble to be translated from one machine to another the OS's had to take the same hurdle every time a new machine was developed. The need and pressure for a common portable language was enormous. There were some unsuccessful projects aimed to solve this problem, like Multics, a joint venture of MIT, General Electric, and Bell Labs. And other developments at DOD's in different countries. But they all came either too late or became too complex to succeed.

But the demise of Multics inspired Dennis Ritchie and Brian Kernigham to develop C in 1972. This was and still is a very strict language that stayed close enough to the machine's internal logic and structure. This new language was reasonable well to read and understand by humans. And because of

this combination the language is fast, compact and became very popular amongst system programmers and commercial software manufacturers.

With that language they also developed UNIX, a generic operating system.

The power of C was that the language had a small language base (vocabulary) but leaned heavily on what they called libraries. Libraries contain machine specific instructions to perform tasks, like the OS does. These libraries were the only parts that had to be redesigned for different machines, or processor families. But, and that was C's strength, the programming interface/language remained the same. Portability was born. Source code could be reused, only to be recompiled when it had to run on other machines.

A classic example of C, printing "Hello World" on your screen:

```
/* helloworld.c */

main()
{
printf('Hello
World\n");
}
```

Soon other language manufacturers sprang on the bandwagon and the software industry leaped ahead. The industry took off like a rocket!

| Predecessor(s) | Year | Name | Chief Developer, Company |
|---|---|---|---|
| **Pre 1950** | | | |
| * | ~1840 | *first program* | Ada Lovelace |
| * | 1945 | **Plankalkül** (concept) | Konrad Zuse |
| **1950s** | | | |
| * | 1952 | **A-0** | Grace Hopper |
| * | 1954 | **Mark I Autocode** | Tony Brooker |
| A-0 | 1954-1955 | **FORTRAN "0"** (concept) | John W. Backus at IBM |
| A-0 | 1954 | **ARITH-MATIC** | Grace Hopper |
| A-0 | 1954 | **MATH-MATIC** | Grace Hopper |
| * | 1954 | **IPL V** (concept) | Allen Newell, Cliff Shaw, Herbert Simon |
| A-0 | 1955 | **FLOW-MATIC** | Grace Hopper |
| IPL | 1956-1958 | **LISP** (concept) | John McCarthy |
| FLOW-MATIC | 1957 | **COMTRAN** | Bob Bemer |
| FORTRAN 0 | 1957 | **FORTRAN "I"** (implementation) | John W. Backus at IBM |
| * | 1957 | **COMIT** (concept) | |
| FORTRAN I | 1958 | **FORTRAN II** | John W. Backus at IBM |
| FORTRAN | 1958 | **ALGOL 58** (**IAL**) | International effort |
| * | 1958 | **IPL V** (implementation) | Allen Newell, Cliff Shaw, Herbert Simon |
| FLOW-MATIC, COMTRAN | 1959 | **COBOL**(concept) | The Codasyl Committee |
| IPL | 1959 | **LISP** (implementation) | John McCarthy |
| | 1959 | **TRAC** (concept) | Mooers |
| **1960s** | | | |
| ALGOL 58 | 1960 | **ALGOL 60** | |
| FLOW-MATIC, COMTRAN | 1960 | **COBOL 61** (implementation) | The Codasyl Committee |
| * | 1961 | **COMIT** (implementation) | |
| FORTRAN II | 1962 | **FORTRAN IV** | |

