Crystal Summer Camp 2018

Crystal Education – TURX

Hardware 硬件
Fundamental 基本
Principles 原理

Contents

目录

- Common Hardware Components
- BIOS / EFI / UEFI / Interrupts / Tables
- Virtualization
- HAL / Drivers / Compatibilities
- Open Hardware with IoT (Raspberry Pi / Arduino)

普通硬件

基本输入输出系统

BIOS

UEFI

硬件中断 表格

虚拟化框架

硬件抽象层/驱动/兼容性

开源硬件

物联网

Ring 0 -> 内核

Ring 3 -> 用户

中断向量表

INT 0X10 调用硬盘

CPU, GPU, Memory

Common Hardware Components

ELF → UNIX

COFF

历史 → 服水多核

Introduction

- 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 computer industry has used the term "central processing unit" at least since the early 1960s.[1] Traditionally, the term "CPU" refers to a processor, more specifically to its processing unit and control unit (CU), distinguishing these core elements of a computer from external components such as main memory and I/O circuitry.[2]

SoC

- 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, one can speak of such single chips as "sockets".[3]

From *Wikipedia*

RISC PowerPC ARM

x86

Operation

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.[d] 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).

Decode

- The instruction that the CPU fetches from memory determines what the CPU will do. In the decode step, performed by the circuitry known as the instruction decoder, the instruction is converted into signals that control other parts of the CPU.

- The way in which the instruction is interpreted is defined by the CPU's instruction set architecture (ISA).[e] Often, one group of bits (that is, a "field") within the instruction, called the opcode, indicates which operation is to be performed, while the remaining fields usually provide supplemental information required for the operation, such as the operands. Those operands may be specified as a constant value (called an immediate value), or as the location of a value that may be a processor 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.

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.

From *Wikipedia*

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.[55] 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.

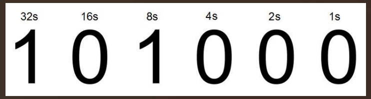
- Rather than totally removing the clock signal, some CPU designs allow certain portions of the device to be asynchronous, such as using asynchronous ALUs in conjunction with superscalar pipelining to achieve some arithmetic performance gains. While it is not altogether clear whether totally asynchronous 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.[56

From *Wikipedia*

DDR3L DDR SDRAM

DATA 8位 → DATA 16位 →

Integer Range



32s	16s	8s	4s	2s	1s
1	0	1	0	0	0

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 8, 16, 32 or 64 bits.

32位 16位 DATA →

TopBench

- 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.[f]
- 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.[g] For example, an 8-bit CPU can directly manipulate integers represented by eight bits, which have a range of 256 (28) 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 232 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. A CPU can have internal data paths shorter than the word size to reduce size and cost. For example, even though the IBM System/360 instruction set was a 32-bit instruction set, the System/360 Model 30 and Model 40 had 8-bit data paths in the arithmetic logical unit, so that a 32-bit add required four cycles, one for each 8 bits of the operands, and, even though the Motorola 68000 series instruction set was a 32-bit instruction set, the Motorola 68000 and Motorola 68010 had 16-bit data paths in the arithmetic logical unit, so that a 32-bit add required two cycles.
- To gain some of the advantages afforded by both lower and higher bit lengths, many instruction sets have different bit widths for integer and floating-point data, allowing CPUs implementing that instruction set to have different bit widths for different portions of the device. For example, the IBM System/360 instruction set was primarily 32 bit, but supported 64-bit floating point values to facilitate greater accuracy and range in floating point numbers.[27] The System/360 Model 65 had an 8-bit adder for decimal and fixed-point binary arithmetic and a 60-bit adder for floating-point arithmetic.[57] 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.

eMMC

16位 实模式 Real Mode

32位 保护模式 Protect Mode

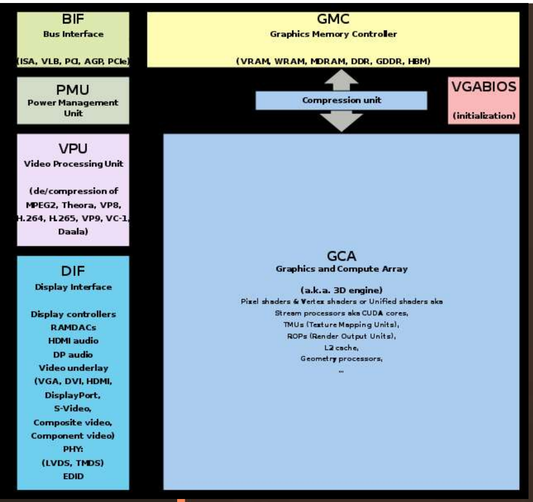
64位 长模式 Long Mode

ax, bx

eax, ebx

r...

4
3
2
1



BIF
Bus Interface
(ISA, VLB, PCI, AGP, PCIe)

GMC
Graphics Memory Controller
(VRAM, WRAM, MDRAM, DDR, GDDR, HBM)

PMU
Power Management Unit

Compression unit

VGABIOS
(initialization)

VPU
Video Processing Unit
(de/compression of MPEG2, Theora, VP8, H.264, H.265, VP9, VC-1, Daala)

GCA
Graphics and Compute Array
(a.k.a. 3D engine)
Pixel shaders & Vertex shaders or Unified shaders aka Stream processors aka CUDA cores,
TMUs (Texture Mapping Units),
ROPs (Render Output Units),
L2 cache,
Geometry processors,
...

DIF
Display Interface
Display controllers
RAMDACs
HDMI audio
DP audio
Video underlay
(VGA, DVI, HDMI, DisplayPort, S-Video, Composite video, Component video)
PHY:
(LVDS, TMDS)
EDID

Components of a GPU

Introduction

- A graphics processing unit (**GPU**) 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 device. 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 efficient than general-purpose CPUs for algorithms where the 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 embedded on the motherboard or—in certain CPUs—on the CPU die.[1]

From *Wikipedia*

Computational functions

- Modern GPUs use most of their transistors to do calculations related to 3D computer graphics. 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. GPUs 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; they are especially suited to other embarrassingly parallel problems.

