SLEZSKÁ UNIVERZITA
FILOZOFICKO-PŘÍRODOVĚDECKÁ FAKULTA V OPAVĚ

Šárka Vavrečková

Study material

Computer Systems Architecture

Institute of Computer Science
Faculty of Philosophy and Science
Silesian University in Opava

Opava
January 23, 2023

Summary: This document is intended for student of the subject *Computer Systems Architecture* at the Institute of Computer Science of the Silesian University in Opava. The subject deals mainly with hardware – the principle of operation of components, diagnostics, maintenance. Although these are lectures, due to the nature of the subject matter covered, practical instruction is also included.

Computer Systems Architecture

RNDr. Šárka Vavrečková, Ph.D.

Institute of Computer Science
Faculty of Philosophy and Science
Silesian University in Opava
Bezručovo nám. 13, Opava

Typeset with LaTeX

Preface

What we can find in this document

Quick preview: In this subject we are interested in the architecture of computer systems (not only computers), hardware communication interfaces, individual components of a computer or other computer system (processor, motherboard, memory, power supply, etc.), their functionality and maintenance.

Some paragraphs or sections are "additional" (marked with purple icons), these are not discussed and they do not appear on the exam – their purpose is to motivate further independent study or experimentation or to assist in the future in acquiring further information. If there is a purple icon in front of a chapter (section) title, it applies to everything in that chapter or section.

Marking

We use the following colourful icons:

- *Quick Preview*, in which we find out what it's going to be about.
- *Keywords*.
- *Study Objectives* for a chapter tell us what new things we will learn in the chapter.
- New *terms* are marked with the blue symbol, seen here on the left. This icon (as well as the following) can be found at the beginning of the paragraph in which the new concept is introduced.
- *Methods* and tools, ways of solving various situations that a computer administrator may find himself in, etc. are also marked with a blue icon.
- Some parts of the text are marked with a purple icon, indicating that they are *optional sections* that are not discussed (mostly; students can request them or study them on their own if they wish). Their purpose is to voluntarily expand students' knowledge of advanced topics that usually don't get much time in class.

- The yellow icon indicates links where you can get *further information* about the topic. Most often, these icons are web links to sites where the authors go into more detail about the topic.
- Red is the icon for *warnings* and notes.

If the amount of text belonging to a certain icon is larger, the whole block is delimited by a space with icons at the beginning and at the end, for example to define a new concept:

Definition

In such an environment, we are defining a concept or explaining a relatively familiar but complex concept with multiple meanings or properties.

Similarly, the environment may look like a longer procedure or a longer note or more links to more information. Other environments may also be used:

Example

This is what the environment looks like with an example, usually of a procedure. The examples are usually annotated to make it clear how to solve them.

Tasks

Questions and tasks, suggestions for testing, which are recommended for practicing the material, are enclosed in this environment. If there are more than one task in the environment, they are numbered.

Contents

Chapter 1

History

Quick preview: This chapter is devoted to the history of computing, ranging from prehistoric mechanical devices to the first calculating machines, through the various generations of computing technology to the modern era. The last section of this chapter discusses the difference between digital and analogue computers.

Keywords: Abacus, logarithm, Antikythera mechanism, Pascaline, Charles Babbage, Differential machine, Analytical machine, Augusta Ada Byron, electromagnetic relay, electron tube, semiconductor effect, transistor, integrated circuit.

Objectives: After studying this chapter, you will gain a basic overview of the development of computing and the building blocks on which current computing is based.

1.1 Technical Prehistory

1.1.1 Mechanical Aids

Mechanical counters were used in various parts of the world. The most known are the following:

- abacus (Europe),
- sčot (Russia), still in use,
- saroban (Japan),
- suan-pchan (China).

