PC Hardware

Contents

1 Computer hardware **1**

1.1 Von Neumann architecture . . . 1

1.2 Sales . . . 1

1.3 Different systems . . . 2

1.3.1 Personal computer . . . 2

1.3.2 Mainframe computer . . . 3

1.3.3 Departmental computing . . . 4

1.3.4 Supercomputer . . . 4

1.4 See also . . . 4

1.5 References . . . 4

1.6 External links . . . 4

2 Central processing unit **5**

2.1 History . . . 5

2.1.1 Transistor and integrated circuit CPUs . . . 6

2.1.2 Microprocessors . . . 7

2.2 Operation . . . 8

2.2.1 Fetch . . . 8

2.2.2 Decode . . . 8

2.2.3 Execute . . . 9

2.3 Design and implementation . . . 9

2.3.1 Control unit . . . 9

2.3.2 Arithmetic logic unit . . . 9

2.3.3 Integer range . . . 10

2.3.4 Clock rate . . . 10

2.3.5 Parallelism . . . 11

2.4 Performance . . . 13

2.5 See also . . . 14

2.6 Notes . . . 14

2.7 References . . . 15

2.8 External links . . . 15

3 Chipset **16**

3.1 Computers . . . 16
3.2 Move toward processor integration in PCs . . . 17
3.3 See also . . . 17
3.4 Notes . . . 17

4 Graphics processing unit 18
4.1 History . . . 18
4.1.1 1980s . . . 18
4.1.2 1990s . . . 19
4.1.3 2000 to 2006 . . . 20
4.1.4 2006 to present . . . 20
4.1.5 GPU companies . . . 20
4.2 Computational functions . . . 21
4.2.1 GPU accelerated video decoding . . . 21
4.3 GPU forms . . . 21
4.3.1 Dedicated graphics cards . . . 21
4.3.2 Integrated graphics solutions . . . 22
4.3.3 Hybrid solutions . . . 22
4.3.4 Stream Processing and General Purpose GPUs (GPGPU) . . . 23
4.3.5 External GPU (eGPU) . . . 23
4.4 Sales . . . 23
4.5 See also . . . 24
4.5.1 Hardware . . . 24
4.5.2 APIs . . . 24
4.5.3 Applications . . . 24
4.6 References . . . 24
4.7 External links . . . 25

5 Random-access memory 26
5.1 History . . . 26
5.2 Types of RAM . . . 27
5.3 Memory hierarchy . . . 28
5.4 Other uses of RAM . . . 28
5.4.1 Virtual memory . . . 28
5.4.2 RAM disk . . . 28
5.4.3 Shadow RAM . . . 28
5.5 Recent developments . . . 28
5.6 Memory wall . . . 29
5.7 See also . . . 29
5.8 Notes and references . . . 29
5.9 External links . . . 30

6 Read-only memory **31**
6.1 History . . . 31
6.1.1 Use for storing programs . . . 32
6.1.2 Use for storing data . . . 33
6.2 Types . . . 33
6.2.1 Semiconductor based . . . 33
6.2.2 Other technologies . . . 34
6.3 Speed . . . 34
6.3.1 Reading . . . 34
6.3.2 Writing . . . 34
6.4 Endurance and data retention . . . 35
6.5 Content images . . . 35
6.6 See also . . . 35
6.7 Terminology . . . 35
6.8 References . . . 35

7 BIOS **36**
7.1 History . . . 36
7.2 User interface . . . 36
7.3 Operation . . . 37
7.3.1 System startup . . . 37
7.3.2 Boot process . . . 37
7.3.3 Boot environment . . . 38
7.4 Extensions (option ROMs) . . . 39
7.4.1 Boot procedure . . . 39
7.4.2 Initialization . . . 40
7.4.3 Physical placement . . . 40
7.5 Operating system services . . . 40
7.5.1 Processor microcode updates . . . 41
7.5.2 Identification . . . 41
7.5.3 Overclocking . . . 41
7.5.4 Modern use . . . 41
7.6 Configuration . . . 41
7.6.1 Setup utility . . . 41
7.6.2 Reprogramming . . . 42
7.7 Hardware . . . 42
7.8 Vendors and products . . . 43
7.9 Security . . . 43
7.10 Alternatives and successors . . . 45
7.11 See also . . . 45
7.12 Notes . . . 45
7.13 References . . . 45

7.14 Further reading . . . 46
7.15 External links . . . 46

8 Bootstrapping **47**
8.1 Etymology . . . 47
8.2 Applications . . . 47
8.2.1 Computing . . . 47
8.2.2 Research . . . 49
8.2.3 Statistics . . . 49
8.2.4 Business . . . 49
8.2.5 Biology . . . 49
8.2.6 Law . . . 49
8.2.7 Linguistics . . . 49
8.2.8 Physics . . . 49
8.2.9 Electronics . . . 49
8.2.10 Electric power grid . . . 50
8.2.11 Cellular networks . . . 50
8.2.12 Media . . . 50
8.3 See also . . . 50
8.4 References . . . 50
8.5 External links . . . 51

9 Firmware **52**
9.1 Origin of the term . . . 52
9.2 Personal computers . . . 52
9.3 Consumer products . . . 53
9.4 Automobiles . . . 53
9.5 Examples . . . 53
9.6 Flashing . . . 54
9.7 Firmware hacking . . . 54
9.7.1 HDD firmware hacks . . . 54
9.8 Security risks . . . 54
9.9 See also . . . 55
9.10 References . . . 55
9.11 External links . . . 55

10 Unified Extensible Firmware Interface **56**
10.1 History . . . 56
10.2 Advantages . . . 56
10.3 Compatibility . . . 57
10.3.1 Processor compatibility . . . 57
10.3.2 Disk device compatibility . . . 57

10.4 Features . . . 58
10.4.1 Services . . . 58
10.4.2 Applications . . . 58
10.4.3 Protocols . . . 58
10.4.4 Device drivers . . . 58
10.4.5 Graphics features . . . 59
10.4.6 EFI System partition . . . 59
10.4.7 Booting . . . 59
10.4.8 Compatibility Support Module . . . 60
10.4.9 UEFI shell . . . 60
10.4.10 Extensions . . . 60
10.5 Implementation and adoption . . . 60
10.5.1 Intel EFI . . . 60
10.5.2 Platforms using EFI/UEFI . . . 60
10.5.3 Operating systems . . . 61
10.5.4 Use of UEFI with virtualization . . . 62
10.6 Applications development . . . 62
10.7 Criticism . . . 62
10.7.1 Secure boot . . . 62
10.7.2 Firmware issues . . . 63
10.8 See also . . . 63
10.9 Notes . . . 63
10.10 References . . . 64
10.11 Further reading . . . 66
10.12 External links . . . 66

11 Bus (computing) 67
11.1 Background and nomenclature . . . 67
11.1.1 Internal bus . . . 68
11.1.2 External bus . . . 68
11.2 Implementation details . . . 68
11.3 History . . . 68
11.3.1 First generation . . . 69
11.3.2 Minis and micros . . . 69
11.3.3 Second generation . . . 70
11.3.4 Third generation . . . 70
11.4 Examples of internal computer buses . . . 70
11.4.1 Parallel . . . 70
11.4.2 Serial . . . 71
11.5 Examples of external computer buses . . . 71
11.5.1 Parallel . . . 71
11.5.2 Serial . . . 71

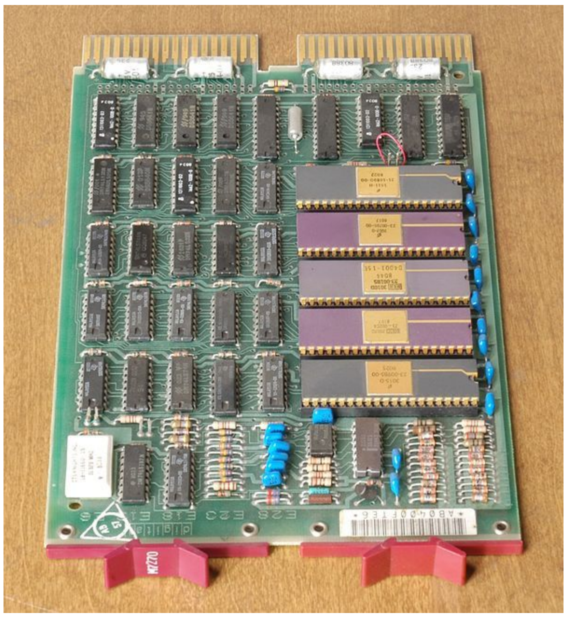
11.6 Examples of internal/external computer buses . . . 71
11.7 See also . . . 71
11.8 References . . . 72
11.9 External links . . . 72
11.10 Text and image sources, contributors, and licenses . . . 73
11.10.1 Text . . . 73
11.10.2 Images . . . 78
11.10.3 Content license . . . 81

Chapter 1

Computer hardware

For other uses, see Hardware.

Computer hardware (usually simply called **hardware** when a computing context is implicit) is the collection of physical elements that constitutes a computer system. Computer hardware is the physical parts or components of a computer, such as the monitor, mouse, keyboard, computer data storage, hard disk drive (HDD), system unit (graphic cards, sound cards, memory, motherboard and chips), and so on, all of which are physical objects that can be touched (that is, they are tangible).[1] In contrast, software is instructions that can be stored and run by hardware.

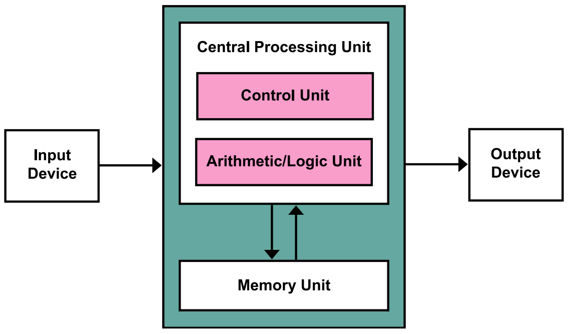

PDP-11 CPU board

Software is any set of machine-readable instructions that directs a computer's processor to perform specific operations. A combination of hardware and software forms a usable computing system.[2]

1.1 Von Neumann architecture

Main article: Von Neumann architecture

The template for all modern computers is the Von Neumann architecture, detailed in a 1945 paper by Hungarian mathematician John von Neumann. This describes a design architecture for an electronic digital computer with subdivisions of a processing unit consisting of an arithmetic logic unit and processor registers, a control unit containing an instruction register and program counter, a memory to store both data and instructions, external mass storage, and input and output mechanisms.[3] The meaning of the term has evolved to mean a stored-program computer in which an instruction fetch and a data operation cannot occur at the same time because they share a common bus. This is referred to as the Von Neumann bottleneck and often limits the performance of the system.[4]



Von Neumann architecture scheme.

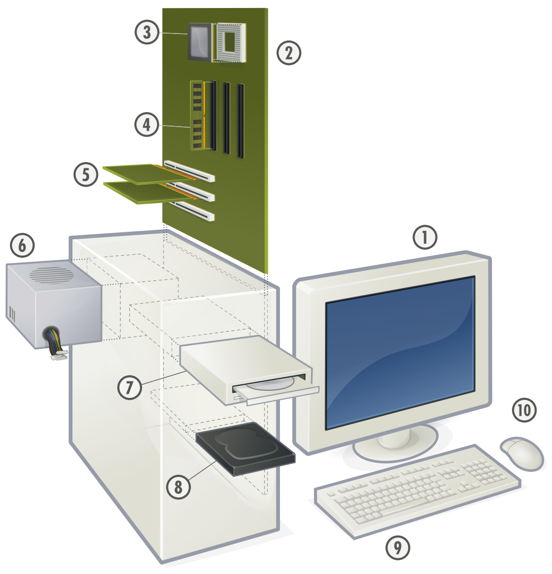
1.2 Sales

For the third consecutive year, U.S. business-to-business channel sales (sales through distributors and commercial resellers) increased, ending 2013 up nearly 6 percent at $1 doller. The impressive growth was the fastest sales increase since the end of the recession. Sales growth accelerated in the second half of the year peaking in fourth quarter with a 6.9 percent increase over the fourth quarter of 2012.[5]

1.3 Different systems

There are a number of different types of computer system in use today.

1.3.1 Personal computer



Hardware of a modern personal computer
1. Monitor 2. Motherboard 3. CPU 4. RAM 5. Expansion cards 6. Power supply 7. Optical disc drive 8. Hard disk drive 9. Keyboard 10. Mouse

Inside a custom-built computer: power supply at the bottom has its own cooling fan.

The personal computer, also known as the PC, is one of the most common types of computer due to its versatility and relatively low price. Laptops are generally very similar, although may use lower-power or reduced size components.

Case

Main article: Computer case

The computer case is a plastic or metal enclosure that houses most of the components. Those found on desktop computers are usually small enough to fit under a desk, however in recent years more compact designs have become more common place, such as the all-in-one style designs from Apple, namely the iMac. Laptops are computers that usually come in a clamshell form factor, again however in more recent years deviations from this form factor have started to emerge such as laptops that have a detachable screen that become tablet computers in their own right.

Power supply

Main article: Power supply unit (computer)

A power supply unit (PSU) converts alternating current (AC) electric power to low-voltage DC power for the internal components of the computer. Laptops are capable of running from a built-in battery, normally for a period of hours.[6]

Motherboard

Main article: Motherboard

The motherboard is the main component of computer. It is a large rectangular board with integrated circuitry that connects the other parts of the computer including the CPU, the RAM, the disk drives(CD, DVD, hard disk, or any others) as well as any peripherals connected via the ports or the expansion slots.

Components directly attached to or part of the motherboard include:

- The **CPU** (Central Processing Unit) performs most of the calculations which enable a computer to function, and is sometimes referred to as the “brain” of the computer. It is usually cooled by a heat sink and fan. Most newer CPUs include an on-die Graphics Processing Unit (GPU).
- The **Chipset**, which includes the north bridge, mediates communication between the CPU and the other components of the system, including main memory.

- The **Random-Access Memory** (RAM) stores the code and data that are being actively accessed by the CPU.
- The **Read-Only Memory** (ROM) stores the BIOS that runs when the computer is powered on or otherwise begins execution, a process known as Bootstrapping, or "booting" or “booting up”. The **BIOS** (Basic Input Output System) includes boot firmware and power management firmware. Newer motherboards use Unified Extensible Firmware Interface (UEFI) instead of BIOS.
- **Buses** connect the CPU to various internal components and to expansion cards for graphics and sound.
- The CMOS battery is also attached to the motherboard. This battery is the same as a watch battery or a battery for a remote to a car’s central locking system. Most batteries are CR2032, which powers the memory for date and time in the BIOS chip.

Expansion cards

Main article: Expansion card

An expansion card in computing is a printed circuit board that can be inserted into an expansion slot of a computer motherboard or backplane to add functionality to a computer system via the expansion bus.

Storage devices

Main article: Computer data storage

Computer data storage, often called storage or memory, refers to computer components and recording media that retain digital data. Data storage is a core function and fundamental component of computers.

Fixed media

Data is stored by a computer using a variety of media. Hard disk drives are found in virtually all older computers, due to their high capacity and low cost, but solid-state drives are faster and more power efficient, although currently more expensive than hard drives, so are often found in more expensive computers. Some systems may use a disk array controller for greater performance or reliability.

Removable media

To transfer data between computers, a USB flash drive or Optical disc may be used. Their usefulness depends on being readable by other systems; the majority of machines have an optical disk drive, and virtually all have a USB port.

Input and output peripherals

Main article: Peripheral

Input and output devices are typically housed externally to the main computer chassis. The following are either standard or very common to many computer systems.

Input

Input devices allow the user to enter information into the system, or control its operation. Most personal computers have a mouse and keyboard, but laptop systems typically use a touchpad instead of a mouse. Other input devices include webcams, microphones, joysticks, and image scanners.

Output device

Output devices display information in a human readable form. Such devices could include printers, speakers, monitors or a Braille embosser.

1.3.2 Mainframe computer

An IBM System z9 mainframe

A mainframe computer is a much larger computer that typically fills a room and may cost many hundreds or thousands of times as much as a personal computer. They are designed to perform large numbers of calculations for governments and large enterprises.

1.3.3 Departmental computing

In the 1960s and 1970s more and more departments started to use cheaper and dedicated systems for specific purposes like process control and laboratory automation.

Main article: Minicomputer

1.3.4 Supercomputer

A supercomputer is superficially similar to a mainframe, but is instead intended for extremely demanding computational tasks. As of November 2013, the fastest supercomputer in the world is the Tianhe-2, in Guangzhou, China.[7]

The term supercomputer does not refer to a specific technology. Rather it indicates the fastest computers available at any given time. In mid 2011, the fastest supercomputers boasted speeds exceeding one petaflop, or 1000 trillion floating point operations per second. Super computers are fast but extremely costly so they are generally used by large organizations to execute computationally demanding tasks involving large data sets. Super computers typically run military and scientific applications. Although they cost millions of dollars, they are also being used for commercial applications where huge amounts of data must be analyzed. For example, large banks employ supercomputers to calculate the risks and returns of various investment strategies, and healthcare organizations use them to analyze giant databases of patient data to determine optimal treatments for various diseases and problems incurring to our country.

1.4 See also

- Open-source computing hardware

1.5 References

[1] "Parts of computer". Microsoft. Retrieved 5 December 2013.

[2] Smither, Roger. "Use of computers in audiovisual archives". UNESCO. Retrieved 5 December 2013.

[3] von Neumann, John (1945). "First Draft of a Report on the EDVAC".

[4] Markgraf, Joey D. (2007). "The Von Neumann bottleneck". Retrieved 24 August 2011.

[5] US B2B Channel sales reach nearly $62 Billion in 2013, by The NPD Group: https://www.npd.com/wps/portal/npd/us/news/press-releases/us-b2bchannel-sales-reach-nearly-62-billion-in-2013-according-to-the-npd-group/

[6] "How long should a laptop battery last?". Computer Hope. Retrieved 9 December 2013.

[7] Alba, Davey. "China's Tianhe-2 Caps Top 10 Supercomputers". IEEE. Retrieved 9 December 2013.

1.6 External links

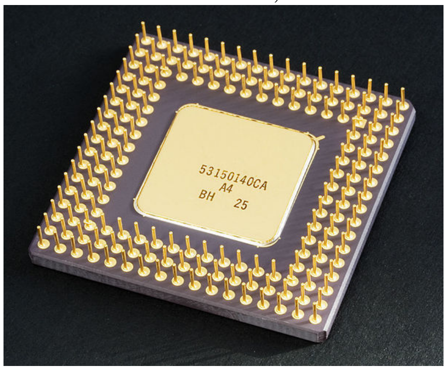
- Media related to Computer hardware at Wikimedia Commons
- Learning materials related to Computer hardware at Wikiversity
- "What You Need to Know About Hardware Beta Tests" Centercode.com

Chapter 2

Central processing unit

“CPU” redirects here. For other uses, see CPU (disambiguation).
“Computer processor” redirects here. For other uses, see Processor (computing).

An Intel 80486DX2 CPU, as seen from above



Bottom side of an Intel 80486DX2

A **central processing unit** (**CPU**) is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions. The term has been used in the computer industry at least since the early 1960s.[1] Traditionally, the term “CPU” refers to a processor and its control unit (CU), distinguishing these core elements of a computer from external components such as main memory and I/O circuitry.[2]

The form, design and implementation of CPUs have changed over the course of their history, but their fundamental operation remains almost unchanged. Principal components of a CPU include the arithmetic logic unit (ALU) that performs arithmetic and logic operations, hardware registers that supply operands to the ALU and store the results of ALU operations, and a control unit that fetches instructions from memory and “executes” them by directing the coordinated operations of the ALU, registers and other components.

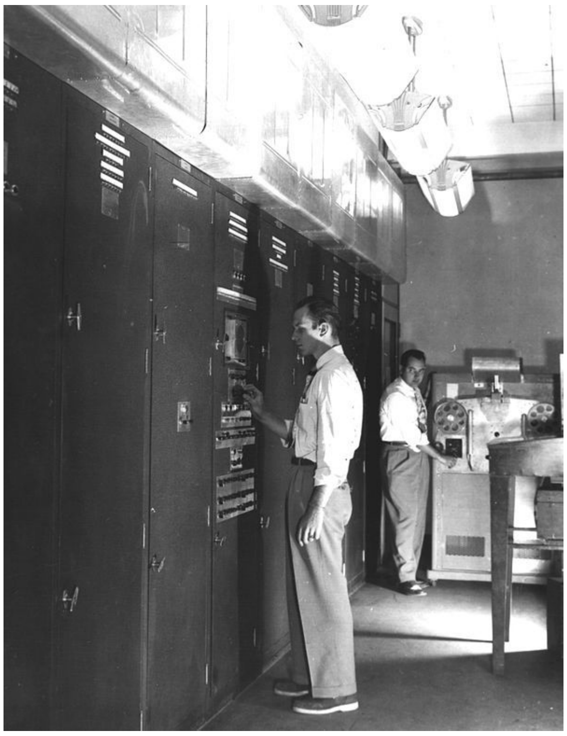
Most modern CPUs are microprocessors, meaning they are contained on a single integrated circuit (IC) chip. An IC that contains a CPU may also contain memory, peripheral interfaces, and other components of a computer; such integrated devices are variously called microcontrollers or systems on a chip (SoC). Some computers employ a multi-core processor, which is a single chip containing two or more CPUs called “cores"; in that context, single chips are sometimes referred to as “sockets”.[3] Array processors or vector processors have multiple processors that operate in parallel, with no unit considered central.

2.1 History

Main article: History of general purpose CPUs

Computers such as the ENIAC had to be physically rewired to perform different tasks, which caused these machines to be called “fixed-program computers”.[4] Since the term “CPU” is generally defined as a device for software (computer program) execution, the earliest devices that could rightly be called CPUs came with the advent of the stored-program computer.

The idea of a stored-program computer was already present in the design of J. Presper Eckert and John William Mauchly's ENIAC, but was initially omitted so that it could be finished sooner. On June 30, 1945, before ENIAC was made, mathematician John von Neumann distributed the paper entitled *First Draft of a Report on the EDVAC*. It was the outline of a stored-program computer that would eventually be completed in August 1949.[5] EDVAC was designed to perform a certain number of instructions (or operations) of various types. Significantly, the programs written for EDVAC were to be stored in high-speed computer memory rather than specified by the physical wiring of the computer. This over-



EDVAC, one of the first stored-program computers.

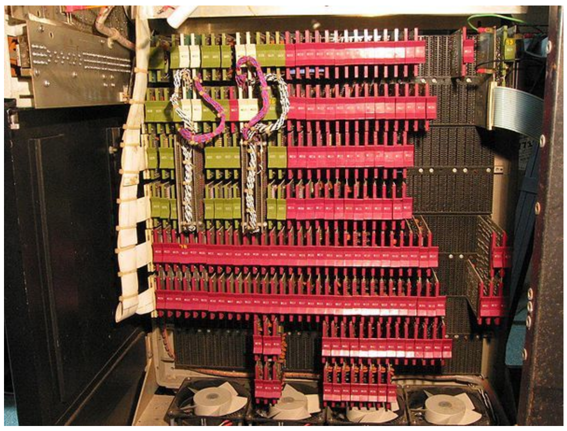
came a severe limitation of ENIAC, which was the considerable time and effort required to reconfigure the computer to perform a new task. With von Neumann's design, the program, or software, that EDVAC ran could be changed simply by changing the contents of the memory. EDVAC, however, was not the first stored-program computer; the Manchester Small-Scale Experimental Machine, a small prototype stored-program computer, ran its first program on 21 June 1948[6] and the Manchester Mark 1 ran its first program during the night of 16–17 June 1949.

Early CPUs were custom-designed as a part of a larger, sometimes one-of-a-kind, computer. However, this method of designing custom CPUs for a particular application has largely given way to the development of mass-produced processors that are made for many purposes. This standardization began in the era of discrete transistor mainframes and minicomputers and has rapidly accelerated with the popularization of the integrated circuit (IC). The IC has allowed increasingly complex CPUs to be designed and manufactured to tolerances on the order of nanometers. Both the miniaturization and standardization of CPUs have increased the presence of digital devices in modern life far beyond the limited application of dedicated computing machines. Modern microprocessors appear in everything from automobiles to cell phones and children's toys.

While von Neumann is most often credited with the design of the stored-program computer because of his design of EDVAC, others before him, such as Konrad Zuse, had suggested and implemented similar ideas. The so-called Harvard architecture of the Harvard Mark I, which was completed before EDVAC, also utilized a stored-program design using punched paper tape rather than electronic memory. The key difference between the von Neumann and Harvard architectures is that the latter separates the storage and treatment of CPU instructions and data, while the former uses the same memory space for both. Most modern CPUs are primarily von Neumann in design, but CPUs with the Harvard architecture are seen as well, especially in embedded applications; for instance, the Atmel AVR microcontrollers are Harvard architecture processors.

Relays and vacuum tubes (thermionic valves) were commonly used as switching elements; a useful computer requires thousands or tens of thousands of switching devices. The overall speed of a system is dependent on the speed of the switches. Tube computers like EDVAC tended to average eight hours between failures, whereas relay computers like the (slower, but earlier) Harvard Mark I failed very rarely.[1] In the end, tube-based CPUs became dominant because the significant speed advantages afforded generally outweighed the reliability problems. Most of these early synchronous CPUs ran at low clock rates compared to modern microelectronic designs (see below for a discussion of clock rate). Clock signal frequencies ranging from 100 kHz to 4 MHz were very common at this time, limited largely by the speed of the switching devices they were built with.

2.1.1 Transistor and integrated circuit CPUs



CPU, core memory, and external bus interface of a DEC PDP-8/I. Made of medium-scale integrated circuits.

The design complexity of CPUs increased as various technologies facilitated building smaller and more reliable electronic devices. The first such improvement came with the advent of the transistor. Transistorized CPUs during the 1950s and 1960s no longer had to be built out of bulky, unreliable, and fragile switching elements like vacuum tubes and electrical relays. With this improvement more complex and reliable CPUs were built onto one or several printed circuit boards containing discrete (individual) components.

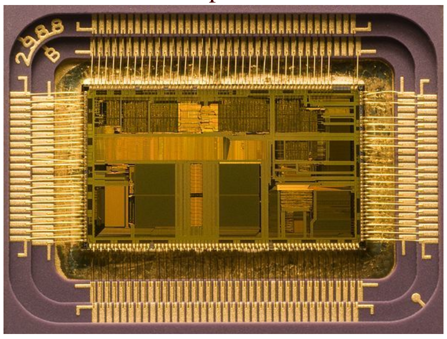
During this period, a method of manufacturing many interconnected transistors in a compact space was developed. The integrated circuit (IC) allowed a large number of transistors to be manufactured on a single semiconductor-based die, or “chip”. At first only very basic non-specialized digital circuits such as NOR gates were miniaturized into ICs. CPUs based upon these “building block” ICs are generally referred to as “small-scale integration” (SSI) devices. SSI ICs, such as the ones used in the Apollo guidance computer, usually contained up to a few score transistors. To build an entire CPU out of SSI ICs required thousands of individual chips, but still consumed much less space and power than earlier discrete transistor designs. As microelectronic technology advanced, an increasing number of transistors were placed on ICs, thus decreasing the quantity of individual ICs needed for a complete CPU. MSI and LSI (medium- and large-scale integration) ICs increased transistor counts to hundreds, and then thousands.

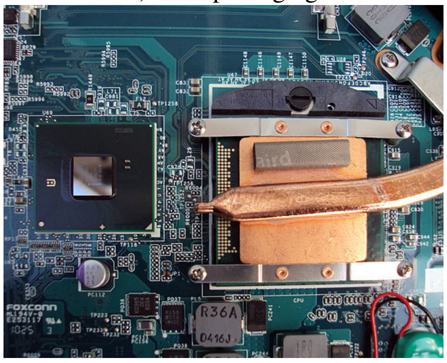
In 1964, IBM introduced its System/360 computer architecture that was used in a series of computers capable of running the same programs with different speed and performance. This was significant at a time when most electronic computers were incompatible with one another, even those made by the same manufacturer. To facilitate this improvement, IBM utilized the concept of a microprogram (often called “microcode”), which still sees widespread usage in modern CPUs.[7] The System/360 architecture was so popular that it dominated the mainframe computer market for decades and left a legacy that is still continued by similar modern computers like the IBM zSeries. In the same year (1964), Digital Equipment Corporation (DEC) introduced another influential computer aimed at the scientific and research markets, the PDP-8. DEC would later introduce the extremely popular PDP-11 line that originally was built with SSI ICs but was eventually implemented with LSI components once these became practical. In stark contrast with its SSI and MSI predecessors, the first LSI implementation of the PDP-11 contained a CPU composed of only four LSI integrated circuits.[8]

Transistor-based computers had several distinct advantages over their predecessors. Aside from facilitating increased reliability and lower power consumption, transistors also allowed CPUs to operate at much higher speeds because of the short switching time of a transistor in comparison to a tube or relay. Thanks to both the increased reliability as well as the dramatically increased speed of the switching elements (which were almost exclusively transistors by this time), CPU clock rates in the tens of megahertz were obtained during this period. Additionally while discrete transistor and IC CPUs were in heavy usage, new high-performance designs like SIMD (Single Instruction Multiple Data) vector processors began to appear. These early experimental designs later gave rise to the era of specialized supercomputers like those made by Cray Inc.

2.1.2 Microprocessors

Main article: Microprocessor


Die of an Intel 80486DX2 microprocessor (actual size: 12×6.75 mm) in its packaging


Intel Core i5 CPU on a Vaio E series laptop motherboard (on the right, beneath the heat pipe).

In the 1970s the fundamental inventions by Federico Faggin (Silicon Gate MOS ICs with self-aligned gates along with his new random logic design methodology) changed the design and implementation of CPUs forever. Since the introduction of the first commercially available microprocessor (the Intel 4004) in 1970, and the first widely used microprocessor (the Intel 8080) in 1974, this class of CPUs has almost completely overtaken all other central processing unit implementation methods. Mainframe and minicomputer manufacturers of the time launched proprietary IC development programs to upgrade their older computer architectures, and eventually produced instruction set compatible microprocessors that were backward-compatible with their older hardware and software. Combined with the advent and eventual success of the ubiquitous personal computer, the term *CPU* is now applied almost exclusively[lower-alpha 1] to microprocessors. Several CPUs (denoted 'cores’) can be combined in a single processing chip.

Previous generations of CPUs were implemented as discrete components and numerous small integrated circuits (ICs) on one or more circuit boards. Microprocessors, on the other hand, are CPUs manufactured on a very small number of ICs; usually just one. The overall smaller

CPU size, as a result of being implemented on a single die, means faster switching time because of physical factors like decreased gate parasitic capacitance. This has allowed synchronous microprocessors to have clock rates ranging from tens of megahertz to several gigahertz. Additionally, as the ability to construct exceedingly small transistors on an IC has increased, the complexity and number of transistors in a single CPU has increased many fold. This widely observed trend is described by Moore's law, which has proven to be a fairly accurate predictor of the growth of CPU (and other IC) complexity.[9]

While the complexity, size, construction, and general form of CPUs have changed enormously since 1950, it is notable that the basic design and function has not changed much at all. Almost all common CPUs today can be very accurately described as von Neumann stored-program machines.[lower-alpha 2] As the aforementioned Moore's law continues to hold true,[9] concerns have arisen about the limits of integrated circuit transistor technology. Extreme miniaturization of electronic gates is causing the effects of phenomena like electromigration and subthreshold leakage to become much more significant. These newer concerns are among the many factors causing researchers to investigate new methods of computing such as the quantum computer, as well as to expand the usage of parallelism and other methods that extend the usefulness of the classical von Neumann model.

2.2 Operation

The fundamental operation of most CPUs, regardless of the physical form they take, is to execute a sequence of stored instructions called a program. The instructions are kept in some kind of computer memory. There are three steps that nearly all CPUs use in their operation: fetch, decode, and execute.

After the execution of an instruction, the entire process repeats, with the next instruction cycle normally fetching the next-in-sequence instruction because of the incremented value in the program counter. If a jump instruction was executed, the program counter will be modified to contain the address of the instruction that was jumped to and program execution continues normally. In more complex CPUs, multiple instructions can be fetched, decoded, and executed simultaneously. This section describes what is generally referred to as the "classic RISC pipeline", which is quite common among the simple CPUs used in many electronic devices (often called microcontroller). It largely ignores the important role of CPU cache, and therefore the access stage of the pipeline.

Some instructions manipulate the program counter rather than producing result data directly; such instructions are generally called "jumps" and facilitate program behavior like loops, conditional program execution (through the use of a conditional jump), and existence of functions.[lower-alpha 3] In some processors, some other instructions change the state of bits in a "flags" register. These flags can be used to influence how a program behaves, since they often indicate the outcome of various operations. For example, in such processors a "compare" instruction evaluates two values and sets or clears bits in the flags register to indicate which one is greater or whether they are equal; one of these flags could then be used by a later jump instruction to determine program flow.

2.2.1 Fetch

The first step, fetch, involves retrieving an instruction (which is represented by a number or sequence of numbers) from program memory. The instruction's location (address) in program memory is determined by a program counter (PC), which stores a number that identifies the address of the next instruction to be fetched. After an instruction is fetched, the PC is incremented by the length of the instruction so that it will contain the address of the next instruction in the sequence.[lower-alpha 4] Often, the instruction to be fetched must be retrieved from relatively slow memory, causing the CPU to stall while waiting for the instruction to be returned. This issue is largely addressed in modern processors by caches and pipeline architectures (see below).

2.2.2 Decode

The instruction that the CPU fetches from memory determines what the CPU has to do. In the decode step, the instruction is broken up into parts that have significance to other portions of the CPU. The way in which the numerical instruction value is interpreted is defined by the CPU's instruction set architecture (ISA).[lower-alpha 5] Often, one group of numbers in the instruction, called the opcode, indicates which operation to perform. The remaining parts of the number usually provide information required for that instruction, such as operands for an addition operation. Such operands may be given as a constant value (called an immediate value), or as a place to locate a value: a register or a memory address, as determined by some addressing mode.

In some CPU designs the instruction decoder is implemented as a hardwired, unchangeable circuit. In others, a microprogram is used to translate instructions into sets of CPU configuration signals that are applied sequentially over multiple clock pulses. In some cases the memory that stores the microprogram is rewritable, making it possible to change the way in which the CPU decodes instructions.

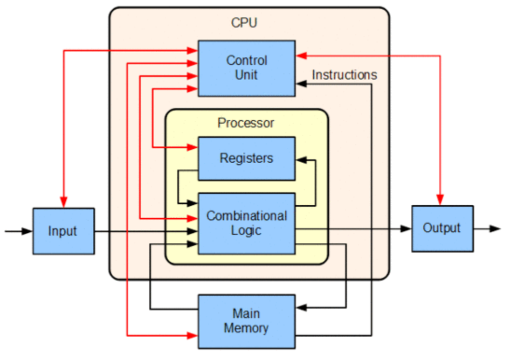
2.2.3 Execute

After the fetch and decode steps, the execute step is performed. Depending on the CPU architecture, this may consist of a single action or a sequence of actions. During each action, various parts of the CPU are electrically connected so they can perform all or part of the desired operation and then the action is completed, typically in response to a clock pulse. Very often the results are written to an internal CPU register for quick access by subsequent instructions. In other cases results may be written to slower, but less expensive and higher capacity main memory.

For example, if an addition instruction is to be executed, the arithmetic logic unit (ALU) inputs are connected to a pair of operand sources (numbers to be summed), the ALU is configured to perform an addition operation so that the sum of its operand inputs will appear at its output, and the ALU output is connected to storage (e.g., a register or memory) that will receive the sum. When the clock pulse occurs, the sum will be transferred to storage and, if the resulting sum is too large (i.e., it is larger than the ALU's output word size), an arithmetic overflow flag will be set.

2.3 Design and implementation

Main article: CPU design
Hardwired into a CPU's circuitry is a set of basic operations it can perform, called an instruction set. Such operations may involve, for example, adding or subtracting two numbers, comparing two numbers, or jumping to a different part of a program. Each basic operation is represented by a particular combination of bits, known as the machine language opcode; while executing instructions in a machine language program, the CPU decides which operation to perform by "decoding" the opcode.



Block diagram of a basic uniprocessor-CPU computer. Black lines indicate data flow, whereas red lines indicate control flow; arrows indicate flow directions.

A complete machine language instruction consists of an opcode and, in many cases, additional bits that specify arguments for the operation (for example, the numbers to be summed in the case of an addition operation). Going up the complexity scale, a machine language program is a collection of machine language instructions that the CPU executes.

The actual mathematical operation for each instruction is performed by a combinational logic circuit within the CPU's processor known as the arithmetic logic unit or ALU. In general, a CPU executes an instruction by fetching it from memory, using its ALU to perform an operation, and then storing the result to memory. Beside the instructions for integer mathematics and logic operations, various other machine instructions exists, such as those for loading data from memory and storing it back, branching operations, and mathematical operations on floating-point numbers performed by the CPU's floating-point unit (FPU).[10]

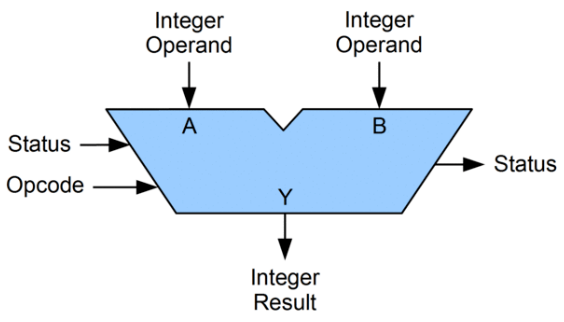
2.3.1 Control unit

Main article: Control unit

The control unit of the CPU contains circuitry that uses electrical signals to direct the entire computer system to carry out stored program instructions. The control unit does not execute program instructions; rather, it directs other parts of the system to do so. The control unit communicates with both the ALU and memory.

2.3.2 Arithmetic logic unit

Main article: Arithmetic logic unit
The arithmetic logic unit (ALU) is a digital circuit within the processor that performs integer arithmetic and bitwise logic operations. The inputs to the ALU are the data words to be operated on (called operands), status information from previous operations, and a code from the control unit indicating which operation to perform. Depending on the instruction being executed, the operands

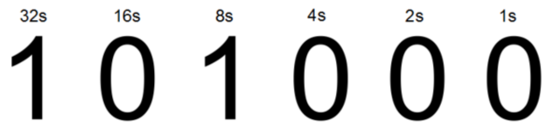


Symbolic representation of an ALU and its input and output signals may come from internal CPU registers or external memory, or they may be constants generated by the ALU itself.

When all input signals have settled and propagated through the ALU circuitry, the result of the performed operation appears at the ALU's outputs. The result consists of both a data word, which may be stored in a register or memory, and status information that is typically stored in a special, internal CPU register reserved for this purpose.

2.3.3 Integer range

Every CPU represents numerical values in a specific way. For example, some early digital computers represented numbers as familiar decimal (base 10) numeral system values, and others have employed more unusual representations such as ternary (base three). Nearly all modern CPUs represent numbers in binary form, with each digit being represented by some two-valued physical quantity such as a "high" or "low" voltage.[lower-alpha 6]



A six-bit word containing the binary encoded representation of decimal value 40. Most modern CPUs employ word sizes that are a power of two, for example eight, 16, 32 or 64 bits.

Related to numeric representation is the size and precision of integer numbers that a CPU can represent. In the case of a binary CPU, this is measured by the number of bits (significant digits of a binary encoded integer) that the CPU can process in one operation, which is commonly called "word size", "bit width", "data path width", "integer precision", or "integer size". A CPU's integer size determines the range of integer values it can directly operate on.[lower-alpha 7] For example, an 8-bit CPU can directly manipulate integers represented by eight bits, which have a range of 256 (2^8) discrete integer values.

Integer range can also affect the number of memory locations the CPU can directly address (an address is an integer value representing a specific memory location). For example, if a binary CPU uses 32 bits to represent a memory address then it can directly address 2^{32} memory locations. To circumvent this limitation and for various other reasons, some CPUs use mechanisms (such as bank switching) that allow additional memory to be addressed.

CPUs with larger word sizes require more circuitry and consequently are physically larger, cost more, and consume more power (and therefore generate more heat). As a result, smaller 4- or 8-bit microcontrollers are commonly used in modern applications even though CPUs with much larger word sizes (such as 16, 32, 64, even 128-bit) are available. When higher performance is required, however, the benefits of a larger word size (larger data ranges and address spaces) may outweigh the disadvantages.

To gain some of the advantages afforded by both lower and higher bit lengths, many CPUs are designed with different bit widths for different portions of the device. For example, the IBM System/370 used a CPU that was primarily 32 bit, but it used 128-bit precision inside its floating point units to facilitate greater accuracy and range in floating point numbers.[7] Many later CPU designs use similar mixed bit width, especially when the processor is meant for general-purpose usage where a reasonable balance of integer and floating point capability is required.

2.3.4 Clock rate

Main article: Clock rate

Most CPUs are synchronous circuits, which means they employ a clock signal to pace their sequential operations. The clock signal is produced by an external oscillator circuit that generates a consistent number of pulses each second in the form of a periodic square wave. The frequency of the clock pulses determines the rate at which a CPU executes instructions and, consequently, the faster the clock, the more instructions the CPU will execute each second.