- With the emergence of deep learning, the importance of GPUs has increased. In research done by Indigo, it was found that while training deep learning neural networks, GPUs can be 250 times faster than CPUs. That's a difference between one day of training and almost 8 months and 10 days of training. The explosive growth of Deep Learning in recent years has been attributed to the emergence of general purpose GPUs.[citation needed]However, beginning in early 2017, GPUs have begun to face some competition from Field Programmable Gate Arrays (FPGAs). FPGAs can also accelerate Machine Learning and Artificial Intelligence workloads. As AI applications mature, data for training models is not necessarily ordered in the neat, array-based, data that GPUs are best at handling, so FPGAs are also used for rapidly processing repetitive functions. Introduced at Hot Chips 2017, Microsoft's Project BrainWave is one example of real-time AI that uses FPGAs in an acceleration platform rather than GPUs.[58]

From *Wikipedia*

Computational functions

GPU accelerated video decoding

- 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.

From *Wikipedia*

Specific GPUs

Most GPUs are designed for a specific usage, real-time 3D graphics or other mass calculations:

- Gaming
  1. GeForce GTX
  2. nVidia Titan X
  3. Radeon HD
  4. Radeon r5, r7, r9 and RX series,
- Cloud Gaming
  1. nVidia Grid
  2. Radeon Sky
- Workstation
  1. nVidia Quadro
  2. nVidia Titan X
  3. AMD FirePro
  4. Radeon Pro
- Cloud Workstation
  1. nVidia Tesla
  2. AMD FireStream
- Artificial Intelligence Cloud
  1. nVidia Tesla
  2. Radeon Instinct
- Automated/Driverless car
  1. nVidia Drive PX

From *Wikipedia*

ROM

Introduction

内存 外 HDD
CD/DVD
CMOS
IS

- In computing, memory refers to the computer hardware integrated circuits that store information for immediate use in a computer; it is synonymous with the term "primary storage". Computer memory operates at a high speed, for example random-access memory (RAM), as a distinction from storage that provides slow-to-access information but offers higher capacities. If needed, contents of the computer memory can be transferred to secondary storage, through a memory management technique called "virtual memory". An archaic synonym for memory is store.[1]

- The term "memory", meaning "primary storage" or "main memory", is often associated with addressable semiconductor memory, i.e. integrated circuits consisting of silicon-based transistors, used for example as primary storage but also other purposes in computers and other digital electronic devices. There are two main kinds of semiconductor memory, volatile and non-volatile. Examples of non-volatile memory are flash memory (used as secondary memory) and ROM, PROM, EPROM and EEPROM memory (used for storing firmware such as BIOS). Examples of volatile memory are primary storage, which is typically dynamic random-access memory (DRAM), and fast CPU cache memory, which is typically static random-access memory (SRAM) that is fast but energy-consuming, offering lower memory areal density than DRAM.

- Most semiconductor memory is organized into memory cells or bistable flip-flops, each storing one bit (0 or 1). Flash memory organization includes both one bit per memory cell and multiple bits per cell (called MLC, Multiple Level Cell). The memory cells are grouped into words of fixed word length, for example 1, 2, 4, 8, 16, 32, 64 or 128 bit. Each word can be accessed by a binary address of N bit, making it possible to store 2 raised by N words in the memory. This implies that processor registers normally are not considered as memory, since they only store one word and do not include an addressing mechanism.

- Typical secondary storage devices are hard disk drives and solid-state drives.

From *Wikipedia*

Volatile memory

易

- Volatile memory is computer memory that requires power to maintain the stored information. Most modern semiconductor volatile memory is either static RAM (SRAM) or dynamic RAM (DRAM). SRAM retains its contents as long as the power is connected and is easy for interfacing, but uses six transistors per bit. Dynamic RAM is more complicated for interfacing and control, needing regular refresh cycles to prevent losing its contents, but uses only one transistor and one capacitor per bit, allowing it to reach much higher densities and much cheaper per-bit costs.
- SRAM is not worthwhile for desktop system memory, where DRAM dominates, but is used for their cache memories. SRAM is commonplace in small embedded systems, which might only need tens of kilobytes or less. Forthcoming volatile memory technologies that aim at replacing or competing with SRAM and DRAM include Z-RAM and A-RAM.

From *Wikipedia*

Non-volatile memory

非易

- Non-volatile memory is computer memory that can retain the stored information even when not powered. Examples of non-volatile memory include read-only memory (see ROM), flash memory, most types of magnetic computer storage devices (e.g. hard disk drives, floppy disks and magnetic tape), optical discs, and early computer storage methods such as paper tape and punched cards.
- Forthcoming non-volatile memory technologies include FeRAM, CBRAM, PRAM, STT-RAM, SONOS, RRAM, racetrack memory, NRAM, 3D XPoint, and millipede memory.

From *Wikipedia*

Virtual memory

- Virtual memory is a system where all physical memory is controlled by the operating system. When a program needs memory, it requests it from the operating system. The operating system then decides what physical location to place the memory in.
- This offers several advantages. Computer programmers no longer need to worry about where the memory is physically stored or whether the user's computer will have enough memory. It also allows multiple types of memory to be used. For example, some memory can be stored in physical RAM chips while other memory is stored on a hard drive (e.g. in a swapfile), functioning as an extension of the cache hierarchy. This drastically increases the amount of memory available to programs. The operating system will place actively used memory in physical RAM, which is much faster than hard disks. When the amount of RAM is not sufficient to run all the current programs, it can result in a situation where the computer spends more time moving memory from RAM to disk and back than it does accomplishing tasks; this is known as thrashing.
- Virtual memory systems usually include protected memory, but this is not always the case.

From *Wikipedia*

Protected memory

OSASK

- Protected memory is a system where each program is given an area of memory to use and is not permitted to go outside that range. Use of protected memory greatly enhances both the reliability and security of a computer system.
- Without protected memory, it is possible that a bug in one program will alter the memory used by another program. This will cause that other program to run off of corrupted memory with unpredictable results. If the operating system's memory is corrupted, the entire computer system may crash and need to be rebooted. At times programs intentionally alter the memory used by other programs. This is done by viruses and malware to take over computers.
- Protected memory assigns programs their own areas of memory. If the operating system detects that a program has tried to alter memory that does not belong to it, the program is terminated. This way, only the offending program crashes, and other programs are not affected by the error.
- Protected memory systems almost always include virtual memory as well.