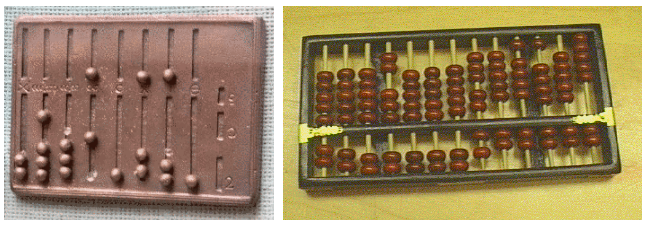


Figure 1.1: Roman abacus and Japanese saroban[1]

The first mentions of similar counters are 3000 years old from China and India.

1.1.2 First Machines

The *Antikythera mechanism* is a 2000 year old machine that was found in a sunken ancient Greek ship. The mechanics are designed with gears and made of bronze. It is uncertain what purpose it was used for, probably astronomical calculations.

[1] Sources: http://abakus.navajo.cz/, http://www.fi.muni.cz/usr/jkucera/pv109/xdavidov.html.

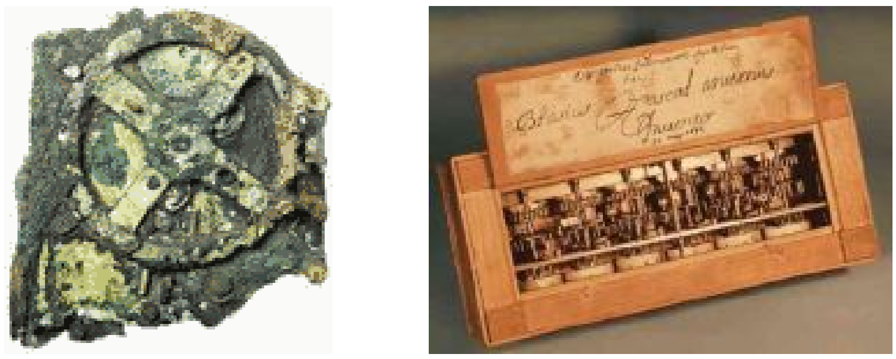


Figure 1.2: Antikythera mechanism and Pascalina[2]

Mechanical calculators were manually powered. The most famous:

- *Pascalina* (1642) – Blaise Pascal built a working mechanical calculator from gears for addition and subtraction (for his father, a tax official),
- *Step Calculator* – Gottfried Wilhelm Leibnitz constructed a mechanical calculator based on the binary system, it was also able to multiply, divide and square.

Later, electricity calculators were developed:

- electromechanical – gears powered by electricity,
- electronic – electronic circuits, there were programmable models.

Programmable devices are the next generation of computing machines. However, programmability was only at a limited level, essentially the ability to work with variable data stored on something that resembled in principle the later punched labels. The most famous are the following:

- *Joseph Marie Jacquard* constructed a loom with a woven pattern stored on a punched label (the "jacquard" pattern is named after him),
- *Herman Hollerith* – a machine for processing statistical data, also using punched labels.

1.2 History

Table 1.1 shows brief overview of the development of computer technology.

1.2.1 Zero generation

Charles Babbage, a British mathematician, was one of the first brilliant inventors in the field of computing. His life's goal was to automate computational processes to save time and, above all, to eliminate errors (which humans often avoid when doing calculations by hand).

During his lifetime, Babbage managed to build and commission several steam-powered machines, proving that this goal was achievable. However, his most interesting machines unfortunately remained only "on paper" – he could not build them for financial reasons, but also because they required very fine mechanical work, which was far from common at the time.

Difference Engine (1822) is one of these machines that Babbage did not manage to finish (he only managed to complete some components). Its purpose was to count polynomial functions up to the

[2]Resources: http://wbdno.wordpress.com/2010/02/06/the-antikythera-mechanism-research-project/, http://michele.nireblog.com/file/237571.

Generation	When	Components
0	19th–20th century 40s of the 20th century	mechanical, gears electromagnetic relays
1	50s	vacuum tubes
2	50s–60s	transistors
3	end of 60s	integrated circuits
$3\frac{1}{2}$	1972 (70s of the 20th century)	integrated circuits (high integration)
4	from 1981 until now	integrated circuits (very high integration)

Table 1.1: Overview of computer history

10th power, and Babbage made do with gears – it was a purely mechanical machine driven by a crank, and the counting was ingeniously done using only the operation of addition.