| | | | |
|---|---|---|---|
| * | 1962 | **APL** (concept) | Iverson |
| ALGOL 58 | 1962 | **MAD** | Arden, *et. al.* |
| ALGOL 60 | 1962 | **SIMULA** (concept) | |
| FORTRAN II, COMIT | 1962 | **SNOBOL** | Griswold, *et al.* |
| ALGOL 60 | 1963 | **CPL** | Barron, Strachey, *et al.* |
| SNOBOL | 1962 | **SNOBOL4** (concept) | Griswold, *et al.* |
| ALGOL 60 | 1963 | **ALGOL 68** (concept) | van Wijngaarden, *et al.* |
| ALGOL 58 | 1963 | **JOSS I** | Cliff Shaw, RAND |
| CPL, LISP | 1964 | **COWSEL** | Burstall, Popplestone |
| ALGOL 60, COBOL, FORTRAN | 1964 | **PL/I** (concept) | IBM |
| FORTRAN II, JOSS | 1964 | **BASIC** | Kemeny and Kurtz |
| | 1964 | **TRAC** (implementation) | Mooers |
| | 1964? | **IITRAN** | |
| JOSS | 1965 | **TELCOMP** | BBN |
| JOSS I | 1966 | **JOSS II** | Chuck Baker, RAND |
| FORTRAN IV | 1966 | **FORTRAN 66** | |
| LISP | 1966 | **ISWIM** | Landin |
| ALGOL 60 | 1966 | **CORAL66** | |
| CPL | 1967 | **BCPL** | Richards |
| FORTRAN, TELCOMP | 1967 | **MUMPS** | Massachusetts General Hospital |
| * | 1967 | **APL** (implementation) | Iverson |
| ALGOL 60 | 1967 | **SIMULA 67** (implementation) | Dahl, Myhrhaug, Nygaard at Norsk Regnesentral |
| SNOBOL | 1967 | **SNOBOL4** (implementation) | Griswold, *et al.* |
| COWSEL | 1968 | **POP-1** | Burstall, Popplestone |
| | 1968 | **FORTH** (concept) | Moore |
| LISP | 1968 | **LOGO** | Papert |
| ALGOL 60 | 1969 | **ALGOL 68** (implementation) | van Wijngaarden, *et al.* |
| ALGOL 60, COBOL, FORTRAN | 1969 | **PL/I** (implementation) | IBM |
| **1970s** | | | |
| | 1970? | **FORTH** (implementation) | Moore |
| POP-1 | 1970 | **POP-2** | |
| ALGOL 60 | 1971 | **Pascal** | Wirth, Jensen |
| SIMULA 67 | 1972 | **Smalltalk-72** | Xerox PARC |
| B, BCPL, ALGOL 68 | 1972 | **C** | Ritchie |
| * | 1972 | **INTERCAL** | |
| 2-level W-Grammar | 1972 | **Prolog** | Colmerauer |
| Pascal, BASIC | 1973 | **COMAL** | Christensen, Løfstedt |
| BASIC | 1974 | **GRASS** | DeFanti |
| Business BASIC | 1974 | **BASIC FOUR** | BASIC FOUR CORPORATION |
| LISP | 1975 | **Scheme** | Sussman, Steele |
| Pascal | 1975? | **Modula** | Wirth |
| BASIC | 1975 | **Altair BASIC** | Gates, Allen |
| Smalltalk-72 | 1976 | **Smalltalk-76** | Xerox PARC |
| C, FORTRAN | 1976 | **Ratfor** | Kernighan |
| * | 1977 | **FP** | John Backus |
| * | 1977 | **Bourne Shell** (*sh*) | Bourne |
| MUMPS | 1977 | **Standard MUMPS** | |
| FORTRAN IV | 1978 | **FORTRAN 77** | |
| Modula | 1978? | **Modula-2** | Wirth |
| * | 1978? | **MATLAB** | Moler at the University of New Mexico |
| * | 1978 | **VISICALC** | Bricklin, Frankston at VisiCorp |
| PL/I, BASIC, EXEC 2 | 1979 | **REXX** | Cowlishaw |
| C, SNOBOL | 1979 | **Awk** | Aho, Weinberger, Kernighan |
| * | 1979 | **Vulcan dBase-II** | Ratliff |
| ALGOL 68 | 1979 | **Green** | Ichbiah *et al.* at US Dept of Defense |
| **1980s** | | | |
| C, SIMULA 67 | 1980 | **C with Classes** | Stroustrup |
| Smalltalk-76 | 1980 | **Smalltalk-80** | Xerox PARC |
| Smalltalk, C | 1982 | **Objective-C** | Brad Cox |
| Green | 1983 | **Ada 83** | U.S. Department of Defense |
| C with Classes | 1983 | **C++** | Stroustrup |