To ensure proper operation of the CPU, the clock period is longer than the maximum time needed for all signals to propagate (move) through the CPU. In setting the clock period to a value well above the worst-case propagation delay, it is possible to design the entire CPU and the way it moves data around the "edges" of the rising and falling clock signal. This has the advantage of simplifying the CPU significantly, both from a design perspective and a component-count perspective. However, it also carries the disadvantage that the entire CPU must wait on its slowest elements, even though some portions of it are much faster. This limitation has largely been compensated for by various methods of increasing CPU parallelism (see below).

However, architectural improvements alone do not solve all of the drawbacks of globally synchronous CPUs. For example, a clock signal is subject to the delays of any other electrical signal. Higher clock rates in increasingly complex CPUs make it more difficult to keep the clock signal in phase (synchronized) throughout the entire unit. This has led many modern CPUs to require multiple identical clock signals to be provided to avoid delaying a single signal significantly enough to cause the CPU to malfunction. Another major issue, as clock rates increase dramatically, is the amount of heat that is dissipated by the CPU. The constantly changing clock causes many components to switch regardless of whether they are being used at that time. In general, a component that is switching uses more energy than an element in a static state. Therefore, as clock rate increases, so does energy consumption, causing the CPU to require more heat dissipation in the form of CPU cooling solutions.

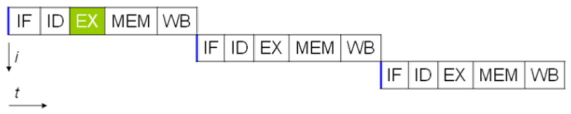
One method of dealing with the switching of unneeded components is called clock gating, which involves turning off the clock signal to unneeded components (effectively disabling them). However, this is often regarded as difficult to implement and therefore does not see common usage outside of very low-power designs. One notable recent CPU design that uses extensive clock gating is the IBM PowerPC-based Xenon used in the Xbox 360; that way, power requirements of the Xbox 360 are greatly reduced.[11] Another method of addressing some of the problems with a global clock signal is the removal of the clock signal altogether. While removing the global clock signal makes the design process considerably more complex in many ways, asynchronous (or clockless) designs carry marked advantages in power consumption and heat dissipation in comparison with similar synchronous designs. While somewhat uncommon, entire asynchronous CPUs have been built without utilizing a global clock signal. Two notable examples of this are the ARM compliant AMULET and the MIPS R3000 compatible MiniMIPS.

Rather than totally removing the clock signal, some CPU designs allow certain portions of the device to be asynchronous, such as using asynchronous ALUs in conjunction with superscalar pipelining to achieve some arithmetic performance gains. While it is not altogether clear whether totally asynchronous designs can perform at a comparable or better level than their synchronous counterparts, it is evident that they do at least excel in simpler math operations. This, combined with their excellent power consumption and heat dissipation properties, makes them very suitable for embedded computers.[12]

2.3.5 Parallelism

Main article: Parallel computing

The description of the basic operation of a CPU offered



Model of a subscalar CPU. Notice that it takes fifteen cycles to complete three instructions.

in the previous section describes the simplest form that a CPU can take. This type of CPU, usually referred to as *subscalar*, operates on and executes one instruction on one or two pieces of data at a time.

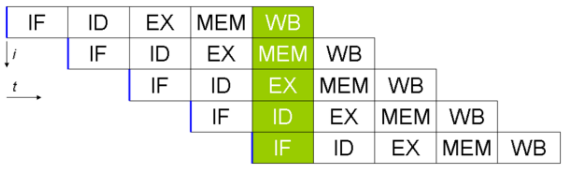
This process gives rise to an inherent inefficiency in subscalar CPUs. Since only one instruction is executed at a time, the entire CPU must wait for that instruction to complete before proceeding to the next instruction. As a result, the subscalar CPU gets “hung up” on instructions which take more than one clock cycle to complete execution. Even adding a second execution unit (see below) does not improve performance much; rather than one pathway being hung up, now two pathways are hung up and the number of unused transistors is increased. This design, wherein the CPU’s execution resources can operate on only one instruction at a time, can only possibly reach *scalar* performance (one instruction per clock). However, the performance is nearly always subscalar (less than one instruction per cycle).

Attempts to achieve scalar and better performance have resulted in a variety of design methodologies that cause the CPU to behave less linearly and more in parallel. When referring to parallelism in CPUs, two terms are generally used to classify these design techniques. Instruction level parallelism (ILP) seeks to increase the rate at which instructions are executed within a CPU (that is, to increase the utilization of on-die execution resources), and thread level parallelism (TLP) purposes to increase the number of threads (effectively individual programs) that a CPU can execute simultaneously. Each methodology differs both in the ways in which they are implemented, as well as the relative effectiveness they afford in increasing the CPU’s performance for an application.[lower-alpha 8]

Instruction-level parallelism

Main articles: Instruction pipelining and Superscalar

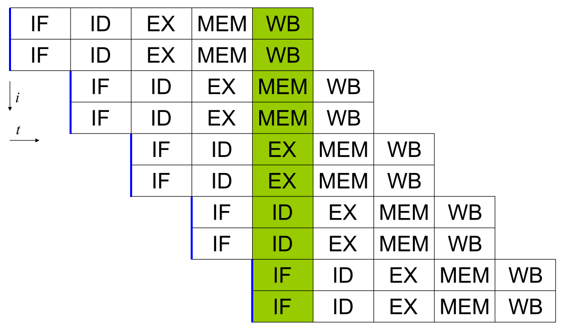
One of the simplest methods used to accomplish in-



Basic five-stage pipeline. In the best case scenario, this pipeline can sustain a completion rate of one instruction per cycle.

creased parallelism is to begin the first steps of instruction fetching and decoding before the prior instruction finishes executing. This is the simplest form of a technique known as instruction pipelining, and is utilized in almost all modern general-purpose CPUs. Pipelining allows more than one instruction to be executed at any given time by breaking down the execution pathway into discrete stages. This separation can be compared to an assembly line, in which an instruction is made more complete at each stage until it exits the execution pipeline and is retired.

Pipelining does, however, introduce the possibility for a situation where the result of the previous operation is needed to complete the next operation; a condition often termed data dependency conflict. To cope with this, additional care must be taken to check for these sorts of conditions and delay a portion of the instruction pipeline if this occurs. Naturally, accomplishing this requires additional circuitry, so pipelined processors are more complex than subscalar ones (though not very significantly so). A pipelined processor can become very nearly scalar, inhibited only by pipeline stalls (an instruction spending more than one clock cycle in a stage).



A simple superscalar pipeline. By fetching and dispatching two instructions at a time, a maximum of two instructions per cycle can be completed.

Further improvement upon the idea of instruction pipelining led to the development of a method that decreases the idle time of CPU components even further. Designs that are said to be *superscalar* include a long instruction pipeline and multiple identical execution units.[13] In a superscalar pipeline, multiple instructions are read and passed to a dispatcher, which decides whether or not the instructions can be executed in parallel (simultaneously). If so they are dispatched to available execution units, resulting in the ability for several instructions to be executed simultaneously. In general, the more instructions a superscalar CPU is able to dispatch simultaneously to waiting execution units, the more instructions will be completed in a given cycle.

Most of the difficulty in the design of a superscalar CPU architecture lies in creating an effective dispatcher. The dispatcher needs to be able to quickly and correctly determine whether instructions can be executed in parallel, as well as dispatch them in such a way as to keep as many execution units busy as possible. This requires that the instruction pipeline is filled as often as possible and gives rise to the need in superscalar architectures for significant amounts of CPU cache. It also makes hazard-avoiding techniques like branch prediction, speculative execution, and out-of-order execution crucial to maintaining high levels of performance. By attempting to predict which branch (or path) a conditional instruction will take, the CPU can minimize the number of times that the entire pipeline must wait until a conditional instruction is completed. Speculative execution often provides modest performance increases by executing portions of code that may not be needed after a conditional operation completes. Out-of-order execution somewhat rearranges the order in which instructions are executed to reduce delays due to data dependencies. Also in case of Single Instructions Multiple Data — a case when a lot of data from the same type has to be processed, modern processors can disable parts of the pipeline so that when a single instruction is executed many times, the CPU skips the fetch and decode phases and thus greatly increases performance on certain occasions, especially in highly monotonous program engines such as video creation software and photo processing.

In the case where a portion of the CPU is superscalar and part is not, the part which is not suffers a performance penalty due to scheduling stalls. The Intel P5 Pentium had two superscalar ALUs which could accept one instruction per clock each, but its FPU could not accept one instruction per clock. Thus the P5 was integer superscalar but not floating point superscalar. Intel's successor to the P5 architecture, P6, added superscalar capabilities to its floating point features, and therefore afforded a significant increase in floating point instruction performance.

Both simple pipelining and superscalar design increase a CPU's ILP by allowing a single processor to complete execution of instructions at rates surpassing one instruction per cycle (IPC).[lower-alpha 9] Most modern CPU designs are at least somewhat superscalar, and nearly all general purpose CPUs designed in the last decade are superscalar. In later years some of the emphasis in designing high-ILP computers has been moved out of the CPU's hardware and into its software interface, or ISA. The strategy of the very long instruction word (VLIW) causes some ILP to become implied directly by the software, reducing the amount of work the CPU must perform to boost ILP and thereby reducing the design's complexity.

Thread-level parallelism

Another strategy of achieving performance is to execute multiple programs or threads in parallel. This area of research is known as parallel computing. In Flynn's taxonomy, this strategy is known as Multiple Instructions-Multiple Data or MIMD.

One technology used for this purpose was multiprocessing (MP). The initial flavor of this technology is known as symmetric multiprocessing (SMP), where a small number of CPUs share a coherent view of their memory system. In this scheme, each CPU has additional hardware to maintain a constantly up-to-date view of memory. By avoiding stale views of memory, the CPUs can cooperate on the same program and programs can migrate from one CPU to another. To increase the number of cooperating CPUs beyond a handful, schemes such as non-uniform memory access (NUMA) and directory-based coherence protocols were introduced in the 1990s. SMP systems are limited to a small number of CPUs while NUMA systems have been built with thousands of processors. Initially, multiprocessing was built using multiple discrete CPUs and boards to implement the interconnect between the processors. When the processors and their interconnect are all implemented on a single silicon chip, the technology is known as a multi-core processor.

It was later recognized that finer-grain parallelism existed with a single program. A single program might have several threads (or functions) that could be executed separately or in parallel. Some of the earliest examples of this technology implemented input/output processing such as direct memory access as a separate thread from the computation thread. A more general approach to this technology was introduced in the 1970s when systems were designed to run multiple computation threads in parallel. This technology is known as multi-threading (MT). This approach is considered more cost-effective than multiprocessing, as only a small number of components within a CPU is replicated to support MT as opposed to the entire CPU in the case of MP. In MT, the execution units and the memory system including the caches are shared among multiple threads. The downside of MT is that the hardware support for multithreading is more visible to software than that of MP and thus supervisor software like operating systems have to undergo larger changes to support MT. One type of MT that was implemented is known as block multithreading, where one thread is executed until it is stalled waiting for data to return from external memory. In this scheme, the CPU would then quickly switch to another thread which is ready to run, the switch often done in one CPU clock cycle, such as the UltraSPARC Technology. Another type of MT is known as simultaneous multithreading, where instructions of multiple threads are executed in parallel within one CPU clock cycle.

For several decades from the 1970s to early 2000s, the focus in designing high performance general purpose CPUs was largely on achieving high ILP through technologies such as pipelining, caches, superscalar execution, out-of-order execution, etc. This trend culminated in large, power-hungry CPUs such as the Intel Pentium 4. By the early 2000s, CPU designers were thwarted from achieving higher performance from ILP techniques due to the growing disparity between CPU operating frequencies and main memory operating frequencies as well as escalating CPU power dissipation owing to more esoteric ILP techniques.

CPU designers then borrowed ideas from commercial computing markets such as transaction processing, where the aggregate performance of multiple programs, also known as throughput computing, was more important than the performance of a single thread or program.

This reversal of emphasis is evidenced by the proliferation of dual and multiple core CMP (chip-level multiprocessing) designs and notably, Intel's newer designs resembling its less superscalar P6 architecture. Late designs in several processor families exhibit CMP, including the x86-64 Opteron and Athlon 64 X2, the SPARC UltraSPARC T1, IBM POWER4 and POWER5, as well as several video game console CPUs like the Xbox 360's triple-core PowerPC design, and the PS3's 7-core Cell microprocessor.

Data parallelism

Main articles: Vector processor and SIMD

A less common but increasingly important paradigm of CPUs (and indeed, computing in general) deals with data parallelism. The processors discussed earlier are all referred to as some type of scalar device.[lower-alpha 10] As the name implies, vector processors deal with multiple pieces of data in the context of one instruction. This contrasts with scalar processors, which deal with one piece of data for every instruction. Using Flynn's taxonomy, these two schemes of dealing with data are generally referred to as SIMD (single instruction, multiple data) and SISD (single instruction, single data), respectively. The great utility in creating CPUs that deal with vectors of data lies in optimizing tasks that tend to require the same operation (for example, a sum or a dot product) to be performed on a large set of data. Some classic examples of these types of tasks are multimedia applications (images, video, and sound), as well as many types of scientific and engineering tasks. Whereas a scalar CPU must complete the entire process of fetching, decoding, and executing each instruction and value in a set of data, a vector CPU can perform a single operation on a comparatively large set of data with one instruction. Of course, this is only possible when the application tends to require many steps which apply one operation to a large set of data.

Most early vector CPUs, such as the Cray-1, were associated almost exclusively with scientific research and cryptography applications. However, as multimedia has largely shifted to digital media, the need for some form of SIMD in general-purpose CPUs has become significant. Shortly after inclusion of floating point execution units started to become commonplace in general-purpose processors, specifications for and implementations of SIMD execution units also began to appear for general-purpose CPUs. Some of these early SIMD specifications like HP's Multimedia Acceleration eXtensions (MAX) and Intel's MMX were integer-only. This proved to be a significant impediment for some software developers, since many of the applications that benefit from SIMD primarily deal with floating point numbers. Progressively, these early designs were refined and remade into some of the common, modern SIMD specifications, which are usually associated with one ISA. Some notable modern examples are Intel's SSE and the PowerPC-related AltiVec (also known as VMX).[lower-alpha 11]

2.4 Performance

Further information: Computer performance and Benchmark (computing)

The *performance* or *speed* of a processor depends on, among many other factors, the clock rate (generally given in multiples of hertz) and the instructions per clock (IPC), which together are the factors for the instructions per second (IPS) that the CPU can perform.[14] Many reported IPS values have represented "peak" execution rates on artificial instruction sequences with few branches, whereas realistic workloads consist of a mix of instructions and applications, some of which take longer to execute than others. The performance of the memory hierarchy also greatly affects processor performance, an issue barely considered in MIPS calculations. Because of these problems, various standardized tests, often called "benchmarks" for this purpose—such as SPECint – have been developed to attempt to measure the real effective performance in commonly used applications.

Processing performance of computers is increased by using multi-core processors, which essentially is plugging two or more individual processors (called *cores* in this sense) into one integrated circuit.[15] Ideally, a dual core processor would be nearly twice as powerful as a single core processor. In practice, the performance gain is far smaller, only about 50%, due to imperfect software algorithms and implementation.[16] Increasing the number of cores in a processor (i.e. dual-core, quad-core, etc.) increases the workload that can be handled. This means that the processor can now handle numerous asynchronous events, interrupts, etc. which can take a toll on the CPU when overwhelmed. These cores can be thought of as different floors in a processing plant, with each floor handling a different task. Sometimes, these cores will handle the same tasks as cores adjacent to them if a single core is not enough to handle the information.

Due to specific capabilities of modern CPUs, such as hyper-threading and uncore, which involve sharing of actual CPU resources while aiming at increased utilization, monitoring performance levels and hardware utilization gradually became a more complex task. As a response, some CPUs implement additional hardware logic that monitors actual utilization of various parts of a CPU and provides various counters accessible to software; an example is Intel's *Performance Counter Monitor* technology.[3]

2.5 See also

- Accelerated processing unit
- Addressing mode
- CISC
- Computer bus
- Computer engineering
- CPU core voltage
- CPU socket
- Digital signal processor
- Hyper-threading
- List of CPU architectures
- Microprocessor
- Multi-core processor
- Protection ring
- RISC
- Stream processing
- True Performance Index
- Wait state

2.6 Notes

[1] Integrated circuits are now used to implement all CPUs, except for a few machines designed to withstand large electromagnetic pulses, say from a nuclear weapon.

[2] The so-called "von Neumann" memo expounded the idea of stored programs, which for example may be stored on punched cards, paper tape, or magnetic tape.

[3] Some early computers like the Harvard Mark I did not support any kind of "jump" instruction, effectively limiting the complexity of the programs they could run. It is largely for this reason that these computers are often not considered to contain a proper CPU, despite their close similarity to stored-program computers.

[4] Since the program counter counts *memory addresses* and not *instructions*, it is incremented by the number of memory units that the instruction word contains. In the case of simple fixed-length instruction word ISAs, this is always the same number. For example, a fixed-length 32-bit instruction word ISA that uses 8-bit memory words would always increment the PC by four (except in the case of jumps). ISAs that use variable-length instruction words increment the PC by the number of memory words corresponding to the last instruction's length.

[5] Because the instruction set architecture of a CPU is fundamental to its interface and usage, it is often used as a classification of the "type" of CPU. For example, a "PowerPC CPU" uses some variant of the PowerPC ISA. A system can execute a different ISA by running an emulator.

[6] The physical concept of voltage is an analog one by nature, practically having an infinite range of possible values. For the purpose of physical representation of binary numbers, two specific ranges of voltages are defined, one for logic '0' and another for logic '1'. These ranges are dictated by design considerations such as noise margins and characteristics of the devices used to create the CPU.

[7] While a CPU's integer size sets a limit on integer ranges, this can (and often is) overcome using a combination of software and hardware techniques. By using additional memory, software can represent integers many magnitudes larger than the CPU can. Sometimes the CPU's ISA will even facilitate operations on integers larger than it can natively represent by providing instructions to make large integer arithmetic relatively quick. This method of dealing with large integers is slower than utilizing a CPU with higher integer size, but is a reasonable trade-off in cases where natively supporting the full integer range needed would be cost-prohibitive. See Arbitrary-precision arithmetic for more details on purely software-supported arbitrary-sized integers.

[8] Neither ILP nor TLP is inherently superior over the other; they are simply different means by which to increase CPU parallelism. As such, they both have advantages and disadvantages, which are often determined by the type of software that the processor is intended to run. High-TLP CPUs are often used in applications that lend themselves well to being split up into numerous smaller applications, so-called "embarrassingly parallel problems". Frequently, a computational problem that can be solved quickly with high TLP design strategies like SMP take significantly more time on high ILP devices like superscalar CPUs, and vice versa.

[9] Best-case scenario (or peak) IPC rates in very superscalar architectures are difficult to maintain since it is impossible to keep the instruction pipeline filled all the time. Therefore, in highly superscalar CPUs, average sustained IPC is often discussed rather than peak IPC.

[10] Earlier the term scalar was used to compare the IPC (instructions per cycle) count afforded by various ILP methods. Here the term is used in the strictly mathematical sense to contrast with vectors. See scalar (mathematics) and Vector (geometric).

[11] Although SSE/SSE2/SSE3 have superseded MMX in Intel's general purpose CPUs, later IA-32 designs still support MMX. This is usually accomplished by providing most of the MMX functionality with the same hardware that supports the much more expansive SSE instruction sets.

2.7 References

[1] Weik, Martin H. (1961). "A Third Survey of Domestic Electronic Digital Computing Systems". Ballistic Research Laboratory.

[2] Kuck, David (1978). *Computers and Computations, Vol 1*. John Wiley & Sons, Inc. p. 12. ISBN 0471027162.

[3] Thomas Willhalm; Roman Dementiev; Patrick Fay (December 18, 2014). "Intel Performance Counter Monitor – A better way to measure CPU utilization". *software.intel.com*. Retrieved February 17, 2015.

[4] Regan, Gerard. *A Brief History of Computing*. p. 66. ISBN 1848000839. Retrieved 26 November 2014.

[5] "First Draft of a Report on the EDVAC". Moore School of Electrical Engineering, University of Pennsylvania. 1945.

[6] Enticknap, Nicholas (Summer 1998), "Computing's Golden Jubilee", *Resurrection* (The Computer Conservation Society) (20), ISSN 0958-7403, retrieved 19 April 2008

[7] Amdahl, G. M., Blaauw, G. A., & Brooks, F. P. Jr. (1964). "Architecture of the IBM System/360". IBM Research.

[8] "LSI-11 Module Descriptions". *LSI-11, PDP-11/03 user's manual* (2nd ed.). Maynard, Massachusetts: Digital Equipment Corporation. November 1975. pp. 4–3.

[9] "Excerpts from A Conversation with Gordon Moore: Moore's Law" (PDF). Intel. 2005. Retrieved 2012-07-25.

[10] Ian Wienand (September 3, 2013). "Computer Science from the Bottom Up, Chapter 3. Computer Architecture" (PDF). *bottomupcs.com*. Retrieved January 7, 2015.

[11] Brown, Jeffery (2005). "Application-customized CPU design". IBM developerWorks. Retrieved 2005-12-17.

[12] Garside, J. D., Furber, S. B., & Chung, S-H (1999). "AMULET3 Revealed". University of Manchester Computer Science Department. Archived from the original on December 10, 2005.

[13] Huynh, Jack (2003). "The AMD Athlon XP Processor with 512KB L2 Cache". University of Illinois — Urbana-Champaign. pp. 6–11. Retrieved 2007-10-06.

[14] "CPU Frequency". *CPU World Glossary*. CPU World. 25 March 2008. Retrieved 1 January 2010.

[15] "What is (a) multi-core processor?". *Data Center Definitions*. SearchDataCenter.com. 27 March 2007. Retrieved 1 January 2010.

[16] "Quad Core Vs. Dual Core". *http://www.buzzle.com/". Retrieved 26 November 2014.*

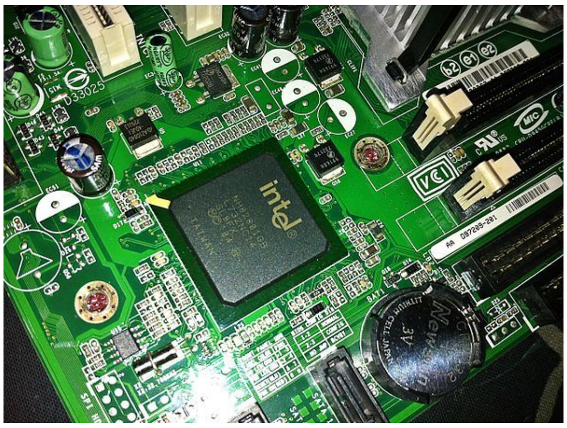
2.8 External links

- How Microprocessors Work at HowStuffWorks.
- 25 Microchips that shook the world – an article by the Institute of Electrical and Electronics Engineers.

Chapter 3

Chipset

In a computer system, a **chipset** is a set of electronic components in an integrated circuit that manages the data flow between the processor, memory and peripherals. It is usually found on the motherboard. Chipsets are usually designed to work with a specific family of microprocessors. Because it controls communications between the processor and external devices, the chipset plays a crucial role in determining system performance.



Intel ICH7 Southbridge on Intel D945GCPE Desktop Board

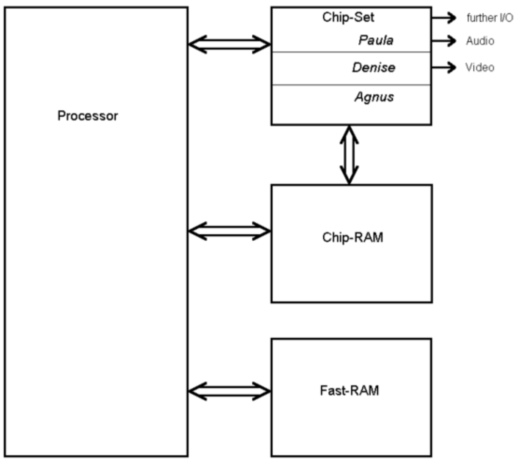
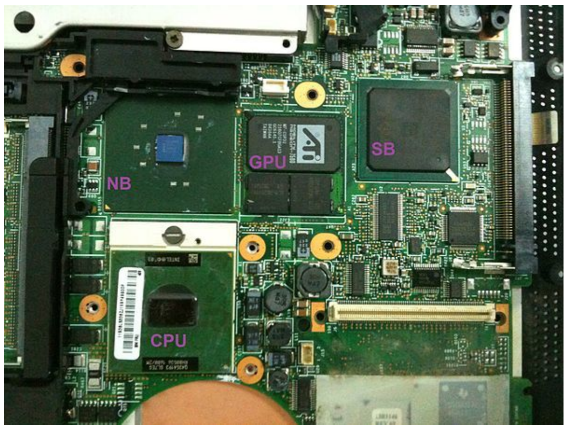


Diagram of Commodore Amiga's Original Chip Set



A part of an IBM T42 laptop motherboard. CPU: Central processing unit. NB: Northbridge. GPU: Graphics processing unit. SB: Southbridge.

3.1 Computers

In computing, the term *chipset* commonly refers to a set of specialized chips on a computer's motherboard or an expansion card. In personal computers, the first chipset for the IBM PC AT of 1984 was the NEAT chipset developed by Chips and Technologies for the Intel 80286 CPU.

In home computers, game consoles and arcade-game hardware of the 1980s and 1990s, the term chipset was used for the custom audio and graphics chips. Examples include the Commodore Amiga's Original Chip Set or SEGA's System 16 chipset.

Based on Intel Pentium-class microprocessors, the term *chipset* often refers to a specific pair of chips on the motherboard: the *northbridge* and the *southbridge*. The northbridge links the CPU to very high-speed devices, especially RAM and graphics controllers, and the southbridge connects to lower-speed peripheral buses (such as PCI or ISA). In many modern chipsets, the southbridge contains some on-chip integrated peripherals, such as Ethernet, USB, and audio devices.

Motherboards and their chipsets often come from different manufacturers. As of 2015, manufacturers of chipsets for x86 motherboards include AMD, Broadcom, Intel, NVIDIA, SiS and VIA Technologies. Apple computers and Unix workstations have traditionally used custom-designed chipsets. Some server manufacturers also develop custom chipsets for their products.

In the 1980s Chips and Technologies pioneered the manufacturing of chipsets for PC-compatible computers. Computer systems produced since then often share commonly used chipsets, even across widely disparate computing specialties. For example, the NCR 53C9x, a low-cost chipset implementing a SCSI interface to storage devices, could be found in Unix machines such as the MIPS Magnum, embedded devices, and personal computers.

3.2 Move toward processor integration in PCs

Traditionally in x86 computers, the processor's primary connection to the rest of the machine is through the motherboard chipset's northbridge. The northbridge is directly responsible for communications with high-speed devices (system memory and primary expansion buses, such as PCIe, AGP and PCI cards, being common examples) and conversely any system communication back to the processor. This connection between the processor and northbridge is traditionally known as the front side bus (FSB). Requests to resources not directly controlled by the northbridge are offloaded to the southbridge, with the northbridge being an intermediary between the processor and the southbridge. The southbridge traditionally handles "everything else", generally lower-speed peripherals and board functions (the largest being hard disk and storage connectivity) such as USB, parallel and serial communications. The connection between the northbridge and southbridge does not have a common name, but is usually a high-speed interconnect proprietary to the chipset vendor.

Before 2003, any interaction between a CPU and main memory or an expansion device such as a graphics card(s) — whether AGP, PCI or integrated into the motherboard — was directly controlled by the northbridge IC on behalf of the processor. This made processor performance highly dependent on the system chipset, especially the northbridge's memory performance and ability to shuttle this information back to the processor. In 2003, however, AMD's introduction of the Athlon 64-bit series of processors[1] changed this. The Athlon64 marked the introduction of an integrated memory controller being incorporated into the processor itself thus allowing the processor to directly access and handle memory, negating the need for a traditional northbridge to do so. Intel followed suit in 2008 with the release of its Core i series CPUs and the X58 platform. In newer processors integration has further increased, primarily inclusion of the system's primary PCIe controller and integrated graphics directly on the CPU itself. As fewer functions are left un-handled by the processor, chipset vendors have condensed the remaining northbridge and southbridge functions into a single chip. Intel's version of this is the "Platform Controller Hub" (PCH), effectively an enhanced southbridge for the remaining peripherals as traditional northbridge duties, such as memory controller, expansion bus (PCIe) interface, and even on-board video controller, are integrated into the CPU itself.

3.3 See also

- Very-large-scale integration or VLSI
- List of Intel chipsets
- Comparison of AMD chipsets
- Comparison of ATI chipsets
- Comparison of Nvidia chipsets
- VIA chipsets
- Silicon Integrated Systems
- Acer Laboratories Incorporated
- Southbridge
- Northbridge

3.4 Notes

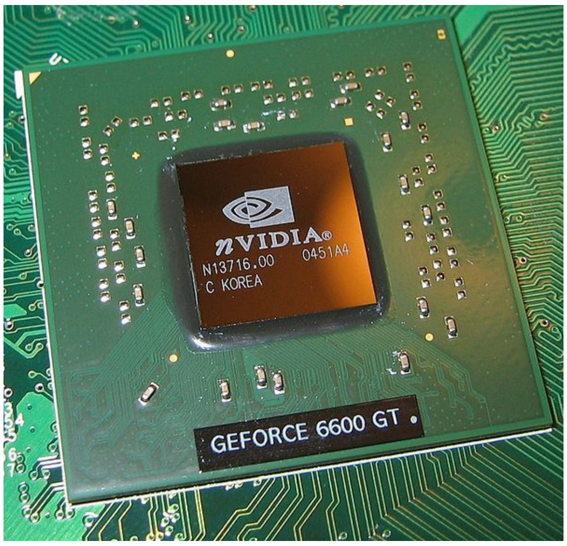
[1] "AMD Ushers in Era of Cinematic Computing with the AMD Athlon 64 FX Processor". 2003-09-23. Retrieved 2006-07-04.

Chapter 4

Graphics processing unit

Not to be confused with Graphics card.
"GPU" redirects here. For other uses, see GPU (disambiguation).

A **graphics processor unit** (**GPU**), also occasionally called **visual processor unit** (**VPU**), is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Modern GPUs are very efficient at manipulating computer graphics and image processing, and their highly parallel structure makes them more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel. In a personal computer, a GPU can be present on a video card, or it can be on the motherboard or—in certain CPUs—on the CPU die.[1]



GeForce 6600GT (NV43) GPU

The term GPU was popularized by Nvidia in 1999, who marketed the GeForce 256 as "the world's first 'GPU', or Graphics Processing Unit, a single-chip processor with integrated transform, lighting, triangle setup/clipping, and rendering engines that are capable of processing a minimum of 10 million polygons per second". Rival ATI Technologies coined the term visual processing unit or VPU with the release of the Radeon 9700 in 2002.

4.1 History

Arcade system boards have been using specialized graphics chips since the 1970s. Fujitsu's MB14241 video shifter was used to accelerate the drawing of sprite graphics for various 1970s arcade games from Taito and Midway, such as *Gun Fight* (1975), *Sea Wolf* (1976) and *Space Invaders* (1978).[2][3][4] The Namco Galaxian arcade system in 1979 used specialized graphics hardware supporting RGB color, multi-colored sprites and tilemap backgrounds.[5] The Galaxian hardware was widely used during the golden age of arcade video games, by game companies such as Namco, Centuri, Gremlin, Irem, Konami, Midway, Nichibutsu, Sega and Taito.[6][7] In the home video game console market, the Atari 2600 in 1977 used a video shifter called the Television Interface Adaptor.

4.1.1 1980s

See also: Video Display Controller, List of home computers by video hardware and Sprite (computer graphics)

In 1985, the Commodore Amiga featured a GPU advanced for a personal computer at the time. It supported line draw, area fill, and included a type of stream processor called a blitter which accelerated the movement, manipulation and combination of multiple arbitrary bitmaps. Also included was a coprocessor with its own (primitive) instruction set capable of directly invoking a sequence of graphics operations without CPU intervention. Prior to this and for quite some time after, many other personal computer systems instead used their main, general-purpose CPU to handle almost every aspect of drawing the display, short of generating the final video signal.

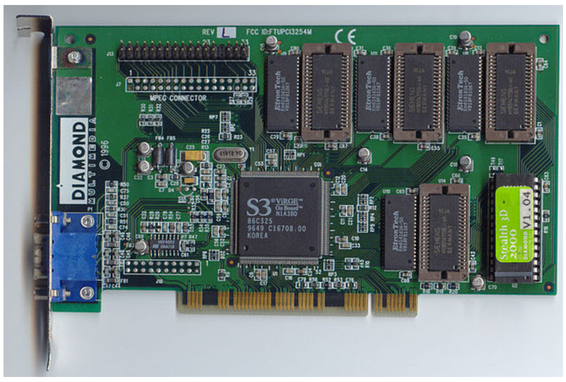
In 1986, Texas Instruments released the TMS34010, the first microprocessor with on-chip graphics capabilities. It could run general-purpose code, but it had a very graphics-oriented instruction set. In 1990-1991, this chip would become the basis of the Texas Instruments Graphics Architecture ("TIGA") Windows accelerator cards.

In 1987, the IBM 8514 graphics system was released as one of the first video cards for IBM PC compatibles to implement fixed-function 2D primitives in electronic hardware. The same year, Sharp released the X68000, which used a custom graphics chipset[8] that was powerful for a home computer at the time, with a 65,536 color palette and hardware support for sprites, scrolling and multiple playfields,[9] eventually serving as a development machine for Capcom's CP System arcade board. Fujitsu later competed with the FM Towns computer, released in 1989 with support for a full 16,777,216 color palette.[10]

In 1988, the first dedicated polygonal 3D graphics boards were introduced in arcades with the Namco System 21[11] and Taito Air System.[12]

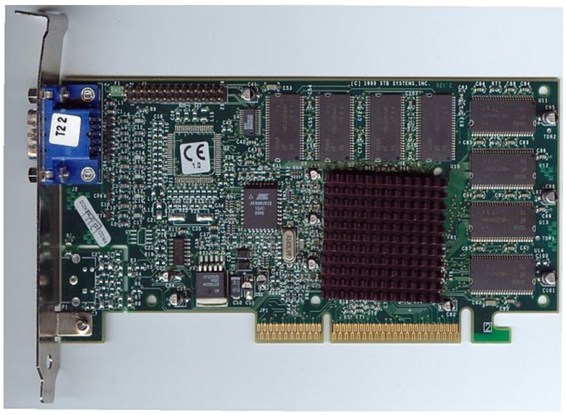
4.1.2 1990s

Tseng Labs ET4000/W32p



S3 Graphics ViRGE

In 1991, S3 Graphics introduced the *S3 86C911*, which its designers named after the Porsche 911 as an implication of the performance increase it promised. The 86C911 spawned a host of imitators: by 1995, all major PC graphics chip makers had added 2D acceleration support to their chips. By this time, fixed-function *Windows accelerators* had surpassed expensive general-purpose graphics coprocessors in Windows performance, and these coprocessors faded away from the PC market.



Voodoo3 2000 AGP card

Throughout the 1990s, 2D GUI acceleration continued to evolve. As manufacturing capabilities improved, so did the level of integration of graphics chips. Additional application programming interfaces (APIs) arrived for a variety of tasks, such as Microsoft's WinG graphics library for Windows 3.x, and their later DirectDraw interface for hardware acceleration of 2D games within Windows 95 and later.

In the early- and mid-1990s, CPU-assisted real-time 3D graphics were becoming increasingly common in arcade, computer and console games, which led to an increasing public demand for hardware-accelerated 3D graphics. Early examples of mass-marketed 3D graphics hardware can be found in arcade system boards such as the Sega Model 1, Namco System 22, and Sega Model 2, and the fifth-generation video game consoles such as the Saturn, PlayStation and Nintendo 64. Arcade systems such as the Sega Model 2 and Namco Magic Edge Hornet Simulator were capable of hardware T&L (transform, clipping, and lighting) years before appearing in consumer graphics cards.[13][14] Fujitsu, which worked on the Sega Model 2 arcade system,[15] began working on integrating T&L into a single LSI solution for use in home computers in 1995.[16][17][18]

In the PC world, notable failed first tries for low-cost 3D graphics chips were the S3 *ViRGE*, ATI *Rage*, and Matrox *Mystique*. These chips were essentially previous-generation 2D accelerators with 3D features bolted on. Many were even pin-compatible with the earlier-generation chips for ease of implementation and minimal cost. Initially, performance 3D graphics were possible only with discrete boards dedicated to accelerating 3D functions (and lacking 2D GUI acceleration entirely) such as the PowerVR and the 3dfx *Voodoo*. However, as manufacturing technology continued to progress, video, 2D GUI acceleration and 3D functionality were all integrated into one chip. Rendition's *Verite* chipsets were among the first to do this well enough to be worthy of note. In 1997, Rendition went a step further by collaborating with Hercules and Fujitsu on a "Thriller Conspiracy" project which combined a Fujitsu FXG-1 Pinolite geometry processor with a Vérité V2200 core to create a graphics card with a full T&L engine years before Nvidia's GeForce 256. This card, designed to reduce the load placed upon the system's CPU, never made it to market.

OpenGL appeared in the early '90s as a professional graphics API, but originally suffered from performance issues which allowed the Glide API to step in and become a dominant force on the PC in the late '90s.[19] However, these issues were quickly overcome and the Glide API fell by the wayside. Software implementations of OpenGL were common during this time, although the influence of OpenGL eventually led to widespread hardware support. Over time, a parity emerged between features offered in hardware and those offered in OpenGL. DirectX became popular among Windows game developers during the late 90s. Unlike OpenGL, Microsoft insisted on providing strict one-to-one support of hardware. The approach made DirectX less popular as a standalone graphics API initially, since many GPUs provided their own specific features, which existing OpenGL applications were already able to benefit from, leaving DirectX often one generation behind. (See: Comparison of OpenGL and Direct3D.)

Over time, Microsoft began to work more closely with hardware developers, and started to target the releases of DirectX to coincide with those of the supporting graphics hardware. Direct3D 5.0 was the first version of the burgeoning API to gain widespread adoption in the gaming market, and it competed directly with many more-hardware-specific, often proprietary graphics libraries, while OpenGL maintained a strong following. Direct3D 7.0 introduced support for hardware-accelerated transform and lighting (T&L) for Direct3D, while OpenGL had this capability already exposed from its inception. 3D accelerator cards moved beyond being just simple rasterizers to add another significant hardware stage to the 3D rendering pipeline. The Nvidia *GeForce 256* (also known as NV10) was the first consumer-level card released on the market with hardware-accelerated T&L, while professional 3D cards already had this capability. Hardware transform and lighting, both already existing features of OpenGL, came to consumer-level hardware in the '90s and set the precedent for later pixel shader and vertex shader units which were far more flexible and programmable.

4.1.3 2000 to 2006

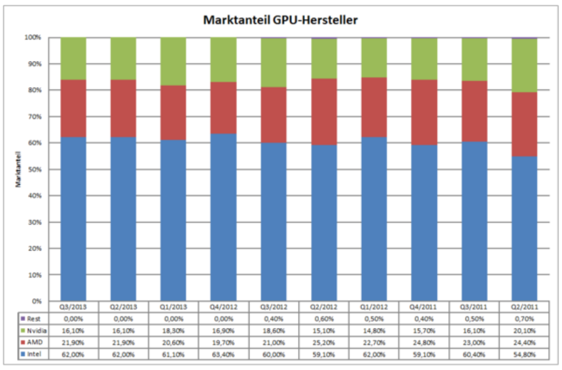
With the advent of the OpenGL API and similar functionality in DirectX, GPUs added shading to their capabilities. Each pixel could now be processed by a short program that could include additional image textures as inputs, and each geometric vertex could likewise be processed by a short program before it was projected onto the screen. Nvidia was first to produce a chip capable of programmable shading, the *GeForce 3* (code named NV20). By October 2002, with the introduction of the ATI *Radeon 9700* (also known as R300), the world's first Direct3D 9.0 accelerator, pixel and vertex shaders could implement looping and lengthy floating point math, and in general were quickly becoming as flexible as CPUs, and orders of magnitude faster for image-array operations. Pixel shading is often used for things like bump mapping, which adds texture, to make an object look shiny, dull, rough, or even round or extruded.[20]

4.1.4 2006 to present

With the introduction of the GeForce 8 series, which was produced by Nvidia, and then new generic stream processing unit GPUs became a more generalized computing device. Today, parallel GPUs have begun making computational inroads against the CPU, and a subfield of research, dubbed GPU Computing or GPGPU for *General Purpose Computing on GPU*, has found its way into fields as diverse as machine learning,[21] oil exploration, scientific image processing, linear algebra,[22] statistics,[23] 3D reconstruction and even stock options pricing determination. Over the years, the energy consumption of GPUs has increased and to manage it, several techniques have been proposed.[24]

Nvidia's CUDA platform was the earliest widely adopted programming model for GPU computing. More recently OpenCL has become broadly supported. OpenCL is an open standard defined by the Khronos Group which allows for the development of code for both GPUs and CPUs with an emphasis on portability.[25] OpenCL solutions are supported by Intel, AMD, Nvidia, and ARM, and according to a recent report by Evan's Data, OpenCL is the GPGPU development platform most widely used by developers in both the US and Asia Pacific.