The differential machine has only recently been built and fully operational, even several times. In 1991, the so-called Difference Engine No. 2 was completed according to Babbage's plans and is now part of the collections at the Science Museum in London. In 2003, a printer was added to it, also according to Babbage's original plans. In 2002, the Difference Engine was built in the USA, and today this five-tonne machine can be found in California and its copy also in London.

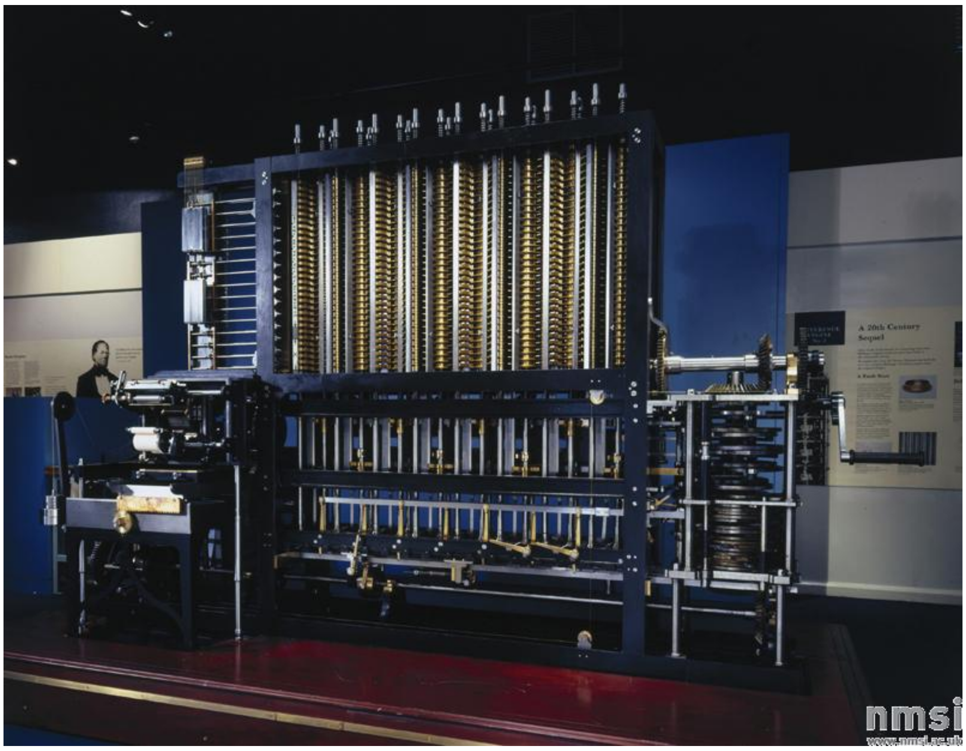


Figure 1.3: Difference Engine[3]

✎ The *Analytical Engine* is considered to be the world's first universal calculating machine (universal in the sense that it can simulate any other machine without the need for hardware rebuilding, just with software modification), it was already much more advanced than the Differential Engine.

[3]Resource: http://collectionsonline.nmsi.ac.uk/browser.php?m=objects&kv=62748&i=123977.

It was a program-controlled mechanical digital computer (driven by steam, no longer by crank), it had its own processor, two registers, it could also do conditional jumps and cycles. In addition to arithmetic operations, it could also solve algebraic and numerical equations, evaluate the results and adjust the course of further calculation to them. He used the analogy of punched labels, in which Babbage was inspired by the previously mentioned Jaquard.

The analytical machine remained only on paper, but attempts to build it have been made throughout history. Until recently, it was more likely that parts of it could be built, which was started by the inventor's son (Henry Babbage). The problems were both financial and technological (gears were breaking, etc.).