From *Wikipedia*

Introduction

LCD

CRT

- A computer monitor is an output device which displays information in pictorial form. A monitor usually comprises the display device, circuitry, casing, and power supply. The display device in modern monitors is typically a thin film transistor liquid crystal display (TFT-LCD) with LED backlighting having replaced cold-cathode fluorescent lamp (CCFL) backlighting. Older monitors used a cathode ray tube (CRT). Monitors are connected to the computer via VGA, Digital Visual Interface (DVI), HDMI, DisplayPort, Thunderbolt, low-voltage differential signaling (LVDS) or other proprietary connectors and signals.

USB3.0

- Originally, computer monitors were used for data processing while television receivers were used for entertainment. From the 1980s onwards, computers (and their monitors) have been used for both data processing and entertainment, while televisions have implemented some computer functionality. The common aspect ratio of televisions, and computer monitors, has changed from 4:3 to 16:10, to 16:9.

600X800 21:9

- Modern computer monitors are easily interchangeable with conventional television sets. However, as computer monitors do not necessarily include components such as a television tuner and speakers, it may not be possible to use a computer monitor as a television without external components.[1]

From *Wikipedia*

Liquid crystal display

液晶

- There are multiple technologies that have been used to implement liquid crystal displays (LCD). Throughout the 1990s, the primary use of LCD technology as computer monitors was in laptops where the lower power consumption, lighter weight, and smaller physical size of LCD's justified the higher price versus a CRT. Commonly, the same laptop would be offered with an assortment of display options at increasing price points: (active or passive) monochrome, passive color, or active matrix color (TFT). As volume and manufacturing capability have improved, the monochrome and passive color technologies were dropped from most product lines.
- TFT-LCD is a variant of LCD which is now the dominant technology used for computer monitors.[4]
- The first standalone LCDs appeared in the mid-1990s selling for high prices. As prices declined over a period of years they became more popular, and by 1997 were competing with CRT monitors. Among the first desktop LCD computer monitors was the Eizo L66 in the mid-1990s, the Apple Studio Display in 1998, and the Apple Cinema Display in 1999. In 2003, TFT-LCDs outsold CRTs for the first time, becoming the primary technology used for computer monitors.[3] The main advantages of LCDs over CRT displays are that LCD's consume less power, take up much less space, and are considerably lighter. The now common active matrix TFT-LCD technology also has less flickering than CRTs, which reduces eye strain.[5] On the other hand, CRT monitors have superior contrast, have a superior response time, are able to use multiple screen resolutions natively, and there is no discernible flicker if the refresh rate is set to a sufficiently high value. LCD monitors have now very high temporal accuracy and can be used for vision research.[6]
- High dynamic range (HDR) has been implemented into high-end LCD monitors to improve color accuracy. Since around the late 2000s, widescreen LCD monitors have become popular, in part due to television series, motion pictures and video games transitioning to high-definition (HD), which makes standard-width monitors unable to display them correctly as they either stretch or crop HD content. These types of monitors may also display it in the proper width, however they usually fill the extra space at the top and bottom of the image with black bars. Other advantages of widescreen monitors over standard-width monitors is that they make work more productive by displaying more of a user's documents and images, and allow displaying toolbars with documents. They also have a larger viewing area, with a typical widescreen monitor having a 16:9 aspect ratio, compared to the 4:3 aspect ratio of a typical standard-width monitor.

From *Wikipedia*

Aspect ratio

比例

- Until about 2003, most computer monitors had a 4:3 aspect ratio and some had 5:4. Between 2003 and 2006, monitors with 16:9 and mostly 16:10 (8:5) aspect ratios became commonly available, first in laptops and later also in standalone monitors. Reasons for this transition was productive uses for such monitors, i.e. besides widescreen computer game play and movie viewing, are the word processor display of two standard letter pages side by side, as well as CAD displays of large-size drawings and CAD application menus at the same time.[7][8] In 2008 16:10 became the most common sold aspect ratio for LCD monitors and the same year 16:10 was the mainstream standard for laptops and notebook computers.[9]
- In 2010 the computer industry started to move over from 16:10 to 16:9 because 16:9 was chosen to be the standard high-definition television display size, and because they were cheaper to manufacture.
- In 2011 non-widescreen displays with 4:3 aspect ratios were only being manufactured in small quantities. According to Samsung this was because the "Demand for the old 'Square monitors' has decreased rapidly over the last couple of years," and "I predict that by the end of 2011, production on all 4:3 or similar panels will be halted due to a lack of demand."[10]

From *Wikipedia*

Resolution

- The resolution for computer monitors has increased over time. From 320x200 during the early 1980s, to 1024x768 during the late 1990s. Since 2009, the most commonly sold resolution for computer monitors is 1920x1080.[11] Before 2013 top-end consumer LCD monitors were limited to 2560x1600 at 30 in (76 cm), excluding Apple products and CRT monitors.[12] Apple introduced 2880x1800 with Retina MacBook Pro at 15.4 in (39 cm) on June 12, 2012, and introduced a 5120x2880 Retina iMac at 27 in (69 cm) on October 16, 2014. By 2015 most major display manufacturers had released 3840x2160 resolution displays.

From *Wikipedia*

DPI

- Dots per inch (DPI, or dpi)[1] is a measure of spatial printing or video or image scanner dot density, in particular the number of individual dots that can be placed in a line within the span of 1 inch (2.54 cm).

- Monitors do not have dots, but do have pixels; the closely related concept for monitors and images is pixels per inch or PPI. Many resources, including the Android developer guide, use the terms DPI and PPI interchangeably.

- In printing, DPI (dots per inch) refers to the output resolution of a printer or imagesetter, and PPI (pixels per inch) refers to the input resolution of a photograph or image. DPI refers to the physical dot density of an image when it is reproduced as a real physical entity, for example printed onto paper. A digitally stored image has no inherent physical dimensions, measured in inches or centimeters. Some digital file formats record a DPI value, or more commonly a PPI (pixels per inch) value, which is to be used when printing the image. This number lets the printer or software know the intended size of the image, or in the case of scanned images, the size of the original scanned object. For example, a bitmap image may measure 1,000 × 1,000 pixels, a resolution of 1 megapixel. If it is labeled as 250 PPI, that is an instruction to the printer to print it at a size of 4 × 4 inches. Changing the PPI to 100 in an image editing program would tell the printer to print it at a size of 10 × 10 inches. However, changing the PPI value would not change the size of the image in pixels which would still be 1,000 × 1,000. An image may also be resampled to change the number of pixels and therefore the size or resolution of the image, but this is quite different from simply setting a new PPI for the file.