4.1.5 GPU companies



	Q3/2013	Q2/2013	Q1/2013	Q4/2012	Q3/2012	Q2/2012	Q1/2012	Q4/2011	Q3/2011	Q2/2011
🟪 Rest	0,00%	0,00%	0,00%	0,00%	0,40%	0,60%	0,50%	0,40%	0,50%	0,70%
🟩 Nvidia	16,10%	16,10%	18,30%	16,90%	18,60%	15,10%	14,80%	15,70%	16,10%	20,10%
🟥 AMD	21,90%	21,90%	20,60%	19,70%	21,00%	25,20%	22,70%	24,80%	23,00%	24,40%
🟦 Intel	62,00%	62,00%	61,10%	63,40%	60,00%	59,10%	62,00%	59,10%	60,40%	54,80%

GPU manufacturers market share

Many companies have produced GPUs under a number of brand names. In 2009, Intel, Nvidia and AMD/ATI were the market share leaders, with 49.4%, 27.8% and

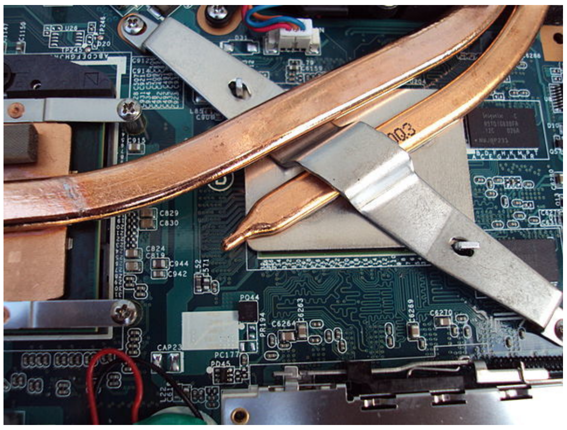
20.6% market share respectively. However, those numbers include Intel's integrated graphics solutions as GPUs. Not counting those numbers, Nvidia and ATI control nearly 100% of the market as of 2008.[26] In addition, S3 Graphics[27] (owned by VIA Technologies) and Matrox[28] produce GPUs.

4.2 Computational functions

Modern GPUs use most of their transistors to do calculations related to 3D computer graphics. They were initially used to accelerate the memory-intensive work of texture mapping and rendering polygons, later adding units to accelerate geometric calculations such as the rotation and translation of vertices into different coordinate systems. Recent developments in GPUs include support for programmable shaders which can manipulate vertices and textures with many of the same operations supported by CPUs, oversampling and interpolation techniques to reduce aliasing, and very high-precision color spaces. Because most of these computations involve matrix and vector operations, engineers and scientists have increasingly studied the use of GPUs for non-graphical calculations.

In addition to the 3D hardware, today's GPUs include basic 2D acceleration and framebuffer capabilities (usually with a VGA compatibility mode). Newer cards like AMD/ATI HD5000-HD7000 even lack 2D acceleration; it has to be emulated by 3D hardware.

4.2.1 GPU accelerated video decoding



The ATI HD5470 GPU (above) features UVD 2.1 which enables it to decode AVC and VC-1 video formats

Most GPUs made since 1995 support the YUV color space and hardware overlays, important for digital video playback, and many GPUs made since 2000 also support MPEG primitives such as motion compensation and iDCT. This process of hardware accelerated video decoding, where portions of the video decoding process and video post-processing are offloaded to the GPU hardware, is commonly referred to as “GPU accelerated video decoding”, “GPU assisted video decoding”, “GPU hardware accelerated video decoding” or “GPU hardware assisted video decoding”.

More recent graphics cards even decode high-definition video on the card, offloading the central processing unit. The most common APIs for GPU accelerated video decoding are DxVA for Microsoft Windows operating system and VDPAU, VAAPI, XvMC, and XvBA for Linux-based and UNIX-like operating systems. All except XvMC are capable of decoding videos encoded with MPEG-1, MPEG-2, MPEG-4 ASP (MPEG-4 Part 2), MPEG-4 AVC (H.264 / DivX 6), VC-1, WMV3/WMV9, Xvid / OpenDivX (DivX 4), and DivX 5 codecs, while XvMC is only capable of decoding MPEG-1 and MPEG-2.

Video decoding processes that can be accelerated

The video decoding processes that can be accelerated by today's modern GPU hardware are:

- Motion compensation (mocomp)
- Inverse discrete cosine transform (iDCT)
    - Inverse telecine 3:2 and 2:2 pull-down correction
- Inverse modified discrete cosine transform (iMDCT)
- In-loop deblocking filter
- Intra-frame prediction
- Inverse quantization (IQ)
- Variable-length decoding (VLD), more commonly known as slice-level acceleration
- Spatial-temporal deinterlacing and automatic interlace/progressive source detection
- Bitstream processing (Context-adaptive variable-length coding/Context-adaptive binary arithmetic coding) and perfect pixel positioning.

4.3 GPU forms

4.3.1 Dedicated graphics cards

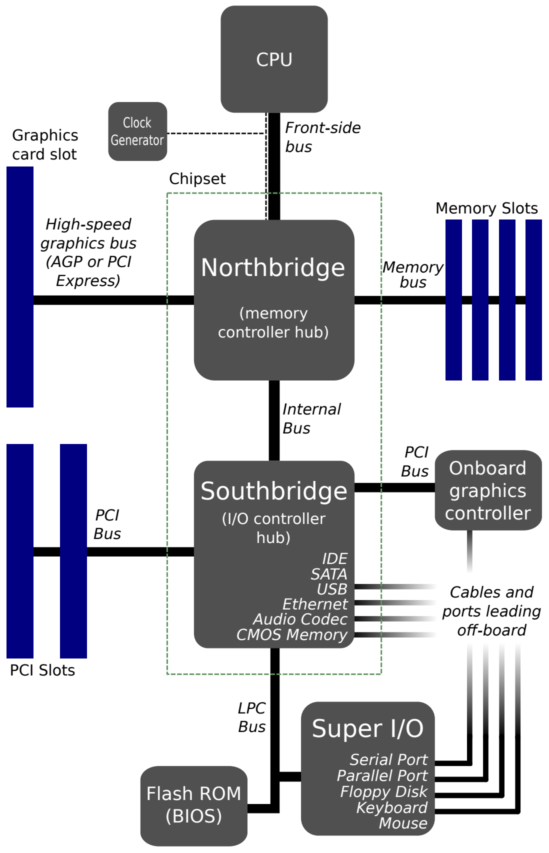
Main article: Video card

The GPUs of the most powerful class typically interface with the motherboard by means of an expansion slot such as PCI Express (PCIe) or Accelerated Graphics Port (AGP) and can usually be replaced or upgraded with relative ease, assuming the motherboard is capable of supporting the upgrade. A few graphics cards still use Peripheral Component Interconnect (PCI) slots, but their bandwidth is so limited that they are generally used only when a PCIe or AGP slot is not available.

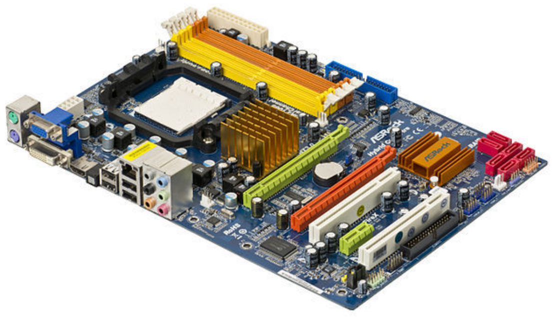
A dedicated GPU is not necessarily removable, nor does it necessarily interface with the motherboard in a standard fashion. The term "dedicated" refers to the fact that dedicated graphics cards have RAM that is dedicated to the card's use, not to the fact that *most* dedicated GPUs are removable. Dedicated GPUs for portable computers are most commonly interfaced through a non-standard and often proprietary slot due to size and weight constraints. Such ports may still be considered PCIe or AGP in terms of their logical host interface, even if they are not physically interchangeable with their counterparts.

Technologies such as SLI by Nvidia and CrossFire by AMD allow multiple GPUs to draw images simultaneously for a single screen, increasing the processing power available for graphics.

4.3.2 Integrated graphics solutions



Layout



A motherboard with integrated graphics, which has HDMI, VGA and DVI outs.

Integrated graphics solutions, *shared graphics solutions*, or *integrated graphics processors* (IGP) utilize a portion of a computer's system RAM rather than dedicated graphics memory. IGPs can be integrated onto the motherboard as part of the chipset, or within the same die as CPU (like AMD APU or Intel HD Graphics). Some of AMD's IGPs use dedicated sideport memory on certain motherboards. Computers with integrated graphics account for 90% of all PC shipments.[29] These solutions are less costly to implement than dedicated graphics solutions, but tend to be less capable. Historically, integrated solutions were often considered unfit to play 3D games or run graphically intensive programs but could run less intensive programs such as Adobe Flash. Examples of such IGPs would be offerings from SiS and VIA circa 2004.[30] However, modern integrated graphics processors such as AMD Accelerated Processing Unit and Intel HD Graphics are more than capable of handling 2D graphics or low stress 3D graphics.

As a GPU is extremely memory intensive, an integrated solution may find itself competing for the already relatively slow system RAM with the CPU, as it has minimal or no dedicated video memory. IGPs can have up to 29.856 GB/s of memory bandwidth from system RAM, however graphics cards can enjoy up to 264 GB/s of bandwidth between its RAM and GPU core. This bandwidth is what is referred to as the memory bus and can be performance limiting. Older integrated graphics chipsets lacked hardware transform and lighting, but newer ones include it.[31][32]

4.3.3 Hybrid solutions

This newer class of GPUs competes with integrated graphics in the low-end desktop and notebook markets. The most common implementations of this are ATI's HyperMemory and Nvidia's TurboCache.

Hybrid graphics cards are somewhat more expensive than integrated graphics, but much less expensive than dedicated graphics cards. These share memory with the system and have a small dedicated memory cache, to make up for the high latency of the system RAM. Technologies within PCI Express can make this possible. While these solutions are sometimes advertised as having as much as 768MB of RAM, this refers to how much can be shared with the system memory.

4.3.4 Stream Processing and General Purpose GPUs (GPGPU)

Main articles: GPGPU and Stream processing

It is becoming increasingly common to use a general purpose graphics processing unit as a modified form of stream processor. This concept turns the massive computational power of a modern graphics accelerator's shader pipeline into general-purpose computing power, as opposed to being hard wired solely to do graphical operations. In certain applications requiring massive vector operations, this can yield several orders of magnitude higher performance than a conventional CPU. The two largest discrete (see "Dedicated graphics cards" above) GPU designers, ATI and Nvidia, are beginning to pursue this approach with an array of applications. Both Nvidia and ATI have teamed with Stanford University to create a GPU-based client for the Folding@home distributed computing project, for protein folding calculations. In certain circumstances the GPU calculates forty times faster than the conventional CPUs traditionally used by such applications.[33][34]

GPGPU can be used for many types of embarrassingly parallel tasks including ray tracing. They are generally suited to high-throughput type computations that exhibit data-parallelism to exploit the wide vector width SIMD architecture of the GPU.

Furthermore, GPU-based high performance computers are starting to play a significant role in large-scale modelling. Three of the 10 most powerful supercomputers in the world take advantage of GPU acceleration.[35]

NVIDIA cards support API extensions to the C programming language such as CUDA ("Compute Unified Device Architecture") and OpenCL. CUDA is specifically for NVIDIA GPUs whilst OpenCL is designed to work across a multitude of architectures including GPU, CPU and DSP (using vendor specific SDKs). These technologies allow specified functions (kernels) from a normal C program to run on the GPU's stream processors. This makes C programs capable of taking advantage of a GPU's ability to operate on large matrices in parallel, while still making use of the CPU when appropriate. CUDA is also the first API to allow CPU-based applications to directly access the resources of a GPU for more general purpose computing without the limitations of using a graphics API.

Since 2005 there has been interest in using the performance offered by GPUs for evolutionary computation in general, and for accelerating the fitness evaluation in genetic programming in particular. Most approaches compile linear or tree programs on the host PC and transfer the executable to the GPU to be run. Typically the performance advantage is only obtained by running the single active program simultaneously on many example problems in parallel, using the GPU's SIMD architecture.[36][37] However, substantial acceleration can also be obtained by not compiling the programs, and instead transferring them to the GPU, to be interpreted there.[38][39] Acceleration can then be obtained by either interpreting multiple programs simultaneously, simultaneously running multiple example problems, or combinations of both. A modern GPU (*e.g.* 8800 GTX or later) can readily simultaneously interpret hundreds of thousands of very small programs.

4.3.5 External GPU (eGPU)

An external GPU is a graphics processor located outside of the housing of the computer. External Graphics Processors are often used with laptop computers. Laptops might have a substantial amount of RAM and a sufficiently powerful Central Processing Unit(CPU), but often lack a powerful graphics processor (and instead have a less powerful, but energy efficient on-board graphics chip). On-board graphics chips are often not powerful enough for playing the latest games, or for other tasks (video editing, ...).

Therefore it is desirable to be able to attach to some external PCIe bus of a notebook. That may be an x1 2.0 5Gbit/s expresscard or mPCIe (wifi) port or a 10Gbit/s/16Gbit/s Thunderbolt1/Thunderbolt2 port. Those ports being only available on certain candidate notebook systems.[40][41]

External GPU's have had little official vendor support. Promising solutions such as Silverstone T004 (aka ASUS XG2)[42] and MSI GUS-II[43] were never released to the general public. MSI's Gamedock [44] promising to deliver a full x16 external PCIe bus to a purpose built compact 13" MSI GS30 notebook. Lenovo and Magma partnering in Sep-2014 to deliver official Thunderbolt eGPU support.[45]

This has not stopped enthusiasts from creating their own DIY eGPU solutions.[46][47] expresscard/mPCIe eGPU adapters/enclosures are usually acquired from BPlus (PE4C, PE4L, PE4C),[48] or EXP GDC.[49] native Thunderbolt eGPU adaptere/enclosures acquired from One Stop Systems,[50] AKiTiO,[51] Sonnet (often rebadge as Other World Computing — OWC) and FirmTek.

4.4 Sales

In 2013, 438.3 million GPUs were shipped globally and the forecast for 2014 was 414.2 million.[52]

4.5 See also

- Brute force attack
- Computer graphics
- Computer hardware
- Computer monitor
- Central processing unit
- GPU cache
- Physics processing unit (PPU)
- Ray tracing hardware
- Video card
- Video Display Controller
- Video game console
- Virtualized GPU

4.5.1 Hardware

- Comparison of AMD graphics processing units
- Comparison of Nvidia graphics processing units
- Comparison of Intel graphics processing units
- Intel GMA
- Larrabee
- Nvidia PureVideo - the bit-stream technology from Nvidia used in their graphics chips to accelerate video decoding on hardware GPU with DXVA.
- UVD (Unified Video Decoder) - is the video decoding bit-stream technology from ATI Technologies to support hardware (GPU) decode with DXVA.

4.5.2 APIs

- OpenGL API
- DirectX Video Acceleration (DxVA) API for Microsoft Windows operating-system.
- Mantle (API)
- Video Acceleration API (VA API)
- VDPAU (Video Decode and Presentation API for Unix)
- X-Video Bitstream Acceleration (XvBA), the X11 equivalent of DXVA for MPEG-2, H.264, and VC-1
- X-Video Motion Compensation, the X11 equivalent for MPEG-2 video codec only

4.5.3 Applications

- GPU cluster
- Mathematica includes built-in support for CUDA and OpenCL GPU execution
- MATLAB acceleration using the Parallel Computing Toolbox and MATLAB Distributed Computing Server,[53] as well as 3rd party packages like Jacket.
- Molecular modeling on GPU
- Deeplearning4j, open-source, distributed deep learning for Java. Machine vision and textual topic modelling toolkit.

4.6 References

[1] Denny Atkin. "Computer Shopper: The Right GPU for You". Retrieved 2007-05-15.

[2] "mame/8080bw.c at master ⍰ mamedev/mame ⍰ GitHub". *GitHub*.

[3] "mame/mw8080bw.c at master ⍰ mamedev/mame ⍰ GitHub". *GitHub*.

[4] "Arcade/SpaceInvaders – Computer Archeology". *computerarcheology.com*.

[5] "mame/galaxian.c at master ⍰ mamedev/mame ⍰ GitHub". *GitHub*.

[6] "mame/galaxian.c at master ⍰ mamedev/mame ⍰ GitHub". *GitHub*.

[7] "MAME - src/mame/drivers/galdrvr.c". *archive.org*. Archived from the original on 3 January 2014.

[8] http://nfggames.com/games/x68k/

[9] "musem ~ Sharp X68000". Old-computers.com. Retrieved 2015-01-28.

[10] "Hardcore Gaming 101: Retro Japanese Computers: Gaming's Final Frontier". *hardcoregaming101.net*.

[11] "System 16 - Namco System 21 Hardware (Namco)". *system16.com*.

[12] "System 16 - Taito Air System Hardware (Taito)". *system16.com*.

[13] "System 16 - Namco Magic Edge Hornet Simulator Hardware (Namco)". *system16.com*.

[14] "MAME - src/mame/video/model2.c". *archive.org*. Archived from the original on 4 January 2013.

[15] "System 16 - Sega Model 2 Hardware (Sega)". *system16.com*.

[16] http://www.hotchips.org/wp-content/uploads/hc_archives/hc07/3_Tue/HC7.S5/HC7.5.1.pdf

[17] http://www.fujitsu.com/downloads/MAG/vol33-2/paper08.pdf

[18] “Fujitsu Develops World’s First Three Dimensional Geometry Processor”. *fujitsu.com*.

[19] 3dfx Glide API

[20] Søren Dreijer. “Bump Mapping Using CG (3rd Edition)". Retrieved 2007-05-30.

[21] “Large-scale deep unsupervised learning using graphics processors”. Dl.acm.org. 2009-06-14. doi:10.1145/1553374.1553486. Retrieved 2014-01-21.

[22] “Linear algebra operators for GPU implementation of numerical algorithms”, Kruger and Westermann, International Conf. on Computer Graphics and Interactive Techniques, 2005

[23] “ABC-SysBio—approximate Bayesian computation in Python with GPU support”, Liepe et al., Bioinformatics, (2010), 26:1797-1799

[24] "A Survey of Methods for Analyzing and Improving GPU Energy Efficiency", Mittal et al., ACM Computing Surveys, 2014.

[25] “OpenCL - The open standard for parallel programming of heterogeneous systems”. *khronos.org*.

[26] “GPU sales strong as AMD gains market share”. *techreport.com*.

[27] “Products”. S3 Graphics. Retrieved 2014-01-21.

[28] “Matrox Graphics - Products - Graphics Cards”. Matrox.com. Retrieved 2014-01-21.

[29] Gary Key. “AnandTech - µATX Part 2: Intel G33 Performance Review”. *anandtech.com*.

[30] Tim Tscheblockov. “Xbit Labs: Roundup of 7 Contemporary Integrated Graphics Chipsets for Socket 478 and Socket A Platforms”. Retrieved 2007-06-03.

[31] Bradley Sanford. “Integrated Graphics Solutions for Graphics-Intensive Applications”. Retrieved 2007-09-02.

[32] Bradley Sanford. “Integrated Graphics Solutions for Graphics-Intensive Applications”. Retrieved 2007-09-02.

[33] Darren Murph. “Stanford University tailors Folding@home to GPUs”. Retrieved 2007-10-04.

[34] Mike Houston. “Folding@Home - GPGPU”. Retrieved 2007-10-04.

[35] “Top500 List - June 2012 | TOP500 Supercomputer Sites”. Top500.org. Retrieved 2014-01-21.

[36] John Nickolls. “Stanford Lecture: Scalable Parallel Programming with CUDA on Manycore GPUs”.

[37] S Harding and W Banzhaf. “Fast genetic programming on GPUs”. Retrieved 2008-05-01.

[38] W Langdon and W Banzhaf. “A SIMD interpreter for Genetic Programming on GPU Graphics Cards”. Retrieved 2008-05-01.

[39] V. Garcia and E. Debreuve and M. Barlaud. Fast k nearest neighbor search using GPU. In Proceedings of the CVPR Workshop on Computer Vision on GPU, Anchorage, Alaska, USA, June 2008.

[40] “eGPU candidate system list”. *Tech-Inferno Forums*.

[41] Neil Mohr. “How to make an external laptop graphics adaptor”. *TechRadar*.

[42] "[THUNDERBOLT NEWS] Silverstone T004... Now the ASUS XG2”. *Tech-Inferno Forums*.

[43] “MSI’s GUS II: External Thunderbolt GPU”. *notebookreview.com*.

[44] “MSI eGPU dock in the works for GS30?". *Tech-Inferno Forums*.

[45] “Lenovo + Magma partnership delivers official Thunderbolt eGPU support”. *Tech-Inferno Forums*.

[46] “DIY eGPU on Tablet PC’s: experiences, benchmarks, setup, ect...”. *tabletpcreview.com*.

[47] “Implementations Hub: TB, EC, mPCIe”. *Tech-Inferno Forums*.

[48] BPlus eGPU adapters

[49] "□□-□□□□-□□□". *taobao.com*.

[50] Jim Galbraith (28 March 2014). “Expo Notes: Thunderbolt takes over”. *Macworld*.

[51] “US$200 AKiTiO Thunder2 PCIe Box (16Gbps-TB2)". *Tech-Inferno Forums*.

[52] “Graphics chips market is showing some life”. *TG Daily*. August 20, 2014. Retrieved August 22, 2014.

[53] “MATLAB Adds GPGPU Support”. 2010-09-20.

4.7 External links

- NVIDIA - What is GPU computing?
- The *GPU Gems* book series
- - a Graphics Hardware History
- General-Purpose Computation Using Graphics Hardware
- How GPUs work
- GPU Caps Viewer - Video card information utility
- OpenGPU-GPU Architecture(In Chinese)
- ARM Mali GPUs Overview
- GPU Rendering Magazine

Chapter 5

Random-access memory

"RAM" redirects here. For the Daft Punk album, see Random Access Memories. For other uses of the word, see Ram (disambiguation).

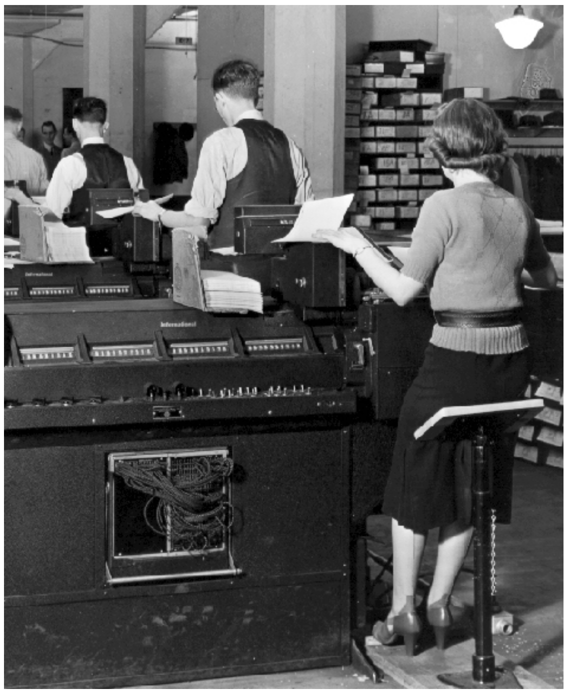
Random-access memory (**RAM** /ræm/) is a form of computer data storage. A random-access memory device allows data items to be read and written in roughly the same amount of time regardless of the order in which data items are accessed.[1] In contrast, with other direct-access data storage media such as hard disks, CD-RWs, DVD-RWs and the older drum memory, the time required to read and write data items varies significantly depending on their physical locations on the recording medium, due to mechanical limitations such as media rotation speeds and arm movement delays.

Example of writable volatile random-access memory: Synchronous Dynamic RAM modules, primarily used as main memory in personal computers, workstations, and servers.

Today, random-access memory takes the form of integrated circuits. RAM is normally associated with volatile types of memory (such as DRAM memory modules), where stored information is lost if power is removed, although many efforts have been made to develop non-volatile RAM chips.[2] Other types of non-volatile memory exist that allow random access for read operations, but either do not allow write operations or have limitations on them. These include most types of ROM and a type of flash memory called *NOR-Flash*.

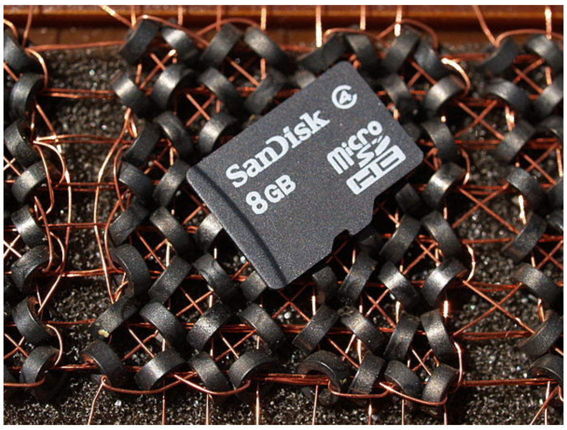
Integrated-circuit RAM chips came into the market in the late 1960s, with the first commercially available DRAM chip, the Intel 1103, introduced in October 1970.[3]

5.1 History

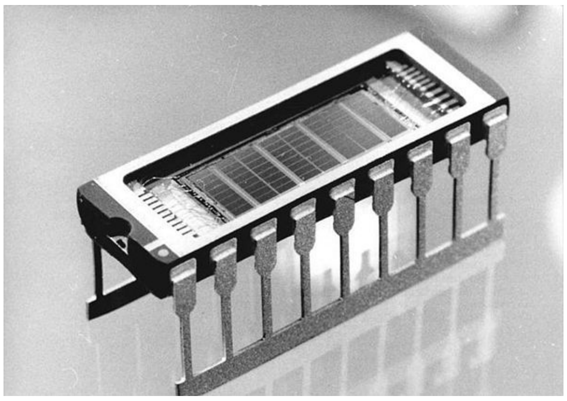


These IBM tabulating machines from the 1930s used mechanical counters to store information

Early computers used relays, mechanical counters[4] or delay lines for main memory functions. Ultrasonic delay lines could only reproduce data in the order it was written. Drum memory could be expanded at relatively low cost but efficient retrieval of memory items required knowledge of the physical layout of the drum to optimize speed. Latches built out of vacuum tube triodes, and later, out of discrete transistors, were used for smaller and faster memories such as registers. Such registers were relatively large and too costly to use for large amounts of data; generally only a few dozen or few hundred bits of such memory could be provided.



A portion of a core memory with a modern flash RAM SD card on top



1 Megabit chip – one of the last models developed by VEB Carl Zeiss Jena in 1989

The first practical form of random-access memory was the Williams tube starting in 1947. It stored data as electrically charged spots on the face of a cathode ray tube. Since the electron beam of the CRT could read and write the spots on the tube in any order, memory was random access. The capacity of the Williams tube was a few hundred to around a thousand bits, but it was much smaller, faster, and more power-efficient than using individual vacuum tube latches. Developed at the University of Manchester in England, the Williams tube provided the medium on which the first electronically stored-memory program was implemented in the Manchester Small-Scale Experimental Machine (SSEM) computer, which first successfully ran a program on 21 June 1948.[5] In fact, rather than the Williams tube memory being designed for the SSEM, the SSEM was a testbed to demonstrate the reliability of the memory.[6][7]

Magnetic-core memory was invented in 1947 and developed up until the mid-1970s. It became a widespread form of random-access memory, relying on an array of magnetized rings. By changing the sense of each ring's magnetization, data could be stored with one bit stored per ring. Since every ring had a combination of address wires to select and read or write it, access to any memory location in any sequence was possible.

Magnetic core memory was the standard form of memory system until displaced by solid-state memory in integrated circuits, starting in the early 1970s. Robert H. Dennard invented dynamic random-access memory (DRAM) in 1968; this allowed replacement of a 4 or 6-transistor latch circuit by a single transistor for each memory bit, greatly increasing memory density at the cost of volatility. Data was stored in the tiny capacitance of each transistor, and had to be periodically refreshed every few milliseconds before the charge could leak away.

Prior to the development of integrated read-only memory (ROM) circuits, *permanent* (or *read-only*) random-access memory was often constructed using diode matrices driven by address decoders, or specially wound core rope memory planes.

5.2 Types of RAM

The two main forms of modern RAM are static RAM (SRAM) and dynamic RAM (DRAM). In SRAM, a bit of data is stored using the state of a six transistor memory cell. This form of RAM is more expensive to produce, but is generally faster and requires less power than DRAM and, in modern computers, is often used as cache memory for the CPU. DRAM stores a bit of data using a transistor and capacitor pair, which together comprise a DRAM memory cell. The capacitor holds a high or low charge (1 or 0, respectively), and the transistor acts as a switch that lets the control circuitry on the chip read the capacitor's state of charge or change it. As this form of memory is less expensive to produce than static RAM, it is the predominant form of computer memory used in modern computers.

Both static and dynamic RAM are considered *volatile*, as their state is lost or reset when power is removed from the system. By contrast, read-only memory (ROM) stores data by permanently enabling or disabling selected transistors, such that the memory cannot be altered. Writeable variants of ROM (such as EEPROM and flash memory) share properties of both ROM and RAM, enabling data to persist without power and to be updated without requiring special equipment. These persistent forms of semiconductor ROM include USB flash drives, memory cards for cameras and portable devices, etc. ECC memory (which can be either SRAM or DRAM) includes special circuitry to detect and/or correct random faults (memory errors) in the stored data, using parity bits or error correction code.

In general, the term *RAM* refers solely to solid-state memory devices (either DRAM or SRAM), and more specifically the main memory in most computers. In optical storage, the term DVD-RAM is somewhat of a misnomer since, unlike CD-RW or DVD-RW it does not need to be erased before reuse. Nevertheless a DVD-

RAM behaves much like a hard disc drive if somewhat slower.

5.3 Memory hierarchy

Main article: Memory hierarchy

One can read and over-write data in RAM. Many computer systems have a memory hierarchy consisting of CPU registers, on-die SRAM caches, external caches, DRAM, paging systems and virtual memory or swap space on a hard drive. This entire pool of memory may be referred to as "RAM" by many developers, even though the various subsystems can have very different access times, violating the original concept behind the *random access* term in RAM. Even within a hierarchy level such as DRAM, the specific row, column, bank, rank, channel, or interleave organization of the components make the access time variable, although not to the extent that rotating storage media or a tape is variable. The overall goal of using a memory hierarchy is to obtain the higher possible average access performance while minimizing the total cost of the entire memory system (generally, the memory hierarchy follows the access time with the fast CPU registers at the top and the slow hard drive at the bottom).

In many modern personal computers, the RAM comes in an easily upgraded form of modules called memory modules or DRAM modules about the size of a few sticks of chewing gum. These can quickly be replaced should they become damaged or when changing needs demand more storage capacity. As suggested above, smaller amounts of RAM (mostly SRAM) are also integrated in the CPU and other ICs on the motherboard, as well as in hard-drives, CD-ROMs, and several other parts of the computer system.

5.4 Other uses of RAM

In addition to serving as temporary storage and working space for the operating system and applications, RAM is used in numerous other ways.

5.4.1 Virtual memory

Main article: virtual memory

Most modern operating systems employ a method of extending RAM capacity, known as "virtual memory". A portion of the computer's hard drive is set aside for a *paging file* or a *scratch partition*, and the combination of physical RAM and the paging file form the system's total memory. (For example, if a computer has 2 GB of RAM and a 1 GB page file, the operating system has 3 GB total memory available to it.) When the system runs low on physical memory, it can "swap" portions of RAM to the paging file to make room for new data, as well as to read previously swapped information back into RAM. Excessive use of this mechanism results in thrashing and generally hampers overall system performance, mainly because hard drives are far slower than RAM.

5.4.2 RAM disk

Main article: RAM disk

Software can "partition" a portion of a computer's RAM, allowing it to act as a much faster hard drive that is called a RAM disk. A RAM disk loses the stored data when the computer is shut down, unless memory is arranged to have a standby battery source.

5.4.3 Shadow RAM

Sometimes, the contents of a relatively slow ROM chip are copied to read/write memory to allow for shorter access times. The ROM chip is then disabled while the initialized memory locations are switched in on the same block of addresses (often write-protected). This process, sometimes called *shadowing*, is fairly common in both computers and embedded systems.

As a common example, the BIOS in typical personal computers often has an option called "use shadow BIOS" or similar. When enabled, functions relying on data from the BIOS's ROM will instead use DRAM locations (most can also toggle shadowing of video card ROM or other ROM sections). Depending on the system, this may not result in increased performance, and may cause incompatibilities. For example, some hardware may be inaccessible to the operating system if shadow RAM is used. On some systems the benefit may be hypothetical because the BIOS is not used after booting in favor of direct hardware access. Free memory is reduced by the size of the shadowed ROMs.[8]

5.5 Recent developments

Several new types of *non-volatile* RAM, which will preserve data while powered down, are under development. The technologies used include carbon nanotubes and approaches utilizing the magnetic tunnel effect. Amongst the 1st generation MRAM, a 128 KiB (128 × 2^{10} bytes) magnetic RAM (MRAM) chip was manufactured with 0.18 µm technology in the summer of 2003. In June 2004, Infineon Technologies unveiled a 16 MiB (16 × 2^{20} bytes) prototype again based on 0.18 µm technology. There are two 2nd generation techniques currently in development: Thermal Assisted Switching (TAS)[9] which is being developed by Crocus Technology, and Spin Torque Transfer (STT) on which Crocus, Hynix, IBM, and several other companies are working.[10] Nantero built a functioning carbon nanotube memory prototype 10 GiB (10×2^{30} bytes) array in 2004. Whether some of these technologies will be able to eventually take a significant market share from either DRAM, SRAM, or flash-memory technology, however, remains to be seen.

Since 2006, "solid-state drives" (based on flash memory) with capacities exceeding 256 gigabytes and performance far exceeding traditional disks have become available. This development has started to blur the definition between traditional random-access memory and “disks”, dramatically reducing the difference in performance.

Some kinds of random-access memory, such as “Eco-RAM”, are specifically designed for server farms, where low power consumption is more important than speed.[11]

5.6 Memory wall

The “memory wall” is the growing disparity of speed between CPU and memory outside the CPU chip. An important reason for this disparity is the limited communication bandwidth beyond chip boundaries. From 1986 to 2000, CPU speed improved at an annual rate of 55% while memory speed only improved at 10%. Given these trends, it was expected that memory latency would become an overwhelming bottleneck in computer performance.[12]

CPU speed improvements slowed significantly partly due to major physical barriers and partly because current CPU designs have already hit the memory wall in some sense. Intel summarized these causes in a 2005 document.[13]

> “First of all, as chip geometries shrink and clock frequencies rise, the transistor leakage current increases, leading to excess power consumption and heat... Secondly, the advantages of higher clock speeds are in part negated by memory latency, since memory access times have not been able to keep pace with increasing clock frequencies. Third, for certain applications, traditional serial architectures are becoming less efficient as processors get faster (due to the so-called Von Neumann bottleneck), further undercutting any gains that frequency increases might otherwise buy. In addition, partly due to limitations in the means of producing inductance within solid state devices, resistance-capacitance (RC) delays in signal transmission are growing as feature sizes shrink, imposing an additional bottleneck that frequency increases don't address.”

The RC delays in signal transmission were also noted in Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures which projects a maximum of 12.5% average annual CPU performance improvement between 2000 and 2014.

A different concept is the processor-memory performance gap, which can be addressed by 3D computer chips that reduce the distance between the logic and memory aspects that are further apart in a 2D chip.[14] Memory subsystem design requires a focus on the gap, which is widening over time.[15] The main method of bridging the gap is the use of caches; small amounts of high-speed memory that houses recent operations and instructions nearby the processor, speeding up the execution of those operations or instructions in cases where they are called upon frequently. Multiple levels of caching have been developed in order to deal with the widening of the gap, and the performance of high-speed modern computers are reliant on evolving caching techniques.[16] These can prevent the loss of performance that the processor has, as it takes less time to perform the computation it has been initiated to complete.[17] There can be up to a 53% difference between the growth in speed of processor speeds and the lagging speed of main memory access.[18]

5.7 See also

- CAS latency (CL)
- Hybrid Memory Cube
- Multi-channel memory architecture
- Registered/buffered memory
- RAM parity
- Memory Interconnect/RAM buses
- Memory geometry

5.8 Notes and references

[1] “RAM, ROM, and Flash Memory”. *For Dummies*. Retrieved March 1, 2015.

[2] Gallagher, Sean. “Memory that never forgets: non-volatile DIMMs hit the market”. Ars Technica.

[3] Bellis, Mary. “The Invention of the Intel 1103”.

[4] “IBM Archives -- FAQ’s for Products and Services”. *ibm.com*.

[5] Napper, Brian, *Computer 50: The University of Manchester Celebrates the Birth of the Modern Computer*, retrieved 26 May 2012

[6] Williams, F.C.; Kilburn, T. (Sep 1948), “Electronic Digital Computers”, *Nature* **162** (4117): 487, doi:10.1038/162487a0. Reprinted in *The Origins of Digital Computers*

[7] Williams, F.C.; Kilburn, T.; Tootill, G.C. (Feb 1951), "Universal High-Speed Digital Computers: A Small-Scale Experimental Machine", *Proc. IEE* **98** (61): 13–28, doi:10.1049/pi-2.1951.0004.

[8] "Shadow Ram". Retrieved 2007-07-24.

[9] The Emergence of Practical MRAM http://www.crocus-technology.com/pdf/BH%20GSA%20Article.pdf

[10] "Tower invests in Crocus, tips MRAM foundry deal". *EE-Times*.

[11] "EcoRAM held up as less power-hungry option than DRAM for server farms" by Heather Clancy 2008

[12] The term was coined in .

[13] "Platform 2015: Intel® Processor and Platform Evolution for the Next Decade". March 2, 2005.

[14] Rainer Waser (2012). *Nanoelectronics and Information Technology*. John Wiley & Sons. p. 790. Retrieved March 31, 2014.

[15] Chris Jesshope and Colin Egan (2006). *Advances in Computer Systems Architecture: 11th Asia-Pacific Conference, ACSAC 2006, Shanghai, China, September 6-8, 2006, Proceedings*. Springer. p. 109. Retrieved March 31, 2014.

[16] Ahmed Amine Jerraya and Wayne Wolf (2005). *Multiprocessor Systems-on-chips*. Morgan Kaufmann. p. 90-91. Retrieved March 31, 2014.

[17] *Impact of Advances in Computing and Communications Technologies on Chemical Science and Technology*. National Academy Press. 1999. p. 110. Retrieved March 31, 2014.

[18] Celso C. Ribeiro and Simone L. Martins (2004). *Experimental and Efficient Algorithms: Third International Workshop, WEA 2004, Angra Dos Reis, Brazil, May 25-28, 2004, Proceedings, Volume 3*. Springer. p. 529. Retrieved March 31, 2014.

5.9 External links

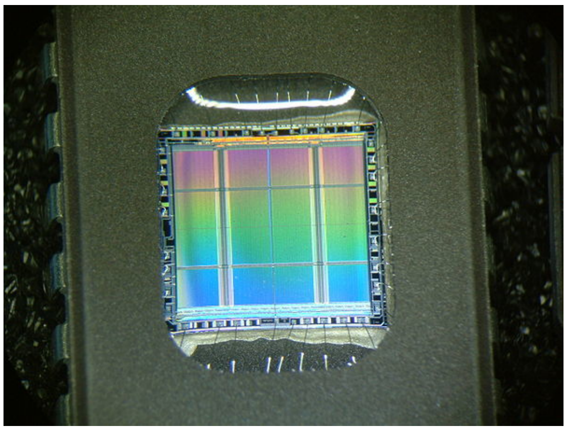
- Media related to RAM at Wikimedia Commons
- Memory Prices (1957–2014)

Chapter 6

Read-only memory

"Read Only Memory" redirects here. For the EP by Chrome, see Read Only Memory (EP).
The notion of read-only data can also refer to file system permissions.

Read-only memory (**ROM**) is a class of storage medium used in computers and other electronic devices. Data stored in ROM can only be modified slowly, with difficulty, or not at all, so it is mainly used to distribute firmware (software that is very closely tied to specific hardware, and unlikely to need frequent updates).



an EPROM

Strictly, *read-only memory* refers to memory that is hard-wired, such as diode matrix and the later mask ROM. Although discrete circuits can be altered (in principle), Integrated Circuits (ICs) cannot and are useless if the data is bad. The fact that such memory can never be changed is a large drawback; more recently, *ROM* commonly refers to memory that is read-only in normal operation, while reserving the fact of some possible way to change it.

Other types of non-volatile memory such as erasable programmable read only memory (EPROM) and electrically erasable programmable read-only memory (EEPROM or Flash ROM) are sometimes referred to, in an abbreviated way, as "read-only memory" (ROM); although these types of memory can be erased and re-programmed multiple times, writing to this memory takes longer and may require different procedures than reading the memory.[1] When used in this less precise way, "ROM" indicates a *non-volatile* memory which serves functions typically provided by mask ROM, such as storage of program code and nonvolatile data.

6.1 History

Many game consoles use interchangeable ROM cartridges, allowing for one system to play multiple games.

Read-only memory was used for Jacquard looms.[2]

The simplest type of solid state ROM is as old as semiconductor technology itself. Combinational logic gates can be joined manually to map n-bit **address** input onto arbitrary values of m-bit **data** output (a look-up table). With the invention of the integrated circuit came mask ROM. Mask ROM consists of a grid of word lines (the address input) and bit lines (the data output), selectively joined together with transistor switches, and can represent an arbitrary look-up table with a regular physical layout and predictable propagation delay.

In mask ROM, the data is physically encoded in the circuit, so it can only be programmed during fabrication.