The Analytical Engine has another first – the world's first programs were written for it. The first programmer was a woman – *Augusta Ada King, Countess of Lovelace* (also Countess Ada Lovelace) born *Augusta Ada Byron* (she was the daughter of the famous English poet Lord G.G. Byron). The first program in the world created by her is a program for calculating Bernoulli numbers.

Remark

The *Ada programming language* was named after Countess Ada, which was a language somewhat similar to Pascal, but with strong security mechanisms (this language was developed with the support of the US military).

Additional information

- Zero Generation Computers: https://justsciencestuff.wordpress.com/2013/09/08/67/
- The Babbage Engines: https://www.computerhistory.org/babbage/engines/

A major step forward was the invention of the *electromagnetic relay*. Among the computers of the zero generation built using relays were, for example, the computers of the German inventor *Conrad Zuse*, who invented in the 1930s and 1940s successively the Z1, Z2, Z3 and Z4 computers, from Z2 onwards also using relays.

Of these, the most beneficial was the Z3 of 1941, which was the first to function pretty much as it should. It was the first electronic digital programmable computer consisting of approximately 2000 relays weighing one ton. Its purpose was to assist in the design of German bombs, ironically its "end of life" was caused by a bomb (but not a German one).

Among others, we should mention the American computer *MARK I* by Howard Aiken (1943, weighing 35 tons). We can see it in Figure 1.4.

1.2.2 First Generation

First-generation computers had the following features:

- it used vacuum tubes as its base,
- no operating system, no higher-level programming languages, each machine had its own assembler,
- the computer always processed a single task given by a (live) operator.

[4]Resource: http://www.columbia.edu/acis/history/mark1.html

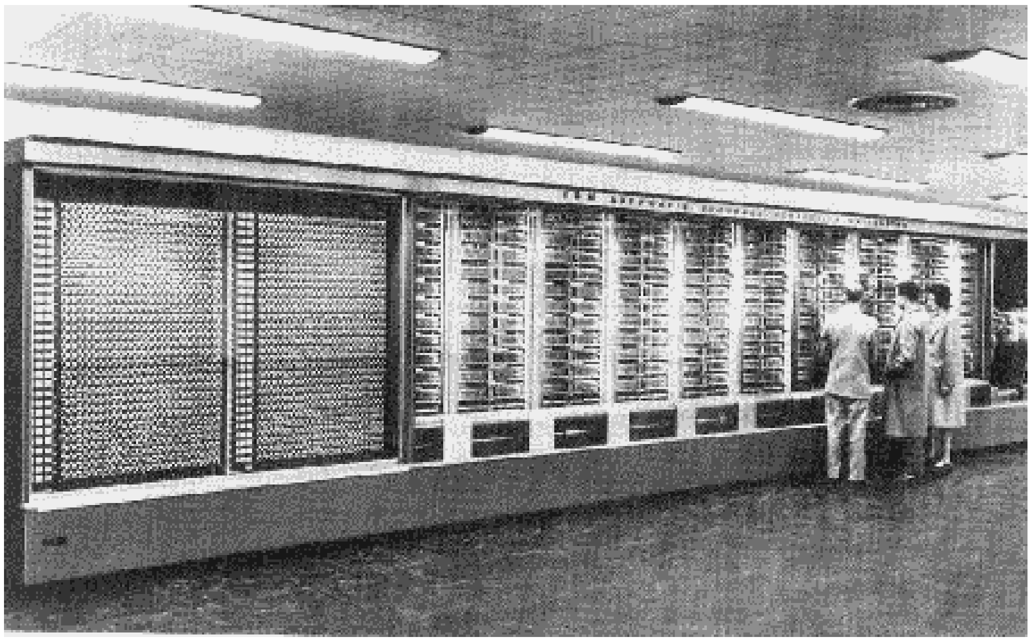


Figure 1.4: Harvard MARK 1[4]

It was basically a big set of cabinets taking up a lot of space. They still need a whole room.

A *vacuum tube* consists of a cathode, anode and conducting wires, all enclosed in a glass tube with exhausted air (i.e. in a vacuum). When the cathode is heated, electrons are emitted by the cathode and are attracted to the anode, thus passing current.