- For vector images, there is no equivalent of resampling an image when it is resized, and there is no PPI in the file because it is resolution independent (prints equally well at all sizes). However, there is still a target printing size. Some image formats, such as Photoshop format, can contain both bitmap and vector data in the same file. Adjusting the PPI in a Photoshop file will change the intended printing size of the bitmap portion of the data and also change the intended printing size of the vector data to match. This way the vector and bitmap data maintain a consistent size relationship when the target printing size is changed. Text stored as outline fonts in bitmap image formats is handled in the same way. Other formats, such as PDF, are primarily vector formats which can contain images, potentially at a mixture of resolutions. In these formats the target PPI of the bitmaps is adjusted to match when the target print size of the file is changed. This is the converse of how it works in a primarily bitmap format like Photoshop, but has exactly the same result of maintaining the relationship between the vector and bitmap portions of the data.

From *Wikipedia*

VESA

- VESA (/ˈviːsə/), formally known as Video Electronics Standards Association, is a technical standards organization for computer display standards. The organization was incorporated in California in July 1989[1] and has its office in San Jose, California.[1][2] It claims a membership of over 225 companies.[3]
- In November 1988, NEC Home Electronics announced its creation of the association to develop and promote a Super VGA computer display standard as a successor to IBM's proprietary Video Graphics Array (VGA) display standard. Super VGA would enable graphics display resolutions up to 800 × 600 pixels, compared to VGA's maximum resolution of 640 × 480 pixels—a 56% increase.[4]
- The organization has since issued several additional standards related to computer video display.

From *Wikipedia*

More

HDD rpm cache
SDD TLC MLC SLC

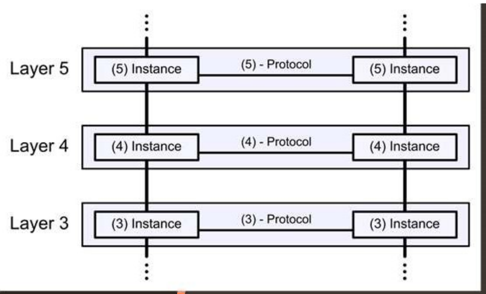
- Hard disk
- Motherboard

From *Wikipedia*

BIOS, UEFI, Interrupt

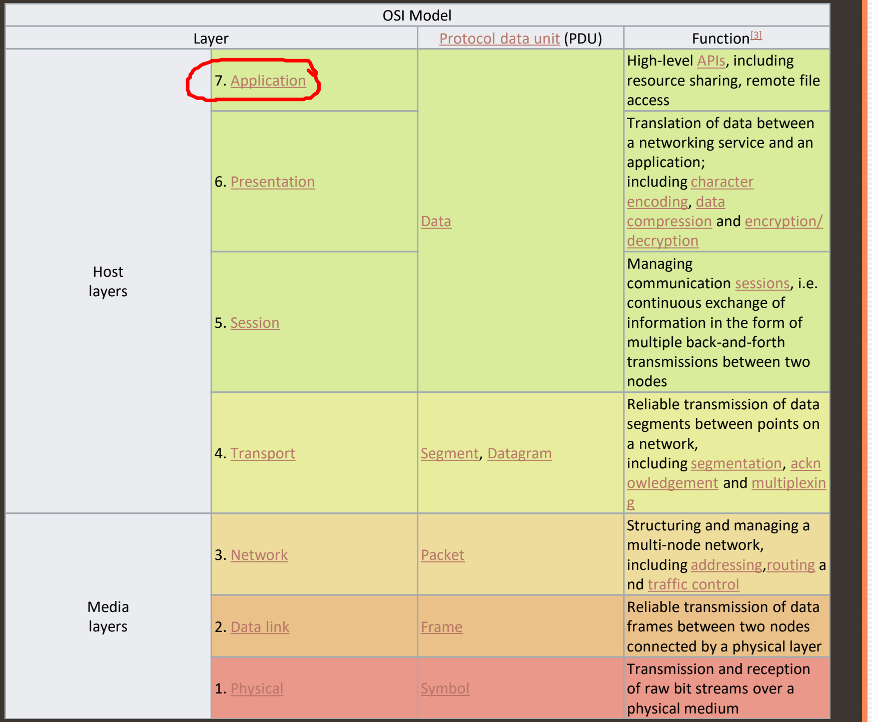
Fundamental Basement of Software

OSI model



Communication in the OSI-Model (example with layers 3 to 5)

- The Open Systems Interconnection model (OSI model) is a conceptual model that characterizes and standardizes the communication functions of a telecommunication or computing system without regard to its underlying internal structure and technology. Its goal is the interoperability of diverse communication systems with standard protocols. The model partitions a communication system into abstraction layers. The original version of the model defined seven layers.
- A layer serves the layer above it and is served by the layer below it. For example, a layer that provides error-free communications across a network provides the path needed by applications above it, while it calls the next lower layer to send and receive packets that comprise the contents of that path. Two instances at the same layer are visualized as connected by a horizontal connection in that layer.



OSI Model			
Layer		Protocol data unit (PDU)	Function[3]
Host layers	7. Application	Data	High-level APIs, including resource sharing, remote file access
	6. Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
	5. Session		Managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
	4. Transport	Segment, Datagram	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
Media layers	3. Network	Packet	Structuring and managing a multi-node network, including addressing,routing and traffic control
	2. Data link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer
	1. Physical	Symbol	Transmission and reception of raw bit streams over a physical medium

POST (Power-on self-test)

BIOS

- BIOS (/ˈbaɪɒs/ BY-oss; an acronym for Basic Input/Output System and also known as the System BIOS, ROM BIOS or PC BIOS) is non-volatile firmware used to perform hardware initialization during the booting process (power-on startup), and to provide runtime services for operating systems and programs.[1] The BIOS firmware comes pre-installed on a personal computer's system board, and it is the first software to run when powered on. The name 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 has been reverse engineered by companies looking to create compatible systems. The interface of that original system serves as a de facto standard.
- The BIOS in modern PCs initializes and tests the system hardware components, and loads a boot loader from a mass memory device which then initializes an operating system. In the era of MS-DOS, the BIOS provided a hardware abstraction layer for the keyboard, display, and other input/output (I/O) devices that standardized an interface to application programs and the operating system. More recent operating systems do not use the BIOS after loading, instead accessing the hardware components directly.
- Most BIOS implementations are 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, end-user 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. Furthermore, a BIOS upgrade that fails can brick the motherboard permanently, unless the system includes some form of backup for this case.
- Unified Extensible Firmware Interface (UEFI) is a successor to BIOS, aiming to address its technical shortcomings.[4]

From *Wikipedia*

0x7c00 → 512 B

i386
↑ 80

BIOS Operation

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).[10] 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 maintenance program into RAM 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. (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.[11][12] 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 (as it was used in the manufacturing environment[citation needed]).