This leads to a number of serious disadvantages:

1. It is only economical to buy mask ROM in large quantities, since users must contract with a foundry to produce a custom design.

2. The turnaround time between completing the design for a mask ROM and receiving the finished product is long, for the same reason.

3. Mask ROM is impractical for R&D work since designers frequently need to modify the contents of memory as they refine a design.

4. If a product is shipped with faulty mask ROM, the only way to fix it is to recall the product and physically replace the ROM in every unit shipped.

Subsequent developments have addressed these shortcomings. PROM, invented in 1956, allowed users to program its contents exactly once by physically altering its structure with the application of high-voltage pulses. This addressed problems 1 and 2 above, since a company can simply order a large batch of fresh PROM chips and program them with the desired contents at its designers' convenience. The 1971 invention of EPROM essentially solved problem 3, since EPROM (unlike PROM) can be repeatedly reset to its unprogrammed state by exposure to strong ultraviolet light. EEPROM, invented in 1983, went a long way to solving problem 4, since an EEPROM can be programmed in-place if the containing device provides a means to receive the program contents from an external source (for example, a personal computer via a serial cable). Flash memory, invented at Toshiba in the mid-1980s, and commercialized in the early 1990s, is a form of EEPROM that makes very efficient use of chip area and can be erased and reprogrammed thousands of times without damage.

All of these technologies improved the flexibility of ROM, but at a significant cost-per-chip, so that in large quantities mask ROM would remain an economical choice for many years. (Decreasing cost of reprogrammable devices had almost eliminated the market for mask ROM by the year 2000.) Rewriteable technologies were envisioned as replacements for mask ROM.

The most recent development is NAND flash, also invented at Toshiba. Its designers explicitly broke from past practice, stating plainly that "the aim of NAND Flash is to replace hard disks,"[3] rather than the traditional use of ROM as a form of non-volatile primary storage. As of 2007, NAND has partially achieved this goal by offering throughput comparable to hard disks, higher tolerance of physical shock, extreme miniaturization (in the form of USB flash drives and tiny microSD memory cards, for example), and much lower power consumption.

6.1.1 Use for storing programs

Every stored-program computer may use a form of non-volatile storage (that is, storage that retains its data when power is removed) to store the initial program that runs when the computer is powered on or otherwise begins execution (a process known as bootstrapping, often abbreviated to "booting" or "booting up"). Likewise, every non-trivial computer needs some form of mutable memory to record changes in its state as it executes.

Forms of read-only memory were employed as non-volatile storage for programs in most early stored-program computers, such as ENIAC after 1948. (Until then it was not a stored-program computer as every program had to be manually wired into the machine, which could take days to weeks.) Read-only memory was simpler to implement since it needed only a mechanism to read stored values, and not to change them in-place, and thus could be implemented with very crude electromechanical devices (see historical examples below). With the advent of integrated circuits in the 1960s, both ROM and its mutable counterpart static RAM were implemented as arrays of transistors in silicon chips; however, a ROM memory cell could be implemented using fewer transistors than an SRAM memory cell, since the latter needs a latch (comprising 5-20 transistors) to retain its contents, while a ROM cell might consist of the absence (logical 0) or presence (logical 1) of one transistor connecting a bit line to a word line.[4] Consequently, ROM could be implemented at a lower cost-per-bit than RAM for many years.

Most home computers of the 1980s stored a BASIC interpreter or operating system in ROM as other forms of non-volatile storage such as magnetic disk drives were too costly. For example, the Commodore 64 included 64 KB of RAM and 20 KB of ROM contained a BASIC interpreter and the "KERNAL" of its operating system. Later home or office computers such as the IBM PC XT often included magnetic disk drives, and larger amounts of RAM, allowing them to load their operating systems from disk into RAM, with only a minimal hardware initialization core and bootloader remaining in ROM (known as the BIOS in IBM-compatible computers). This arrangement allowed for a more complex and easily upgradeable operating system.

In modern PCs, "ROM" (or flash) is used to store the basic bootstrapping firmware for the main processor, as well as the various firmware needed to internally control self-contained devices such as graphic cards, hard disks, DVD drives, TFT screens, etc., in the system. Today, many of these "read-only" memories – especially the BIOS – are often replaced with Flash memory (see below), to permit in-place reprogramming should the need for a firmware upgrade arise. However, simple and mature sub-systems (such as the keyboard or some communication controllers in the integrated circuits on the main board, for example) may employ mask ROM or OTP (one-time programmable).

ROM and successor technologies such as flash are prevalent in embedded systems. These are in everything from industrial robots to home appliances and consumer electronics (MP3 players, set-top boxes, etc.) all of which are designed for specific functions, but are based on general-purpose microprocessors. With software usually tightly coupled to hardware, program changes are rarely needed in such devices (which typically lack hard disks for reasons of cost, size, or power consumption). As of 2008, most products use Flash rather than mask ROM, and many provide some means for connecting to a PC for firmware updates; for example, a digital audio player might be updated to support a new file format. Some hobbyists have taken advantage of this flexibility to reprogram consumer products for new purposes; for example, the iPodLinux and OpenWrt projects have enabled users to run full-featured Linux distributions on their MP3 players and wireless routers, respectively.

ROM is also useful for binary storage of cryptographic data, as it makes them difficult to replace, which may be desirable in order to enhance information security.

6.1.2 Use for storing data

Since ROM (at least in hard-wired mask form) cannot be modified, it is really only suitable for storing data which is not expected to need modification for the life of the device. To that end, ROM has been used in many computers to store look-up tables for the evaluation of mathematical and logical functions (for example, a floating-point unit might tabulate the sine function in order to facilitate faster computation). This was especially effective when CPUs were slow and ROM was cheap compared to RAM.

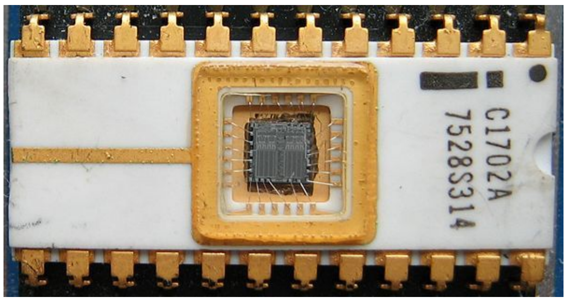
Notably, the display adapters of early personal computers stored tables of bitmapped font characters in ROM. This usually meant that the text display font could not be changed interactively. This was the case for both the CGA and MDA adapters available with the IBM PC XT.

The use of ROM to store such small amounts of data has disappeared almost completely in modern general-purpose computers. However, Flash ROM has taken over a new role as a medium for mass storage or secondary storage of files.

6.2 Types

6.2.1 Semiconductor based

Classic *mask-programmed ROM* chips are integrated circuits that physically encode the data to be stored, and thus it is impossible to change their contents after fabrication. Other types of non-volatile solid-state memory permit some degree of modification:



The first EPROM, an Intel 1702, with the die and wire bonds clearly visible through the erase window.

- *Programmable read-only memory* (PROM), or *one-time programmable ROM* (OTP), can be written to or *programmed* via a special device called a *PROM programmer*. Typically, this device uses high voltages to permanently destroy or create internal links (fuses or antifuses) within the chip. Consequently, a PROM can only be programmed once.
- *Erasable programmable read-only memory* (EPROM) can be erased by exposure to strong ultraviolet light (typically for 10 minutes or longer), then rewritten with a process that again needs higher than usual voltage applied. Repeated exposure to UV light will eventually wear out an EPROM, but the *endurance* of most EPROM chips exceeds 1000 cycles of erasing and reprogramming. EPROM chip packages can often be identified by the prominent quartz “window” which allows UV light to enter. After programming, the window is typically covered with a label to prevent accidental erasure. Some EPROM chips are factory-erased before they are packaged, and include no window; these are effectively PROM.
- *Electrically erasable programmable read-only memory* (EEPROM) is based on a similar semiconductor structure to EPROM, but allows its entire contents (or selected *banks*) to be electrically erased, then rewritten electrically, so that they need not be removed from the computer (or camera, MP3 player, etc.). Writing or *flashing* an EEPROM is much slower (milliseconds per bit) than reading from a ROM or writing to a RAM (nanoseconds in both cases).
  - *Electrically alterable read-only memory* (EAROM) is a type of EEPROM that can be modified one bit at a time. Writing is a very slow process and again needs higher voltage (usually around 12 V) than is used for read access. EAROMs are intended for applications that require infrequent and only partial rewriting. EAROM may be used as non-volatile storage for critical system setup information; in many applications, EAROM has been supplanted by CMOS RAM supplied

by mains power and backed-up with a lithium battery.

- *Flash memory* (or simply *flash*) is a modern type of EEPROM invented in 1984. Flash memory can be erased and rewritten faster than ordinary EEPROM, and newer designs feature very high endurance (exceeding 1,000,000 cycles). Modern NAND flash makes efficient use of silicon chip area, resulting in individual ICs with a capacity as high as 32 GB as of 2007; this feature, along with its endurance and physical durability, has allowed NAND flash to replace magnetic in some applications (such as USB flash drives). Flash memory is sometimes called *flash ROM* or *flash EEPROM* when used as a replacement for older ROM types, but not in applications that take advantage of its ability to be modified quickly and frequently.

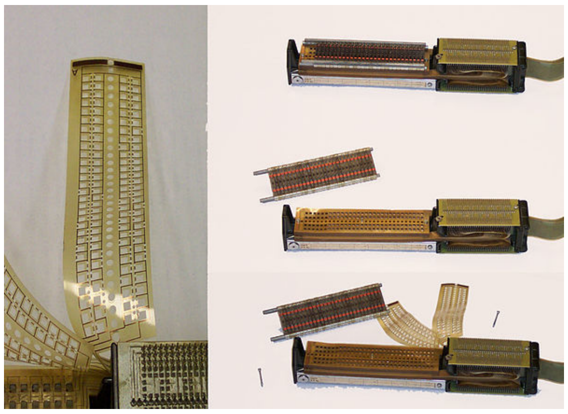
By applying write protection, some types of reprogrammable ROMs may temporarily become read-only memory.

6.2.2 Other technologies

There are other types of non-volatile memory which are not based on solid-state IC technology, including:

- Optical storage media, such CD-ROM which is read-only (analogous to masked ROM). CD-R is Write Once Read Many (analogous to PROM), while CD-RW supports erase-rewrite cycles (analogous to EEPROM); both are designed for backwards-compatibility with CD-ROM.

Historical examples



Transformer matrix ROM (TROS), from the IBM System 360/20

- Diode matrix ROM, used in small amounts in many computers in the 1960s as well as electronic desk calculators and keyboard encoders for terminals. This ROM was programmed by installing discrete semiconductor diodes at selected locations between a matrix of *word line traces* and *bit line traces* on a printed circuit board.

- Resistor, capacitor, or transformer matrix ROM, used in many computers until the 1970s. Like diode matrix ROM, it was programmed by placing components at selected locations between a matrix of *word lines* and *bit lines*. ENIAC's Function Tables were resistor matrix ROM, programmed by manually setting rotary switches. Various models of the IBM System/360 and complex peripheral devices stored their microcode in either capacitor (called *BCROS* for *balanced capacitor read-only storage* on the 360/50 and 360/65, or *CCROS* for *charged capacitor read-only storage* on the 360/30) or transformer (called *TROS* for *transformer read-only storage* on the 360/20, 360/40 and others) matrix ROM.

- Core rope, a form of transformer matrix ROM technology used where size and weight were critical. This was used in NASA/MIT's Apollo Spacecraft Computers, DEC's PDP-8 computers, and other places. This type of ROM was programmed by hand by weaving “word line wires” inside or outside of ferrite transformer cores.

- Dimond Ring stores, in which wires are threaded through a sequence of large ferrite rings that function only as sensing devices. These were used in TXE telephone exchanges.

- The perforated metal character mask ("stencil") in Charactron cathode ray tubes, which was used as ROM to shape a wide electron beam to form a selected character shape on the screen either for display or a scanned electron beam to form a selected character shape as an overlay on a video signal.

6.3 Speed

6.3.1 Reading

Although the relative speed of RAM vs. ROM has varied over time, as of 2007 large RAM chips can be read faster than most ROMs. For this reason (and to allow uniform access), ROM content is sometimes copied to RAM or **shadowed** before its first use, and subsequently read from RAM.

6.3.2 Writing

For those types of ROM that can be electrically modified, writing speed is always much slower than reading speed, and it may need unusually high voltage, the movement of jumper plugs to apply write-enable signals, and special lock/unlock command codes. Modern NAND Flash achieves the highest write speeds of any rewritable ROM technology, with speeds as high as 15 MB/s (or 70 ns/bit), by allowing (needing) large blocks of memory cells to be written simultaneously.

6.4 Endurance and data retention

Because they are written by forcing electrons through a layer of electrical insulation onto a floating transistor gate, rewriteable ROMs can withstand only a limited number of write and erase cycles before the insulation is permanently damaged. In the earliest EAROMs, this might occur after as few as 1,000 write cycles, while in modern Flash EEPROM the **endurance** may exceed 1,000,000, but it is by no means infinite. This limited endurance, as well as the higher cost per bit, means that Flash-based storage is unlikely to completely supplant magnetic disk drives in the near future.

The timespan over which a ROM remains accurately readable is not limited by write cycling. The **data retention** of EPROM, EAROM, EEPROM, and Flash *may* be limited by charge leaking from the floating gates of the memory cell transistors. Leakage is accelerated by high temperatures or radiation. Masked ROM and fuse/antifuse PROM do not suffer from this effect, as their data retention depends on physical rather than electrical permanence of the integrated circuit (although *fuse re-growth* was once a problem in some systems).

6.5 Content images

Main article: ROM image

The contents of ROM chips in video game console cartridges can be extracted with special software or hardware devices. The resultant memory dump files are known as **ROM images**, and can be used to produce duplicate cartridges, or in console emulators. The term originated when most console games were distributed on cartridges containing ROM chips, but achieved such widespread usage that it is still applied to images of newer games distributed on CD-ROMs or other optical media.

ROM images of commercial games usually contain copyrighted software. The unauthorized copying and distribution of copyrighted software is usually a violation of copyright laws (in some jurisdictions, duplication of ROM cartridges for backup purposes may be considered fair use). Nevertheless, there is a thriving community engaged in the illegal distribution and trading of such software and abandonware. In such circles, the term “ROM images” is sometimes shortened simply to “ROMs” or sometimes changed to “romz” to highlight the connection with "warez".

6.6 See also

- EEPROM
- EPROM
- Flash memory
- PROM
- Random access memory
- Write-only memory

6.7 Terminology

- EEPROM: electrically erasable programmable read-only memory
- EPROM: erasable programmable read-only memory
- PROM: programmable read-only memory

6.8 References

[1] Definition of: flash ROM, PC Magazine.

[2] “History of Computation - Babbage, Boole, Hollerith”. Paul E. Dunne.

[3] See page 6 of Toshiba’s 1993 *NAND Flash Applications Design Guide*.

[4] See chapters on “Combinatorial Digital Circuits” and “Sequential Digital Circuits” in Millman & Grable, *Microelectronics,* 2nd ed.

Chapter 7

BIOS

This article is about the BIOS as found in IBM PC compatibles. For the general concept, see firmware. For similar programs on non-PC systems, see booting. For other uses, see Bios (disambiguation).

The **BIOS** (/ˈbaɪ.ɒs/, an acronym for **Basic Input/Output System** and also known as the **System BIOS**, **ROM BIOS** or **PC BIOS**) is a type of firmware used during the booting process (power-on startup) on IBM PC compatible computers.[1] The BIOS firmware is built into PCs, and it is the first software they run when powered on. The name itself originates from the Basic Input/Output System used in the CP/M operating system in 1975.[2][3] Originally proprietary to the IBM PC, the BIOS was reverse engineered by companies looking to create compatible systems and the interface of that original system serves as a *de facto* standard.

The fundamental purposes of the BIOS are to initialize and test the system hardware components, and to load a boot loader or an operating system from a mass memory device. The BIOS additionally provides an abstraction layer for the hardware, i.e. a consistent way for application programs and operating systems to interact with the keyboard, display, and other input/output devices. Variations in the system hardware are hidden by the BIOS from programs that use BIOS services instead of directly accessing the hardware. MS-DOS (PC DOS), which was the dominant PC operating system from the early 1980s until the mid 1990s, relied on BIOS services for disk, keyboard, and text display functions. MS Windows NT, Linux, and other protected mode operating systems in general ignore the abstraction layer provided by the BIOS and do not use it after loading, instead accessing the hardware components directly.

Every BIOS implementation is specifically designed to work with a particular computer or motherboard model, by interfacing with various devices that make up the complementary system chipset. Originally, BIOS firmware was stored in a ROM chip on the PC motherboard; in modern computer systems, the BIOS contents are stored on flash memory so it can be rewritten without removing the chip from the motherboard. This allows easy updates to the BIOS firmware so new features can be added or bugs can be fixed, but it also creates a possibility for the computer to become infected with BIOS rootkits.

Unified Extensible Firmware Interface (UEFI) was designed as a successor to BIOS, aiming to address its technical shortcomings.[4] As of 2014, new PC hardware predominantly ships with UEFI firmware.

7.1 History

The term BIOS (Basic Input/Output System) was invented by Gary Kildall[5] and first appeared in the CP/M operating system in 1975,[2][3][6][7] describing the machine-specific part of CP/M loaded during boot time that interfaces directly with the hardware.[3] (A CP/M machine usually has only a simple boot loader in its ROM.)

Versions of MS-DOS, PC DOS or DR-DOS contain a file called variously "IO.SYS", "IBMBIO.COM", “IBMBIO.SYS”, or "DRBIOS.SYS"; this file is known as the "DOS BIOS" (also known as "DOS I/O System") and contains the lower-level hardware-specific part of the operating system. Together with the underlying hardware-specific, but operating system-independent “System BIOS”, which resides in ROM, it represents the analogous to the "CP/M BIOS".

With the introduction of PS/2 machines, IBM divided the System BIOS into real-mode and protected mode portions. The real-mode portion was meant to provide backward-compatibility with existing operating systems such as DOS, and therefore was named “CBIOS” (for Compatibility BIOS), whereas the “ABIOS” (for Advanced BIOS) provided new interfaces specifically suited for multitasking operating systems such as OS/2.

7.2 User interface

The BIOS of the original IBM PC XT had no interactive user interface. Error codes or messages were displayed on the screen, or coded series of sounds were generated to signal errors (when the POST had not proceeded to the point of successfully initializing a video display adapter). Options on the PC and XT were set by switches and jumpers on the main board and on peripheral cards. Starting around the mid-1990s, it became typical for the BIOS ROM to include a *"BIOS configuration utility"* (BCU[8]) or "BIOS setup utility", accessed at system power-up by a particular key sequence. This program allowed the user to set system configuration options, of the type formerly set using DIP switches, through an interactive menu system controlled through the keyboard. In the interim period, IBM-compatible PCs—including the IBM AT—held configuration settings in battery-backed RAM and used a bootable configuration program on disk, not in the ROM, to set the configuration options contained in this memory. The disk was supplied with the computer, and if it was lost the system settings could not be changed.

A modern Wintel-compatible computer provides a setup routine essentially unchanged in nature from the ROM-resident BIOS setup utilities of the late 1990s; the user can configure hardware options using the keyboard and video display. Also, when errors occur at boot time, a modern BIOS usually displays user-friendly error messages, often presented as pop-up boxes in a TUI style, and offers to enter the BIOS setup utility or to ignore the error and proceed if possible. Instead of battery-backed RAM, the modern Wintel machine may store the BIOS configuration settings in flash ROM, perhaps the same flash ROM that holds the BIOS itself.

7.3 Operation

7.3.1 System startup

Early Intel processors started at physical address 000FFFF0h. When a modern x86 microprocessor is reset, it starts in pseudo 16-bit real mode, initializing most registers to zero. The code segment register is initialized with selector F000h, base FFFF0000h, and limit FFFFh, so that execution starts at 4 GB minus 16 bytes (FFFFFFF0h).[9] The platform logic maps this address into the system ROM, mirroring address 000FFFF0h.

If the system has just been powered up or the reset button was pressed ("cold boot"), the full power-on self-test (POST) is run. If Ctrl+Alt+Delete was pressed ("warm boot"), a special flag value is stored in nonvolatile BIOS memory ("CMOS") before the processor is reset, and after the reset the BIOS startup code detects this flag and does not run the POST. This saves the time otherwise used to detect and test all memory.

The POST checks, identifies, and initializes system devices such as the CPU, RAM, interrupt and DMA controllers and other parts of the chipset, video display card, keyboard, hard disk drive, optical disc drive and other basic hardware.

Early IBM PCs had a little-known routine in the POST that would attempt to download a program from into RAM through the keyboard port. (Note that no serial or parallel ports were standard on early IBM PCs, but a keyboard port of either the XT or AT / PS/2 type has been standard on practically every PC and clone.) If the download was apparently successful, the BIOS would verify a checksum on it and then run it.[10] This feature was intended for factory test or diagnostic purposes. While it was of limited utility outside of factory or repair facilities, it could be used in a proprietary way to boot the PC as a satellite system to a host machine (which is essentially the same technical way it was used, if it was used, in the manufacturing environment).

7.3.2 Boot process

After the option ROM scan is completed and all detected ROM modules with valid checksums have been called, or immediately after POST in a BIOS version that does not scan for option ROMs, the BIOS calls INT 19h to start boot processing. Post-boot, Programs loaded can also call INT 19h to reboot the system, but they must be careful to disable interrupts and other asynchronous hardware processes that may interfere with the BIOS rebooting process, or else the system may hang or crash while it is rebooting. Unique to the original IBM BIOS was that it would attempt to load a maintenance program through the keyboard port before performing any other elements of the boot process, such as before scanning for option ROMs or executing a boot loader.[11]

When INT 19h is called, the BIOS attempts to locate boot loader software held on a storage device designated as a "boot device", such as a hard disk, a floppy disk, CD, or DVD. It loads and executes the first boot software it finds, giving it control of the PC.[12] This is the process that is known as *booting* (sometimes informally called "booting up"), which is short for "bootstrapping".

The BIOS selects candidate boot devices using information collected by POST and configuration information from EEPROM, CMOS RAM or, in the earliest PCs, DIP switches. Following the boot priority sequence in effect, BIOS checks each device in order to see if it is bootable. For a disk drive or a device that logically emulates a disk drive, such as a USB Flash drive or perhaps a tape drive, to perform this check the BIOS attempts to load the first sector (boot sector) from the disk into RAM at memory address 0x0000:0x7C00. If the sector cannot be read (due to a missing or unformatted disk, or due to a hardware failure), the BIOS considers the device unbootable and proceeds to check the next device. If the sector is read successfully, some BIOSes will also check for the boot sector signature 0x55 0xAA in the last two bytes of the (512 byte long) sector, before accepting a boot sector and considering the device bootable.[nb 1]

The BIOS proceeds to test each device sequentially until a bootable device is found, at which time the BIOS transfers control to the loaded sector with a jump instruction to its first byte at address 0x0000:0x7C00 (exactly 1 KiB below the 32 KiB mark); see MBR invocation and VBR invocation. (This location is one reason that an IBM PC requires at least 32 KiB of RAM in order to be equipped with a disk system; with 31 KiB or less, it would be impossible to boot from any disk, removable or fixed, using the BIOS boot protocol.) Most, but not all, BIOSes load the drive number (as used by INT 13h) of the boot drive into CPU register DL before jumping to the first byte of the loaded boot sector.

Note well that the BIOS does not interpret or process the contents of the boot sector other than to possibly check for the boot sector signature in the last two bytes; all interpretation of data structures like MBR partition tables and so-called BIOS Parameter Blocks is done by the boot program in the boot sector itself or by other programs loaded through the boot process and is beyond the scope of BIOS. Nothing about BIOS predicates these data structures or impedes their replacement or improvement.

A non-disk device such as a network adapter attempts booting by a procedure that is defined by its option ROM or the equivalent integrated into the motherboard BIOS ROM. As such, option ROMs may also influence or supplant the boot process defined by the motherboard BIOS ROM.

Boot priority

The user can control the boot process, to cause one medium to be booted instead of another when two or more bootable media are present, by taking advantage of the boot priority implemented by the BIOS. For example, most computers have a hard disk that is bootable, but usually there is a removable-media drive that has higher boot priority, so the user can cause a removable disk to be booted, simply by inserting it, without removing the hard disk drive or altering its contents to make it unbootable.

In most modern BIOSes, the boot priority order of all potentially bootable devices can be freely configured by the user through the BIOS configuration utility. In older BIOSes, limited boot priority options are selectable; in the earliest BIOSes, a fixed priority scheme was implemented, with floppy disk drives first, fixed disks (i.e. hard disks) second, and typically no other boot devices supported, subject to modification of these rules by installed option ROMs. The BIOS in an early PC also usually would only boot from the first floppy disk drive or the first hard disk drive, even if there were two drives of either type installed. All more advanced boot priority sequences evolved as incremental improvements on this basic system.

Historically the BIOS would try to boot from a floppy drive first and a hard disk second. The default for CD or DVD booting is an extension of this. With the El Torito optical media boot standard, the optical drive actually emulates a 3.5" high-density floppy disk to the BIOS for boot purposes. Optical disks are a special case, because their lowest level of data organization is typically a fairly high-level file system (e.g. ISO 9660 for CD-ROM).

Reading the “first sector” of a CD-ROM or DVD-ROM is not a simply defined operation like it is on a floppy disk or a hard disk. Furthermore, the complexity of the medium makes it difficult to write a useful boot program in one sector, even though optical media sectors are typically 2048 bytes each, four times the standard 512-byte size of floppy and legacy hard disk sectors. Therefore, optical media booting uses the El Torito standard, which specifies a way for an optical disk to contain an image of a high-density (1.44 MB) floppy disk and for the drive to provide access to this disk image in a simple manner that emulates floppy disk drive operations. Therefore, CD-ROM drives boot as emulated floppy disk drives; the bootable virtual floppy disk can contain software that provides access to the optical medium in its native format.

Boot failure

The behavior if the BIOS does not find a bootable device has varied as personal computers developed. The original IBM PC and XT had Microsoft Cassette BASIC in ROM, and if no bootable device was found, ROM BASIC was started by calling INT 18h. Therefore, barring a hardware failure, an original IBM PC or XT would never fail to boot, either into BASIC or from disk (or through an option ROM). One model of the original IBM PC was available with no disk drive; a cassette recorder could be attached via the cassette port on the rear, for loading and saving BASIC programs to tape. Since few programs used BASIC in ROM, clone PC makers left it out; then a computer that failed to boot from a disk would display “No ROM BASIC” and halt (in response to INT 18h).

Later computers would display a message like “No bootable disk found"; some would prompt for a disk to be inserted and a key to be pressed, and when a key was pressed they would restart the boot process. A modern BIOS may display nothing or may automatically enter the BIOS configuration utility when the boot process fails. Unlike earlier BIOSes, modern versions are often written with the assumption that if the computer cannot be booted from a hard disk, the user will not have software that they want to boot from removable media instead. (Lately, typically it will only be a specialist computer technician who does that, only to get the computer back into a condition where it can be booted from the hard disk.)

7.3.3 Boot environment

The environment for the boot program is very simple: the CPU is in real mode and the general-purpose and segment registers are undefined, except CS, SS, SP, and DL. CS is always zero and IP is initially 0x7C00. Because boot programs are always loaded at this fixed address, there is no need or motivation for a boot program to be relocatable. DL contains the drive number, as used with INT 13h, of the boot device, unless the BIOS is one that does not set the drive number in DL – and then DL is undefined. SS:SP points to a valid stack that is presumably large enough to support hardware interrupts, but otherwise SS and SP are undefined. (A stack must be already set up in order for interrupts to be serviced, and interrupts must be enabled in order for the system timer-tick interrupt, which BIOS always uses at least to maintain the time-of-day count and which it initializes during POST, to be active and for the keyboard to work. The keyboard works even if the BIOS keyboard service is not called; keystrokes are received and placed in the 15-character type-ahead buffer maintained by BIOS.) The boot program must set up its own stack (or at least MS-DOS 6 acts like it must), because the size of the stack set up by BIOS is unknown and its location is likewise variable; although the boot program can investigate the default stack by examining SS:SP, it is easier and shorter to just unconditionally set up a new stack.

At boot time, all BIOS services are available, and the memory below address 0x00400 contains the interrupt vector table. BIOS POST has initialized the system timers (8253 or 8254 IC), interrupt controller(s), DMA controller(s), and other motherboard/chipset hardware as necessary to bring all BIOS services to ready status. DRAM refresh for all system DRAM in conventional memory and extended memory, but not necessarily expanded memory, has been set up and is running. The interrupt vectors corresponding to the BIOS interrupts have been set to point at the appropriate entry points in the BIOS, hardware interrupt vectors for devices initialized by the BIOS have been set to point to the BIOS-provided ISRs, and some other interrupts, including ones that BIOS generates for programs to hook, have been set to a default dummy ISR that immediately returns. The BIOS maintains a reserved block of system RAM at addresses 0x00400–0x004FF with various parameters initialized during the POST. All memory at and above address 0x00500 can be used by the boot program; it may even overwrite itself.

7.4 Extensions (option ROMs)

Peripheral cards such as some hard disk drive controllers and some video display adapters have their own BIOS extension option ROMs, which provide additional functionality to BIOS. Code in these extensions runs before the BIOS boots the system from mass storage. These ROMs typically test and initialize hardware, add new BIOS services, and augment or replace existing BIOS services with their own versions of those services. For example, a SCSI controller usually has a BIOS extension ROM that adds support for hard drives connected through that controller. Some video cards have extension ROMs that replace the video services of the motherboard BIOS with their own video services. BIOS extension ROMs gain total control of the machine, so they can in fact do anything, and they may never return control to the BIOS that invoked them. An extension ROM could in principle contain an entire operating system or an application program, or it could implement an entirely different boot process such as booting from a network. Operation of an IBM-compatible computer system can be completely changed by removing or inserting an adapter card (or a ROM chip) that contains a BIOS extension ROM.

The motherboard BIOS typically contains code to access hardware components necessary for bootstrapping the system, such as the keyboard, display, and storage. In addition, plug-in adapter cards such as SCSI, RAID, network interface cards, and video boards often include their own BIOS (e.g. Video BIOS), complementing or replacing the system BIOS code for the given component. Even devices built into the motherboard can behave in this way; their option ROMs can be stored as separate code on the main BIOS flash chip, and upgraded either in tandem with, or separately from, the main BIOS.

An add-in card requires an option ROM if the card is not supported by the main BIOS and the card needs to be initialized or made accessible through BIOS services before the operating system can be loaded (usually this means it is required in the bootstrapping process). Even when it is not required, an option ROM can allow an adapter card to be used without loading driver software from a storage device after booting begins – with an option ROM, no time is taken to load the driver, the driver does not take up space in RAM nor on hard disk, and the driver software on the ROM always stays with the device so the two cannot be accidentally separated. Also, if the ROM is on the card, both the peripheral hardware and the driver software provided by the ROM are installed together with no extra effort to install the software. An additional advantage of ROM on some early PC systems (notably including the IBM PCjr) was that ROM was faster than main system RAM. (On modern systems, the case is very much the reverse of this, and BIOS ROM code is usually copied (“shadowed”) into RAM so it will run faster.)

There are many methods and utilities for examining the contents of various motherboard BIOS and expansion ROMs, such as Microsoft DEBUG or the Unix dd.

7.4.1 Boot procedure

If an expansion ROM wishes to change the way the system boots (such as from a network device or a SCSI adapter for which the BIOS has no driver code) in a cooperative way, it can use the *BIOS Boot Specification* (BBS) API to register its ability to do so. Once the expansion ROMs have registered using the BBS APIs, the user can select among the available boot options from within the

BIOS's user interface. This is why most BBS compliant PC BIOS implementations will not allow the user to enter the BIOS's user interface until the expansion ROMs have finished executing and registering themselves with the BBS API. The specification can be downloaded from the ACPICA website. The official title is BIOS Boot Specification (Version 1.01, 11 January 1996).[13]

Also, if an expansion ROM wishes to change the way the system boots unilaterally, it can simply hook INT 19h or other interrupts normally called from interrupt 19h, such as INT 13h, the BIOS disk service, to intercept the BIOS boot process. Then it can replace the BIOS boot process with one of its own, or it can merely modify the boot sequence by inserting its own boot actions into it, by preventing the BIOS from detecting certain devices as bootable, or both. Before the BIOS Boot Specification was promulgated, this was the only way for expansion ROMs to implement boot capability for devices not supported for booting by the native BIOS of the motherboard.

7.4.2 Initialization

After the motherboard BIOS completes its POST, most BIOS versions search for option ROM modules, also called BIOS extension ROMs, and execute them. The motherboard BIOS scans for extension ROMs in a portion of the "upper memory area" (the part of the x86 real-mode address space at and above address 0xA0000) and runs each ROM found, in order. To discover memory-mapped ISA option ROMs, a BIOS implementation scans the real-mode address space from 0x0C0000 to 0x0F0000 on 2 KiB boundaries, looking for a two-byte ROM *signature*: 0x55 followed by 0xAA. In a valid expansion ROM, this signature is followed by a single byte indicating the number of 512-byte blocks the expansion ROM occupies in real memory, and the next byte is the option ROM's entry point (also known as its "entry offset"). A checksum of the specified number of 512-byte blocks is calculated, and if the ROM has a valid checksum, the BIOS transfers control to the entry address, which in a normal BIOS extension ROM should be the beginning of the extension's initialization routine.

At this point, the extension ROM code takes over, typically testing and initializing the hardware it controls and registering interrupt vectors for use by post-boot applications. It may use BIOS services (including those provided by previously initialized option ROMs) to provide a user configuration interface, to display diagnostic information, or to do anything else that it requires. While the actions mentioned are typical behaviors of BIOS extension ROMs, each option ROM receives total control of the computer and may do anything at all, as noted with more detail in the Extensions section below; it is possible that an option ROM will not return to BIOS, pre-empting the BIOS's boot sequence altogether.

An option ROM should normally return to the BIOS after completing its initialization process. Once (and if) an option ROM returns, the BIOS continues searching for more option ROMs, calling each as it is found, until the entire option ROM area in the memory space has been scanned.

7.4.3 Physical placement

Option ROMs normally reside on adapter cards. However, the original PC, and perhaps also the PC XT, have a spare ROM socket on the motherboard (the "system board" in IBM's terms) into which an option ROM can be inserted, and the four ROMs that contain the BASIC interpreter can also be removed and replaced with custom ROMs which can be option ROMs. The IBM PCjr is unique among PCs in having two ROM cartridge slots on the front. Cartridges in these slots map into the same region of the upper memory area used for option ROMs, and the cartridges can contain option ROM modules that the BIOS would recognize. The cartridges can also contain other types of ROM modules, such as BASIC programs, that are handled differently. One PCjr cartridge can contain several ROM modules of different types, possibly stored together in one ROM chip.

7.5 Operating system services

The BIOS ROM is customized to the particular manufacturer's hardware, allowing low-level services (such as reading a keystroke or writing a sector of data to diskette) to be provided in a standardized way to programs, including operating systems. For example, an IBM PC might have either a monochrome or a color display adapter (using different display memory addresses and hardware), but a single, standard, BIOS system call may be invoked to display a character at a specified position on the screen in text mode or graphics mode.

The BIOS provides a small library of basic input/output functions to operate peripherals (such as the keyboard, rudimentary text and graphics display functions and so forth). When using MS-DOS, BIOS services could be accessed by an application program (or by MS-DOS) by executing an INT 13h interrupt instruction to access disk functions, or by executing one of a number of other documented BIOS interrupt calls to access video display, keyboard, cassette, and other device functions.

Operating systems and executive software that are designed to supersede this basic firmware functionality provide replacement software interfaces to application software. Applications can also provide these services to themselves. This began even in the 1980s under MS-DOS, when programmers observed that using the BIOS video services for graphics display was very slow. To increase the speed of screen output, many programs by-passed the BIOS and programmed the video display hardware directly. Other graphics programmers, particularly but not exclusively in the demoscene, observed that there were technical capabilities of the PC display adapters that were not supported by the IBM BIOS and could not be taken advantage of without circumventing it. Since the AT-compatible BIOS ran in Intel real mode, operating systems that ran in protected mode on 286 and later processors required hardware device drivers compatible with protected mode operation to replace BIOS services.

In modern personal computers running modern operating systems the BIOS is used only during booting and initial loading of system software. Before the operating system's first graphical screen is displayed, input and output are typically handled through BIOS. A boot menu such as the textual menu of Windows, which allows users to choose an operating system to boot, to boot into the safe mode, or to use the last known good configuration, is displayed through BIOS and receives keyboard input through BIOS.

However, it is also important to note that modern PCs can still boot and run legacy operating systems such as MS-DOS or DR-DOS that rely heavily on BIOS for their console and disk I/O. Thus, while not as central as they once were, the BIOS services are still important.

7.5.1 Processor microcode updates

Intel processors have reprogrammable microcode since the P6 microarchitecture.[14][15] The BIOS may contain patches to the processor microcode that fix errors in the initial processor microcode; reprogramming is not persistent, thus loading of microcode updates is performed each time the system is powered up. Without reprogrammable microcode, an expensive processor swap would be required;[16] for example, the Pentium FDIV bug became an expensive fiasco for Intel as it required a product recall because the original Pentium processor's defective microcode could not be reprogrammed.

7.5.2 Identification

Some BIOSes contain a *software licensing description table* (SLIC), a digital signature placed inside the BIOS by the manufacturer, for example Dell. The SLIC is inserted into the ACPI table and contains no active code.[17][18]

Computer manufacturers that distribute OEM versions of Microsoft Windows and Microsoft application software can use the SLIC to authenticate licensing to the OEM Windows Installation disk and system recovery disc containing Windows software. Systems having a SLIC can be preactivated with an OEM product key, and they verify an XML formatted OEM certificate against the SLIC in the BIOS as a means of self-activating (see System Locked Preinstallation, SLP). If a user performs a fresh install of Windows, they will need to have possession of both the OEM key (either SLP or COA) and the digital certificate for their SLIC in order to bypass activation.[17] This can be achieved if the user performs a restore using a pre-customised image provided by the OEM. Power users can copy the necessary certificate files from the OEM image, decode the SLP product key, then perform SLP activation manually. Cracks for non-genuine Windows distributions usually edit the SLIC or emulate it in order to bypass Windows activation.

7.5.3 Overclocking

Some BIOS implementations allow overclocking, an action in which the CPU is adjusted to a higher clock rate than its manufacturer rating for guaranteed capability. Overclocking may, however, seriously compromise system reliability in insufficiently cooled computers and generally shorten component lifespan. Overclocking, when incorrectly performed, may also cause components to overheat so quickly that they mechanically destroy themselves.[19]

7.5.4 Modern use

Some operating systems, for example MS-DOS, rely on the BIOS to carry out most input/output tasks within the PC.[20]

Because the BIOS still runs in 16-bit real mode, calling BIOS services directly is inefficient for protected-mode operating systems. BIOS services are not used by modern multitasking GUI operating systems after they initially load, so the importance of the primary part of BIOS is greatly reduced from what it was initially.

Later BIOS implementations took on more complex functions, by including interfaces such as Advanced Configuration and Power Interface (ACPI); these functions include power management, hot swapping, and thermal management. At the same time, since 2010 BIOS technology is in a transitional process toward the UEFI.[4]

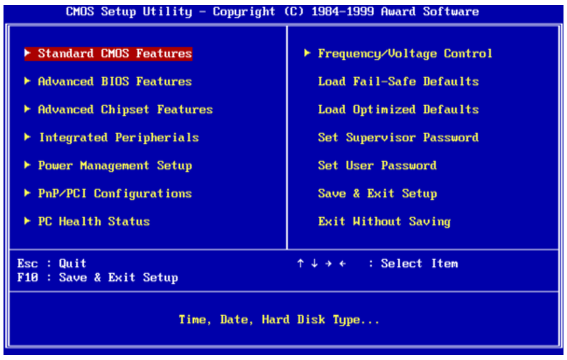
7.6 Configuration

7.6.1 Setup utility

Historically, the BIOS in the IBM PC and XT had no built-in user interface. The BIOS versions in earlier PCs (XT-class) were not software configurable; instead, users set the options via DIP switches on the motherboard. Later computers, including all IBM-compatibles with 80286 CPUs, had a battery-backed nonvolatile BIOS memory (CMOS RAM chip) that held BIOS settings.[21] These settings, such as video-adapter type, memory size, and hard-disk parameters, could only be configured by running a configuration program from a disk, not built into the ROM. A special “reference diskette” was inserted in an IBM AT to configure settings such as memory size.

Early BIOS versions did not have passwords or boot-device selection options. The BIOS was hard-coded to boot from the first floppy drive, or, if that failed, the first hard disk. Access control in early AT-class machines was by a physical keylock switch (which was not hard to defeat if the computer case could be opened). Anyone who could switch on the computer could boot it.