| | | | |
|---|---|---|---|
| Pascal | 1983 | **Turbo Pascal** | Hejlsberg |
| BASIC | 1983 | **True BASIC** | Kemeny, Kurtz at Dartmouth University |
| sh | 1984? | **Korn Shell** (*ksh*) | Dave Korn |
| * | 1984 | **Standard ML** | |
| dBase | 1984 | **CLIPPER** | Nantucket |
| LISP | 1984 | **Common Lisp** | Guy Steele and many others |
| 1977MUMPS | 1985 | **1984 MUMPS** | |
| dBase | 1985 | **PARADOX** | Borland |
| Interpress | 1985 | **PostScript** | Warnock |
| BASIC | 1985 | **QuickBASIC** | Microsoft |
| | 1986 | **Miranda** | David Turner at University of Kent |
| | 1986 | **LabVIEW** | National Instruments |
| SIMULA 67 | 1986 | **Eiffel** | Meyer |
| | 1986 | **Informix-4GL** | Informix |
| C | 1986 | **PROMAL** | |
| Smalltalk | 1987 | **Self** (concept) | Sun Microsystems Inc. |
| * | 1987 | **HyperTalk** | Apple |
| * | 1987 | **SQL-87** | |
| C, sed, awk, sh | 1987 | **Perl** | Wall |
| MATLAB | 1988 | **Octave** | |
| dBase-III | 1988 | **dBase-IV** | |
| Awk, Lisp | 1988 | **Tcl** | Ousterhout |
| REXX | 1988 | **Object REXX** | Simon Nash |
| Ada | 1988 | **SPARK** | Bernard A. Carré |
| Turbo Pascal | 1989 | **Turbo Pascal OOP** | Borland |
| C | 1989 | **Standard C89/90** | ANSI X3.159-1989 (adopted by ISO in 1990) |
| Modula-2 | 1989 | **Modula-3** | Cardeli, et al. |
| Modula-2 | 1989 | **Oberon** | Wirth |
| **1990s** | | | |
| Oberon | 1990 | **Object Oberon** | Wirth |
| APL, FP | 1990 | **J** | Iverson, R. Hui at Iverson Software |
| Miranda | 1990 | **Haskell** | |
| 1984 MUMPS | 1990 | **1990 MUMPS** | |
| Fortran 77 | 1991 | **Fortran 90** | |
| Object Oberon | 1991 | **Oberon-2** | Wirth |
| ABC | 1991 | **Python** | Van Rossum |
| | 1991 | **Q** | |
| QuickBASIC | 1991 | **Visual Basic** | Alan Cooper at Microsoft |
| SQL-87 | 1992 | **SQL-92** | |
| Turbo Pascal OOP | 1992 | **Borland Pascal** | |
| ksh | 1993? | **Z Shell** (*zsh*) | |
| Smalltalk | 1993? | **Self** (implementation) | Sun Microsystems Inc. |
| Forth | 1993 | **FALSE** | Oortmerrsen |
| FALSE | 1993 | **Brainfuck** | Mueller |
| HyperTalk | 1993 | **Revolution Transcript** | |
| HyperTalk | 1993 | **AppleScript** | Apple |
| APL, Lisp | 1993 | **K** | Whitney |
| Smalltalk, Perl | 1993 | **Ruby** | |
| | 1993 | **Lua** | Waldemar Celes *et al.* at Tecgraf, PUC-Rio |
| C | 1993 | **ZPL** | Chamberlain *et al.* at University of Washington |
| Lisp | 1994 | **Dylan** | many people at Apple Computer |
| Ada 83 | 1995 | **Ada 95** | ISO |
| Borland Pascal | 1995 | **Delphi** | Anders Hejlsberg at Borland |
| C, SIMULA67 OR C++, Smalltalk | 1995 | **Java** | James Gosling at Sun Microsystems |
| 1990MUMPS | 1995 | **1995 MUMPS** | |
| Self, Java | 1995? | **LiveScript** | Brendan Eich at Netscape |
| Fortran 90 | 1996 | **Fortran 95** | |
| REXX | 1996 | **NetRexx** | Cowlishaw |
| LiveScript | 1997? | **JavaScript** | Brendan Eich at Netscape |
| SML 84 | 1997 | **SML 97** | |
| PHP 3 | 1997 | **PHP** | |
| Scheme | 1997 | **Pico** | Free University of Brussels |