From *Wikipedia*

BIOS Operation

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.

- 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.[13] 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 sector (which is 512 bytes long), 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.

From *Wikipedia*

BIOS Operation

HEX
OCT
BIN

- 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).

HLT

- 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.)

From *Wikipedia*

BIOS Operation

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.

From *Wikipedia*

Extensible Firmware Interface (EFI) and Unified EFI (UEFI)

Defining the Interface Between the Operating System and Platform Firmware

- The Unified EFI (UEFI) Specification (previously known as the EFI Specification) defines an interface between an operating system and platform firmware.
- The interface consists of data tables that contain platform-related information, boot service calls, and runtime service calls that are available to the operating system and its loader. These provide a standard environment for booting an operating system and running pre-boot applications.
- The UEFI Specification was primarily intended for the next generation of IA architecture–based computers, and is an outgrowth of the "Intel® Boot Initiative" (IBI) program that began in 1998.
- Intel's original version of this specification was publicly named EFI, ending with the EFI 1.10 version.
- In 2005, The Unified EFI Forum was formed as an industry-wide organization to promote adoption and continue the development of the EFI Specification. Using the EFI 1.10 Specification as the starting point, this industry group released the following specifications, renamed Unified EFI.
- The current version of the UEFI Specification can be found at the UEFI web site.

From *Intel Website*

UEFI

- The Unified Extensible Firmware Interface (UEFI) is a specification that defines a software interface between an operating system and platform ~~firmware. UEFI replaces the Basic Input/Output System (BIOS) firmware interface originally present in all IBM PC-~~compatible personal computers,[1][2] with most UEFI firmware implementations providing legacy support for BIOS services. UEFI can support remote diagnostics and repair of computers, even with no operating system installed.[3]
- Intel developed the original Extensible Firmware Interface (EFI) specification. Some of the EFI's practices and data formats mirror those from Microsoft Windows.[4][5] In 2005, UEFI deprecated EFI 1.10 (the final release of EFI). The Unified EFI Forum is the industry body that manages the UEFI specification.

From *Wikipedia*

UEFI

- Advantages
  - The interface defined by the EFI specification includes data tables that contain platform information, and boot 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 use large disks (over 2 TB) with a GUID Partition Table (GPT)[13][a]
  - CPU-independent architecture[a]
  - ~~CPU-independent drivers~~[a]
  - Flexible ~~pre-OS environment, including network capability~~
  - ~~Modular design~~
  - Backward and forward compatibility

From *Wikipedia*

UEFI

- Processor compatibility
  - As of version 2.5, processor bindings exist for Itanium, x86, x86-64, ARM (AArch32) and ARM64 (AArch64).[14] Only little-endian processors can be supported.[15] Unofficial UEFI support is under development for POWERPC64 by implementing TianoCore on top of OPAL,[16] the OpenPOWER abstraction layer, running in little-endian mode.[17] Similar projects exist for MIPS[18] and RISC-V.[19] As of UEFI 2.7, RISC-V processor bindings have been officially established for 32, 64 and 128-bit modes.[20]
  - Standard PC BIOS is limited to a 16-bit processor mode and 1 MB of addressable memory space, resulting from the design based on the IBM 5150 that used a 16-bit Intel 8088 processor.[6][21] 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).[6][22] 64-bit UEFI firmware implementations support long mode, which allows applications in the preboot execution environment to use 64-bit addressing to get direct access to all of the machine's memory.[23]
  - UEFI requires the firmware and operating system loader (or kernel) to be size-matched; for example, a 64-bit UEFI firmware implementation can load only a 64-bit operating system (OS) 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).[24]:sections 2.3.2 and 2.3.4 As of version 3.15, the Linux kernel supports 64-bit kernels to be booted on 32-bit UEFI firmware implementations running on x86-64 CPUs, with UEFI handover support from a UEFI boot loader as the requirement.[25] 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.[26][27]

From *Wikipedia*

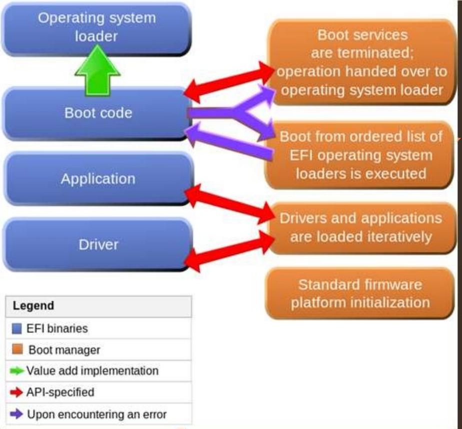
UEFI

- Disk device compatibility
    - In addition to the standard PC disk partition scheme that uses a master boot record (MBR), UEFI also works with a new partitioning scheme called GUID Partition Table (GPT), which is free from many of the limitations of MBR. In particular, the MBR limits on the number and size of disk partitions (up to four primary partitions per disk, and up to 2 TiB (2 × 240 bytes) per disk) are relaxed.[28] More specifically, GPT allows for a maximum disk and partition size of 8 ZiB (8 × 270 bytes).[28][29]
  - Linux
    - Support for GPT in Linux is enabled by turning on the option CONFIG_EFI_PARTITION (EFI GUID Partition Support) during kernel configuration.[30] 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.[31] As GPT incorporates the protective MBR, a BIOS-based computer can boot from a GPT disk using a GPT-aware boot loader stored in the protective MBR's bootstrap code area.[29] In the 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 (GUID) in GPT scheme is 21686148-6449-6E6F-744E-656564454649 and is used by GRUB only in BIOS-GPT setups. From GRUB's perspective, no such partition type exists in case of MBR partitioning. This partition is not required if the system is UEFI-based because no embedding of the second-stage code is needed in that case.[13][29][31]
    - UEFI systems can access GPT disks and boot directly from them, which allows Linux to use UEFI boot methods. 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.[32][33][34] 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.[29][31][35]
    - For backward compatibility, most UEFI implementations also support booting from MBR-partitioned disks, through the Compatibility Support Module (CSM) that provides legacy BIOS compatibility.[36] In that case, booting Linux on UEFI systems is the same as on legacy BIOS-based systems.
  - Microsoft Windows
    - The 64-bit versions of Windows 7 and later can boot from disks with a partition size larger than 2 TB.