Later, 386-class computers started integrating the BIOS setup utility in the ROM itself, alongside the BIOS code; these computers usually boot into the BIOS setup utility if a certain key or key combination is pressed, otherwise the BIOS POST and boot process are executed.



```
          CMOS Setup Utility - Copyright (C) 1984-1999 Award Software

  ► Standard CMOS Features              ► Frequency/Voltage Control
  ► Advanced BIOS Features                Load Fail-Safe Defaults
  ► Advanced Chipset Features             Load Optimized Defaults
  ► Integrated Peripherals                Set Supervisor Password
  ► Power Management Setup                Set User Password
  ► PnP/PCI Configurations                Save & Exit Setup
  ► PC Health Status                      Exit Without Saving

 Esc : Quit                          ↑ ↓ → ←   : Select Item
 F10 : Save & Exit Setup

                         Time, Date, Hard Disk Type...
```

Award BIOS setup utility on a standard PC

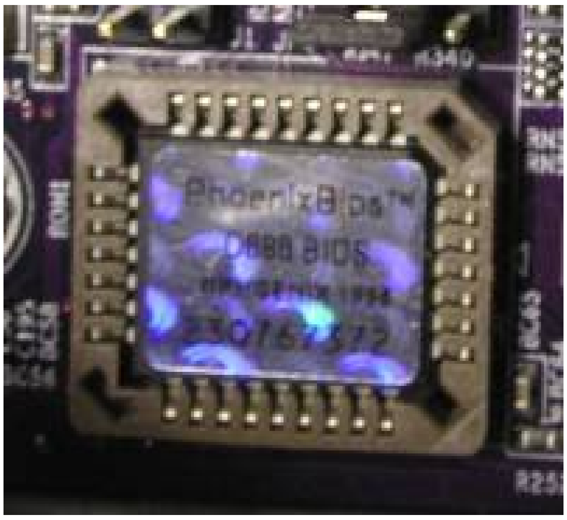
A modern BIOS setup utility has a menu-based user interface (UI) accessed by pressing a certain key on the keyboard when the PC starts. Usually the key is advertised for short time during the early startup, for example “Press F1 to enter CMOS setup”. The actual key depends on specific hardware. Features present in the BIOS setup utility typically include:

- Configuring the hardware components, including setting their various operating modes and frequencies (for example, selecting how the storage controllers are visible to the operating system, or overlocking the CPU)
- Setting the system clock
- Enabling or disabling system components
- Selecting which devices are potential boot devices, and in which order booting from them will be attempted
- Setting various passwords, such as a password for securing access to the BIOS user interface functions itself and preventing malicious users from booting the system from unauthorized portable storage devices, a password for booting the system, or a hard disk drive password that limits access to it and stays assigned even if the hard disk drive is moved to another computer.

7.6.2 Reprogramming

In modern PCs the BIOS is stored in rewritable memory, allowing the contents to be replaced or “rewritten”. This rewriting of the contents is sometimes termed *flashing*, based on the common use of a kind of EEPROM known technically as “flash EEPROM” and colloquially as “flash memory”. It can be done by a special program, usually provided by the system’s manufacturer, or at POST, with a BIOS image in a hard drive or USB flash drive. A file containing such contents is sometimes termed “a BIOS image”. A BIOS might be reflashed in order to upgrade to a newer version to fix bugs or provide improved performance or to support newer hardware, or a reflashing operation might be needed to fix a damaged BIOS.

7.7 Hardware



PhoenixBIOS D686. This BIOS chip is housed in a PLCC package in a socket.

The original IBM PC BIOS (and cassette BASIC) was stored on mask-programmed read-only memory (ROM) chips in sockets on the motherboard. ROMs could be replaced, but not altered, by users. To allow for updates, many compatible computers used re-programmable memory devices such as EPROM and later flash memory devices. According to Robert Braver, the president of the BIOS manufacturer Micro Firmware, **Flash BIOS** chips became common around 1995 because the electrically erasable PROM (EEPROM) chips are cheaper and easier to program than standard ultraviolet erasable PROM (EPROM) chips. Flash chips are programmed (and re-programmed) in-circuit, while EPROM chips need to be removed from the motherboard for re-programming.[22] BIOS versions are upgraded to take advantage of newer versions of hardware and to correct bugs in previous revisions of BIOSes.[23]

Beginning with the IBM AT, PCs supported a hardware clock settable through BIOS. It had a century bit which allowed for manually changing the century when the year 2000 happened. Most BIOS revisions created in 1995 and nearly all BIOS revisions in 1997 supported the year 2000 by setting the century bit automatically when the clock rolled past midnight, December 31, 1999.[24]

The first flash chips were attached to the ISA bus. Starting in 1997, the BIOS flash moved to the LPC bus, a functional replacement for ISA, following a new standard implementation known as "firmware hub" (FWH). In 2006, the first systems supporting a Serial Peripheral Interface (SPI) appeared, and the BIOS flash memory moved again.

The size of the BIOS, and the capacity of the ROM, EEPROM, or other media it may be stored on, has increased over time as new features have been added to the code; BIOS versions now exist with sizes up to 16 megabytes. For contrast, the original IBM PC BIOS was contained in an 8 KiB mask ROM. Some modern motherboards are including even bigger NAND flash memory ICs on board which are capable of storing whole compact operating systems, such as some Linux distributions. For example, some ASUS motherboards included SplashTop Linux embedded into their NAND flash memory ICs.[25] However, the idea of including an operating system along with BIOS in the ROM of a PC is not new; in the 1980s, Microsoft offered a ROM option for MS-DOS, and it was included in the ROMs of some PC clones such as the Tandy 1000 HX.

Another type of firmware chip was found on the IBM PC AT and early compatibles. In the AT, the keyboard interface was controlled by a microcontroller with its own programmable memory. On the IBM AT, that was a 40-pin socketed device, while some manufacturers used an EPROM version of this chip which resembled an EPROM. This controller was also assigned the A20 gate function to manage memory above the one-megabyte range; occasionally an upgrade of this "keyboard BIOS" was necessary to take advantage of software that could use upper memory.

The BIOS may contain components such as the Memory Reference Code (MRC), which is responsible for handling memory timings and related hardware settings.[26]:8[27]

7.8 Vendors and products

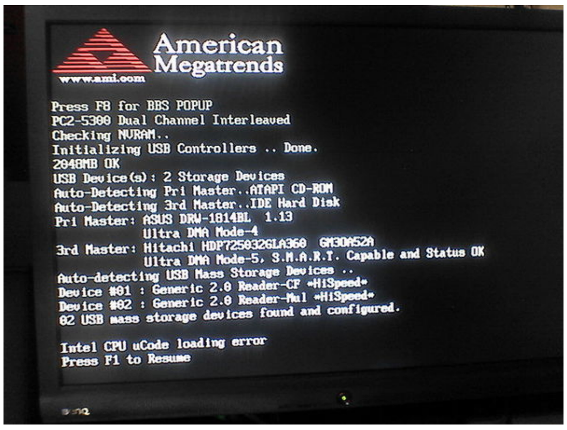
IBM published the entire listings of the BIOS for its original PC, PC XT, PC AT, and other contemporary PC models, in an appendix of the Technical Reference manual for each machine type. The effect of the publication of the BIOS listings is that anyone can see exactly what a definitive BIOS does and how it does it. Phoenix Technologies was the first company to write a fully compatible and completely legal BIOS through clean-room reverse engineering.

New standards grafted onto the BIOS are usually without complete public documentation or any BIOS listings. As a result, it is not as easy to learn the intimate details about the many non-IBM additions to BIOS as about the core BIOS services.

Most PC motherboard suppliers license a BIOS "core" and toolkit from a commercial third-party, known as an "independent BIOS vendor" or IBV. The motherboard manufacturer then customizes this BIOS to suit its own hardware. For this reason, updated BIOSes are normally obtained directly from the motherboard manufacturer. Major BIOS vendors include American Megatrends (AMI), Insyde Software, Phoenix Technologies and Byosoft. Former vendors include Award Software and Microid Research which were acquired by Phoenix Technologies in 1998; Phoenix later phased out the Award Brand name. General Software, which was also acquired by Phoenix in 2007, sold BIOS for Intel processor based embedded systems.

The open source community increased their effort to develop a replacement for proprietary BIOSes and their future incarnations with an open sourced counterpart through the coreboot and OpenBIOS/Open Firmware projects. AMD provided product specifications for some chipsets, and Google is sponsoring the project. Motherboard manufacturer Tyan offers coreboot next to the standard BIOS with their Opteron line of motherboards. MSI and Gigabyte Technology have followed suit with the MSI K9ND MS-9282 and MSI K9SD MS-9185 resp. the M57SLI-S4 models.

7.9 Security

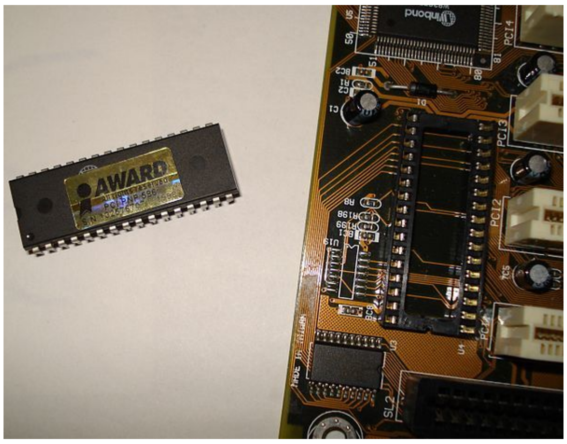


```
Press F8 for BBS POPUP
PC2-5300 Dual Channel Interleaved
Checking NVRAM..
Initializing USB Controllers .. Done.
2048MB OK
USB Device(s): 2 Storage Devices
Auto-Detecting Pri Master..ATAPI CD-ROM
Auto-Detecting 3rd Master..IDE Hard Disk
Pri Master: ASUS DRW-1814BL  1.13
            Ultra DMA Mode-4
3rd Master: Hitachi HDP725032GLA360  GM30A52A
            Ultra DMA Mode-5, S.M.A.R.T. Capable and Status OK
Auto-detecting USB Mass Storage Devices ..
Device #01 : Generic 2.0 Reader-CF *HiSpeed*
Device #02 : Generic 2.0 Reader-Mul *HiSpeed*
02 USB mass storage devices found and configured.

Intel CPU uCode loading error
Press F1 to Resume
```

An American Megatrends BIOS showing a "Intel CPU uCode Loading Error" after a failed attempt to upload microcode patches into the CPU.

EEPROM chips are advantageous because they can be easily updated by the user; hardware manufacturers frequently issue BIOS updates to upgrade their products, improve compatibility and remove bugs. However, this advantage had the risk that an improperly executed or



Detached BIOS Chip

aborted BIOS update could render the computer or device unusable. To avoid these situations, more recent BIOSes use a “boot block"; a portion of the BIOS which runs first and must be updated separately. This code verifies if the rest of the BIOS is intact (using hash checksums or other methods) before transferring control to it. If the boot block detects any corruption in the main BIOS, it will typically warn the user that a recovery process must be initiated by booting from removable media (floppy, CD or USB memory) so the user can try flashing the BIOS again. Some motherboards have a *backup* BIOS (sometimes referred to as DualBIOS boards) to recover from BIOS corruptions.

There are at least four known BIOS attack viruses, two of which were for demonstration purposes. The first one found in the wild was *Mebromi*, targeting Chinese users.

The first BIOS virus was CIH, whose name matches the initials of its creator, Chen Ing Hau. CIH was also called the “Chernobyl Virus”, because its payload date was 1999-04-26, the 13th anniversary of the Chernobyl accident. CIH appeared in mid-1998 and became active in April 1999. It was able to erase flash ROM BIOS content. Often, infected computers could no longer boot, and people had to remove the flash ROM IC from the motherboard and reprogram it. CIH targeted the then-widespread Intel i430TX motherboard chipset and took advantage of the fact that the Windows 9x operating systems, also widespread at the time, allowed direct hardware access to all programs.

Modern systems are not vulnerable to CIH because of a variety of chipsets being used which are incompatible with the Intel i430TX chipset, and also other flash ROM IC types. There is also extra protection from accidental BIOS rewrites in the form of boot blocks which are protected from accidental overwrite or dual and quad BIOS equipped systems which may, in the event of a crash, use a backup BIOS. Also, all modern operating systems such as FreeBSD, Linux, OS X, Windows NT-based Windows OS like Windows 2000, Windows XP and newer, do not allow user-mode programs to have direct hardware access.

As a result, as of 2008, CIH has become essentially harmless, at worst causing annoyance by infecting executable files and triggering antivirus software. Other BIOS viruses remain possible, however;[28] since most Windows home users without Windows Vista/7’s UAC run all applications with administrative privileges, a modern CIH-like virus could in principle still gain access to hardware without first using an exploit. The operating system OpenBSD prevents all users from having this access and the grsecurity patch for the linux kernel also prevents this direct hardware access by default, the difference being an attacker requiring a much more difficult kernel level exploit or reboot of the machine.

The second BIOS virus was a technique presented by John Heasman, principal security consultant for UK-based Next-Generation Security Software. In 2006, at the Black Hat Security Conference, he showed how to elevate privileges and read physical memory, using malicious procedures that replaced normal ACPI functions stored in flash memory.

The third BIOS virus was a technique called “Persistent BIOS infection.” It appeared in 2009 at the CanSecWest Security Conference in Vancouver, and at the SyScan Security Conference in Singapore. Researchers Anibal Sacco[29] and Alfredo Ortega, from Core Security Technologies, demonstrated how to insert malicious code into the decompression routines in the BIOS, allowing for nearly full control of the PC at start-up, even before the operating system is booted. The proof-of-concept does not exploit a flaw in the BIOS implementation, but only involves the normal BIOS flashing procedures. Thus, it requires physical access to the machine, or for the user to be root. Despite these requirements, Ortega underlined the profound implications of his and Sacco’s discovery: “We can patch a driver to drop a fully working rootkit. We even have a little code that can remove or disable antivirus.”[30]

Mebromi is a trojan which targets computers with AwardBIOS, Microsoft Windows, and antivirus software from two Chinese companies: Rising Antivirus and Jiangmin KV Antivirus.[31][32][33] Mebromi installs a rootkit which infects the master boot record.

In a December 2013 interview with CBS 60 Minutes, Deborah Plunkett, Information Assurance Director for the US National Security Agency claimed that NSA analysts had uncovered and thwarted a possible BIOS attack by a foreign nation state. The attack on the world’s computers could have allegedly “literally taken down the US economy.” The segment further cites anonymous cyber security experts briefed on the operation as alleging the plot was conceived in China.[34] A later article in The Guardian cast doubt on the likelihood of such a threat, quoting Berkeley computer-science researcher Nicholas Weaver, Matt Blaze, a computer and information sciences professor at the University of Pennsylvania, and cybersecurity expert Robert David Graham in an analysis of the NSA's claims.[35]

7.10 Alternatives and successors

For comparable software on other computer systems, see booting.

As of 2011, the BIOS is being replaced by the more complex Extensible Firmware Interface (EFI) in many new machines. EFI is a specification which replaces the runtime interface of the legacy BIOS. Initially written for the Itanium architecture, EFI is now available for x86 and x86-64 platforms; the specification development is driven by The Unified EFI Forum, an industry Special Interest Group. EFI booting has been supported in only Microsoft Windows versions supporting GPT,[36] the Linux kernel 2.6.1 and later, and Mac OS X on Intel-based Macs.[37]

Other alternatives to the functionality of the "Legacy BIOS" in the x86 world include coreboot.

A number of larger, more powerful servers and workstations use a platform-independent Open Firmware (IEEE-1275) based on the Forth programming language; it is included with Sun's SPARC computers, IBM's RS/6000 line, and other PowerPC systems such as the CHRP motherboards, along with the x86-based OLPC XO-1. Later x86-based personal computer operating systems, like Windows NT, use their own, native drivers; this makes it much easier to extend support to new hardware.

7.11 See also

- e820
- Extended System Configuration Data
- Double boot
- Plug and play
- Ralf Brown's Interrupt List
- System Management BIOS
- VESA BIOS Extensions
- XDK Debug BIOS

7.12 Notes

[1] The signature at offset +0x1FE in boot sectors is 0x55 0xAA, that is 0x55 at offset +0x1FE and 0xAA at offset +0x1FF. Since little-endian representation must be assumed in the context of IBM PC compatible machines, this can be written as 16-bit word 0xAA55 in programs for x86 processors (note the swapped order), whereas it would have to be written as 0x55AA in programs for other CPU architectures using a big-endian representation. Since this has been mixed up numerous times in books and even in original Microsoft reference documents, this article uses the offset-based byte-wise on-disk representation to avoid any possible misinterpretation.

7.13 References

[1] "Ref — System BIOS". *PCGuide*. Retrieved 6 December 2014.

[2] Kildall, Gary A. (June 1975), *CP/M 1.1 or 1.2 BIOS and BDOS for Lawrence Livermore Laboratories*

[3] Kildall, Gary A. (January 1980). "The History of CP/M, THE EVOLUTION OF AN INDUSTRY: ONE PERSON'S VIEWPOINT" (Vol. 5, No. 1, Number 41 ed.). Dr. Dobb's Journal of Computer Calisthenics & Orthodontia. pp. 6–7. Retrieved 2013-06-03.

[4] Bradley, Tony. "R.I.P. BIOS: A UEFI Primer". PC-World. Retrieved 2014-01-27.

[5] Swaine, Michael (1997-04-01). "Gary Kildall and Collegial Entrepreneurship". *Dr. Dobb's Journal*. Retrieved 2006-11-20.

[6] Killian, A. Joseph "Joe" (2001). "Gary Kildall's CP/M: Some early CP/M history - 1976-1977". Thomas "Todd" Fischer, IMSAI. Retrieved 2013-06-03.

[7] Fraley, Bob; Spicer, Dag (2007-01-26). "Oral History of Joseph Killian, Interviewed by: Bob Fraley, Edited by: Dag Spicer, Recorded: January 26, 2007, Mountain View, California, CHM Reference number: X3879.2007,". Computer History Museum. Retrieved 2013-06-03.

[8] "HP BIOS Configuration Utility". Hewlett-Packard. 2013. Retrieved 2015-01-12.

[9] See Intel® 64 and IA-32 Architectures Software Developer's Manual, volume 3, section 9.1.2

[10] page 5-27 *IBM Personal Computer Hardware Reference Library Technical Reference*, 1984, publication number 6361459

[11] "IBM 5162 PC XT286 TechRef 68X2537 Technical Reference manual" (PDF). August 1986. p. 35 (System BIOS A-5). Retrieved 2014-12-11.

[12] How StuffWorks: What BIOS Does.

[13] *BIOS Boot Specification (Version 1.01, 11 January 1996)*

[14] Mueller, Scott (2001-06-08). "Processor Update Feature | Microprocessor Types and Specifications". InformIT. Retrieved 2014-04-15.

[15] "Linux* Processor Microcode Data File". *Download Center*. Downloadcenter.intel.com. 2009-09-23. Retrieved 2014-04-15.

[16] Scott Mueller, *Upgrading and repairing PCs 15th edition*, Que Publishing, 2003 ISBN 0-7897-2974-1, pages 109-110

[17] "How SLP and SLIC Works". *guytechie.com*. 2010-02-25. Retrieved 2015-02-03.

[18] "Create and add an OEM ACPI SLIC table module to a congatec BIOS" (PDF). *congatec.com*. 2011-06-16. Retrieved 2015-02-03.

[19] Whitson Gordon. "A Beginner's Introduction to Overclocking Your Intel Processor". *Lifehacker*. Gawker Media. Retrieved 6 December 2014.

[20] Smart Computing Article - What Is The BIOS? - Computing Basics July 1994 • Vol.5 Issue 7

[21] Torres, Gabriel (24 November 2004). "Introduction and Lithium Battery". *Replacing the Motherboard Battery*. hardwaresecrets.com. Retrieved June 20, 2013.

[22] "Decoding RAM & ROM." *Smart Computing*. June 1997. Volume 8, Issue 6.

[23] "Upgrading Your Flash BIOS For Plug And Play." *Smart Computing*. March 1996. Volume 7, Issue 3.

[24] "Time To Check BIOS." *Smart Computing*. April 1999. Volume 7, Issue 4.

[25] SplashTop's Instant-On Linux Desktop | Geek.com

[26] Posted by Alex Watson, possibly repost from original content on custompc.com [unclear]. "The life and times of the modern motherboard". *2007-11-27*. Retrieved 2 February 2013.

[27] David Hilber, Jr. (August 2009). "Considerations for Designing an Embedded Intel Architecture System with System Memory Down ®". Intel. Retrieved 2 February 2013.

[28] New BIOS Virus Withstands HDD Wipes, March 27, 2009. Marcus Yam. Tom's Hardware US

[29] Sacco, Anibal; Alfredo Ortéga. "Persistent BIOS Infection". *Exploiting Stuff*. Retrieved 2010-02-06.

[30] Fisher, Dennis. "Researchers unveil persistent BIOS attack methods". *Threat Post*. Archived from the original on 30 January 2010. Retrieved 2010-02-06.

[31] Giuliani, Marco. "Mebromi: the first BIOS rootkit in the wild". *blog*. Retrieved 2011-09-19.

[32] "360□□"BMW□□"□□□□□□". *blog*. Retrieved 2011-09-19.

[33] Yuan, Liang. "Trojan.Mebromi". *Threat Response*. Retrieved 2011-09-19.

[34] "How did 60 Minutes get cameras into a spy agency?". CBS News. Retrieved 2014-04-15.

[35] Spencer Ackerman in Washington (2013-12-16). "NSA goes on 60 Minutes: the definitive facts behind CBS's flawed report | World news". theguardian.com. Retrieved 2014-01-27.

[36] "Windows and GPT FAQ". *microsoft.com*. Microsoft. Retrieved 6 December 2014.

[37] "Extensible Firmware Interface (EFI) and Unified EFI (UEFI)". *Intel*. Retrieved 6 December 2014.

7.14 Further reading

- *IBM Personal Computer Technical Reference* (Revised ed.). IBM Corporation. March 1983.
- *IBM Personal Computer AT Technical Reference*. IBM Personal Computer Hardware Reference Library. 0, 1, 2 (Revised ed.). IBM Corporation. March 1986 [1984-03]. 1502494, 6139362, 6183310, 6183312, 6183355, 6280070, 6280099.
- Phoenix Technologies, Ltd. (1989) [1987]. *System BIOS for IBM PC/XT/AT Computers and Compatibles — The Complete Guide to ROM-Based System Software*. Phoenix Technical Reference Series (1st ed.). Addison Wesley Publishing Company, Inc. ISBN 0-201-51806-6.
- Phoenix Technologies, Ltd. (1989) [1987]. *CBIOS for IBM PS/2 Computers and Compatibles — The Complete Guide to ROM-Based System Software for DOS*. Phoenix Technical Reference Series (1st ed.). Addison Wesley Publishing Company, Inc. ISBN 0-201-51804-X.
- Phoenix Technologies, Ltd. (1989) [1987]. *ABIOS for IBM PS/2 Computers and Compatibles — The Complete Guide to ROM-Based System Software for OS/2*. Phoenix Technical Reference Series (1st ed.). Addison Wesley Publishing Company, Inc. ISBN 0-201-51805-8.
- BIOS Disassembly Ninjutsu Uncovered, 1st edition (freely available book, PDF)

7.15 External links

- List of BIOS options
- How BIOS Works
- Persistent BIOS Infection - Phrack #66
- Preventing BIOS Failures Using Intel Boot Block Flash Memory (December 1998)
- BIOS Boot Specification 1.01 (January 1996)
- Implementing a Plug and Play BIOS Using Intel's Boot Block Flash Memory (February 1995)

Chapter 8

Bootstrapping

"Bootstrap" redirects here. For a UI web design tool called "Bootstrap", see Bootstrap (front-end framework). For other uses, see Bootstrapping (disambiguation).

In general parlance, **bootstrapping** usually refers to the starting of a self-sustaining process that is supposed to proceed without external input. In computer technology the term (usually shortened to **booting**) usually refers to the process of loading the basic software into the memory of a computer after power-on or general reset, especially the operating system which will then take care of loading other software as needed.

The term appears to have originated in the early 19th century United States (particularly in the phrase "pull oneself over a fence by one's bootstraps"), to mean an absurdly impossible action, an adynaton.[1][2][3]

8.1 Etymology

A pair of boots with one bootstrap visible

Tall boots may have a tab, loop or handle at the top known as a bootstrap, allowing one to use fingers or a boot hook tool to help pulling the boots on. The saying "to pull oneself up by one's bootstraps"[3] was already in use during the 19th century as an example of an impossible task. The idiom dates at least to 1834, when it appeared in the *Workingman's Advocate*: "It is conjectured that Mr. Murphee will now be enabled to hand himself over the Cumberland river or a barn yard fence by the straps of his boots."[4] In 1860 it appeared in a comment on metaphysical philosophy: "The attempt of the mind to analyze itself [is] an effort analogous to one who would lift himself by his own bootstraps."[5] Bootstrap as a metaphor, meaning to better oneself by one's own unaided efforts, was in use in 1922.[6] This metaphor spawned additional metaphors for a series of self-sustaining processes that proceed without external help.[7]

The term is sometimes attributed to a story in Rudolf Erich Raspe's *The Surprising Adventures of Baron Munchausen*, but in that story Baron Munchausen pulls himself (and his horse) out of a swamp by his hair (specifically, his pigtail), not by his bootstraps – and no explicit reference to bootstraps has been found elsewhere in the various versions of the Munchausen tales.[4]

8.2 Applications

8.2.1 Computing

Software loading and execution

Main article: Booting

Booting is the process of starting a computer, specifically in regards to starting its software. The process involves a chain of stages, in which at each stage a smaller simpler program loads and then executes the larger more complicated program of the next stage. It is in this sense that the computer "pulls itself up by its bootstraps", i.e. it improves itself by its own efforts. Booting is a chain of events that starts with execution of hardware-based procedures and may then hand-off to firmware and software which is loaded into main memory. Booting often involves processes such as performing self-tests, loading configuration settings, loading a BIOS, resident monitors, a hypervisor, an operating system, or utility software.

The computer term bootstrap began as a metaphor in the 1950s. In computers, pressing a bootstrap button caused a hardwired program to read a bootstrap program from an input unit. The computer would then execute the bootstrap program, which caused it to read more program instructions. It became a self-sustaining process that proceeded without external help from manually entered instructions. As a computing term, bootstrap has been used since at least 1953.[8]

Software development

Bootstrapping can also refer to the development of successively more complex, faster programming environments. The simplest environment will be, perhaps, a very basic text editor (e.g., ed) and an assembler program. Using these tools, one can write a more complex text editor, and a simple compiler for a higher-level language and so on, until one can have a graphical IDE and an extremely high-level programming language.

Historically, bootstrapping also refers to an early technique for computer program development on new hardware. The technique described in this paragraph has been replaced by the use of a cross compiler executed by a pre-existing computer. Bootstrapping in program development began during the 1950s when each program was constructed on paper in decimal code or in binary code, bit by bit (1s and 0s), because there was no high-level computer language, no compiler, no assembler, and no linker. A tiny assembler program was hand-coded for a new computer (for example the IBM 650) which converted a few instructions into binary or decimal code: A1. This simple assembler program was then rewritten in its just-defined assembly language but with extensions that would enable the use of some additional mnemonics for more complex operation codes. The enhanced assembler's source program was then assembled by its predecessor's executable (A1) into binary or decimal code to give A2, and the cycle repeated (now with those enhancements available), until the entire instruction set was coded, branch addresses were automatically calculated, and other conveniences (such as conditional assembly, macros, optimisations, etc.) established. This was how the early assembly program SOAP (Symbolic Optimal Assembly Program) was developed. Compilers, linkers, loaders, and utilities were then coded in assembly language, further continuing the bootstrapping process of developing complex software systems by using simpler software.

The term was also championed by Doug Engelbart to refer to his belief that organizations could better evolve by improving the process they use for improvement (thus obtaining a compounding effect over time). His SRI team that developed the NLS hypertext system applied this strategy by using the tool they had developed to improve the tool.

Compilers

Main article: Bootstrapping (compilers)

The development of compilers for new programming languages first developed in an existing language but then rewritten in the new language and compiled by itself, is another example of the bootstrapping notion. Using an existing language to bootstrap a new language is one way to solve the "chicken or the egg" causality dilemma.

Installers

Main article: Installation (computer programs)

During the installation of computer programs it is sometimes necessary to update the installer or package manager itself. The common pattern for this is to use a small executable bootstrapper file (e.g. setup.exe) which updates the installer and starts the real installation after the update. Sometimes the bootstrapper also installs other prerequisites for the software during the bootstrapping process.

Overlay networks

Main article: Bootstrapping node

A bootstrapping node, also known as a rendezvous host,[9] is a node in an overlay network that provides initial configuration information to newly joining nodes so that they may successfully join the overlay network.[10][11]

Discrete event simulation

Main article: Discrete event simulation

A type of computer simulation called discrete event simulation represents the operation of a system as a chronological sequence of events. A technique called *bootstrapping the simulation model* is used, which bootstraps initial data points using a pseudorandom number generator to schedule an initial set of pending events, which schedule additional events, and with time, the distribution of event times approaches its steady state—the bootstrapping behavior is overwhelmed by steady-state behavior.

Artificial intelligence and machine learning

Main articles: Bootstrap aggregating and Recursive self improvement

Bootstrapping is a technique used to iteratively improve a classifier's performance. Seed AI is a hypothesized type of artificial intelligence capable of recursive self-improvement. Having improved itself, it would become better at improving itself, potentially leading to an exponential increase in intelligence. No such AI is known to exist, but it remains an active field of research.

Seed AI is a significant part of some theories about the technological singularity: proponents believe that the development of seed AI will rapidly yield ever-smarter intelligence (via bootstrapping) and thus a new era.

8.2.2 Research

Main article: Information retrieval

Bootstrapping is a database searching technique. One may perform an inexact search (using keywords, for instance) and retrieve numerous "hits", some of which will be on-target. When the researcher looks at a relevant document that comes through in the mix, subject headings will be located within the document. The researcher can then execute a new search using authorized subject headings that will yield more focused, pinpointed results.

8.2.3 Statistics

Main articles: Bootstrapping (statistics) and Bootstrapping populations

Bootstrapping is a resampling technique used to obtain estimates of summary statistics.

8.2.4 Business

Bootstrapping in business means starting a business without external help or capital. Such startups fund the development of their company through internal cash flow and are cautious with their expenses.[12] Generally at the start of a venture, a small amount of money will be set aside for the bootstrap process.[13] Bootstrapping can also be a supplement for econometric models.[14] Bootstrapping was also expanded upon in the book *Bootstrap Business*, by Richard Christiansen.

- Startups can grow by reinvesting profits in its own growth if bootstrapping costs are low and return on investment is high. This financing approach allows owners to maintain control of their business, enables them to focus on customers instead of investors, and forces them to spend with discipline. [15]
- Leveraged buyouts, or highly leveraged or "bootstrap" transactions, occur when an investor acquires a controlling interest in a company's equity and where a significant percentage of the purchase price is financed through leverage, i.e., borrowing
- Bootstrapping in finance refers to the method to create the spot rate curve
- Operation Bootstrap (*Operación Manos a la Obra*) refers to the ambitious projects that industrialized Puerto Rico in the mid-20th century

8.2.5 Biology

Richard Dawkins in his book *River Out of Eden*[16] used the computer bootstrapping concept to explain how biological cells differentiate: "Different cells receive different combinations of chemicals, which switch on different combinations of genes, and some genes work to switch other genes on or off. And so the bootstrapping continues, until we have the full repertoire of different kinds of cells."

Phylogenetics

Bootstrapping analysis gives a way to judge the strength of support for clades on phylogenetic trees. A number is written by a node, which reflects the percentage of bootstrap trees which also resolve the clade at the endpoints of that branch.[17]

8.2.6 Law

Main article: Bootstrapping (law)

Bootstrapping is a rule preventing the admission of hearsay evidence in conspiracy cases.

8.2.7 Linguistics

Main article: Bootstrapping (linguistics)

Bootstrapping is a theory of language acquisition.

8.2.8 Physics

Main article: Bootstrap model

Bootstrapping is using very general consistency criteria to determine the form of a quantum theory from some assumptions on the spectrum of particles.

8.2.9 Electronics

Main article: Bootstrapping (electronics)

Bootstrapping is a form of positive feedback in analog circuit design.

8.2.10 Electric power grid

Main article: Black start

An electric power grid is almost never brought down intentionally. Generators and power stations are started and shut down as necessary. A typical power station requires power for start up prior to being able to generate power. This power is obtained from the grid, so if the entire grid is down these stations cannot be started.

Therefore to get a grid started, there must be at least a small number of power stations that can start entirely on their own. A black start is the process of restoring a power station to operation without relying on external power. In the absence of grid power, one or more black starts are used to bootstrap the grid.

8.2.11 Cellular networks

Main articles: Bootstrapping Server Function and Generic Bootstrapping Architecture

A Bootstrapping Server Function (BSF) is an intermediary element in cellular networks which provides application independent functions for mutual authentication of user equipment and servers unknown to each other and for 'bootstrapping' the exchange of secret session keys afterwards. The term 'bootstrapping' is related to building a security relation with a previously unknown device first and to allow installing security elements (keys) in the device and the BSF afterwards.

8.2.12 Media

A media bootstrap is the process whereby a story or meme is deliberately (but artificially) produced by self and peer-referential journalism, originally within a tight circle of media content originators, often commencing with stories written within the same media organization. This story is then expanded into a general media “accepted wisdom” with the aim of having it accepted as self-evident “common knowledge” by the reading, listening and viewing publics. The key feature of a media bootstrap is that as little hard, verifiable, external evidence as possible is used to support the story, preference being given to the citation (often unattributed) of other media stories, i.e. “journalists interviewing journalists”.

Because the campaign is usually originated and at least initially concocted internally by a media organization with a particular agenda in mind, within a closed loop of reportage and opinionation, the campaign is said to have “pulled itself up by its own bootstraps”.

A bootstrap campaign should be distinguished from a genuine news story of genuine interest, such as a natural disaster that kills thousands, or the death of a respected public figure. It is legitimate for these stories to be given coverage across all media platforms. What distinguishes a bootstrap from a real story is the contrived and organized manner in which the bootstrap appears to come out of nowhere. A bootstrap commonly claims to be tapping a hitherto unrecognized phenomenon within society.

As self-levitating by pulling on one’s bootstraps is physically impossible, this is often used by the bootstrappers themselves to deny the possibility that the bootstrap campaign is indeed concocted and artificial. They assert that it has arisen via a groundswell of public opinion. Media campaigns that are openly admitted as concocted (e.g. a public service campaign titled “Let’s Clean Up Our City”) are usually ignored by other media organizations for reasons related to competition. On the other hand the true bootstrap welcomes the participation of other media organizations, indeed encourages it, as this participation gains the bootstrap notoriety and, most importantly, legitimacy.

8.3 See also

- Bootstrap paradox
- Conceptual metaphor
- Horatio Alger myth
- Münchhausen trilemma
- Positive feedback
- Robert A. Heinlein's short story "By His Bootstraps"
- The Bootstrap Alliance, an institute founded by Douglas Engelbart

8.4 References

[1] World Wide Words: Boot, Michael Quinion

[2] “bootstraps--speculation/questions” (Mailing list). 2005-08-28.

[3] “figurative 'bootstraps'" (Mailing list). 2005-08-11.

[4] Jan Freeman, Bootstraps and Baron Munchausen, *Boston.com*, January 27, 2009

[5] Jan Freeman, The unkindliest cut, *Boston.com*, January 25, 2009

[6] *Ulysses* cited in the Oxford English Dictionary

[7] Phrase Finder

[8] Buchholz, Werner (1953). “The System Design of the IBM Type 701 Computer”. *Proceedings of the I.R.E.* **41** (10): 1273.

[9] Francis, Paul (2000-04-02). "Yoid: Extending the Internet Multicast Architecture". www.aciri.org. Retrieved 2008-12-24.

[10] Traversat et al. (2006-06-20). "US Patent 7,065,579". Retrieved 2008-12-23.

[11] Saxena et al. (2003). "Admission Control in Peer-to-Peer: Design and Performance Evaluation". In ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN) 2003. Retrieved 2008-12-24.

[12] The Art of the Bootstrap, Venture Beat

[13] Bootstrap

[14] J. Scott Armstrong (2001). "Judgmental Bootstrapping: Inferring Experts= Rules for Forecasting". *Principles of Forecasting: A Handbook for Researchers and Practitioners* (Kluwer Academic Publishers).

[15] Bootstrapping in Entrepreneurship

[16] Richard Dawkins, *River Out of Eden*, pages 23-25, 1995 (paper) ISBN 0-465-06990-8

[17] Bradley Efron, Elizabeth Halloran, and Susan Holmes (1996). "Bootstrap confidence levels for phylogenetic trees". *PNAS* **93** (23). Retrieved 11 June 2013.

8.5 External links

- Pull straps for boots
- Dictionary.com entries for Bootstrap
- Freedictionary.com entries for Bootstrap
- A video talk by Douglas Engelbart on Bootstrapping on YouTube
- Engelbart Institute on Bootstrapping Strategies
- Bootstrap sampling distributions for radiocarbon dating

Chapter 9

Firmware

A typical firmware-controlled device: a television remote control. Consumer products like this have been using firmware since the 1970s.

In electronic systems and computing, **firmware** is “the combination of a hardware device, e.g. an integrated circuit, and computer instructions and data that reside as read only software on that device”. As a result, firmware usually cannot be modified during normal operation of the device.[1] Typical examples of devices containing firmware are embedded systems (such as traffic lights, consumer appliances, and digital watches), computers, computer peripherals, mobile phones, and digital cameras. The firmware contained in these devices provides the control program for the device.

Firmware is held in non-volatile memory devices such as ROM, EPROM, or flash memory. Changing the firmware of a device may rarely or never be done during its economic lifetime; some firmware memory devices are permanently installed and cannot be changed after manufacture. Common reasons for updating firmware include fixing bugs or adding features to the device. This may require ROM integrated circuits to be physically replaced, or flash memory to be reprogrammed through a special procedure.[2] Firmware such as the ROM BIOS of a personal computer may contain only elementary basic functions of a device and may only provide services to higher-level software. Firmware such as the program of an embedded system may be the only program that will run on the system and provide all of its functions.

Before integrated circuits, other firmware devices included a discrete semiconductor diode matrix. The Apollo guidance computer had firmware consisting of a specially manufactured core memory plane, called "core rope memory", where data were stored by physically threading wires through (1) or around (0) the core storing each data bit.[3]

9.1 Origin of the term

Ascher Opler coined the term “firmware” in a 1967 *Datamation* article.[4] Originally, it meant the contents of a writable control store (a small specialized high speed memory), containing microcode that defined and implemented the computer’s instruction set, and that could be reloaded to specialize or modify the instructions that the central processing unit (CPU) could execute. As originally used, firmware contrasted with hardware (the CPU itself) and software (normal instructions executing on a CPU). It was not composed of CPU machine instructions, but of lower-level microcode involved in the implementation of machine instructions. It existed on the boundary between hardware and software; thus the name “firmware”.

Still later, popular usage extended the word “firmware” to denote anything ROM-resident, including processor machine-instructions for BIOS, bootstrap loaders, or specialized applications.

Until the mid-1990s, updating firmware typically involved replacing a storage medium containing firmware, usually a socketed ROM integrated circuit. Flash memory allows firmware to be updated without physically removing an integrated circuit from the system. An error during the update process may make the device non-functional, or “bricked”.

9.2 Personal computers

In some respects, the various firmware components are as important as the operating system in a working computer. However, unlike most modern operating systems,

ROM BIOS firmware on a Baby AT motherboard

firmware rarely has a well-evolved automatic mechanism of updating itself to fix any functionality issues detected after shipping the unit.

The BIOS may be "manually" updated by a user, using a small utility program. In contrast, firmware in storage devices (harddisks, DVD drives, flash storage) rarely gets updated, even when flash (rather than ROM) storage is used for the firmware; there are no standardized mechanisms for detecting or updating firmware versions.

Most computer peripherals are themselves special-purpose computers. Devices such as printers, scanners, cameras, USB drives, have firmware stored internally. Some devices may permit field replacement of firmware.

Some low-cost peripherals no longer contain non-volatile memory for firmware, and instead rely on the host system to transfer the device control program from a disk file or CD.[5]

9.3 Consumer products

As of 2010 most portable music players support firmware upgrades. Some companies use firmware updates to add new playable file formats (codecs); iriver added Vorbis playback support this way, for instance. Other features that may change with firmware updates include the GUI or even the battery life. Most mobile phones have a Firmware Over The Air firmware upgrade capability for much the same reasons; some may even be upgraded to enhance reception or sound quality, illustrating the fact that firmware is used at more than one level in complex products (in a CPU-like microcontroller versus in a digital signal processor, in this particular case).

9.4 Automobiles

Since 1996 most automobiles have employed an on-board computer and various sensors to detect mechanical problems. As of 2010 modern vehicles also employ computer-controlled ABS systems and computer-operated Transmission Control Units (TCU). The driver can also get in-dash information while driving in this manner, such as real-time fuel-economy and tire-pressure readings. Local dealers can update most vehicle firmware.