| | | | |
|---|---|---|---|
| Smalltalk-80, Self | 1997 | **Squeak Smalltalk** | Alan Kay, *et al.* at Apple Computer |
| JavaScript | 1997? | **ECMAScript** | ECMA TC39-TG1 |
| C++, Standard C | 1998 | **Standard C++** | ANSI/ISO Standard C++ |
| Prolog | 1998 | **Erlang** | Open Source Erlang at Ericsson |
| Standard C89/90 | 1999 | **Standard C99** | ISO/IEC 9899:1999 |
| **2000s** | | | |
| FP, Forth | 2000 | **Joy** | von Thun |
| C, C++ | 2000 | **D** | Walter Bright at Digital Mars |
| C, C++, Java | 2000 | **C#** | Anders Hejlsberg at Microsoft (ECMA) |
| Whitespace | 2003 | **Whitespace** | Brady and Morris |
| Perl, C++ | 2003 | **S2** | Fitzpatrick, Atkins |
| C#, ML, MetaHaskell | 2003 | **Nemerle** | University of Wroclaw |
| J, FL, K | 2003 | **NGL** | E. Herrera at Tlällian |
| Joy, Forth, Lisp | 2003 | **Factor** | Slava Pestov |
| Fortran 95 | 2004 | **Fortran 2003** | |
| Python, C#, Ruby | 2004 | **Boo** | Rodrigo B. de Oliveira |

## 3.3. History of Operating System

The history of computer operating systems recapitulates to a degree, the recent history of computing. Operating systems provide a set of functions needed and used by most applications, and provide the necessary linkages to control a computer's hardware. Without an operating system, each program would have to have drivers for your video card, sound card, hard drive, and other peripherals.

### 3.3.1. The mainframe era

Early operating systems were very diverse, with each vendor producing one or more operating systems specific to their particular hardware. This state of affairs continued until the 1960s when IBM developed the System/360 series of machines; although there were enormous performance differences across the range, all the machines ran essentially the same operating system, OS/360.

OS/360 evolved to become successively MFT, MVT, SVS, MVS, MVS/XA, MVS/ESA, OS/390 and z/OS, that includes the UNIX kernel as well as a huge amount of new functions required by modern mission-critical applications running on the zSeries mainframes.

### 3.3.2. Minicomputers and the rise of UNIX

The UNIX operating system was developed at AT&T Bell Laboratories. Because it was essentially free in early editions, easily obtainable, and easily modified, it achieved wide acceptance. It also became a requirement within the Bell systems operating companies. Since it was written in a high level language, when that language was ported to a new machine architecture it was also able to be ported. This portability permitted it to become the choice for a second generation of minicomputers and the first generation of workstations. By widespread use it exemplified the idea of a operating system that was conceptually the same across various hardware platforms. It still was owned by AT&T and that limited its use to groups or corporations who could afford to license it.

### 3.3.3. The personal computer era: Apple, DOS and beyond

The development of microprocessors made inexpensive computing available for the small business and hobbyist, which in turn led to the widespread use of interchangeable hardware components using a common interconnection (such as the S-100, SS-50, Apple II, ISA, and PCI buses), and an increasing need for 'standard' operating systems to control them. The most important of the early OSes on these machines was Digital Research's CP/M-80 for the 8080 / 8085 / Z-80 CPUs. It was based on several

Digital Equipment Corporation operating systems, mostly for the PDP-11 architecture. MS-DOS (or PC-DOS when supplied by IBM) was based originally on CP/M-80. Each of these machines had a small boot program in ROM which loaded the OS itself from disk.

The decreasing cost of display equipment and processors made it practical to provide graphical user interfaces for many operating systems, such as the generic X Window System that is provided with many UNIX systems, or other graphical systems such as Microsoft Windows, the RadioShack Color Computer's OS-9, Commodore's AmigaOS, Level II, Apple's Mac OS, or even IBM's OS/2. The original GUI was developed at Xerox Palo Alto Research Center in the early '70s (the Alto computer system) and imitated by many vendors.

# 4. History of Network

The earliest idea of a computer network intended to allow general communication between users of various computers was the ARPANET, the world's first packet switching network, which first went online in 1969.

The Internet's roots lie within the ARPANET, which not only was the intellectual forerunner of the Internet, but was also initially the core network in the collection of networks in the Internet, as well as an important tool in developing the Internet (being used for communication between the groups working on internetworking research).

## 4.1. Motivation for the Internet

The need for an internetwork appeared with ARPA's sponsorship, by Robert E. Kahn, of the development of a number of innovative networking technologies; in particular, the first packet radio networks (inspired by the ALOHA network), and a satellite packet communication program. Later, local area networks (LANs) would also join the mix.

Connecting these disparate networking technologies was not possible with the kind of protocols used on the ARPANET, which depended on the exact nature of the subnetwork. A wholly new kind of networking architecture was needed.