From *Wikipedia*

UEFI


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

From *Wikipedia*

- Applications
  - Beyond loading an OS, UEFI can run UEFI applications, which reside as files on the EFI System Partition. They can be executed from the UEFI command shell, by the firmware's boot manager, or by other UEFI applications. UEFI applications can be developed and installed independently of the system manufacturer.
  - A type of UEFI application is an OS loader such as GRUB, rEFInd, Gummiboot, and Windows Boot Manager; which loads an OS file into memory and executes it. Also, an OS loader can provide a user interface to allow the selection of another UEFI application to run. Utilities like the UEFI shell are also UEFI applications.

UEFI- GPT

UEFI

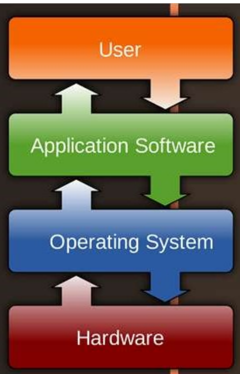
- Booting
  - UEFI booting
    - Unlike BIOS, UEFI does not rely on a boot sector, defining instead a boot manager as part of the UEFI specification. When a computer is powered on, the boot manager checks the boot configuration and based on its settings, loads into memory and then executes the specified OS loader or operating system kernel. The boot configuration is defined by variables stored in NVRAM, including variables that indicate the file system paths to OS loaders and OS kernels.
    - OS loaders can be automatically detected by UEFI, which enables easy booting from removable devices such as USB flash drives. This automated detection relies on standardized file paths to the OS loader, with the path varying depending on the computer architecture. The format of the file path is defined as <EFI_SYSTEM_PARTITION>/EFI/BOOT/BOOT<MACHINE_TYPE_SHORT_NAME>.EFI; for example, the file path to the OS loader on an x86-64 system is /efi/BOOT/BOOTX64.EFI,[24] and efi\boot\bootaa64.efi on ARM64 architecture.
    - Booting UEFI systems from GPT-partitioned disks is commonly called UEFI-GPT booting. Despite the fact that the UEFI specification requires MBR partition tables to be fully supported,[24] some UEFI firmware implementations immediately switch to the BIOS-based CSM booting depending on the type of boot disk's partition table, effectively preventing UEFI booting to be performed from EFI System partitions on MBR-partitioned disks.[36] Such a boot scheme is commonly called UEFI-MBR.
    - It is also common for a boot manager to have a textual user interface so the user can select the desired OS (or system utility) from a list of available boot options.
  - CSM booting
    - To ensure backward compatibility, most UEFI firmware implementations on PC-class machines also support booting in legacy BIOS mode from MBR-partitioned disks, through the Compatibility Support Module (CSM) that provides legacy BIOS compatibility. In this 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.[36]
    - BIOS-style booting from MBR-partitioned disks is commonly called BIOS-MBR, regardless of it being performed on UEFI or legacy BIOS-based systems. Furthermore, booting legacy BIOS-based systems from GPT disks is also possible, and such a boot scheme is commonly called BIOS-GPT.
    - The Compatibility Support Module allows legacy operating systems and some option ROMs that do not support UEFI to still be used.[42] It 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.[42]
    - In November 2017, Intel announced that it planned to phase out support for CSM by 2020.[43]
- Secure boot
  - The UEFI 2.3.1 Errata C specification (or higher) defines 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. The mechanical details of how precisely these drivers are to be signed are not specified.[48] 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.[49] 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.[50]
  - Secure boot is supported by Windows 8 and 8.1, Windows Server 2012, and 2012 R2, and Windows 10, VMware vSphere 6.5[51] and a number of Linux distributions including Fedora (since version 18), openSUSE (since version 12.3), RHEL (since RHEL 7), CentOS (since CentOS 7[52]) and Ubuntu (since version 12.04.2).[53] As of January 2017, FreeBSD support is in a planning stage.[54]

From *Wikipedia*

UEFI

- UEFI shell
  - UEFI provides a shell environment, which can be used to execute other UEFI applications, including UEFI boot loaders.[34] 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).[55][56][57]
  - Source code for a UEFI shell can be downloaded from the Intel's TianoCore UDK2010 / EDK2 SourceForge project.[58] 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.[55][59][60]
  - 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.[61][62] 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.[57][61][63][64]
- ~~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.

From *Wikipedia*



Interrupt

- In system programming, an interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. An interrupt alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing. The processor responds by suspending its current activities, saving its state, and executing a function called an interrupt handler (or an interrupt service routine, ISR) to deal with the event. This interruption is temporary, and, after the interrupt handler finishes, the processor resumes normal activities.[1] There are two types of interrupts: hardware interrupts and software interrupts.

- Hardware interrupts are used by devices to communicate that they require attention from the operating system.[2] Internally, hardware interrupts are implemented using electronic alerting signals that are sent to the processor from an external device, which is either a part of the computer itself, such as a disk controller, or an external peripheral. For example, pressing a key on the keyboard or moving the mouse triggers hardware interrupts that cause the processor to read the keystroke or mouse position. Unlike the software type (described below), hardware interrupts are asynchronous and can occur in the middle of instruction execution, requiring additional care in programming. The act of initiating a hardware interrupt is referred to as an interrupt request (IRQ).

- A software interrupt is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which causes an interrupt when it is executed. The former is often called a trap or exception and is used for errors or events occurring during program execution that are exceptional enough that they cannot be handled within the program itself. For example, a divide-by-zero exception will be thrown if the processor's arithmetic logic unit is commanded to divide a number by zero as this instruction is an error and impossible. The operating system will catch this exception, and can choose to abort the instruction. Software interrupt instructions can function similarly to subroutine calls and are used for a variety of purposes, such as to request services from device drivers, like interrupts sent to and from a disk controller to request reading or writing of data to and from the disk.