9.5 Examples

Examples of firmware include:

- In consumer products:
  - Timing and control systems for washing machines
  - Controlling sound and video attributes, as well as the channel list, in modern TVs
  - EPROM chips used in the Eventide H-3000 series of digital music processors
- In computers:
  - The BIOS found in IBM-compatible personal computers
  - The (U)EFI-compliant firmware used on Itanium systems, Intel-based computers from Apple, and many Intel desktop computer motherboards
  - Open Firmware, used in SPARC-based computers from Sun Microsystems and Oracle Corporation, PowerPC-based computers from Apple, and computers from Genesi
  - ARCS, used in computers from Silicon Graphics
  - Kickstart, used in the Amiga line of computers (POST, hardware init + Plug and Play auto-configuration of peripherals, kernel, etc.)
  - RTAS (Run-Time Abstraction Services), used in computers from IBM
  - The Common Firmware Environment (CFE)
- In routers and firewalls:
  - LibreWRT – a 100% free software router distribution based on the Linux-libre kernel
  - IPFire – an open-source firewall/router distribution based on the Linux kernel
  - fli4l – an open-source firewall/router distribution based on the Linux kernel
  - OpenWrt – an open-source firewall/router distribution based on the Linux kernel
  - m0n0wall – an embedded firewall distribution of FreeBSD
- In NAS systems:
  - NAS4Free – an open-source NAS operating system based on FreeBSD 9.1
  - Openfiler – a open-source NAS operating system based on the Linux kernel

9.6 Flashing

Flashing[6] involves the overwriting of existing firmware or data on EEPROM modules present in an electronic device with new data.[6] This can be done to upgrade a device[7] or to change the provider of a service associated with the function of the device, such as changing from one mobile phone service provider to another or installing a new operating system. If firmware is upgradable, it is often done via a program from the provider, and will often allow the old firmware to be saved before upgrading so it can be reverted to if the process fails, or if the newer version performs worse.

9.7 Firmware hacking

Sometimes third parties create an unofficial new or modified ("aftermarket") version of firmware to provide new features or to unlock hidden functionality. Examples include:

- Rockbox for digital audio players.
- CHDK[8] and Magic Lantern[8] for Canon digital cameras.
- Nikon Hacker project for Nikon EXPEED DSLRs.
- LibreWRT project for Ben Nanonote, Buffalo WZR-HP-G300NH and other computers with minimal resources. [9]
- Many third-party firmware projects for wireless routers, including:
  - OpenWrt, and its derivatives such as DD-WRT.[8]
  - RouterTech – for ADSL modem/routers based on the Texas Instruments AR7 chipset (with the Pspboot or Adam2 bootloader).
- Firmware that allows DVD drives to be region-free.
- SamyGO, modified firmware for Samsung televisions.[10]
- Many homebrew projects for gaming consoles. These often unlock general-purpose computing functionality in previously limited devices (e.g., running Doom on iPods).

Most firmware hacks are free software.

These hacks usually take advantage of the firmware update facility on many devices to install or run themselves. Some, however, must resort to exploits in order to run, because the manufacturer has attempted to lock the hardware to stop it from running unlicensed code.

9.7.1 HDD firmware hacks

The Moscow-based Kaspersky Lab discovered that a group of developers it refers to as the "Equation Group" has developed hard disk drive firmware modifications for various drive models, containing a trojan horse that allows data to be stored on the drive in locations that will not be erased even if the drive is formatted or wiped.[11] Although the Kaspersky Lab report did not explicitly claim that this group is part of the United States National Security Agency (NSA), evidence obtained from the code of various Equation Group software suggests that they are part of the NSA.[12][13]

Researchers from the Kaspersky Lab categorized the undertakings by Equation Group as the most advanced hacking operation ever uncovered, also documenting around 500 infections caused by the Equation Group in at least 42 countries.

9.8 Security risks

Mark Shuttleworth, founder of the Ubuntu Linux distribution, has described proprietary firmware as a security risk,[14] saying that "firmware on your device is the NSA's best friend" and calling firmware "a trojan horse of monumental proportions". He has pointed out that low-quality, nonfree firmware is a major threat to system security:[14] "Your biggest mistake is to assume that the NSA is the only institution abusing this position of trust – in fact, it's reasonable to assume that all firmware is a cesspool of insecurity, courtesy of incompetence of the highest degree from manufacturers, and competence of the highest degree from a very wide range of such agencies".

As a solution to this problem, he has called for declarative firmware. Declarative firmware would describe "hardware linkage and dependencies" and "should not include executable code".[14]

Custom firmware hacks have also focused on injecting malware into devices such as smartphones or USB devices. One such smartphone injection was demonstrated on the Symbian OS at MalCon,[15][16] a hacker convention. A USB device firmware hack called *BadUSB* was presented at Black Hat USA 2014 conference,[17] demonstrating how a USB flash drive microcontroller can be reprogrammed to spoof various other device types in order to take control of a computer, exfiltrate data, or spy on the user.[18][19] Other security researchers have worked further on how to exploit the principles behind BadUSB,[20] releasing at the same time the source code of hacking tools that can be used to modify the behavior of USB flash drives.[21]

9.9 See also

- ROM image
- UEFI
- Coreboot
- Microcode
- Binary blob
- Bootloader
- Aftermarket firmware category

9.10 References

[1] "Glossary of Computer System Software Development Terminology (8/95)". *fda.gov*. FDA. November 25, 2014. Retrieved March 1, 2015.

[2] "What is firmware?". incepator.pinzaru.ro. Retrieved 2013-06-14.

[3] Dag Spicer (August 12, 2000). "One Giant Leap: The Apollo Guidance Computer". Dr. Dobbs. Retrieved August 24, 2012.

[4] Opler, Ascher (January 1967). "Fourth-Generation Software". *Datamation* **13** (1): 22–24.

[5] Corbet, Jonathan; Rubini, Alessandro; Kroah-Hartman, Greg (2005). *Linux Device Drivers*. O'Reilly Media. p. 405. ISBN 0596005903.

[6] "Flashing Firmware". Tech-Faq.com. Retrieved July 8, 2011.

[7] "HTC Developer Center". HTC. Archived from the original on April 26, 2011. Retrieved July 8, 2011.

[8] "Custom Firmware Rocks!". 2009-08-05. Retrieved 2009-08-13.

[9] http://librewrt.org/index.php?title=Hardware_Support. Missing or empty |title= (help)

[10] "SamyGO: replacing television firmware". LWN.net. 2009-11-14. Retrieved 2009-12-11.

[11] "Equation Group: The Crown Creator of Cyber-Espionage". Kaspersky Lab. February 16, 2015.

[12] Dan Goodin (February 2015). "How "omnipotent" hackers tied to NSA hid for 14 years—and were found at last". *Ars Technica*.

[13] "Breaking: Kaspersky Exposes NSA's Worldwide, Backdoor Hacking of Virtually All Hard-Drive Firmware". Daily Kos. February 17, 2015.

[14] Shuttleworth, Mark (March 17, 2014). "ACPI, firmware and your security".

[15] "We will be back soon!". Malcon.org. Retrieved 2013-06-14.

[16] "Hacker plants back door in Symbian firmware". H-online.com. 2010-12-08. Archived from the original on 21 May 2013. Retrieved 2013-06-14.

[17] "Why the Security of USB Is Fundamentally Broken". Wired.com. 2014-07-31. Retrieved 2014-08-04.

[18] "BadUSB - On Accessories that Turn Evil". Black-Hat.com. Retrieved 2014-08-06.

[19] Karsten Nohl; Sascha Krißler; Jakob Lell (2014-08-07). "BadUSB – On accessories that turn evil" (PDF). *sr-labs.de*. Retrieved 2014-08-23.

[20] "BadUSB Malware Released - Infect millions of USB Drives". *The Hacking Post - Latest hacking News & Security Updates*. Retrieved 7 October 2014.

[21] "The Unpatchable Malware That Infects USBs Is Now on the Loose". *WIRED*. Retrieved 7 October 2014.

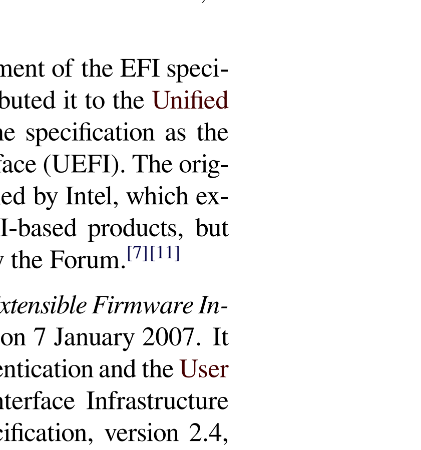
9.11 External links

- BadUSB - On Accessories that Turn Evil on YouTube, by Karsten Nohl and Jakob Lell
- Phison 2251-03 (2303) Custom Firmware & Existing Firmware Patches (BadUSB)
- Hard disk hacking (includes an analysis of feasible security exploits through firmware modifications, in eight parts)
- Snake on a keyboard (firmware modifications, in seven parts)

Chapter 10

Unified Extensible Firmware Interface

UEFI Logo



Extensible Firmware Interface's position in the software stack.

The **Unified Extensible Firmware Interface** (**UEFI**, pronounced as an initialism U-E-F-I or like "unify" without the *n*[lower-alpha 1]) is a specification that defines a software interface between an operating system and platform firmware. UEFI is meant to replace the Basic Input/Output System (BIOS) firmware interface, originally present in all IBM PC-compatible personal computers.[2][3] In practice, most UEFI firmware images provide legacy support for BIOS services. UEFI can support remote diagnostics and repair of computers, even without another operating system.[4]

Intel developed the original **EFI** (**Extensible Firmware Interface**) specification. Some of the EFI's practices and data formats mirror those from Microsoft Windows.[5][6] In 2005, UEFI deprecated EFI 1.10 (the final release of EFI). The Unified EFI Forum manages the UEFI specification.

10.1 History

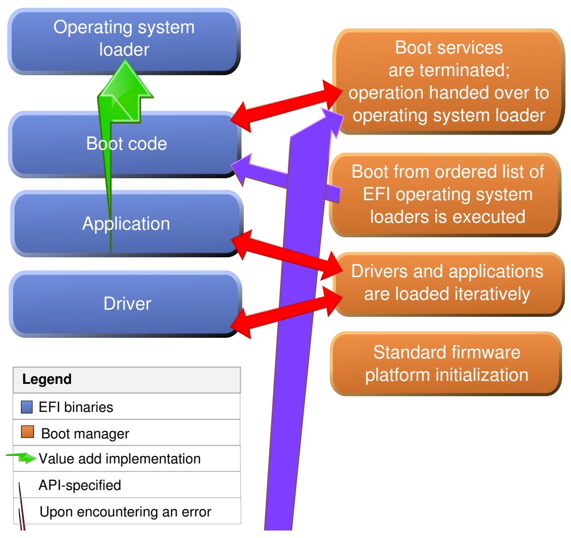
The original motivation for EFI came during early development of the first Intel–HP Itanium systems in the mid-1990s. BIOS limitations (such as 16-bit processor mode, 1 MB addressable space and PC AT hardware) were unacceptable for the larger server platforms Itanium was targeting.[7] The effort to address these concerns began in 1998 and was initially called *Intel Boot Initiative*;[8] it was later renamed to EFI.[9][10]

In July 2005, Intel ceased development of the EFI specification at version 1.10, and contributed it to the Unified EFI Forum, which has evolved the specification as the Unified Extensible Firmware Interface (UEFI). The original EFI specification remains owned by Intel, which exclusively provides licenses for EFI-based products, but the UEFI specification is owned by the Forum.[7][11]

Version 2.1 of the UEFI (*Unified Extensible Firmware Interface*) specification was released on 7 January 2007. It added cryptography, network authentication and the User Interface Architecture (Human Interface Infrastructure in UEFI). The current UEFI specification, version 2.4, was approved in July 2013.

10.2 Advantages

The interface defined by the EFI specification includes data tables that contain platform information, and boot


Operating system loader
Boot code
Application
Driver
Boot services are terminated; operation handed over to operating system loader
Boot from ordered list of EFI operating system loaders is executed
Drivers and applications are loaded iteratively
Standard firmware platform initialization
Legend
EFI binaries
Boot manager
Value add implementation
API-specified
Upon encountering an error

Interaction between the EFI boot manager and EFI drivers

and runtime services that are available to the OS loader and OS. UEFI firmware provides several technical advantages over a traditional BIOS system:[12]

- Ability to boot from large disks (over 2 TB) with a GUID Partition Table (GPT)[13][lower-alpha 2]
- CPU-independent architecture[lower-alpha 2]
- CPU-independent drivers[lower-alpha 2]
- Flexible pre-OS environment, including network capability
- Modular design

10.3 Compatibility

10.3.1 Processor compatibility

As of version 2.4, processor bindings exist for Itanium, x86, x86-64, ARM (AArch32) and ARM64 (AArch64).[14] Only little-endian processors can be supported.[15]

A normal PC BIOS is limited to a 16-bit processor mode and 1 MB of addressable space due to the design being based on the IBM 5150, which used the 16-bit Intel 8088.[7][16] In comparison, the processor mode in a UEFI environment can be either 32-bit (x86-32, AArch32) or 64-bit (x86-64, Itanium, and AArch64).[7][17] 64-bit UEFI firmware implementations understand long mode, which allows applications in the pre-boot execution environment to have direct access to all of the memory using 64-bit addressing.[18]

UEFI requires the firmware and operating system loader (or kernel) to be size-matched; for example, a 64-bit UEFI firmware implementation can only load a 64-bit UEFI operating system boot loader or kernel. After the system transitions from "Boot Services" to "Runtime Services", the operating system kernel takes over. At this point, the kernel can change processor modes if it desires, but this bars usage of the runtime services (unless the kernel switches back again).[19]:sections 2.3.2 and 2.3.4 As of version 3.15, Linux kernel supports booting of 64-bit kernels on 32-bit UEFI firmware implementations running on x86-64 CPUs, with *UEFI handover* support from a UEFI boot loader as the requirement.[20] UEFI handover protocol deduplicates the UEFI initialization code between the kernel and UEFI boot loaders, leaving the initialization to be performed only by the Linux kernel's *UEFI boot stub*.[21][22]

10.3.2 Disk device compatibility

See also: GPT § Operating systems support and Protective MBR

In addition to the standard PC disk partition scheme, which uses a master boot record (MBR), UEFI works with a new partitioning scheme: GUID Partition Table (GPT). GPT is free from many of the limitations of MBR. In particular, the MBR limits on the number and size of disk partitions (up to 4 primary partitions per disk, up to 2 TB (2×2^{40} bytes) per disk) are relaxed.[23] GPT allows for a maximum disk and partition size of 8 ZB (8×2^{70} bytes).[23][24]

The UEFI specification explicitly requires support for FAT32 for EFI System partitions (ESPs), and FAT16 or FAT12 for removable media;[19]:section 12.3 specific implementations may support other file systems.

Linux

See also: EFI System partition and Linux

Support for GPT in Linux is enabled by turning on the option CONFIG_EFI_PARTITION (EFI GUID Partition Support) during kernel configuration.[25] This option allows Linux to recognize and use GPT disks *after* the system firmware passes control over the system to Linux.

For reverse compatibility, Linux can use GPT disks in BIOS-based systems for both data storage and booting, as both GRUB 2 and Linux are GPT-aware. Such a setup is usually referred to as *BIOS-GPT*.[26] As GPT incorporates the protective MBR, a BIOS-based computer can boot from a GPT disk using GPT-aware boot loader stored in the protective MBR's bootstrap code area.[24] In case of GRUB, such a configuration requires a BIOS Boot partition for GRUB to embed its second-stage code due to absence of the post-MBR gap in GPT partitioned disks (which is taken over by the GPT's *Primary Header* and *Primary Partition Table*). Commonly 1 MiB in size, this partition's Globally Unique Identifier in GPT scheme is 21686148-6449-6E6F-744E-656564454649 and it is used by GRUB only in BIOS-GPT setups. From the GRUB's perspective, no such partition type exists in case of MBR partitioning. This partition is not required if the system is UEFI based, as there is no such embedding of the second-stage code in that case.[13][24][26]

UEFI systems can access GPT disks and directly boot from them, simplifying things and allowing UEFI boot methods for Linux. Booting Linux from GPT disks on UEFI systems involves creation of an EFI System partition (ESP), which contains UEFI applications such as bootloaders, operating system kernels, and utility software.[27][28][29] Such a setup is usually referred to as *UEFI-GPT*, while ESP is recommended to be at least 512 MiB in size and formatted with a FAT32 filesystem for maximum compatibility.[24][26][30]

For backwards compatibility, most of the UEFI implementations also support booting from MBR-partitioned disks, through the Compatibility Support Module (CSM) which provides legacy BIOS compatibility.[31] In that case, booting Linux on UEFI systems is the same as on legacy BIOS-based systems.

Microsoft Windows

The 64-bit versions of Microsoft Windows Vista[32] and later, 32-bit versions of Windows 8, and the Itanium versions of Windows XP and Server 2003 can boot from disks with a partition size larger than 2 TB.

10.4 Features

10.4.1 Services

EFI defines two types of services: *boot services* and *runtime services*. Boot services are only available while the firmware owns the platform (before the ExitBootServices call). Boot services include text and graphical consoles on various devices, and bus, block and file services. Runtime services are still accessible while the operating system is running; they include services such as date, time and NVRAM access.

In addition, the *Graphics Output Protocol* (GOP) provides limited runtime services support; see also Graphics features section below. The operating system is permitted to directly write to the framebuffer provided by GOP during runtime mode. However, the ability to change video modes is lost after transitioning to runtime services mode until the OS graphics driver is loaded.

Variable services UEFI variables provide a way to store data, in particular non-volatile data, that is shared between platform firmware and operating systems or UEFI applications. Variable namespaces are identified by GUIDs, and variables are key/value pairs. For example, variables can be used to keep crash messages in NVRAM after a crash for the operating system to retrieve after a reboot.[33]

Time services UEFI provides device-independent time services. Time services include support for timezone and daylight saving fields, which allow the hardware real-time clock to be set to local time or UTC.[34] On machines using a PC-AT real-time clock, the clock still has to be set to local time for compatibility with BIOS-based Windows.[6]

10.4.2 Applications

Independently of loading an operating system, UEFI has the ability to run standalone *UEFI applications*, which can be developed and installed independently of the system manufacturer. UEFI applications reside as files on the ESP and can be started directly by the firmware's boot manager, or by other UEFI applications. One class of the UEFI applications are the operating system loaders, such as rEFInd, Gummiboot, and Windows Boot Manager; they start a specific operating system and optionally provide a user interface for the selection of another UEFI application to run. Utilities like the UEFI shell are also UEFI applications.

10.4.3 Protocols

EFI defines protocols as a set of software interfaces used for communication between two binary modules. All EFI drivers must provide services to others via protocols.

10.4.4 Device drivers

In addition to standard architecture-specific device drivers, the EFI specification provides for a processor-independent device driver environment, called *EFI byte code* or *EBC*. System firmware is required by the UEFI specification to carry an interpreter for any EBC images that reside in or are loaded into the environment. In that sense, EBC is similar to Open Firmware, the hardware-independent firmware used in PowerPC-based Apple Macintosh and Sun Microsystems SPARC computers, among others.

Some architecture-specific (non-EBC) EFI device driver types can have interfaces for use from the operating system. This allows the OS to rely on EFI for basic graphics and network functions until OS specific drivers are loaded.

10.4.5 Graphics features

The EFI specification defined a UGA (Universal Graphic Adapter) protocol as a way to support device-independent graphics. UEFI did not include UGA and replaced it with GOP (Graphics Output Protocol), with the explicit goal of removing VGA hardware dependencies. The two are similar.[35]

UEFI 2.1 defined a "Human Interface Infrastructure" (HII) to manage user input, localized strings, fonts, and forms (in the HTML sense). These enable original equipment manufacturers (OEMs) or independent BIOS vendors (IBVs) to design graphical interfaces for pre-boot configuration; UEFI itself does not define a user interface.

Most early UEFI firmware implementations were console-based, but as early as 2007 some implementations featured a graphical user interface.[36]

10.4.6 EFI System partition

Main article: EFI System partition

EFI System partition, often abbreviated to ESP, is a data storage device partition that is used in computers adhering to the UEFI specification. Accessed by the UEFI firmware when a computer is powered up, it stores UEFI applications and the files these applications need to run, including operating system kernels. Supported partition table schemes include MBR and GPT, as well as El Torito volumes on optical disks.[19]:section 2.6.2 For the use on ESPs, UEFI defines a specific version of the FAT file system, which encompasses FAT32 file systems on ESPs, and FAT16 and FAT12 on removable media.[19]:section 12.3 The ESP provides space for a boot sector as part of the BIOS backward compatibility.[31]

10.4.7 Booting

UEFI booting

Unlike BIOS, UEFI does not rely on a boot sector, defining instead a *boot manager* as part of the UEFI specification. When a computer is powered on, the boot manager checks the boot configuration and, based on its settings, loads and executes the specified operating system loader or operating system kernel. The boot configuration is a set of global-scope variables stored in NVRAM, including the boot variables that indicate the paths to operating system loaders or kernels, which as a component class of UEFI applications are stored as files on the firmware-accessible EFI System partition (ESP).

Operating system loaders can also be automatically detected by an UEFI implementation, what enables easy booting from removable devices such as USB flash drives. This automated detection relies on a standardized file path to the operating system loader, with the path depending on the computer architecture. Format of the file path is defined as <EFI_SYSTEM_PARTITION>/BOOT/BOOT<MACHINE_TYPE_SHO for example, on an x86-64 computer the path is /efi/BOOT/BOOTX64.EFI.[19]

Booting UEFI systems from GPT-partitioned disks is commonly called *UEFI-GPT booting*. Additionally, it is common for an UEFI implementation to include a user interface to the boot manager, allowing the user to manually select the desired operating system (or system utility) from the list of available boot options and load it.

CSM booting

For backwards compatibility, most of the UEFI firmware implementations on PC-class machines also support booting in legacy BIOS mode from MBR-partitioned disks, through the *Compatibility Support Module (CSM)* which provides legacy BIOS compatibility. In that scenario, booting is performed in the same way as on legacy BIOS-based systems, by ignoring the partition table and relying on the content of a boot sector.[31]

BIOS booting from MBR-partitioned disks is commonly called *BIOS-MBR*, regardless of it being performed on UEFI or legacy BIOS-based systems. As a side note, booting legacy BIOS-based systems from GPT disks is also possible, and it is commonly called *BIOS-GPT*.

Despite the fact MBR partition tables are required to be fully supported within the UEFI specification,[19] some UEFI firmwares immediately switch to the BIOS-based CSM booting depending on the type of boot disk's partition table, thus preventing UEFI booting to be performed from EFI System partitions on MBR-partitioned disks.[31] Such a scheme is commonly called *UEFI-MBR*.

Network booting

UEFI specification includes support for booting over network through the Preboot eXecution Environment (PXE). Underlying network protocols include Internet Protocol (IPv4 and IPv6), User Datagram Protocol (UDP), Dynamic Host Configuration Protocol (DHCP) and Trivial File Transfer Protocol (TFTP).[19][37]

Also included is support for boot images remotely stored on storage area networks (SANs), with Internet Small Computer System Interface (iSCSI) and Fibre Channel over Ethernet (FCoE) as supported protocols for accessing the SANs.[19][38][39]

Secure boot

See also: Secure boot criticism

The UEFI 2.2 specification adds a protocol known as *secure boot*, which can secure the boot process by preventing the loading of drivers or OS loaders that are not signed with an acceptable digital signature. When secure boot is enabled, it is initially placed in "setup" mode, which allows a public key known as the "Platform key" (PK) to be written to the firmware. Once the key is written, secure boot enters "User" mode, where only drivers and loaders signed with the platform key can be loaded by the firmware. Additional "Key Exchange Keys" (KEK) can be added to a database stored in memory to allow other certificates to be used, but they must still have a connection to the private portion of the Platform key.[40] Secure boot can also be placed in "Custom" mode, where additional public keys can be added to the system that do not match the private key.[41]

Secure boot is supported by Windows 8, Windows Server 2012, FreeBSD, and a number of Linux distributions including Fedora, OpenSuse, and Ubuntu.[42]

10.4.8 Compatibility Support Module

The *Compatibility Support Module* (CSM) is a component of the UEFI firmware that provides legacy BIOS compatibility by emulating a BIOS environment, allowing legacy operating systems and some option ROMs that do not support UEFI to still be used.[43]

CSM also provides required legacy System Management Mode (SMM) functionality, called *CompatibilitySmm*, as an addition to features provided by the UEFI SMM. This is optional, and highly chipset and platform specific. An example of such a legacy SMM functionality is providing USB legacy support for keyboard and mouse, by emulating their classic PS/2 counterparts.[43]

10.4.9 UEFI shell

UEFI provides a shell environment, which can be used to execute other UEFI applications, including UEFI boot loaders.[29] Apart from that, commands available in the UEFI shell can be used for obtaining various other information about the system or the firmware, including getting the memory map (memmap), modifying boot manager variables (bcfg), running partitioning programs (diskpart), loading UEFI drivers, and editing text files (edit).[44][45][46]

Source code for a UEFI shell can be downloaded from the Intel's TianoCore UDK2010 / EDK2 SourceForge project.[47] Shell v2 works best in UEFI 2.3+ systems and is recommended over the shell v1 in those systems. Shell v1 should work in all UEFI systems.[44][48][49]

Methods used for launching UEFI shell depend on the manufacturer and model of the system motherboard. Some of them already provide a direct option in firmware setup for launching, e.g. compiled x86-64 version of the shell needs to be made available as <EFI_SYSTEM_PARTITION>/SHELLX64.EFI.
Some other systems have an already embedded UEFI shell which can be launched by appropriate key press combinations.[50][51] For other systems, the solution is either creating an appropriate USB flash drive or adding manually (bcfg) a boot option associated with the compiled version of shell.[46][50][52][53]

10.4.10 Extensions

Extensions to EFI can be loaded from virtually any non-volatile storage device attached to the computer. For example, an original equipment manufacturer (OEM) can distribute systems with an EFI partition on the hard drive, which would add additional functions to the standard EFI firmware stored on the motherboard's ROM.

10.5 Implementation and adoption

10.5.1 Intel EFI

Intel's implementation of EFI is the *Intel Platform Innovation Framework*, codenamed "Tiano." Tiano runs on Intel's XScale, Itanium and IA-32 processors, and is proprietary software, although a portion of the code has been released under the BSD license or Eclipse Public License (EPL) as TianoCore. TianoCore can be used as a payload for coreboot.[54]

Phoenix Technologies' implementations of UEFI include its SecureCore and SecureCore Tiano products.[55] American Megatrends offers its own UEFI firmware implementation known as Aptio,[56] while Insyde Software offers InsydeH2O, its own implementation of Tiano.[57]

10.5.2 Platforms using EFI/UEFI

Intel's first Itanium workstations and servers, released in 2000, implemented EFI 1.02.

Hewlett-Packard's first Itanium 2 systems, released in 2002, implemented EFI 1.10; they were able to boot Windows, Linux, FreeBSD and HP-UX; OpenVMS added UEFI capability in June, 2003.

In January 2006, Apple Inc. shipped its first Intel-based Macintosh computers. These systems used EFI instead of Open Firmware, which had been used on its previous PowerPC-based systems.[58] On 5 April 2006, Apple first released Boot Camp, which produces a Windows drivers disk and a non-destructive partitioning tool to allow the installation of Windows XP or Vista without requiring a reinstallation of Mac OS X. A firmware update was also released that added BIOS compatibility to its EFI implementation. Subsequent Macintosh models shipped with the newer firmware.[59]

During 2005, more than one million Intel systems shipped with Intel's implementation of UEFI.[60] New mobile, desktop and server products, using Intel's implementation of UEFI, started shipping in 2006. For instance, boards that use the Intel 945 chipset series use Intel's UEFI firmware implementation.

Since 2005, EFI has also been implemented on non-PC architectures, such as embedded systems based on XScale cores.[60]

The EDK (EFI Developer Kit) includes an NT32 target, which allows EFI firmware and EFI applications to run within a Windows application. But no direct hardware access is allowed by EDK NT32. This means only a subset of EFI application and drivers can be executed at the EDK NT32 target.

In 2008, more x86-64 systems adopted UEFI. While many of these systems still allow booting only the BIOS-based OSes via the Compatibility Support Module (CSM) (thus not appearing to the user to be UEFI-based), other systems started to allow booting UEFI-based OSes. For example, IBM x3450 server, MSI motherboards with ClickBIOS, all HP EliteBook Notebook and Tablet PCs, newer HP Compaq Notebook PCs (e.g., 6730b, 6735b, etc.).

In 2009, IBM shipped System x machines (x3550 M2, x3650 M2, iDataPlex dx360 M2) and BladeCenter HS22 with UEFI capability. Dell shipped PowerEdge T610, R610, R710, M610 and M710 servers with UEFI capability. More commercially available systems are mentioned in a UEFI whitepaper.[61]

In 2011, major vendors (such as ASRock, Asus, Gigabyte, and MSI) launched several consumer-oriented motherboards using the Intel 6-series LGA 1155 chipset and AMD 9 Series AM3+ chipsets with UEFI.[62]

With the release of Windows 8 in October 2012, Microsoft's certification requirements now require that computers include firmware that implements the UEFI specification. Furthermore, if the computer supports the "Connected Standby" feature of Windows 8 (which allows devices to have power management comparable to smartphones, with an almost instantaneous return from standby mode), then the firmware is not permitted to contain a Compatibility Support Module (CSM). As such, systems that support Connected Standby are incapable of booting Legacy BIOS operating systems.[63][64]

10.5.3 Operating systems

An operating system that can be booted from a (U)EFI is called a (U)EFI-aware OS, defined by (U)EFI specification. Here the term *booted from a (U)EFI* means directly booting the system using a (U)EFI OS loader stored on any storage device. The default location for the operating system loader is <EFI_SYSTEM_PARTITION>/BOOT/BOOT<MACHINE_TYPE_SHORT_NAME>.EFI, where short name of the machine type can be IA32, X64, IA64, ARM or AA64.[19] Some operating systems vendors may have their own boot loaders. They may also change the default boot location.

- The Linux kernel has been able to use EFI at boot time since early 2000,[65] using the elilo EFI boot loader or, more recently, EFI versions of GRUB.[66] Grub+Linux also supports booting from a GUID partition table without UEFI.[13] The distribution Ubuntu added support for UEFI secure boot as of version 12.10.[67] Further, the Linux kernel can be compiled with the option to run as an EFI bootloader on its own through the EFI bootstub feature.
- HP-UX has used (U)EFI as its boot mechanism on IA-64 systems since 2002.
- HP OpenVMS has used (U)EFI on IA-64 since its initial evaluation release in December 2003, and for production releases since January 2005.[68]
- Apple uses EFI for its line of Intel-based Macs. Mac OS X v10.4 Tiger and Mac OS X v10.5 Leopard implement EFI v1.10 in 32-bit mode even on newer 64-bit CPUs, but full support arrived with Mac OS X v10.8 Mountain Lion.[69]
- The Itanium versions of Windows 2000 (Advanced Server Limited Edition and Datacenter Server Limited Edition) implemented EFI 1.10 in 2002. MS Windows Server 2003 for IA-64, MS Windows XP 64-bit Edition and Windows 2000 Advanced Server Limited Edition, all of which are for the Intel Itanium family of processors, implement EFI, a requirement of the platform through the DIG64 specification.[70]
- Microsoft introduced UEFI for x86-64 Windows operating systems with Windows Server 2008 and Windows Vista Service Pack 1 so the 64-bit versions of Windows 7 are compatible with EFI. 32-bit UEFI was originally not supported since vendors did not have any interest in producing native 32-bit UEFI firmware because of the mainstream status of 64-bit computing.[71] Windows 8 includes further optimizations for UEFI systems, including a faster startup, 32-bit support, and secure boot support.[72][73]
- On March 5, 2013, the FreeBSD Foundation awarded a grant to a developer seeking to add UEFI support to the FreeBSD kernel and bootloader.[74] The changes were initially stored in a discrete branch of the FreeBSD source code, but were merged into the mainline source on April 4, 2014 (revision 264095); the changes include support in the installer as well.[75]
- Oracle Solaris 11.1 and later support UEFI boot for x86 systems with UEFI firmware version 2.1 or later. GRUB 2 is used as the boot loader on x86.[76]

10.5.4 Use of UEFI with virtualization

- HP Integrity Virtual Machines provides UEFI boot on HP Integrity Servers. It also provides a virtualized UEFI environment for the guest UEFI-aware OSes.
- Intel hosts an Open Virtual Machine Firmware project on SourceForge.[77]
- VMware Fusion 3 software for Mac OS X can boot Mac OS X Server virtual machines using EFI. VMware Workstation *unofficially* supports EFI, but it needs to be manually enabled by editing the vmx file, and as of 2012 Secure Boot is not yet supported.[78] ESXi/vSphere 5.0 officially support UEFI.[79]
- VirtualBox has implemented UEFI since 3.1,[80] but limited to Unix/Linux operating systems (does not work with Windows Vista x64 and Windows 7 x64).[81][82]
- QEMU can be used with the Open Virtual Machine Firmware (OVMF) provided by TianoCore.[83]
- The VMware ESXi version 5 hypervisor, part of VMware vSphere, supports virtualized EFI as an alternative to BIOS inside a virtual machine.
- Second generation of the Microsoft Hyper-V virtual machine supports virtualized UEFI.[84]

10.6 Applications development

EDK2 Application Development Kit (EADK) makes it possible to use standard C library functions in UEFI applications. EADK can be freely downloaded from the Intel's TianoCore UDK2010 / EDK2 SourceForge project. As an example, a port of the Python interpreter is made available as an UEFI application by using the EADK.[85]

A minimalistic “Hello world” C program written using EADK looks similar to its usual C counterpart:

#include <Uefi.h> #include <Library/UefiLib.h> #include <Library/ShellCEntryLib.h> EFI_STATUS EFIAPI ShellAppMain(IN UINTN Argc, IN CHAR16 **Argv) { Print(L"hello, world\n”); return EFI_SUCCESS; }

10.7 Criticism

Numerous digital rights activists have protested against UEFI. Ronald G. Minnich, a co-author of coreboot, and Cory Doctorow, a digital rights activist, have criticized EFI as an attempt to remove the ability of the user to truly control the computer.[86][87] It does not solve any of the BIOS’s long-standing problems of requiring two different drivers—one for the firmware and one for the operating system—for most hardware.[88]

Open source project TianoCore also provides the UEFI interfaces.[89] TianoCore lacks the specialized drivers that initialize chipset functions, which are instead provided by Coreboot, of which TianoCore is one of many payload options. The development of Coreboot requires cooperation from chipset manufacturers to provide the specifications needed to develop initialization drivers.

10.7.1 Secure boot

See also: Windows 8 § Reception and Hardware restrictions § Secure boot

In 2011, Microsoft announced that computers certified to run its Windows 8 operating system had to ship with secure boot enabled using a Microsoft private key. Following the announcement, the company was accused by critics and free software/open source advocates (including the Free Software Foundation) of trying to use the secure boot functionality of UEFI to hinder or outright prevent the installation of alternative operating systems such as Linux. Microsoft denied that the secure boot requirement was intended to serve as a form of lock-in, and clarified its requirements by stating that Intel-based systems certified for Windows 8 must allow secure boot to enter custom mode or be disabled, but not on systems using the ARM architecture.[41][90]

Other developers raised concerns about the legal and practical issues of implementing support for secure boot on Linux systems in general. Former Red Hat developer Matthew Garrett noted that conditions in the GNU General Public License version 3 may prevent the use of the GRUB bootloader without a distribution’s developer disclosing the private key (however, the Free Software Foundation has since clarified its position, assuring that the responsibility to make keys available was held by the hardware manufacturer),[67] and that it would also be difficult for advanced users to build custom kernels that could function with secure boot enabled without self-signing them.[90] Other developers suggested that signed builds of Linux with another key could be provided, but noted that it would be difficult to persuade OEMs to ship their computers with the required key alongside the Microsoft key.[3]

Several major Linux distributions have developed different implementations for secure boot. Matthew Garrett himself developed a minimal bootloader known as shim; a pre-compiled, signed bootloader that allows the user to individually trust keys provided by distributors.[91] Ubuntu 12.10 uses an older version of shim pre-configured for use with Canonical's own key that verifies only the bootloader and allows unsigned kernels to be loaded; developers believed that the practice of signing only the bootloader is more feasible, since a trusted kernel is effective at securing only the user space, and not the pre-boot state for which secure boot is designed to add protection. That also allows users to build their own kernels and use custom kernel modules as well, without the need to reconfigure the system.[67][92][93] Canonical also maintains its own private key to sign installations of Ubuntu pre-loaded on certified OEM computers that run the operating system, and also plans to enforce a secure boot requirement as well—requiring both a Canonical key and a Microsoft key (for compatibility reasons) to be included in their firmware. Fedora also uses shim, but requires that both the kernel and its modules be signed as well.[92]

It has been disputed whether the kernel and its modules must be signed as well; while the UEFI specifications do not require it, Microsoft has asserted that their contractual requirements do, and that it reserves the right to revoke any certificates used to sign code that can be used to compromise the security of the system.[93] In February 2013, another Red Hat developer attempted to submit a patch to the Linux kernel that would allow it to parse Microsoft's authenticode signing using a master X.509 key embedded in PE files signed by Microsoft. However, the proposal was criticized by Linux creator Linus Torvalds, who attacked Red Hat for supporting Microsoft's control over the secure boot infrastructure.[94]

On March 26, 2013, the Spanish free software development group Hispalinux filed a formal complaint with the European Commission, contending that Microsoft's secure boot requirements on OEM systems were "obstructive" and anti-competitive.[95]

At the Black Hat conference in August 2013, a group of security researchers presented a series of exploits in specific vendor implementations of UEFI that could be used to exploit secure boot.[96]

Windows 10 will allow OEMs to not offer the ability to configure or disable secure boot on x86 systems.[97]

10.7.2 Firmware issues

The increased prominence of UEFI firmware in devices has also led to a number of technical issues blamed on their respective implementations.[98]

Following the release of Windows 8 in late 2012, it was discovered that certain Lenovo computer models with secure boot had firmware that was hardcoded to allow only executables named "Windows Boot Manager" or "Red Hat Enterprise Linux" to load, regardless of any other setting.[99] Other issues were encountered by several Toshiba laptop models with secure boot that were missing certain certificates required for its proper operation.[98]

In January 2013, a bug surrounding the UEFI implementation on some Samsung laptops was publicized, which caused them to be bricked after installing a Linux distribution in UEFI mode. While potential conflicts with a kernel module designed to access system features on Samsung laptops were initially blamed (also prompting kernel maintainers to disable the module on UEFI systems as a safety measure), Matthew Garrett uncovered that the bug was actually triggered by storing too many UEFI variables to memory, and that the bug could also be triggered under Windows as well under special conditions. In conclusion, he determined that the offending kernel module had caused kernel message dumps to be written to the firmware, thus triggering the bug.[33][100][101]

10.8 See also

- Booting
- GNU GRUB
- Master boot record (MBR)
- GUID Partition Table (GPT)
- EFI System partition (ESP)
- BIOS Boot partition
- Advanced Configuration and Power Interface (ACPI)
- coreboot
- Open Firmware
- OpenBIOS
- Platform Initialization Specification
- System Management BIOS (SMBIOS)
- System Management Mode (SMM)
- Trusted Platform Module (TPM)
- Unified EFI Forum

10.9 Notes

[1] Various pronunciations have existed for UEFI; according to the UK *PC Pro Magazine*, the following pronunciations are in use: "weffy" (PC Pro), "U-E-F-I" (Microsoft), "you-fee", and "you-ef-fee". The magazine also notes the lack of agreement on the pronunciation.[1]

[2] Large disk support and features such as Advanced Configuration and Power Interface (ACPI) and System Management BIOS (SMBIOS) were subsequently implemented in BIOS-based systems.

10.10 References

[1] "UEFI BIOS Explained". pcpro.co.uk. 2013-05-03. Retrieved 2014-07-05.

[2] Michael Kinney (1 September 2000). "Solving BIOS Boot Issues with EFI" (PDF). pp. 47–50. Retrieved 14 September 2010.

[3] "MS denies secure boot will exclude Linux". The Register. 23 September 2011. Retrieved 24 September 2011.

[4] "The 30-year-long Reign of BIOS is Over: Why UEFI W... - Input Output". H30565.www3.hp.com. Archived from the original on 2013-06-26. Retrieved 2012-03-06.

[5] IBM PC Real Time Clock should run in UT. Cl.cam.ac.uk. Retrieved on 2013-10-30.

[6] Matthew Garrett (Jan 19, 2012). "EFI and Linux: the future is here, and it's awful - Matthew Garrett". *linux.conf.au 2012*. Retrieved 2 April 2012.

[7] "Emulex UEFI Implementation Delivers Industry-leading Features for IBM Systems" (PDF). Emulex. Retrieved 14 September 2010.

[8] *Extensible Firmware Interface (EFI) and Unified EFI (UEFI)*, Intel, archived from the original on 2010-01-05

[9] Wei, Dong (2006), "foreword", *Beyond BIOS*, Intel Press, ISBN 978-0-9743649-0-2

[10] "1.10 Specification overview", *Extensible Firmware Interface*, Intel

[11] *About*, Unified EFI Forum, Q: What is the relationship between EFI and UEFI? A: The UEFI specification is based on the EFI 1.10 specification published by Intel with corrections and changes managed by the Unified EFI Forum. Intel still holds the copyright on the EFI 1.10 specification, but has contributed it to the Forum so that the Forum can evolve it. There will be no future versions of the EFI specification, but customers who license it can still use it under the terms of their license from Intel. The license to the Unified EFI Specification comes from the Forum, not from Intel

[12] "UEFI and Windows". Microsoft. 15 September 2009. Retrieved 14 September 2010.

[13] "Installation". *3.4 BIOS installation*. GNU GRUB. Retrieved 2013-09-25.

[14] UEFI Specification 2.4, section 2.3

[15] UEFI specification 2.3.1, section 1.8.1.