## 4.2. Early Internet work

Kahn recruited Vint Cerf of University of California, Los Angeles to work with him on the problem, and they soon worked out a fundamental reformulation, where the differences between network protocols were hidden by using a common internetwork protocol, and instead of the network being responsible for reliability, as in the ARPANET, the hosts became responsible. Cerf credits Herbert Zimmerman and Louis Pouzin (designer of the CYCLADES network) with important influences on this design. Some accounts also credit the early networking work at Xerox PARC with an important technical influence.

With the role of the network reduced to the bare minimum, it became possible to join almost any networks together, no matter what their characteristics, thereby solving Kahn's initial problem. (One popular saying has it that TCP/IP, the eventual product of Cerf and Kahn's work, will run over "two tin cans and a string".) A computer called a *gateway* (later changed to *router* to avoid confusion with other types of *gateway*) is provided with an interface to each network, and forwards packets back and forth between them.

Happily, this new concept was a perfect fit with the newly emerging local area networks, which were revolutionizing communication between computers within a site.

## 4.3. Early growth

After ARPANET had been up and running for a decade, ARPA looked for another agency to hand off the network to. After all, ARPA's primary business was funding cutting-edge research and development, not running a communications utility. Eventually the network was turned over to the Defense Communications Agency, also part of the Department of Defense.

In 1983, TCP/IP protocols replaced the earlier NCP protocol as the principal protocol of the ARPANET; in 1984, the U.S. military portion of the ARPANet was broken off as a separate network, the MILNET. At the same time, Paul Mockapetris and Jon Postel were working on what would become the Domain Name System.

The early Internet, based around the ARPANET, was government-funded and therefore restricted to non-commercial uses such as research; unrelated commercial use was strictly forbidden. This initially restricted connections to military sites and universities. During the 1980s, the connections expanded to more educational institutions, and even to a growing number of companies such as Digital Equipment Corporation and Hewlett-Packard, which were participating in research projects, or providing services to those who were.

Another branch of the U.S. government, the National Science Foundation, became heavily involved in Internet research in the mid-1980s. The NSFNet backbone, intended to connect and provide access to a number of supercomputing centers established by the NSF, was established in 1986.
At the end of the 1980s, the U.S. Department of Defense decided the network was developed enough for its initial purposes, and decided to stop further funding of the core Internet backbone. The ARPANET was gradually shut down (its last node was turned off in 1989), and NSF, a civilian agency, took over responsibility for providing long-haul connectivity in the U.S.

In another NSF initiative, regional TCP/IP-based networks such as NYSERNet (New York State Education and Research Network) and BARRNet (Bay Area Regional Research Network), grew up and started interconnecting with the nascent Internet. This greatly expanded the reach of the rapidly growing network. On April 30, 1995 the NSF privatized access to the network they had created. It was at this point that the growth of the Internet really took off.

## 4.4. Commercialization and privatization

Parallel to the Internet, other networks were growing. Some were educational and centrally-organized like BITNET and CSNET. Others were a grass-roots mix of school, commercial, and hobby like the UUCP network.

During the late 1980s the first Internet Service Provider (ISP) companies were formed. Companies like PSINet, UUNET, Netcom, and Portal were formed to provide service to the regional research networks and provide alternate network access (like UUCP-based email and Usenet News) to the public. The first dial-up ISP, world.std.com, opened in 1989.

The interest in commercial use of the Internet became a hotly-debated topic. Although commercial use was forbidden, the exact definition of commercial use could be unclear and subjective. Everyone agreed that one company sending an invoice to another company was clearly commercial use, but anything less was up for debate. The alternate networks, like UUCP, had no such restrictions, so many people were skirting grey areas in the interconnection of the various networks.

Many university users were outraged at the idea of non-educational use of their networks. Ironically it was the commercial Internet service providers who brought prices low enough that junior colleges and other schools could afford to participate in the new arenas of education and research.

By 1994, the NSFNet lost its standing as the backbone of the Internet. Other competing commercial providers created their own backbones and interconnections. Regional NAPs (network access points) became the primary interconnections between the many networks. The NSFNet was dropped as the main backbone, and commercial restrictions were gone.

## 4.5. Early applications

E-mail existed as a message service on early time-sharing mainframe computers connected to a number of terminals. Around 1971 it developed into the first system of exchanging addressed messages between different, networked computers; in 1972 Ray Tomlinson introduced the "name@computer" notation that is still used today. E-mail turned into the Internet "killer application" of the 1980s.