- Each interrupt has its own interrupt handler. The number of hardware interrupts is limited by the number of interrupt request (IRQ) lines to the processor, but there may be hundreds of different software interrupts. Interrupts are a commonly used technique for computer multitasking, especially in real-time computing. Such a system is said to be interrupt-driven.[3]

- Interrupts are similar to signals, the difference being that signals are used for IPC, mediated by the kernel (possibly via system calls) and handled by processes, while interrupts are mediated by the processor and handled by the kernel. The kernel may pass an interrupt as a signal to the process that caused it (typical examples are SIGSEGV, SIGBUS, SIGILL and SIGFPE).

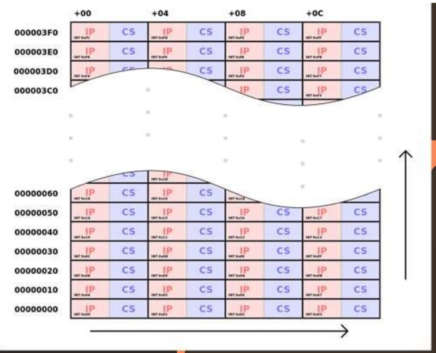
From *Wikipedia*

Interrupt Overview

- Hardware interrupts were introduced as an optimization, eliminating unproductive waiting time in polling loops, waiting for external events. They may be implemented in hardware as a distinct system with control lines, or they may be integrated into the memory subsystem.
- If implemented in hardware, an interrupt controller circuit such as the IBM PC's Programmable Interrupt Controller (PIC) may be connected between the interrupting device and the processor's interrupt pin to multiplex several sources of interrupt onto the one or two CPU lines typically available. If implemented as part of the memory controller, interrupts are mapped into the system's memory address space.
- Interrupts can be categorized into these different types:
  - Maskable interrupt (IRQ): a hardware interrupt that may be ignored by setting a bit in an interrupt mask register's (IMR) bit-mask.
  - Non-maskable interrupt (NMI): a hardware interrupt that lacks an associated bit-mask, so that it can never be ignored. NMIs are used for the highest priority tasks such as timers, especially watchdog timers.
  - Inter-processor interrupt (IPI): a special case of interrupt that is generated by one processor to interrupt another processor in a multiprocessor system.
  - Software interrupt: an interrupt generated within a processor by executing an instruction. Software interrupts are often used to implement system calls because they result in a subroutine call with a CPU ring level change.
  - Spurious interrupt: a hardware interrupt that is unwanted. They are typically generated by system conditions such as electrical interference on an interrupt line or through incorrectly designed hardware.

- Processors typically have an internal interrupt mask which allows software to ignore all external hardware interrupts while it is set. Setting or clearing this mask may be faster than accessing an interrupt mask register (IMR) in a PIC or disabling interrupts in the device itself. In some cases, such as the x86 architecture, disabling and enabling interrupts on the processor itself act as a memory barrier; however, it may actually be slower.
- An interrupt that leaves the machine in a well-defined state is called a precise interrupt. Such an interrupt has four properties:
  - The Program Counter (PC) is saved in a known place.
  - All instructions before the one pointed to by the PC have fully executed.
  - No instruction beyond the one pointed to by the PC has been executed (that is no prohibition on instruction beyond that in PC, it is just that any changes they make to registers or memory must be undone before the interrupt happens).
  - The execution state of the instruction pointed to by the PC is known.
- An interrupt that does not meet these requirements is called an imprecise interrupt.
- The phenomenon where the overall system performance is severely hindered by excessive amounts of processing time spent handling interrupts is called an interrupt storm.

From *Wikipedia*



Interrupt vector table

- For its implementation found in x86 processors, see Interrupt descriptor table.
- An "interrupt vector table" (IVT) is a data structure that associates a list of interrupt handlers with a list of interrupt requests in a table of interrupt vectors. Each entry of the interrupt vector table, called an interrupt vector, is the address of an interrupt handler. While the concept is common across processor architectures, IVTs may be implemented in architecture-specific fashions. For example, a dispatch table is one method of implementing an interrupt vector table.

From *Wikipedia*

Interrupt descriptor table

- The Interrupt Descriptor Table (IDT) is a data structure used by the x86 architecture to implement an interrupt vector table. The IDT is used by the processor to determine the correct response to interrupts and exceptions.
- The details in the description below apply specifically to the x86 architecture and the AMD64 architecture. Other architectures have similar data structures, but may behave differently.
- Use of the IDT is triggered by three types of events: hardware interrupts, software interrupts, and processor exceptions, which together are referred to as "interrupts". The IDT consists of 256 interrupt vectors–the first 32 (0-31 or 00-1F) of which are reserved for processor exceptions.

From *Wikipedia*

Interrupt descriptor table

Ptr = 段 + 偏

- Real mode
  - In the 8086 processor, the interrupt table is called IVT (interrupt vector table). The IVT always resides at the same location in memory, ranging from 0x0000 to 0x03ff, and consists of 256 four-byte real mode far pointers (256 × 4 = 1024 bytes of memory).
  - A real mode pointer is defined as a 16-bit segment and a 16-bit offset into that segment. The segment is expanded internally by the processor to 20 bits by shifting it 4 bits to the left, thus limiting real mode interrupt handlers to the first 1 megabyte of memory. The first 32 vectors are reserved for the processor's internal exceptions, and hardware interrupts may be mapped to any of the vectors by way of a programmable interrupt controller.
  - On the 80286 and later, the size and locations of the IVT can be changed in the same way as it is done with the IDT in protected mode, i.e. via the LIDT instruction, though it does not change the format of it. The 80286 also introduced the high memory area, which raises the address limit in real mode by 65520 bytes.
  - A commonly used x86 real mode interrupt is INT 10, the Video BIOS code to handle primitive screen drawing functions such as pixel drawing and changing the screen resolution.