[16] Hardwidge, Ben (1 June 2010). "LBA explained — Solving the 3TB Problem?". bit-tech. Retrieved 18 June 2010.

[17] Brian Richardson (10 May 2010). "Ask a BIOS Guy: "Why UEFI"". Intel Architecture Blog. Retrieved 18 June 2010.

[18] Gary Simpson. "UEFI Momentum — The AMD perspective" (PPTX). AMD. Archived from the original on 2014-01-04. Retrieved 2014-09-20.

[19] "UEFI Specifications (version 2.4 and older)" (PDF). Unified EFI, Inc. June 2013. Retrieved 2013-09-25.

[20] "Linux kernel 3.15, Section 1.3. EFI 64-bit kernels can be booted from 32-bit firmware". *kernelnewbies.org*. 2014-06-08. Retrieved 2014-06-15.

[21] "x86, efi: Handover Protocol". LWN.net. 2012-07-19. Retrieved 2014-06-15.

[22] "Linux kernel documentation: Documentation/efi-stub.txt". kernel.org. 2014-02-01. Retrieved 2014-06-15.

[23] "FAQ: Drive Partition Limits" (PDF). UEFI Forum. Retrieved 9 June 2010.

[24] Roderick W. Smith (2012-07-03). "Make the most of large drives with GPT and Linux". IBM. Retrieved 2013-09-25.

[25] "block/partitions/Kconfig (3.11.1)". *CONFIG_EFI_PARTITION (line #247)*. kernel.org. Retrieved 2013-09-25.

[26] "GRUB". *BIOS systems*. Arch Linux. Retrieved 2013-09-25.

[27] "GRUB and the boot process on UEFI-based x86 systems". redhat.com. Retrieved 2013-11-14.

[28] "UEFI Booting 64-bit Redhat Enterprise Linux 6". fpmurphy.com. September 2010. Retrieved 2013-11-14.

[29] "UEFI Bootloaders". Arch Linux. Retrieved 2013-09-25.

[30] "Unified Extensible Firmware Interface". *EFI System Partition*. Arch Linux. Retrieved 2013-09-25.

[31] "UEFI system booting from MBR partition table and GRUB legacy". Arch Linux Forums. June 2012. Retrieved 2013-10-06.

[32] For Microsoft Windows Vista (x64), it is possible only if installed from an installation DVD of Microsoft Windows Vista (x64) with its service pack 1 or 2 integrated.

[33] "Samsung UEFI bug: Notebook bricked from Windows". The H. Retrieved 27 February 2013.

[34] UEFI specification, section 7.3

[35] "Intel Embedded Graphics Drivers FAQ: BIOS and firmware". Intel. Retrieved 2014-05-19.

[36] Intel shows PC booting Windows with UEFI firmware

[37] "Red Hat Enterprise Linux 6 Installation Guide". *30.2.2. Configuring PXE boot for EFI*. Red Hat. Retrieved 2013-10-09.

[38] "UEFI Summit" (PDF). *Advances in Pre-OS Networking in UEFI 2.4*. Hewlett-Packard. July 2013. Retrieved 2013-10-09.

[39] "Storage and Network Convergence Using FCoE and iSCSI" (PDF). IBM. July 2012. Retrieved 2013-10-09.

[40] Edge, Jake. "UEFI and "secure boot"". LWN.net. Retrieved 9 September 2012.

[41] "Windows 8 Secure Boot: The Controversy Continues". PC World. Retrieved 9 September 2012.

[42] Matthew Garrett (2012-12-27). "Secure Boot distribution support". Mjg59.dreamwidth.org. Retrieved 2014-03-20.

[43] "Intel® Platform Innovation Framework for EFI" (PDF). *Compatibility Support Module Specification (revision 0.97)*. Intel. 2007-09-04. Retrieved 2013-10-06.

[44] "Unified Extensible Firmware Interface". *UEFI Shell*. Arch Linux. Retrieved 2013-09-25.

[45] "EFI Shells and Scripting". Intel. Retrieved 2013-09-25.

[46] "UEFI Shell Specification Version 2.0, Errata A" (PDF). Unified EFI, Inc. May 2012. Retrieved 2013-09-25.

[47] "TianoCore on SourceForge". Intel. Retrieved 2013-09-25.

[48] "Email Archive: edk2-devel". *[edk2] Inclusion of UEFI shell in Linux distro iso*. SourceForge. 2012. Retrieved 2013-09-25.

[49] "TianoCore on SourceForge". *Shell FAQ*. Intel. Retrieved 2013-09-25.

[50] "Unified Extensible Firmware Interface". *Launching UEFI Shell*. Arch Linux. Retrieved 2013-09-25.

[51] "Basic Instructions for Using EFI for Server Configuration on Intel® Server Boards and Intel® Server Systems" (PDF). Intel. 2008. Retrieved 2013-09-25.

[52] "Unified Extensible Firmware Interface". *bcfg*. Arch Linux. Retrieved 2013-09-25.

[53] "GRUB EFI Examples". *Asus*. Arch Linux. Retrieved 2013-09-25.

[54] "TianoCore - coreboot". Retrieved 25 May 2012.

[55] "SecureCore Tiano™". Phoenix Technologies. Retrieved 14 September 2010.

[56] "Aptio®: The Complete UEFI Product Solution" (PDF). American Megatrends, Inc. Retrieved 8 January 2011.

[57] "InsydeH2O UEFI Framework". Insyde Software Corp. Retrieved 8 January 2011.

[58] Apple Computer. "Universal Binary Programming Guidelines, Second Edition: Extensible Firmware Interface (EFI)"

[59] Apple's Transition from Open Firmware to Extensible Firmware Interface, mactech, 2007.

[60] "Intel® Platform Innovation Framework for UEFI Overview". Intel. Retrieved 14 September 2010.

[61] *Evaluating UEFI using Commercially Available Platforms and Solutions* (PDF), UEFI, 2011-5 Check date values in: |date= (help)

[62] Asus P67 Motherboard Preview.

[63] *Windows Hardware Certification Requirements for Client and Server Systems*, Microsoft, 2013-1, System.Fundamentals.Firmware.CS.UEFISecureBoot.ConnectedStandby ... Platforms shall be UEFI Class Three (see UEFI Industry Group, Evaluating UEFI using Commercially Available Platforms and Solutions, version 0.3, for a definition) with no Compatibility Support Module installed or installable. BIOS emulation and legacy PC/AT boot must be disabled. Check date values in: |date= (help)

[64] "Microsoft: All You Need to Know About Windows 8 on ARM". *PC Magazine*. Retrieved 30 September 2013.

[65] Announcement of release 3.5pre1 by maintainer Brett Johnson made on 2004-02-27.

[66] *EFI version of Grub*, Debian GNU/Linux, retrieved 1 May 2008

[67] "Ubuntu will use GRUB 2 for its Secure Boot implementation". The H Online. Retrieved 28 October 2012.

[68] *OpenVMS Release History*, HP, retrieved 16 September 2008

[69] *rEFIt — Windows Vista and EFI*, SourceForge

[70] "Extensible Firmware Interface", *Windows Server TechCenter*, Microsoft

[71] "Unified Extended Firmware Interface support in Windows Vista". Microsoft. 26 October 2006. Retrieved 12 June 2010. Microsoft determined that vendors would not have any interest in producing native UEFI 32-bit firmware because of the current status of mainstream 64-bit computing and platform costs. Therefore, Microsoft originally did not to ship support for 32-bit UEFI implementations.

[72] "Microsoft Touts Incredible Windows 8 Boot Times". Retrieved September 9, 2011.

[73] Jon Brodkin (21 September 2011). "Windows 8 secure boot could complicate Linux installs". Ars Technica. Retrieved 23 September 2011.

[74] "FreeBSD to get UEFI support". The H. Retrieved 7 March 2013.

[75] "UEFI - FreeBSD Wiki". FreeBSD.org. Retrieved 19 June 2014.

[76] "Oracle Solaris 11.1 — What's New" (PDF). oracle.com. Retrieved 2013-11-04.

[77] *Open Virtual Machine Firmware*, SourceForge

[78] "VMWare Workstation EFI firmware | VMware Communities". Communities.vmware.com. Retrieved 2014-02-28.

[79] "What's New in vSphere 5.0". Vmware.com. Retrieved 2014-02-28.

[80] *3.1 Changelog*, VirtualBox

[81] *Ticket 7702*, VirtualBox

[82] "Statement by sr. software engineer at Oracle", *Forum*, VirtualBox

[83] "Testing secureboot with KVM". FedoraProject. Retrieved 2014-02-28.

[84] "What's New in Hyper-V for Windows Server 2012 R2". MicrosoftTechNet. Retrieved 2013-06-24.

[85] "TianoCore on SourceForge". *EDK2 Application Development Kit (EADK)*. Intel. Retrieved 2013-09-25.

[86] "Interview: Ronald G Minnich". Fosdem. 6 February 2007. Retrieved 14 September 2010.

[87] Doctorow, Cory (2011-12-27), *The Coming War on General Purpose Computation*, retrieved 2013-09-25

[88] "coreboot (aka LinuxBIOS): The Free/Open-Source x86 Firmware". YouTube. 31 October 2008. Retrieved 14 September 2010.

[89] "Welcome", *TianoCore*, SourceForge

[90] "Is Microsoft Blocking Linux Booting on ARM Hardware?". Compurer World UK. Retrieved 2012-03-06.

[91] "Shimming your way to Linux on Windows 8 PCs". ZDNet. Retrieved 26 February 2013.

[92] "Ubuntu details its UEFI secure boot plans". Linux Weekly News. Retrieved 11 September 2012.

[93] "No Microsoft certificate support in Linux kernel says Torvalds". The H. Retrieved 26 February 2013.

[94] "Linus Torvalds: I will not change Linux to "deep-throat Microsoft"". Ars Technica. Retrieved 26 February 2013.

[95] "Exclusive: Open software group files complaint against Microsoft to EU". Reuters. 26 March 2013. Retrieved 26 March 2013.

[96] "Researchers demo exploits that bypass Windows 8 Secure Boot". *IT World*. Retrieved 5 August 2013.

[97] "Windows 10 to make the Secure Boot alt-OS lock out a reality". *Ars Technica*. Retrieved 21 March 2015.

[98] "Linux on Windows 8 PCs: Some progress, but still a nuisance". ZDNet. Retrieved 26 February 2013.

[99] "Lenovo UEFI Only Wants To Boot Windows, RHEL". Phoronix. Retrieved 26 February 2013.

[100] "Linux acquitted in Samsung laptop UEFI deaths". Bittech. Retrieved 26 February 2013.

[101] "Booting Linux using UEFI can brick Samsung laptops". The H. Retrieved 26 February 2013.

10.11 Further reading

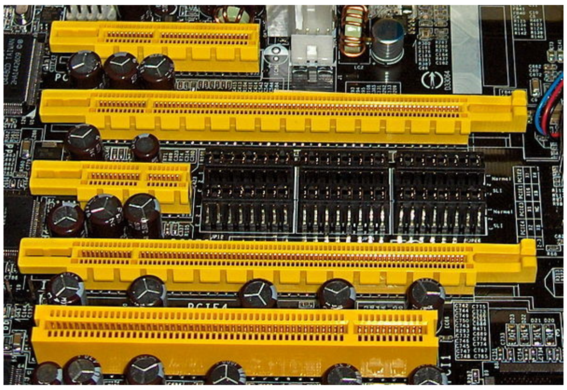
- Zimmer, Vincent; Rothman, Michael; Hale, Robert (10 May 2007). "EFI Architecture". *Dr. Dobb's Journal*. UBM. Retrieved 12 October 2012.
- De Boyne Pollard, Jonathan (11 July 2011). "The EFI boot process". *Frequently Given Answers*. Retrieved 12 October 2012.
- De Boyne Pollard, Jonathan (8 December 2011). "The Windows NT 6 boot process". *Frequently Given Answers*. Retrieved 12 October 2012.
- Smith, Roderick W. (2011). "A BIOS to UEFI Transformation". *Roderick W. Smith's Web Page*. Retrieved 12 October 2012.
- Kothari, Rajiv (21 September 2011). "UEFI – Just How Important It Really Is". *Hardware Secrets*. Retrieved 12 October 2012.
- Fisher, Doug (2011). "UEFI Today: Bootstrapping the Continuum". *Intel Technology Journal* (Intel) **15** (01). ISBN 9781934053430. Retrieved 2013-09-24.

10.12 External links

- Official website
- Intel-sponsored open-source EFI Framework initiative. SourceForge.
- Intel EFI/UEFI portal
- UEFI documents
- Microsoft UEFI Support and Requirements for Windows Operating Systems
- How Windows 8 Hybrid Shutdown / Fast Boot feature works
- Securing the Windows 8 Boot Process

Chapter 11

Bus (computing)



4 PCI Express bus card slots (from top to bottom: x4, x16, x1 and x16), compared to a 32-bit conventional PCI bus card slot (very bottom)

In computer architecture, a **bus** (related to the Latin *omnibus*, meaning “for all”) is a communication system that transfers data between components inside a computer, or between computers. This expression covers all related hardware components (wire, optical fiber, etc.) and software, including communication protocols.[1]

Early computer buses were parallel electrical wires with multiple connections, but the term is now used for any physical arrangement that provides the same logical functionality as a parallel electrical bus. Modern computer buses can use both parallel and bit serial connections, and can be wired in either a multidrop (electrical parallel) or daisy chain topology, or connected by switched hubs, as in the case of USB.

11.1 Background and nomenclature

Computer systems generally consist of three main parts, the central processing unit (CPU) to process data, main memory to hold the data to be processed, and a variety of peripherals to communicate that data with the outside world. An early computer might use a hand-wired CPU of vacuum tubes, a magnetic drum for main memory, and a punch tape and printer for reading and writing data. In a modern system we might find a multi-core CPU, DDR3 SDRAM for memory, a hard drive for secondary storage, a graphics card and LCD display as a display system, a mouse and keyboard for interaction, and a Wi-Fi connection for networking. In both examples, computer buses of one form or another move data between all of these devices.

In most traditional computer architectures, the CPU and main memory tend to be tightly coupled. A microprocessor conventionally is a single chip which has a number of electrical connections on its pins that can be used to select an “address” in the main memory and another set of pins to read and write the data stored at that location. In most cases, the CPU and memory share signalling characteristics and operate in synchrony. The bus connecting the CPU and memory is one of the defining characteristics of the system, and often referred to simply as the system bus.

It is possible to allow peripherals to communicate with memory in the same fashion, attaching adaptors in the form of expansion cards directly to the system bus. This is commonly accomplished through some sort of standardized electrical connector, several of these forming the expansion bus or local bus. However, as the performance differences between the CPU and peripherals varies widely, some solution is generally needed to ensure that peripherals do not slow overall system performance. Many CPUs feature a second set of pins similar to those for communicating with memory, but able to operate at very different speeds and using different protocols. Others use smart controllers to place the data directly in memory, a concept known as direct memory access. Most modern systems combine both solutions, where appropriate.

As the number of potential peripherals grew, using an expansion card for every peripheral became increasingly untenable. This has led to the introduction of bus systems designed specifically to support multiple peripherals. Common examples are the SATA ports in modern computers, which allow a number of hard drives to be connected without the need for a card. However, these high-performance systems are generally too expensive to implement in low-end devices, like a mouse. This has led to the parallel development of a number of low-performance bus systems for these solutions, the most common example being Universal Serial Bus. All such examples may be referred to as peripheral buses, although this terminology is not universal.

In modern systems the performance difference between the CPU and main memory has grown so great that increasing amounts of high-speed memory is built directly into the CPU, known as a cache. In such systems, CPUs communicate using high-performance buses that operate at speeds much greater than memory, and communicate with memory using protocols similar to those used solely for peripherals in the past. These system buses are also used to communicate with most (or all) other peripherals, through adaptors, which in turn talk to other peripherals and controllers. Such systems are architecturally more similar to multicomputers, communicating over a bus rather than a network. In these cases, expansion buses are entirely separate and no longer share any architecture with their host CPU (and may in fact support many different CPUs, as is the case with PCI). What would have formerly been a system bus is now often known as a front-side bus.

Given these changes, the classical terms "system", "expansion" and "peripheral" no longer have the same connotations. Other common categorization systems are based on the buses primary role, connecting devices internally or externally, PCI vs. SCSI for instance. However, many common modern bus systems can be used for both; SATA and the associated eSATA are one example of a system that would formerly be described as internal, while in certain automotive applications use the primarily external IEEE 1394 in a fashion more similar to a system bus. Other examples, like InfiniBand and I^{2}C were designed from the start to be used both internally and externally.

11.1.1 Internal bus

The internal bus, also known as internal data bus, memory bus, system bus or Front-Side-Bus, connects all the internal components of a computer, such as CPU and memory, to the motherboard. Internal data buses are also referred to as a local bus, because they are intended to connect to local devices. This bus is typically rather quick and is independent of the rest of the computer operations.

11.1.2 External bus

The external bus, or expansion bus, is made up of the electronic pathways that connect the different external devices, such as printer etc., to the computer.

11.2 Implementation details

Buses can be parallel buses, which carry data words in parallel on multiple wires, or serial buses, which carry data in bit-serial form. The addition of extra power and control connections, differential drivers, and data connections in each direction usually means that most serial buses have more conductors than the minimum of one used in 1-Wire and UNI/O. As data rates increase, the problems of timing skew, power consumption, electromagnetic interference and crosstalk across parallel buses become more and more difficult to circumvent. One partial solution to this problem has been to double pump the bus. Often, a serial bus can be operated at higher overall data rates than a parallel bus, despite having fewer electrical connections, because a serial bus inherently has no timing skew or crosstalk. USB, FireWire, and Serial ATA are examples of this. Multidrop connections do not work well for fast serial buses, so most modern serial buses use daisy-chain or hub designs.

Network connections such as Ethernet are not generally regarded as buses, although the difference is largely conceptual rather than practical. An attribute generally used to characterize a bus is that power is provided by the bus for the connected hardware. This emphasizes the busbar origins of bus architecture as supplying switched or distributed power. This excludes, as buses, schemes such as serial RS-232, parallel Centronics, IEEE 1284 interfaces and Ethernet, since these devices also needed separate power supplies. Universal Serial Bus devices may use the bus supplied power, but often use a separate power source. This distinction is exemplified by a telephone system with a connected modem, where the RJ11 connection and associated modulated signalling scheme is not considered a bus, and is analogous to an Ethernet connection. A phone line connection scheme is not considered to be a bus with respect to signals, but the Central Office uses buses with cross-bar switches for connections between phones.

However, this distinction—that power is provided by the bus—is not the case in many avionic systems, where data connections such as ARINC 429, ARINC 629, MIL-STD-1553B (STANAG 3838), and EFABus (STANAG 3910) are commonly referred to as "data buses" or, sometimes, "databuses". Such avionic data buses are usually characterized by having several equipments or Line Replaceable Items/Units (LRI/LRUs) connected to a common, shared media. They may, as with ARINC 429, be simplex, i.e. have a single source LRI/LRU or, as with ARINC 629, MIL-STD-1553B, and STANAG 3910, be duplex, allow all the connected LRI/LRUs to act, at different times (half duplex), as transmitters and receivers of data.[2]

11.3 History

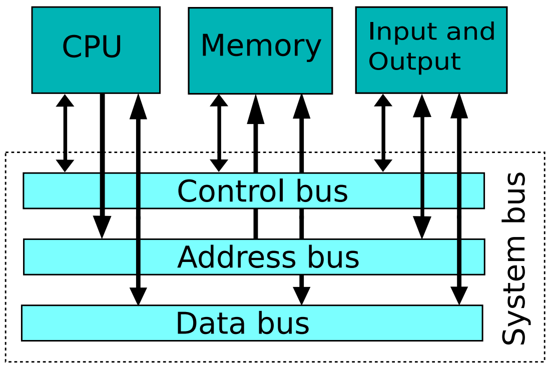
Over time, several groups of people worked on various computer bus standards, including the IEEE Bus Architecture Standards Committee (BASC), the IEEE "Superbus" study group, the open microprocessor initiative (OMI), the open microsystems initiative (OMI), the "Gang of Nine" that developed EISA, etc.

11.3.1 First generation

Early computer buses were bundles of wire that attached computer memory and peripherals. Anecdotally termed the "*digit trunk*",[3] they were named after electrical power buses, or busbars. Almost always, there was one bus for memory, and one or more separate buses for peripherals. These were accessed by separate instructions, with completely different timings and protocols.

One of the first complications was the use of interrupts. Early computer programs performed I/O by waiting in a loop for the peripheral to become ready. This was a waste of time for programs that had other tasks to do. Also, if the program attempted to perform those other tasks, it might take too long for the program to check again, resulting in loss of data. Engineers thus arranged for the peripherals to interrupt the CPU. The interrupts had to be prioritized, because the CPU can only execute code for one peripheral at a time, and some devices are more time-critical than others.

High-end systems introduced the idea of channel controllers, which were essentially small computers dedicated to handling the input and output of a given bus. IBM introduced these on the IBM 709 in 1958, and they became a common feature of their platforms. Other high-performance vendors like Control Data Corporation implemented similar designs. Generally, the channel controllers would do their best to run all of the bus operations internally, moving data when the CPU was known to be busy elsewhere if possible, and only using interrupts when necessary. This greatly reduced CPU load, and provided better overall system performance.



Single system bus

To provide modularity, memory and I/O buses can be combined into a unified system bus.[4] In this case, a single mechanical and electrical system can be used to connect together many of the system components, or in some cases, all of them.

Later computer programs began to share memory common to several CPUs. Access to this memory bus had to be prioritized, as well. The simple way to prioritize interrupts or bus access was with a daisy chain. In this case signals will naturally flow through the bus in physical or logical order, eliminating the need for complex scheduling.

11.3.2 Minis and micros

Digital Equipment Corporation (DEC) further reduced cost for mass-produced minicomputers, and mapped peripherals into the memory bus, so that the input and output devices appeared to be memory locations. This was implemented in the Unibus of the PDP-11 around 1969.[5]

Early microcomputer bus systems were essentially a passive backplane connected directly or through buffer amplifiers to the pins of the CPU. Memory and other devices would be added to the bus using the same address and data pins as the CPU itself used, connected in parallel. Communication was controlled by the CPU, which had read and written data from the devices as if they are blocks of memory, using the same instructions, all timed by a central clock controlling the speed of the CPU. Still, devices interrupted the CPU by signaling on separate CPU pins. For instance, a disk drive controller would signal the CPU that new data was ready to be read, at which point the CPU would move the data by reading the "memory location" that corresponded to the disk drive. Almost all early microcomputers were built in this fashion, starting with the S-100 bus in the Altair 8800 computer system.

In some instances, most notably in the IBM PC, although similar physical architecture can be employed, instructions to access peripherals (in and out) and memory (mov and others) have not been made uniform at all, and still generate distinct CPU signals, that could be used to implement a separate I/O bus.

These simple bus systems had a serious drawback when used for general-purpose computers. All the equipment on the bus has to talk at the same speed, as it shared a single clock.

Increasing the speed of the CPU becomes harder, because the speed of all the devices must increase as well. When it is not practical or economical to have all devices as fast as the CPU, the CPU must either enter a wait state, or work at a slower clock frequency temporarily,[6] to talk to other devices in the computer. While acceptable in embedded systems, this problem was not tolerated for long in general-purpose, user-expandable computers.

Such bus systems are also difficult to configure when constructed from common off-the-shelf equipment. Typically each added expansion card requires many jumpers in order to set memory addresses, I/O addresses, interrupt priorities, and interrupt numbers.

11.3.3 Second generation

“Second generation” bus systems like NuBus addressed some of these problems. They typically separated the computer into two “worlds”, the CPU and memory on one side, and the various devices on the other. A *bus controller* accepted data from the CPU side to be moved to the peripherals side, thus shifting the communications protocol burden from the CPU itself. This allowed the CPU and memory side to evolve separately from the device bus, or just “bus”. Devices on the bus could talk to each other with no CPU intervention. This led to much better “real world” performance, but also required the cards to be much more complex. These buses also often addressed speed issues by being “bigger” in terms of the size of the data path, moving from 8-bit parallel buses in the first generation, to 16 or 32-bit in the second, as well as adding software setup (now standardised as Plug-n-play) to supplant or replace the jumpers.

However these newer systems shared one quality with their earlier cousins, in that everyone on the bus had to talk at the same speed. While the CPU was now isolated and could increase speed, CPUs and memory continued to increase in speed much faster than the buses they talked to. The result was that the bus speeds were now very much slower than what a modern system needed, and the machines were left starved for data. A particularly common example of this problem was that video cards quickly outran even the newer bus systems like PCI, and computers began to include AGP just to drive the video card. By 2004 AGP was outgrown again by high-end video cards and other peripherals and has been replaced by the new PCI Express bus.

An increasing number of external devices started employing their own bus systems as well. When disk drives were first introduced, they would be added to the machine with a card plugged into the bus, which is why computers have so many slots on the bus. But through the 1980s and 1990s, new systems like SCSI and IDE were introduced to serve this need, leaving most slots in modern systems empty. Today there are likely to be about five different buses in the typical machine, supporting various devices.

11.3.4 Third generation

See also: Bus network

“Third generation” buses have been emerging into the market since about 2001, including HyperTransport and InfiniBand. They also tend to be very flexible in terms of their physical connections, allowing them to be used both as internal buses, as well as connecting different machines together. This can lead to complex problems when trying to service different requests, so much of the work on these systems concerns software design, as opposed to the hardware itself. In general, these third generation buses tend to look more like a network than the original concept of a bus, with a higher protocol overhead needed than early systems, while also allowing multiple devices to use the bus at once.

Buses such as Wishbone have been developed by the open source hardware movement in an attempt to further remove legal and patent constraints from computer design.

11.4 Examples of internal computer buses

11.4.1 Parallel

- ASUS Media Bus proprietary, used on some ASUS Socket 7 motherboards
- Computer Automated Measurement and Control (CAMAC) for instrumentation systems
- Extended ISA or EISA
- Industry Standard Architecture or ISA
- Low Pin Count or LPC
- MBus
- MicroChannel or MCA
- Multibus for industrial systems
- NuBus or IEEE 1196
- OPTi local bus used on early Intel 80486 motherboards.
- Conventional PCI
- Parallel ATA (also known as Advanced Technology Attachment, ATA, PATA, IDE, EIDE, ATAPI, etc.) disk/tape peripheral attachment bus
- S-100 bus or IEEE 696, used in the Altair and similar microcomputers
- SBus or IEEE 1496
- SS-50 Bus
- Runway bus, a proprietary front side CPU bus developed by Hewlett-Packard for use by its PA-RISC microprocessor family
- GSC/HSC, a proprietary peripheral bus developed by Hewlett-Packard for use by its PA-RISC microprocessor family
- Precision Bus, a proprietary bus developed by Hewlett-Packard for use by its HP3000 computer family
- STEbus

- STD Bus (for STD-80 [8-bit] and STD32 [16-/32-bit]), FAQ
- Unibus, a proprietary bus developed by Digital Equipment Corporation for their PDP-11 and early VAX computers.
- Q-Bus, a proprietary bus developed by Digital Equipment Corporation for their PDP and later VAX computers.
- VESA Local Bus or VLB or VL-bus
- VMEbus, the VERSAmodule Eurocard bus
- PC/104
- PC/104 Plus
- PC/104 Express
- PCI-104
- PCIe-104
- Zorro II and Zorro III, used in Amiga computer systems

11.4.2 Serial

- 1-Wire
- HyperTransport
- I^{2}C
- PCI Express or PCIe
- Serial ATA (SATA)
- Serial Peripheral Interface Bus or SPI bus
- UNI/O
- SMBus

11.5 Examples of external computer buses

11.5.1 Parallel

- HIPPI HIgh Performance Parallel Interface
- IEEE-488 (also known as GPIB, General-Purpose Interface Bus, and HPIB, Hewlett-Packard Instrumentation Bus)
- PC Card, previously known as *PCMCIA*, much used in laptop computers and other portables, but fading with the introduction of USB and built-in network and modem connections

11.5.2 Serial

- Controller area network ("CAN bus")
- eSATA
- ExpressCard
- Fieldbus
- IEEE 1394 interface (FireWire)
- RS-232
- RS-485
- Thunderbolt
- USB Universal Serial Bus, used for a variety of external devices

11.6 Examples of internal/external computer buses

- Futurebus
- InfiniBand
- PCI Express External Cabling
- QuickRing
- Scalable Coherent Interface (SCI)
- SCSI Small Computer System Interface, disk/tape peripheral attachment bus
- Serial Attached SCSI (SAS) and other serial SCSI buses
- Thunderbolt
- Yapbus, a proprietary bus developed for the Pixar Image Computer

11.7 See also

- Address bus
- Bus contention
- Control bus
- Front-side bus (FSB)
- External Bus Interface (EBI)
- Harvard architecture
- Network On Chip
- List of device bandwidths
- List of network buses
- Software bus

11.8 References

[1] “bus Definition from PC Magazine Encyclopedia”. pcmag.com. 2014-05-29. Retrieved 2014-06-21.

[2] Avionic Systems Standardisation Committee, *Guide to Digital Interface Standards For Military Avionic Applications*, ASSC/110/6/2, Issue 2, September 2003

[3] See the early Australian CSIRAC computer

[4] Linda Null; Julia Lobur (2006). *The essentials of computer organization and architecture* (2nd ed.). Jones & Bartlett Learning. pp. 33,179–181. ISBN 978-0-7637-3769-6.

[5] C. Gordon Bell; R. Cady; H. McFarland; J. O'Laughlin; R. Noonan; W. Wulf (1970). “A New Architecture for Mini-Computers—The DEC PDP-11”. *Spring Joint Computer Conference*: 657–675.

[6] Bray, Andrew C.; Dickens, Adrian C.; Holmes, Mark A. (1983). “28. The One Megahertz bus”. *The Advanced User Guide for the BBC Microcomputer* (zipped PDF). Cambridge, UK: Cambridge Microcomputer Centre. pp. 442–443. ISBN 0-946827-00-1. Retrieved 2008-03-28.

11.9 External links

- Computer hardware buses at DMOZ
- Computer hardware buses and slots pinouts with brief descriptions

11.10 Text and image sources, contributors, and licenses

11.10.1 Text

Foxj, Wikievil666, The Thing That Should Not Be, Rilak, Jan1nad, Knepflerle, Pheeror, Arakunem, Starcraft.nut, Myhellhereorunder, Yabeeno, Matt 118118, Blanchardb, Ridge Runner, Rprpr, Rockfang, Pointillist, Sensiblemayank, Hallo990, Catfish Jim and the soapdish, Excirial, Goodone121, Tomeasy, Koishii1521, Alejandrocaro35, Tyler, Jjtennisman, Ben ben ben ben ben jerry, Dekisugi, XTerminator2000, Netanel h, Deerstop, Aitias, Flipper344, Versus22, Lambtron, SoxBot III, Ginbot86, XLinkBot, Hotcrocodile, Fastily, Gwandoya, Stickee, Rror, Feinoha, Little Mountain 5, Rreagan007, SilvonenBot, NellieBly, Galzigler, Jd027, Torahjerus14, JinJian, Zodon, Lanky217, Dsimic, Tim1980tim, Addbot, Winstonliang6758, Pyfan, Raghavkvp, Mortense, Sdfsakjdhfuioaheif283, Manuel Trujillo Berges, Landon1980, Captain-tucker, LHvU, Peti1212, Turnerj, Ronhjones, Fieldday-sunday, Ironholds, D0762, Darklightning1, Scientus, CanadianLinuxUser, MrOllie, Glane23, Matthewirwin28693, WikiDegausser, Jasper Deng, Beastathon, Brainmachine, Numbo3-bot, Bwrs, DNA to RNA, Tide rolls, Krano, Teles, Gail, Zorrobot, Jarble, Ettrig, Luckas-bot, Yobot, Mcdennis13, Fraggle81, Phatstakks, Pikachu, ArchonMagnus, Mmxx, Knockwood, KamikazeBot, Timir Saxa, AnakngAraw, Radiopathy, Deicool, AnomieBOT, Rubinbot, Jim1138, IRP, Galoubet, Piano non troppo, AdjustShift, Fahadsadah, Kingpin13, RandomAct, Flewis, Materialscientist, The High Fin Sperm Whale, Citation bot, Juhuang, X360Silent, Georgeb92, Wx4sno, Milf&cookies, GB fan, JohnFromPinckney, 04satvinderbi, Xqbot, 4I7.4I7, Cureden, The sock that should not be, Capricorn42, 4twenty42o, Knowitall44, Jsharpminor, Blargarg, GrouchoBot, Nayvik, Solphusion, Wizardist, RibotBOT, Mpgenius, Genius1789, SCΛRECROW, Luciandrei, Sicklight, Jarred.rop1, Full-hyperion, Shadowjams, Dougofborg, Cekli829, GT5162, FrescoBot, Bighead01753, LucienBOT, Rome109, X5UPR3ME STEV3x, VI, Jamesooders, Citation bot 1, Wdfowty, Pooosihole, FMAlchemist36, HRoestBot, Edderso, A412, Coekon, Calmer Waters, Tinton5, BRUTE, RedBot, SpaceFlight89, Footwarrior, Lineslarge, Rzęsor, Pranayrocks23, WardMuylaert, RMartin-2, ئاراس نوری, Peterlunde, Thestraycat57, Ravenperch, Vrenator, Defender of torch, Momma69, Aiken drum, Bitolado, Jeffrd10, Specs112, Diannaa, Tbhotch, Reach Out to the Truth, Jesse V., Minimac, Hornlitz, 11james22, DARTH SIDIOUS 2, Onel5969, Dmytheus, TjBot, Ihateblazing, Dhiraj1984, Noommos, Lauri.pirttiaho, Instigate cjsc (Narine), Thisisafakeaccountfordeletingpages, EmausBot, Orphan Wiki, WikitanvirBot, Mynameiswa, BillyPreset, Akjar13, Racerx11, Wikipelli, Mz7, ZéroBot, John Cline, Cogiati, Daonguyen95, Ida Shaw, Fæ, Imperial Monarch, Pololei, Espin2, Fred Gandt, Stas3717, Wayne Slam, Ferry24.Milan, Arman Cagle, LordJeff, Shrikanthv, GeorgeBarnick, L Kensington, Donner60, Wikiloop, Orange Suede Sofa, Ipsign, Jrstern29, LikeLakers2, Erin2003, 28bot, ClueBot NG, Bionik276, TomBridle, Satellizer, Andreas.Persson, Movses-bot, Millermk, Rajayush78, Cntras, O.Koslowski, Kasirbot, Masssly, Widr, Greg.Kerr01, Ashish Gaikwad, Jorgenev, Mapolat, Adam.Amory.97, Yucel114, Hazal018, HMSSolent, Wbm1058, Nashhinton, Dogan900, Wiki13, MusikAnimal, Amp71, IraChesterfield, Jonas weepel, Kinnngg, Rm1271, Dauntless28, Anchit singla, Camomen3000, Glacialfox, Klilidiplomus, MantridFreeman, Mikepabell, Chip123456, Fylbecatulous, BattyBot, Justincheng12345-bot, WhiteNebula, Computerwoman417, Benjaminjkim, CKsquid, YFdyhbot, Ekonomka, Modwizcode, MadGuy7023, Gdrg22, Tow, Sniper-ass, Dexbot, KuraAbyss, Taylorclem, ZaferXYZ, Frosty, Graphium, Jamesx12345, Trollilols, Botusharov, EdwardJK, Jazzee8845, Andypandyjay, Reatlas, Greengreengreenred, Sancho .308, Thatginernonce, Mndunne, Bobobob1231, DavidLeighEllis, Edwinblack1, Comp.arch, Ugog Nizdast, Fwiki225, Riehutunut, AddWittyNameHere, OccultZone, JammedWords, Lopsidedtriangle, Djgolam, FlashDave, UltraFireFX, Sonyoo1, GeorgeAhad, TKer193, Monkbot, Rudyjuliyanto, MatthewBuchwalder, Jim Carter, Trax support, Sperkowsky, Keonnikpour, Vinaya.mente, QueenFan, TerryAlex, Siraj Khurshied, Anish057, ChamithN, Darknesssnow, KOool23456, Joeseph kake, Sivaprasadmsr, ComsciStudent, Israelg99, TurkeyWriter77, Benji877, TehBacon, Lumpy247, Troll42069, Thefreeencyclopedia1231, Robolamiah, Superbug1000, Dou14208038, Territory war 3 pwner and Anonymous: 1331