The early e-mail system was not limited to the Internet. Gateway machines connected Internet SMTP e-mail with UUCP mail, BITNET, the Fidonet BBS network, and other services. Commercial e-mail providers such as CompuServe and The Source could also be reached.

The second most popular application of the early Internet was Usenet, a system of distributed discussion groups which is still going strong today. Usenet had existed even before the internet, as an application of Unix computers connected by telephone lines via UUCP. The Network News Transfer Protocol (NNTP), similar in flavor to SMTP, slowly replaced UUCP for the relaying of news articles. Today, almost all Usenet traffic is carried over high-speed NNTP servers.

Other early protocols include the File Transfer Protocol (1985), and Telnet (1983), a networked terminal emulator allowing users on one computer to log in to other computers.

## 4.6. Host naming and the DNS

One of the scalability limitations of the early Internet was that there was no distributed way to name hosts, other than with their IP addresses. The Network Information Centre (NIC) maintained a central hosts file, giving names for various reachable hosts. Sites were expected to download this file regularly to have a current list of hosts.

In 1984, Paul Mockapetris devised the Domain Name System (DNS) as an alternative. Domain names (like "wikipedia.org") provided names for hosts that were both globally unique (like IP addresses), memorable (like hostnames), and distributed -- sites no longer had to download a hosts file.

Domain names quickly became a feature of e-mail addresses -- replacing the older bang path notation -- as well as other services. Many years later, they would become the central part of the World Wide Web's URLs, for which see below.

## 4.7. Standards and control

The Internet has developed a significant subculture dedicated to the idea that the Internet is not owned or controlled by any one person, company, group, or organization. Nevertheless, some standardization and control is necessary for anything to function.

Many people wanted to put their ideas into the standards for communication between the computers that made up this network, so a system was devised for putting forward ideas. One would write one's

ideas in a paper called a "Request for Comments" (RFC for short), and let everyone else read it. People commented on and improved those ideas in new RFCs.

With its basis as an educational research project, much of the documentation was written by students or others who played significant roles in developing the network (as part of the original Network Working Group) but did not have official responsibility for defining standards. This is the reason for the very low-key name of "Request for Comments" rather than something like "Declaration of Official Standards".

The first RFC (RFC1) was written on April 7th, 1969. As of 2004 there are over 3500 RFCs, describing every aspect of how the Internet functions.

The liberal RFC publication procedure has engendered confusion about the Internet standardization process, and has led to more formalization of official accepted standards. Acceptance of an RFC by the RFC Editor for publication does not automatically make the RFC into a standard. It may be recognized as such by the IETF only after many years of experimentation, use, and acceptance have proven it to be worthy of that designation. Official standards are numbered with a prefix "STD" and a number, similar to the RFC naming style. However, even after becoming a standard, most are still commonly referred to by their RFC number.

The Internet standards process has been as innovative as the Internet technology. Prior to the Internet, standardization was a slow process run by committees with arguing vendor-driven factions and lengthy delays. In networking in particular, the results were monstrous patchworks of bloated specifications.

The fundamental requirement for a networking protocol to become an Internet standard is the existence of at least two existing, working implementations that inter-operate with each other. This makes sense in retrospect, but it was a new concept at the time. Other efforts built huge specifications with many optional parts and then expected people to go off and implement them, and only later did people find that they did not inter-operate, or worse, the standard was completely impossible to implement.

In the 1980s, the International Organization for Standardization (ISO) documented a new effort in networking called Open Systems Interconnect or OSI. Prior to OSI, networking was completely vendor-developed and proprietary. OSI was a new industry effort, attempting to get everyone to agree to common network standards to provide multi-vendor interoperability. The OSI model was the most important advance in teaching network concepts. However, the OSI protocols or "stack" that were specified as part of the project were a bloated mess. Standards like X.400 for e-mail took up several large books, while Internet e-mail took only a few dozen pages at most in RFC-821 and 822. Most protocols and specifications in the OSI stack, such as token-bus media, CLNP packet delivery, FTAM file transfer, and X.400 e-mail, are long-gone today; in 1996, ISO finally acknowledged that TCP/IP had won and killed the OSI project. Only one OSI standard, X.500 directory service, still survives with significant usage, mainly because the original unwieldy protocol has been stripped away and effectively replaced with LDAP.

Some formal organization is necessary to make everything operate. The first central authority was the NIC (Network Information Centre) at SRI (Stanford Research Institute in Menlo Park, California).