From *Wikipedia*

Interrupt descriptor table

- Protected mode
  - In protected mode, the IDT is an array of 8-byte descriptors stored consecutively in memory and indexed by an interrupt vector. These descriptors may be either interrupt gates, trap gates or task gates. Interrupt and trap gates point to a memory location containing code to execute by specifying both a segment (present in either the GDT or LDT) and an offset within that segment. The only difference between these two is that an interrupt gate will disable further processor handling of hardware interrupts, making it especially suitable to service hardware interrupts, while a trap gate will leave hardware interrupts enabled and is thus mainly used for handling software interrupts and exceptions. Finally, a task gate will cause the currently active task-state segment to be switched, using the hardware task switch mechanism to effectively hand over use of the processor to another program, thread or process.
  - The protected mode IDT may reside anywhere in physical memory. The processor has a special register (IDTR) to store both the physical base address and the length in bytes of the IDT. When an interrupt occurs, the processor multiplies the interrupt vector by 8 and adds the result to the IDT base address. With help of the IDT length, the resulting memory address is then verified to be within the table; if it is too large, an exception is generated. If everything is okay, the 8-byte descriptor stored at the calculated memory location is loaded and actions are taken according to the descriptor's type and contents.
  - A fully populated IDT is 2 KB (256 entries of 8 bytes each) in length. It is not necessary to use all of the possible entries: it is sufficient to populate the IDT up to the highest interrupt vector used, and set the IDT length portion of the IDTR accordingly. Vectors 0-31 are reserved by Intel for processor generated exceptions (general protection fault, page fault, etc.). Though currently only vectors 0-20 are used by the processor, future processors may create incompatibilities for broken software which use these vectors for other purposes.

From *Wikipedia*

Virtualization

Overview

- In computing, virtualization refers to the act of creating a virtual (rather than actual) version of something, including virtual computer hardware platforms, storage devices, and computer network resources.
- Virtualization began in the 1960s, as a method of logically dividing the system resources provided by mainframe computers between different applications. Since then, the meaning of the term has broadened.[1]

From *Wikipedia*

Hardware virtualization

KVM

- Hardware virtualization or platform virtualization refers to the creation of a virtual machine that acts like a real computer with an operating system. Software executed on these virtual machines is separated from the underlying hardware resources. For example, a computer that is running Microsoft Windows may host a virtual machine that looks like a computer with the Ubuntu Linux operating system; Ubuntu-based software can be run on the virtual machine.[2][3]
- In hardware virtualization, the host machine is the actual machine on which the virtualization takes place, and the guest machine is the virtual machine. The words host and guest are used to distinguish the software that runs on the physical machine from the software that runs on the virtual machine. The software or firmware that creates a virtual machine on the host hardware is called a hypervisor or Virtual Machine Manager.
- Different types of hardware virtualization include:
  - Full virtualization – almost complete simulation of the actual hardware to allow software, which typically consists of a guest operating system, to run unmodified.
  - Paravirtualization – a hardware environment is not simulated; however, the guest programs are executed in their own isolated domains, as if they are running on a separate system. Guest programs need to be specifically modified to run in this environment.
  - Hardware-assisted virtualization is a way of improving overall efficiency of virtualization. It involves CPUs that provide support for virtualization in hardware, and other hardware components that help improve the performance of a guest environment.
- Hardware virtualization can be viewed as part of an overall trend in enterprise IT that includes autonomic computing, a scenario in which the IT environment will be able to manage itself based on perceived activity, and utility computing, in which computer processing power is seen as a utility that clients can pay for only as needed. The usual goal of virtualization is to centralize administrative tasks while improving scalability and overall hardware-resource utilization. With virtualization, several operating systems can be run in parallel on a single central processing unit (CPU). This parallelism tends to reduce overhead costs and differs from multitasking, which involves running several programs on the same OS. Using virtualization, an enterprise can better manage updates and rapid changes to the operating system and applications without disrupting the user. "Ultimately, virtualization dramatically improves the efficiency and availability of resources and applications in an organization. Instead of relying on the old model of “one server, one application” that leads to underutilized resources, virtual resources are dynamically applied to meet business needs without any excess fat" (ConsonusTech).
- Hardware virtualization is not the same as hardware emulation. In hardware emulation, a piece of hardware imitates another, while in hardware virtualization, a hypervisor (a piece of software) imitates a particular piece of computer hardware or the entire computer. Furthermore, a hypervisor is not the same as an emulator; both are computer programs that imitate hardware, but their domain of use in language differs.

From *Wikipedia*

Desktop virtualization

- Desktop virtualization is the concept of separating the logical desktop from the physical machine.

- One form of desktop virtualization, virtual desktop infrastructure (VDI), can be thought of as a more advanced form of hardware virtualization. Rather than interacting with a host computer directly via a keyboard, mouse, and monitor, the user interacts with the host computer using another desktop computer or a mobile device by means of a network connection, such as a LAN, Wireless LAN or even the Internet. In addition, the host computer in this scenario becomes a server computer capable of hosting multiple virtual machines at the same time for multiple users.[4]

- As organizations continue to virtualize and converge their data center environment, client architectures also continue to evolve in order to take advantage of the predictability, continuity, and quality of service delivered by their converged infrastructure. For example, companies like HP and IBM provide a hybrid VDI model with a range of virtualization software and delivery models to improve upon the limitations of distributed client computing.[5] Selected client environments move workloads from PCs and other devices to data center servers, creating well-managed virtual clients, with applications and client operating environments hosted on servers and storage in the data center. For users, this means they can access their desktop from any location, without being tied to a single client device. Since the resources are centralized, users moving between work locations can still access the same client environment with their applications and data.[5] For IT administrators, this means a more centralized, efficient client environment that is easier to maintain and able to more quickly respond to the changing needs of the user and business.[6][7]

- Another form, session virtualization, allows multiple users to connect and log into a shared but powerful computer over the network and use it simultaneously. Each is given a desktop and a personal folder in which they store their files.[4] With multiseat configuration, session virtualization can be accomplished using a single PC with multiple monitors, keyboards, and mice connected.

- Thin clients, which are seen in desktop virtualization, are simple and/or cheap computers that are primarily designed to connect to the network. They may lack significant hard disk storage space, RAM or even processing power, but many organizations are beginning to look at the cost benefits of eliminating “thick client” desktops that are packed with software (and require software licensing fees) and making more strategic investments.[8] Desktop virtualization simplifies software versioning and patch management, where the new image is simply updated on the server, and the desktop gets the updated version when it reboots. It also enables centralized control over what applications the user is allowed to have access to on the workstation.

- Moving virtualized desktops into the cloud creates hosted virtual desktops (HVDs), in which the desktop images are centrally managed and maintained by a specialist hosting firm. Benefits include scalability and the reduction of capital expenditure, which is replaced by a monthly operational cost.[9]

From *Wikipedia*