- **Chipset** *Source:* http://en.wikipedia.org/wiki/Chipset?oldid=650476841 *Contributors:* Andre Engels, Michael Hardy, Alfio, Harvester, RickK, David Shay, Robbot, Digital infinity, Presto8, Aechols, Gracefool, Khalid hassani, Ryanaxp, Chowbok, Jonathan Grynspan, Karl-Henner, Maikel, Moxfyre, EagleOne, Discospinster, Pixel8, WegianWarrior, CanisRufus, Kyz, Mairi, Spalding, Polluks, Dungodung, Alansohn, Atlant, Alinor, Suruena, Dtcdthingy, Sciurinæ, Boothy443, Timharwoodx, Rocastelo, Robert K S, Knuckles, Alexthemans, Isnow, Alecv, Kbdank71, Reisio, Ketiltrout, Bubuka, FlaBot, Fivemack, Geimas5, M7bot, Chobot, YurikBot, Vuvar1, Ansell, NawlinWiki, Ethan, Robertvan1, DeadEyeArrow, GaryKlein, Kkmurray, Wknight94, Lt-wiki-bot, Eskimbot, Agentbla, Jerome Charles Potts, Wisden17, Frap, JonHarder, HarisM, JzG, Gobonobo, Robofish, Feureau, CUTKD, Mets501, Kvng, JustinRossi, Amalas, Trentdk, Iliank, Thijs!bot, Barticus88, Ranunculoid, AntiVandalBot, Janus657, MER-C, Geniac, Magioladitis, Twsx, Allstarecho, ClubOranje, Conquerist, BetBot, Jargon777, Cerebral0214, Y2kbug, Jesant13, Whitebox, Half-Blood Auror, D-Kuru, Adam Zivner, VolkovBot, Philip Trueman, OracleGuy01, GL1zdA, Andy Dingley, Hellcat fighter, Bojan PLOJ, Haiviet, Jakeslogan, Btotanes, SieBot, Agarfield2004, Gerakibot, Phe-bot, Green-eyed girl, Oxymoron83, Dlrohrer2003, ClueBot, Rilak, ImperfectlyInformed, Zacharychung, Rprpr, Jusdafax, Arjayay, JimMobo, DumZiBoT, Hyins, Addbot, MrOllie, Sanchayansinha, Oldmountains, Eddau, Xalteva, Tide rolls, Lightbot, Luckas-bot, AnomieBOT, Bobi.1, Tehori, Kingpin13, Materialscientist, SvartMan, LouriePieterse, Xqbot, J G Campbell, RibotBOT, Kyng, Der Falke, Mfwitten, Arndbergmann, Jandalhandler, TobeBot, Borandi, Mattias.Campe, EmausBot, Angrytoast, Tecnova, Klelith, ClueBot NG, Martin Berka, Mudasir005, Wbm1058, Habitmelon, Bcmengel, Arcticaribou, BattyBot, BjornVanB, Sumasdad, C5st4wr6ch, Infiniti4, Julianprescott2604juuly and Anonymous: 134
- **Graphics processing unit** *Source:* http://en.wikipedia.org/wiki/Graphics%20processing%20unit?oldid=653012511 *Contributors:* Taw, Wayne Hardman, Heron, Edward, DopefishJustin, Mahjongg, Nixdorf, Karada, Egil, Andres, Harvester, Lee Cremeans, Furrykef, Tempshill, Wernher, Thue, Topbanana, Stormie, Optim, Robbot, Chealer, Vespristiano, Playwrite, Academic Challenger, Tobias Bergemann, Alan Liefting, Alf Boggis, Paul Pogonyshev, Everyking, Alison, Lurker, DJSupreme23, Gracefool, Rchandra, AlistairMcMillan, Egomaniac, Khalid hassani, Gadfium, Utcursch, Pgan002, Aughtandzero, Quadell, Lockeownzj00, Beland, MFNickster, Simoneau, Trilobite, Imroy, Pixel8, AlexKepler, Berkut, Alistair1978, Pavel Vozenilek, Gronky, Indrian, Evice, Billlion, TOR, CanisRufus, RoyBoy, Drhex, Polluks, Matt Britt, Richi, Kjkolb, Markpapadakis, Kaf, Varuna, Murphykieran, Mc6809e, Hohum, Angelic Wraith, Velella, Suruena, Sciurinæ, Bjorke, Freyr, Marasmusine, Kelly Martin, Woohookitty, Jannex, Ae-a, Macronyx, SCEhardt, Isnow, M412k, Toussaint, Kbdank71, Josh Parris, Tbird20d, Sdornan, Sango123, StuartBrady, FlaBot, Mirror Vax, Arnero, Viznut, Chobot, ShadowHntr, YurikBot, Jtbandes, Locke411, Yyy, ALoopingIcon, Virek, RicReis, Qviri, Panscient, Zephalis, Mike92591, MaxDZ8, Wknight94, Delirium of disorder, Arthur Rubin, D'Agosta, E Wing, Red Jay, David Biddulph, Mikkow, Nekura, Veinor, FearTec, SmackBot, Colinstu, AFBorchert, Bigbluefish, Unyoyega, Jagged 85, Renku, KVDP, Jrockley, Eskimbot, Scott Paeth, Jpvinall, Gilliam, Bluebot, TimBentley, GoldDragon, QTCaptain, Thumperward, Jerome Charles Potts, Octahedron80, Anabus, Tsca.bot, Can't sleep, clown will eat me, Harumphy, Frap, JonHarder, Ruw1090, Easwarno1, Theonlyedge, Cybercobra, Melter, Nakon, Trieste, HarisM, Nitro912gr, Swaaye, Salamurai, HeroTsai, Soumya92, Disavian, Wibbble, Joffeloff, Codepro, Aleenf1, Vuurmeester, Phranq, Cxk271, Sjf, Hu12, Stargaming, Agelu, ScottHolden, Stoakron97, Aeons, Tawkerbot2, Jafet, Braddodson, SkyWalker, Xcentaur, Zarex, Mattdj, Nczempin, Jsmaye, Jesse Viviano, Shandris, Lazulilasher, Sahrin, Pi Guy 31415, Phatom87, Danrok, JJC1138, Gogo Dodo, Scissorhands1203, Soetermans, Mr. XYZ, Tawkerbot4, Bitsmart, Thijs!bot, Wermlandsdata, Mentifisto, Eberhart, AntiVandalBot, Konman72, Gioto, SEG88, Flex Flint, Johan.Seland, Skarkkai, Serpent's Choice, JAnDbot, MER-C, Jdevesa, Arch dude, Kremerica, RubyQ, Vidsi, AndriusG, RBBrittain, Gbrose85, Michaelothomas, Nikevich, I JethroBT, Marmoulak, David Eppstein, Crazyideas21, Frampis, El Krem, UnfriendlyFire, Trusader, R'n'B, J.delanoy, Pharaoh of the Wizards, ChrisfromHouston, Jesant13, Smite-Meister, Gzkn, Xbspiro, M-le-mot-dit, Urzadek, Jo7hs2, EconomistBR, Sugarbat, Spiesr, Canadianbob, Martial75, Lights, VolkovBot, MrRK, TXiKiBoT, Like.liberation, Tr-the-maniac,

load, LaaknorBot, Glass Sword, AndersBot, Jonhjelm, Favonian, Jasper Deng, West.andrew.g, Akyoyo94, Fireaxe888, KaiKemmann, Tide rolls, SDJ, Llakais, טבעת-זרם, LuK3, Megaman en m, Ochib, Frehley, Luckas-bot, ZX81, Yobot, Ptbotgourou, TaBOT-zerem, Snydale, II MusLiM HyBRiD II, Crispmuncher, THEN WHO WAS PHONE?, محبوب عالم, Eric-Wester, AnomieBOT, Ciphers, Rubinbot, 1exec1, Jim1138, IRP, Bmonro, Theqwert, Kingpin13, Law, RandomAct, Flewis, Materialscientist, Bobagoncheese, Blackdogman, Pcb95, Kevin chen2003, Xqbot, Kagemaru16, Capricorn42, Doanjackson, DSisyphBot, Jsharpminor, Ommel, Ubcule, J04n, Frosted14, Wizardist, Newport Beach, SciberDoc, Papercutbiology, Mark Schierbecker, RibotBOT, The Interior, Crashdoom, Amaury, KB Alpha, Masterofabcs, GhalyBot, =Josh.Harris, Savannah Kaylee, CodeMaster123, Chaheel Riens, Alan1000, ELCleanup, Fillepa, Jimfile, ComputerWizerd, Super IT guy, Edgars2007, GliderMaven, FrescoBot, Scarypeep, W Nowicki, Grinters, Sagark86, Eehite, SkyHigher, Ctech72, Pinethicket, Boulaur, Edderso, Callofduty4, Hoo man, RedBot, Phearson, Serols, ALEF7, Gapaddict, Lineslarge, RazielZero, Reconsider the static, Irbisgreif, Lando Calrissian, Hwan051, TobeBot, Shubham18, Yunshui, Compvis, LogAntiLog, Tubby23, Ravenperch, Dinamik-bot, Vrenator, Clarkcj12, SeoMac, Coercorash, Seahorseruler, Bob360bob360, Diannaa, Raykyogrou0, Suffusion of Yellow, Jesse V., Lord of the Pit, DARTH SIDIOUS 2, Mindymoo22, Jfmantis, Addera, DexDor, Prakashkumaraman, DRAGON BOOSTER, Khanbm, Skamecrazy123, EmausBot, Stryn, Twilight1188, Reddysan345, Super48paul, Nintnt, RA0808, Solarra, Hounder4, Wikipelli, K6ka, Anirudh Emani, Savh, 2TerabyteBox, L Kensington, Donner60, Wikiloop, Usb10, Puffin, Damirgraffiti, DASHBotAV, Purpledramallama, Rocketrod1960, ClueBot NG, Mechanical digger, Rich Smith, Smtchahal, MelbourneStar, Satellizer, Kikichugirl, Bped1985, DieSwartzPunkt, Widr, Narwhallrus, Frosty3219, Notting Hill in London, Anupmehra, AlexanderDS, Sw33tkill3r24, Basak327, Aslihanbilgekurt, Oddbodz, Arbraini, Anuragiscool12, Aogus, Hsn6161, Onur074, Sinohayja, HMSSolent, Wbm1058, Zim Lex, 2001:db8, Hikmet483, DBigXray, Wowkiddymage, Vagobot, M0rphzone, Stevethepanda, Dharmuone, CityOfSilver, Bigjackg, Hallows AG, Wiki13, Mark Arsten, Rm1271, Hllomen, Tehh bakery, Vesna Wylde, Pano38, Jayadevp13, TheLurkerMan, Vanischenu, Klilidiplomus, Bsdjkqvfkj, Seesh cabeesh, Maxxdxx, Anbu121, BattyBot, Jeremy112233, Teammm, Pratyya Ghosh, Theamazingswamp, TheCascadian, Cyberbot II, Mediran, Khazar2, EuroCarGT, Marsel92 22, Lakers297065, Theramcerebral, Abhikandoi2000, Will Sandberg, ZaferXYZ, Sourav255, Scottyj200, TwoTwoHello, Frosty, SFK2, Graphium, Sriharsh1234, Milesandkilometrestogo, Liamgaygay, RandomLittleHelper, Raj Singh, Craven, Faizan, Camyoung54, AKATRAZ99, Jacob neale, Daniiielc, Handsome cat, Everymorning, EvergreenFir, Probusiness, DavidLeighEllis, Dannyhacker1, Babitaarora, PasseVivant, My name is not dave, Jianhui67, Miner49er5635, Jackmcbarn, SDTV is beast123, محمد عاصم, Anarcham, DPRoberts534, Joewithers89, Mikeycpud, Yaakovaryeh, Sirspray, UltraFireFX, 7Sidz, Avidee007, LukeSmithlol, Mr Bo Janglez 69, Maxlan1, BethNaught, Kzorq, Swallis11, Scrappyboy88, Cncreate, Ballistic238, NQ, Frinked.tpm, Mjiles85, Gvkmohan5, ChamithN, TAYLORGATE, Crystallizedcarbon, Ggggg1234, Rombom4, Abcdefgjit, Hongkongfoooi, Esquivalience, Brajmohan Pathak, Benji877, Butcrackman, Abu ali-shabat thawadi, AveragelyPro and Anonymous: 1475

- **Read-only memory** *Source:* http://en.wikipedia.org/wiki/Read-only%20memory?oldid=645463645 *Contributors:* Damian Yerrick, Derek Ross, Brion VIBBER, Stephen Gilbert, Drj, Toby Bartels, Ben-Zin, Patrick, RTC, Michael Hardy, TakuyaMurata, Pcb21, Ahoerstemeier, Mac, Glenn, Netsnipe, Timwi, Cjmnyc, WhisperToMe, Wernher, דוד, Traroth, Shantavira, Riddley, Kizor, Boffy b, Gidonb, Blainster, Wikibot, Tobias Bergemann, Wizzy, DavidCary, ShaunMacPherson, Kim Bruning, Bradeos Graphon, Niteowlneils, Sdfisher, Finn-Zoltan, Gracefool, Luigi30, VampWillow, Pne, Bobblewik, TerokNor, Jpeloquin, Ojw, Chmod007, Moxfyre, Zoganes, Lovelac7, ArnoldReinhold, Xezbeth, Quistnix, ESkog, Khalid, CanisRufus, El C, Bobo192, Harald Hansen, Smalljim, .:Ajvol:., Nk, Obradovic Goran, Hooperbloob, Shadoks, Jumbuck, Jic, 119, Atlant, Craigy144, Angelic Wraith, Wtshymanski, Tony Sidaway, ThomasWinwood, Firsfron, Woohookitty, Camw, Optichan, Isnow, Bruns, Mandarax, Graham87, Tangotango, Bruce1ee, DirkvdM, FlaBot, Fresheneesz, CStyle, Chobot, Former user 6, YurikBot, Laurentius, SpuriousQ, Stephenb, Shell Kinney, Yyy, Shanel, NawlinWiki, Wiki alf, Scs, Zwobot, Allens, GrinBot, Krótki, Veinor, SmackBot, Pgk, KocjoBot, Mdd4696, Nscheffey, Brianski, Skizzik, GoneAwayNowAndRetired, JDCMAN, DMS, Mattythewhite, DHN-bot, KieferSkunk, Can't sleep, clown will eat me, Frap, Xmastree, Jmlk17, HarisM, Salamurai, Ugur Basak Bot, Fire emblem, Microchip08, Dicklyon, TheOtherStephan, Axipher, Beno1000, CapitalR, Blehfu, Tawkerbot2, Unionhawk, Dgw, Green caterpillar, DanielRigal, Neelix, Djg2006, Thijs!bot, Epbr123, Barticus88, Bezking, Qwyrxian, Daniel, N5iln, AntiVandalBot, Majorly, JAnDbot, MER-C, Geniac, ZPM, Pedro, Bongwarrior, VoABot II, AtticusX, Soulbot, JaGa, Gwern, MartinBot, R'n'B, Thugofnewbold, Darin-0, Brest, Jesant13, Tiggerjay, Kvdveer, CardinalDan, EEye, Indubitably, Darksideofarollingstone, DoorsAjar, TXiKiBoT, Hqb, GDonato, Falcon8765, Spinningspark, AlleborgoBot, Nagy, EmxBot, SieBot, AS, BotMultichill, Winchelsea, Timhowardriley, Jerryobject, Happysailor, Universalcosmos, Lagrange613, Frappucino, StaticGull, Anchor Link Bot, Dolphin51, ClueBot, Fyyer, The Thing That Should Not Be, Ramstwer, Boing! said Zebedee, Niceguyedc, Muhandes, Posix memalign, OekelWm, Ottawa4ever, Bathis, Joel Saks, Patrick.franklin, XLinkBot, BodhisattvaBot, WikHead, Dsimic, VanMerde, Addbot, Mortense, Fyrael, Atethnekos, Sergei, MrOllie, Favonian, Tassedethe, Tide rolls, Lightbot, Luckas-bot, Ptbotgourou, Nallimbot, KDS4444, Jim1138, Dwayne, School1015895, Kingpin13, ArthurBot, Xqbot, Nasnema, XZeroBot, Knuckx, RibotBOT, The Interior, DaleDe, Salient Edge, Ghoshs89, Thehelpfulbot, Alexf0708, Glider87, Pinethicket, HRoestBot, MastiBot, Ezhuttukari, Amar007sv, TobeBot, Lotje, Ansumang, Minimac, DARTH SIDIOUS 2, Onel5969, EmausBot, Acather96, Dewritech, Vanished user zq46pw21, Wikipelli, Jplcrd, Wayne Slam, Music Sorter, Rombomb12, Bomazi, EvanAmos, Pana Gill, Whoop whoop pull up, ClueBot NG, Aakashgv, Sph3698, O.Koslowski, Aslihanbilgekurt, John.nuttall.jr, Wbm1058, Usman Nasir Khan, Rancher 42, Rynsaha, Rajeevs1992, Bsdjkqvfkj, Seesh cabeesh, Thomasdavies1993, Lukas^{23}, None but shining hours, Makecat-bot, Lugia2453, Coketales, My-2-bits, Luke Watto, MasterTriangle12, Fudgcker33, SantiLak, Happy Attack Dog, A.Minkowiski, Amortias, Gvkmohan5, Wiscipidier, Laudson, Sane007, Brajmohan Pathak, Ondrej34 and Anonymous: 308
- **BIOS** *Source:* http://en.wikipedia.org/wiki/BIOS?oldid=652046163 *Contributors:* Zundark, The Anome, Tarquin, Koyaanis Qatsi, Taw, Jeronimo, Andre Engels, Ted Longstaffe, Christian List, William Avery, Roadrunner, Edward, Michael Hardy, Modster, Tannin, Chinju, MichaelJanich, Ahoerstemeier, Cyan, Chrysalis, Robertkeller, Lenaic, Timwi, Dcoetzee, Dysprosia, Greenrd, WhisperToMe, Lee Cremeans, Rvalles, Jgm, Jimbreed, Fvw, Robbot, Dersonlwd, SchmuckyTheCat, Auric, Pcr, Wikibot, Cek, Iain.mcclatchie, Nagelfar, Alan Liefting, Xyzzyva, Giftlite, DavidCary, Tom harrison, Ferkelparade, Jdavidb, Chris Wood, AlistairMcMillan, Richard cocks, VampWillow, Edcolins, Chowbok, Andycjp, Knutux, Antandrus, OverlordQ, Piotrus, Imlepid, Maximaximax, Henriquevicente, Frau Holle, Salv236, Abdull, Zondor, Trevor MacInnis, Kate, Gazpacho, Mormegil, Hinrik, Discospinster, Rich Farmbrough, Guanabot, FT2, Caesar, Vsmith, ArnoldReinhold, Berkut, Dyl, ESkog, Andrejj, Swid, Evice, CanisRufus, Zenohockey, Bletch, Kwamikagami, Deanos, Bobo192, Cmdrjameson, Mactenchi, Matt Britt, JW1805, Cghost, Towel401, Hesperian, Officiallyover, Lysdexia, Jumbuck, Poweroid, Alansohn, Guy Harris, M7, Lord Pistachio, Cibumamo, Denniss, Wtshymanski, LFaraone, Ringbang, Ceyockey, Forderud, Peter Putzer, Kbolino, Billhpike, OwenX, Mindmatrix, Timharwoodx, Rocastelo, Peng, Pol098, Ruud Koot, WadeSimMiser, Fred J, Srborlongan, Wikiklrsc, Bluemoose, Sega381, Palica, Cuvtixo, Magister Mathematicae, Feydey, Alll, DoubleBlue, Kwharris, RoceKiller, Antimatt, FayssalF, Titoxd, FlaBot, Mirror Vax, SchuminWeb, Flydpnkrtn, Riluve, Jrtayloriv, Antiuser, Bgwhite, Mortenoesterlundjoergensen, YurikBot, Todd Vierling, Logixoul, RussBot, Fabartus, Ineedbettername, Epolk, SpuriousQ, Yuhong, Akamad, Stephenb, Manop, Sikon, Gaius Cornelius, Rsrikanth05, Wimt, NawlinWiki, Aeusoes1, Długosz, Jabencarsey, CecilWard, Cybergoth, Bayerischermann, Closedmouth, Arthur Rubin, KGasso, GraemeL, Snaxe920, GrinBot, CIreland, NetRolller 3D, SmackBot, Verdafolio, Bobet, Gribeco, IEdML, KocjoBot, Thunderboltz, Jedikaiti, Eskimbot, Srnec, Brianski, Ohnoitsjamie, Chris the speller, Michele.alessandrini, Bluebot, Jprg1966, Thumperward, Snori, Nbarth, DHN-bot, Darth Panda, Jimwelch, Audriusa, Can't sleep, clown will eat me, Blah2, Mitchell7man, Frap, Onorem, Burns

flipper, Nixeagle, Zazpot, The tooth, Luigi.a.cruz, Chopin1810, Kukini, SashatoBot, Harryboyles, Rodri316, Lakinekaki, Ocee, Rafert, Shadowlynk, Kashmiri, Minna Sora no Shita, Turanyuksel, Prasannaxd, Slakr, Yodaat, Epimorph, BranStark, Kiwi8, Asmpgmr, Rigurat, Blehfu, Tawkerbot2, Jafet, Drachenstern, CmdrObot, Caomhin, Chrumps, Coolerhead, Basawala, Harej bot, Jesse Viviano, Requestion, WeggeBot, T23c, Michael J. Mullany, Funnyfarmofdoom, Inzy, A876, Dowieite, RohitUpadhyay, Underpants, Inkington, Gimmetrow, Littlegeisha, Thijs!bot, Diophantus, Epbr123, A3RO, TheJosh, RFerreira, Mrpaco, Fx6893, KrakatoaKatie, AntiVandalBot, Thematrixeatsyou, Gioto, Luna Santin, Widefox, LibLord, Myanw, Golgofrinchian, Res2216firestar, Deflective, MER-C, Asnac, Xeno, Kipholbeck, Olipro, Bongwarrior, VoABot II, JamesBWatson, Venkatarao mynampati, Ishi Gustaedr, Tedickey, Axidos, Jim Douglas, Brusegadi, 28421u2232nfenfcenc, Wwmbes, Schevitz, Forcery, Gwern, Mermaid from the Baltic Sea, Owenja, Smokizzy, Ceros, Pharaoh of the Wizards, Uncle Dick, Jesant13, Lincstrunk, Bigdumbdinosaur, KylieTastic, Juliancolton, Vanished user 39948282, Bobo2000, Natl1, TheNewPhobia, Specter01010, CardinalDan, VolkovBot, Damonhill, Rhtc, Indubitably, TXiKiBoT, CoJaBo, Scienceguy2005, Una Smith, Hornet135, Biosman, ^demonBot2, Mannafredo, Haseo9999, ObjectivismLover, Spartacus106, TheAlmightyEgg, Jimmi Hugh, Radawson, Legoktm, TheStarman, Ruraloccur, Illinoisavonlady, Wjl2, SieBot, Josh the Nerd, Phagura, Althena, GrooveDog, Flyer22, Yatsu, Nopetro, Faradayplank, Binary Bird, Mwikgren, BenoniBot, Fratrep, OKBot, Anchor Link Bot, Jfromcanada, Zbisasimone, Loren.wilton, Martarius, Mbssbs, ClueBot, Avenged Eightfold, GorillaWarfare, Helpsloose, Spambrian, The Thing That Should Not Be, Voxpuppet, Rilak, Achadd, GreenSpigot, Mkmk101, Excirial, Goodone121, Bob man801, Boomhauer, Exit jones, Alejandrocaro35, Technobadger, Quentic, Gurpreet007, DanielPharos, Mapar007, Leffmann, Callmejosh, Jtle515, Versus22, SF007, Vanished user uih38riiw4hjlsd, Miami33139, DumZiBoT, LHMike, Nathan Johnson, Dark Mage, Kubus345, Avoided, Skarebo, Mchacotay, Dsimic, Count of Tuscany, Addbot, ConCompS, Dawynn, EjsBot, Fieldday-sunday, Fluffernutter, NjardarBot, Cst17, Download, LaaknorBot, 5 albert square, Tide rolls, Krano, ComputerHotline, Zorrobot, Jarble, Quantumobserver, A:-)Brunuś, Olsen-Fan, ProfessorThunderlips, Legobot, आशीष भटनागर, Yobot, Tohd8BohaithuGh1, Mklbtz, AnotherNitPicker, MrBurns, Backslash Forwardslash, AnomieBOT, Rubinbot, Jim1138, Xiaomao101, Flewis, LiuyuanChen, Limideen, X360Silent, Eumolpo, Lolschoolishorrible, Gsmgm, Sionus, RufusThorne, Sdcoonce, Grim23, Fortunez, TheIntersect, GrouchoBot, АлександрВв, Omnipaedista, ئاسۆ, Kyng, Amaury, Smallman12q, Bo98, Shadowjams, Chaheel Riens, Howard McCay, A.amitkumar, Krj373, Harsisus, Envirocbr school, Sae1962, Mfwitten, DivineAlpha, Cannolis, Citation bot 1, MichaelMatirko, Winterst, Calmer Waters, Skyerise, Btilm, Pnprasanth, Wizzboy34, Reconsider the static, Irbisgreif, BerndSinger, NortyNort, Yunshui, ItsZippy, Jonkerz, Zvn, Sgt. R.K. Blue, Suffusion of Yellow, DARTH SIDIOUS 2, Marky540, Jfmantis, Superfloccinaucinihilipilification, FetchcommsAWB, Tremaster, DASHBot, EmausBot, John of Reading, Gyorokpeter, Gfoley4, Dewritech, Sam Tomato, Valorum27, Wikipelli, Fæ, Coagulans, Kiwi128, Bamyers99, Cm621, Tolly4bolly, Hidbaty223, Palosirkka, Donner60, Orange Suede Sofa, Marcelogomez, Socialservice, Geosak, ClueBot NG, Astatine211, Gareth Griffith-Jones, Matthiaspaul, Aruffell, Jetpropulsion, Nantasatria, Simified, Widr, Mebob85, AlexWorth91, Helpful Pixie Bot, Illuminatii, Titodutta, Wbm1058, Kinaro, BG19bot, Numb3r4, Charliegreen1, MusikAnimal, AvocatoBot, AwamerT, Compfreak7, Wikia360, Sparkie82, Catcheshire, Skweezy, 220 of Borg, Linuxlalala, Metilisopropilisergamida, Rohitonline, Winston Chuen-Shih Yang, BattyBot, Cimorcus, Mcholbe2, Riley Huntley, ChrisGualtieri, YFdyh-bot, Tommy Wernryd, Codename Lisa, Kephir, Lugia2453, Spicyitalianmeatball, Frosty, EvilKeyboardCat, BitError, Acetotyce, PhantomTech, Tentinator, A1n2e3e4t5a, Ï¿½, LukusAreulius, Yaroslavas, Semsi Paco Virchow, Tertius51, Monkbot, Sofia Koutsouveli, Trackteur, EChastain, Absolution provider 1999, Pancho507 and Anonymous: 740

- **Bootstrapping** *Source:* http://en.wikipedia.org/wiki/Bootstrapping?oldid=651208930 *Contributors:* Damian Yerrick, Fnielsen, Adek, Tox, Ryguasu, Patrick, Michael Hardy, Alan Peakall, Nixdorf, Tannin, Graue, TakuyaMurata, Paddu, DavidWBrooks, Ronz, Angela, Randywombat, Nikai, Evercat, Smack, GRAHAMUK, Schneelocke, Dysprosia, Greenrd, Zoicon5, Omegatron, David.Monniaux, Robbot, Chealer, Jredmond, RedWolf, Kadin2048, Dittaeva, Tim Ivorson, Zidane2k1, Robbe, Eliashedberg, Asparagus, TexasDex, Mfc, Alan Liefting, DavidCary, Wolfkeeper, Zigger, Alison, Duncharris, Mboverload, Schweini, Pgan002, Quadell, Vbs, Lumidek, Fintor, Sysy, Brianhe, Rich Farmbrough, Rhobite, Guanabot, Silence, Ponder, CanisRufus, Pjrich, Sietse Snel, RoyBoy, Bobo192, O18, R. S. Shaw, Jerryseinfeld, Perceval, Diego Moya, Andrewpmk, Zippanova, Hu, Ish ishwar, DV8 2XL, Forderud, Oleg Alexandrov, Radiant, Natalya, Thryduulf, Clemmy, Turnstep, Graham87, BD2412, Syndicate, Hulagutten, ErikHaugen, Nirkus, Mirror Vax, Harmil, Krzysiu, Eldred, Jrtayloriv, Chobot, Uriah923, YurikBot, Kafziel, Esco, Conscious, Hellbus, Stephenb, Welsh, Adamrush, Kassie, Asams10, Cengelbart, Flooey, TimHesterberg, Dontaskme, That Guy, From That Show!, Eykanal, Johnmarkh, DocendoDiscimus, SmackBot, VnutZ, TheBilly, Unyoyega, Scott Paeth, Slaniel, Hmains, Bluebot, Ce1984, Thumperward, Nbarth, Chruck, LouScheffer, Adamantios, COMPFUNK2, Valenciano, Andrew Gildehaus, Bgoswami, Bige1977, Lambiam, Cody5, Archimerged, Rsimmonds01, Physis, 16@r, Smith609, Peyre, Snezzy, Phoenixrod, Glaberson, Frank Lofaro Jr., Npdoty, Cydebot, Ahawowow, Rdls01, Kozuch, Pellucid, Ebraminio, Ncjones, Thijs!bot, Seaphoto, SummerPhD, AubreyEllenShomo, Greensburger, Magioladitis, Parsecboy, Transcendence, Pixel ;-), Steven Walling, Rich257, Lenticel, Myshark, Gkklein, Mulangi, Mojodaddy, Gblandst, Tichondrius, STBotD, Philip Trueman, Altermike, Johnson487682, The Thing That Should Not Be, Wesleyneo, Anon lynx, Arjayay, XLinkBot, Galzigler, Dsimic, Addbot, Bhargavasaikumar, Bushfire Bill, SpBot, 1RadicalOne, Yobot, Megatron must be stopped, AnomieBOT, Zk00006, Flewis, Synaptophysin, Jacobspaulsen, Thomas Maierhofer, FrescoBot, Iconmode, Haeinous, Gottesmm, Citation bot 1, Pinethicket, I dream of horses, Achitchcock, Linus Smithson, RjwilmsiBot, EmausBot, Frabrunelle, J.kadlubowska, ZéroBot, Dkevanko, ClueBot NG, Lgaglioti10, Matthiaspaul, Lord Roem, Lemuellio, Denemark, Widr, Boots12, MerlIwBot, Curb Chain, Wbm1058, Walk&check, Atomician, Ugncreative Usergname, ChrisGualtieri, Cynmcc, Mogism, Alt Content, ProfessorHouseCat, Andyhowlett, Bossyandpicky, Postonm, Mei lin fung, AfadsBad, Paul2520, Jamalmunshi, Mario Castelán Castro, Leslie belden, Googleguacamole and Anonymous: 179
- **Firmware** *Source:* http://en.wikipedia.org/wiki/Firmware?oldid=652758618 *Contributors:* Damian Yerrick, Zundark, The Anome, Tarquin, Andre Engels, Enchanter, Hannes Hirzel, Rbrwr, Nixdorf, Mac, Smack, Dysprosia, Myshkin, Furrykef, Wernher, Robbot, Zz, Techtonik, Bkell, Wikibot, Engerim, Alan Liefting, David Gerard, DavidCary, Kenny sh, Antandrus, Beland, Panit, Abdull, Adashiel, Vsmith, Jojit fb, Lysdexia, Alansohn, Guy Harris, Spangineer, Brock, Wtshymanski, TahitiB, Dzordzm, SCEhardt, Isnow, Graham87, Raffaele Megabyte, NeonMerlin, Fred Bradstadt, Titoxd, FlaBot, SchuminWeb, Ayla, Fresheneesz, BMF81, King of Hearts, Chobot, Nastajus, WriterHound, YurikBot, Borgx, MMuzammils, Stephenb, Manop, Sikon, Cunado19, Jabencarsey, Code65536, Bota47, Nlu, Gregzeng, Nikkimaria, GrinBot, Perardi, NetRolller 3D, SmackBot, YellowMonkey, Redslime, F, Wlindley, Hydrogen Iodide, Gribeco, Ultramandk, Evanreyes, Brianski, Jcarroll, Dlohcierekim's sock, Frap, Chlewbot, Steveo1544, LouScheffer, Joema, Decltype, DylanW, NickPenguin, Autopilot, Weatherman1126, AThing, Mr Stephen, UKER, Dicklyon, Bashari, Amitch, The7thmagus, CapitalR, Sph147, Tawkerbot2, Ryt, Nuclearo, Sandeep pranavam, HenkeB, SolarisBigot, Phatom87, O mores, MikeLacey, Normix, Stevag, Thijs!bot, Kubanczyk, Fourchette, Dgies, Icep, Mentifisto, Widefox, Nosbig, JAnDbot, Arch dude, Aki009, ProjectPlatinum, VoABot II, Kuyabribri, Tedickey, SomethingWittyHere, Alex Spade, Gwern, Majesty9012, Herbythyme, Public Menace, Thaurisil, Laurusnobilis, JensRex, LordAnubisBOT, Cometstyles, STBotD, DorganBot, Ertyeryery, Aaronsingh, CardinalDan, Idioma-bot, Deor, Philip Trueman, Rocketmagnet, TyrantX, WolfgangEcker, DennyColt, LeaveSleaves, Mazarin07, Softtest123, Miko3k, Alaniaris, SieBot, WereSpielChequers, Winchelsea, Stonejag, Steven Zhang, OKBot, Dillard421, ClueBot, Meekywiki, ChandlerMapBot, Posix memalign, SchreiberBike, Zappa711, Joel Saks, Darkicebot, S1fw, Rror, Cmr08, Jaymacdonald, Kurniasan, Dsimic, Addbot, Grandscribe, Cptnoremac, Colcolstyles, Fluffernutter, Cst17, Aclews56, ChenzwBot, 5 albert square, Tide rolls, Rainbow will, Another-anomaly, Crt, Luckas-bot, Yobot, Ptbotgourou, Fraggle81,

Legobot II, Crispmuncher, MrBurns, AnomieBOT, Exp HP, Rubinbot, Jim1138, Name5555, Gacpro, Xqbot, Rijndael, The Evil IP address, GrouchoBot, RibotBOT, Universalss, Fobeteh, Erik9, SupportFTP, FrescoBot, Mohandesi, Winterst, SpaceFlight89, Jandalhandler, Siddharthsivakumar, TobeBot, SchreyP, RjwilmsiBot, Jtsandlund, Lopifalko, WikitanvirBot, Chewbaca75, Liquidmetalrob, Doomedtx, Contribute23, The contributor 4783, Chezi-Schlaff, MaGa, ChuispastonBot, VictorianMutant, Kenny Strawn, Rocketrod1960, ClueBot NG, Rajaram Sarangapani, Steve dexon, Wbm1058, BG19bot, Walk&check, Arashium, Mark Arsten, Compfreak7, Rancher 42, Winston Chuen-Shih Yang, BattyBot, Tkbx, Tagremover, Bachware, Ajv39, Marian Robinson, NoBearHere, Melonkelon, Gerardwm, Shaddycrook, Comp.arch, Monkbot, Guglastican, Tanankmaster118, MXocrossIIB, Garfield Garfield, Ggordonbyrne and Anonymous: 317

- **Unified Extensible Firmware Interface** *Source:* http://en.wikipedia.org/wiki/Unified%20Extensible%20Firmware%20Interface?oldid=652969864 *Contributors:* Deb, AdamWill, Leandrod, Edward, Nealmcb, Karada, KAMiKAZOW, Mac, Bueller 007, Julesd, Paulnasca, Nikai, Evercat, Rrostie, Jdstroy, Echoray, Jgm, AnonMoos, Chealer, Rfc1394, Mushroom, Pengo, Knobunc, Fudoreaper, Scott Wilson, Uzume, Aughtandzero, Mako098765, Sedulus, Vina, Ary29, Grm wnr, GreenReaper, Chmod007, Seffer, Imroy, Qutezuce, ArnoldReinhold, Nchaimov, Night Gyr, Bender235, CanisRufus, Jarfil, Rlaager, TheSolomon, Xojo, Foobaz, Dee Earley, Geoff.green, Giraffedata, Cncxbox, Krellis, Pearle, Gary, Guy Harris, CyberSkull, PatrickFisher, Watsonladd, Graingert, Kocio, Snorgy, SteinbDJ, Kbolino, Mindmatrix, Bwallum, Jonathan de Boyne Pollard, Pol098, Tmassey, Eyreland, Alecv, LinkTiger, Scmasaru, Marudubshinki, Kesla, G001, Qwertyus, Kbdank71, Phoenix-forgotten, ZeframCochrane, Rjwilmsi, Fahrenheit451, DRGrim, Sdornan, Bubba73, FlaBot, Mirror Vax, Wikidgood, Master Thief Garrett, Riluve, RobyWayne, Intgr, Ahunt, Psantora, King of Hearts, Shaggyjacobs, The Rambling Man, YurikBot, Todd Vierling, Hairy Dude, RussBot, Blodhevn, Wengier, Yuhong, Jrideout, Romanc19s, Msikma, Jabencarsey, Voidxor, MySchizoBuddy, Leotohill, Drakino, Xpclient, Nozzo, Mellowiz, Waynejkruse10, Palthrow, Smurfy, Memodude, ViperSnake151, Darrenmoffat, Jroddi, SmackBot, CBragg, Maelwys, Gribeco, KocjoBot, Davewild, Isaac Dupree, Thumperward, Keryst, Nbarth, Nbougalis, Frap, Racklever, NitishP, Shadow1, Warren, HarisM, ClarusWorks, Luís Felipe Braga, Smart Fox, A5b, Luigi.a.cruz, ThurnerRupert, SashatoBot, N Vale, Tanadeau, Errorx666, PeterEnnis, Hvn0413, Beetstra, NJA, Msandersen, McDrewn, Raysonho, Lhasapso, Andkore, Mblumber, Xcxin, Tkynerd, Thijs!bot, Daëmon, Hcobb, AntiVandalBot, Gioto, Widefox, QuiteUnusual, Tmopkisn, Wayiran, Ironiridis, LeedsKing, Thomas Linard, Andreas Toth, Benstown, Alphaman, Destynova, Lenschulwitz, EagleFan, Edurant, Seashorewiki, Avijitguharoy, AVRS, STBot, CitizenB, Nikpapag, Esper256, Francis Tyers, CACLF, Valvicus, Jesant13, Little Professor, Remember the dot, Cerberus0, VolkovBot, Aesopos, Jalwikip, OlavN, Felipebm, Brian Eisley, Mezzaluna, Bojan PLOJ, TheAlmightyEgg, MuzikJunky, BotMultichill, Josh the Nerd, Martin Kealey, GeiwTeol, Neophyrigian, Jerryobject, Oxymoron83, Hello71, Noxorc, Jfromcanada, Edifyyo, Treekids, ElectronicsEnthusiast, Martarius, ClueBot, Prohlep, Spambrian, GrandDrake, Shjacks45, Auntof6, Socrates2008, Pot, Arjayay, SDK SDK, Korynd, SF007, DumZiBoT, Missing30, Dsimic, Addbot, Inopia, Ghettoblaster, Jonbryce, AkhtaBot, OBloodyHell, Sergei, Scientus, Bngsudheer, Bernie Kohl, Lightbot, Fiftyquid, Crt, Lundberg85, Kuzetsa, Yobot, Fraggle81, KamikazeBot, Iconv, AnomieBOT, 1exec1, Götz, Pbrunnen, ArthurBot, LilHelpa, Gacpro, Xqbot, Meewam, Etoombs, PraeceptorIP, Ordishj, Mouagip, StefanoIT, FrescoBot, Dilic, UncleNinja, Furshur, Mctpyt, Mfwitten, Hazir, Arekrishna, Winterst, Therealdp, Stefan Weil, Srs5694, Jandalhandler, Klambour, Lotje, Dinamikbot, IlyaMart, Aoidh, Genhuan, RjwilmsiBot, Jlhcpa, Kreig303, EmausBot, John of Reading, RA0808, Zambi007, Cyberkagami, David Markun, Lokpest, TheChampionMan1234, Risthel, Mlorer, ClamDip, Mikhail Ryazanov, ClueBot NG, Freevanx, Jamesw2010, Matthiaspaul, Lostick, Snotbot, Rezabot, Widr, Titodutta, BG19bot, Compfreak7, Mattwick, Dwduback, ChrisGualtieri, 2040tech, Jdratlif5942, Codename Lisa, PeteMaz, Kephir, Calinou1, MartinMichlmayr, Ashwestonr, The Herald, Gogrp89, Someone not using his real name, ScotXW, Flechaig, Monkbot, Sofia Koutsouveli, Sawsmith, JW19335762743 and Anonymous: 353
- **Bus (computing)** *Source:* http://en.wikipedia.org/wiki/Bus%20(computing)?oldid=652616543 *Contributors:* Uriyan, The Anome, Drj, Css, Dachshund, Shd, Aldie, Fubar Obfusco, Deb, SimonP, Maury Markowitz, Heron, Olivier, Michael Hardy, Mahjongg, Nixdorf, Egil, ArnoLagrange, Mac, Nanshu, Glenn, Nikai, Rl, GRAHAMUK, CAkira, Reddi, Magnus.de, Colin Marquardt, Tschild, Saltine, Wernher, Thue, Veghead, Jni, Chuunen Baka, Robbot, MrJones, Fredrik, Scott McNay, RedWolf, Naddy, Merovingian, Hadal, Marc Venot, DavidCary, Kenny sh, Guanaco, Ezhiki, Tom-, Ferdinand Pienaar, AlistairMcMillan, VampWillow, Uzume, Sam Hocevar, Klemen Kocjancic, Mike Rosoft, Olki, Imroy, Slady, JTN, Discospinster, Guanabot, Bert490, Xezbeth, Mjpieters, Horsten, WegianWarrior, Johannes Rohr, Limbo socrates, Violetriga, Tooto, CanisRufus, R. S. Shaw, Brim, SpeedyGonsales, Tritium6, James Foster, MatthewWilcox, Civvi, Hopp, Riana, Fritzpoll, Mailer diablo, Helixblue, Wtshymanski, Proton, Nuno Tavares, Woohookitty, Timharwoodx, Rocastelo, Hbdragon88, JRHorse, Isnow, Cyberman, Wayward, Graham87, Arunib, Kbdank71, Pako, FlaBot, SchuminWeb, Moreati, Numa, RexNL, Nimur, Revolving Bugbear, Intgr, CiaPan, Chobot, DVdm, Bgwhite, YurikBot, Fabartus, CambridgeBayWeather, Rsrikanth05, Pseudomonas, Cpuwhiz11, Jeword, ENeville, Nowa, Pagrashtak, Dijxtra, Maverick Leonhart, Nick, Ospalh, Samir, Scope creep, Wknight94, Johndburger, Phgao, Sean Whitton, Kevin, Curpsbot-unicodify, Tom Morris, Attilios, SmackBot, KelleyCook, Commander Keane bot, Gilliam, Kurykh, Jerome Charles Potts, Craig t moore, Fjmustak, Can't sleep, clown will eat me, Frap, HarisM, Autodmc, A5b, Luigi.a.cruz, SashatoBot, SpareHeadOne, DHR, Mathias-S, Beetstra, Kvng, DabMachine, Iridescent, Ihatethetv, SkyWalker, Raysonho, Nhumfrey, Jesse Viviano, ShoobyD, ST47, Epbr123, Kubanczyk, Sobreira, I do not exist, Doyley, Yettie0711, Andrew sh, Mentifisto, AntiVandalBot, Wayiran, JAnDbot, NapoliRoma, BenB4, Acroterion, Bongwarrior, VoABot II, Wikidudeman, Becksguy, Twsx, Avicennasis, BrianGV, GermanX, Amazonite, AVRS, MartinBot, Dima373, Rpclod, J.delanoy, Cpiral, Pyams, McSly, Peskydan, Cometstyles, CompNerd11, Idioma-bot, Priceman86, VolkovBot, AndyLandy, Philip Trueman, Technopat, Hqb, Anna Lincoln, BotKung, Legoktm, Cowlinator, EmxBot, Regregex, SieBot, Ham Pastrami, Flyer22, Oda Mari, Egrian, Lightmouse, ClueBot, GorillaWarfare, Rilak, No such user, Jusdafax, Sun Creator, Lunchscale, BOTarate, Jonverve, Redhill54, XLinkBot, Dsimic, Deineka, Addbot, GargoyleBot, CanadianLinuxUser, Graham.Fountain, CarsracBot, BepBot, Lightbot, OlEnglish, Mike88chan, Legobot, Hydrofiber, Luckas-bot, Yobot, OrgasGirl, Crispmuncher, Tuxraider reloaded, Mmxx, Nallimbot, IW.HG, Keithbob, Toko50, Citation bot, TinucherianBot II, Capricorn42, Nasaverve, Wearingaredhat, RibotBOT, FrescoBot, קשיו, Krj373, W Nowicki, Amaka555, Yahia.barie, RedBot, MastiBot, Boriss111, Jeffrd10, Cp82, DARTH SIDIOUS 2, RjwilmsiBot, BlakeD360, Midhart90, Brightbulb, EmausBot, John of Reading, Stryn, ValC, Wikipelli, Thecheesykid, Prof Karl, Surya Prakash.S.A., Donner60, Atrivo, 28bot, ClueBot NG, Gilderien, Firowkp, Widr, Oddbodz, Helpful Pixie Bot, Wbm1058, Snaevar-bot, Kokkkikumar, Maxxdxx, ChrisGualtieri, EE JRW, Zeeyanwiki, Lugia2453, Frosty, Greenstruck, Lsmll, LarryWiki2009, Ugog Nizdast, Doenymo, Jianhui67, Reddraggone9, Dmtech, 42315413ferq, Derped Monkey Fish, MonkeysEatSwag, Some Gadget Geek, Dghgdsh and Anonymous: 291

11.10.2 Images

- **File:4Mbit_EPROM_Texas_Instruments_TMS27C040_(1).jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/3/34/4Mbit_EPROM_Texas_Instruments_TMS27C040_%281%29.jpg *License:* CC BY 2.0 *Contributors:* 4Mbit EPROM Texas Instruments TMS27C040 *Original artist:* yellowcloud from Germany
- **File:6600GT_GPU.jpg** *Source:* http://upload.wikimedia.org/wikipedia/commons/4/44/6600GT_GPU.jpg *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* Berkut

11.10.3 Content license