However, the usual size of the electron was a few centimetres, and to build the whole computer meant having enough space, counting on high electricity consumption and thorough cooling (the electrons were "heating" a lot). Because of the high operating temperature and overall complexity, first generation computers were very faulty.

The most famous computers of the first generation include the following:

- *Colossus* (1943) – British computer, assisted in WorldWide War II (decrypted intercepted German dispatches),
- *ENIAC* (1946) – the first all-electron machine (about 17,500 electron tubes), weighing 27 tons,
- *EDVAC* (1951) – unlike ENIAC, EDVAC was binary,
- *UNIVAC* (UNIVersal Automatic Computer, 1951) – the first commercially successful computer; the price was fluctuating (rather rising), gradually exceeding one million USD.

1.2.3 Second Generation

This is a turning point in computing, because it was then that the first truly desktop computers began to appear, and they reached even "ordinary" people (although they were by no means the home computers of today's common size).

Second generation computers were based on transistors. A *Semiconductor* is a solid whose electrical conductivity depends on internal or external conditions, i.e. it is not constant and can be influenced.

The transistor effect was discovered in 1947 at Bell Laboratories. They were awarded the Nobel Prize in Physics in 1956 for this invertion.

✎ In addition to using transistors, second-generation computers also had the following features:

- *batch work system* – programs with data to be processed are stacked by the operator, when one program is finished, the next program from the batch starts processing automatically,
- other *programming languages* besides assembler (e.g. FORTRAN, COBOL).

One of the first transistor computers of the second generation was the *PDP-1* by DEC. It was about the size of a cabinet and was designed for larger companies (55 units were sold, it was a commercially very successful machine). UNIX was created for this computer.

1.2.4 Third Generation

Third-generation computers are based on integrated circuits.

✎ *Integrated circuit* is an electronic component integrating smaller components (transistors, resistors, capacitors, etc.) on a single semiconductor wafer (usually silicon), in a plastic case or otherwise protected. It was invented at Texas Instruments in 1958.

Properties of third-generation computers:

- first operating systems (CP/M, etc.),
- higher-level programming languages (e.g. ALGOL, LISP, Pascal, BASIC),
- floppy disk drive (8" by IBM, in 1971),
- possibility of parallel data processing.

The main representatives of this generation are IBM System 360, Siemens, Tesla 200 and 300.

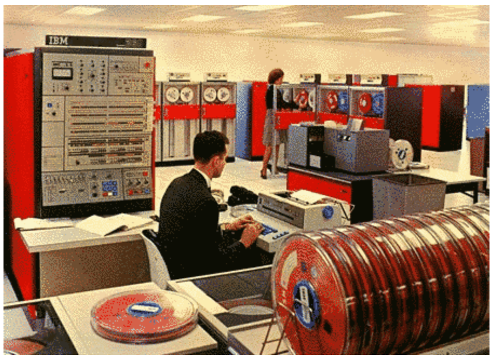


Figure 1.5: IBM System 360[5]

✎ Mainly because of the relatively rapid technical development, we also distinguish the *generation three and a half*. It had the following characteristics:

- integrated circuits of high integration,
- microprocessors, minicomputers, terminals, display,
- first microprocessors by Intel and Motorola.

The first microprocessor was the *Intel 4004* (development started in 1971). This 4-bit microprocessor was originally commissioned by the Japanese company Busicom for a calculator. Later, Intel realized the potential of this product and continued its development.

1.2.5 Fourth Generation

The fourth generation lasts practically to this day. The main feature is the use of *integrated circuits of very high integration*. The first CD-ROM drive appeared (1984).

Available personal computers were 8-bit (Altair, IBM, Apple, Commodore, Atari, ZX Sinclair), later 16-bit (IBM, Apple, ...). The first standard for computer power – AT (Advanced Technology) – also appeared. The first personal computer to meet this standard was the IBM PC AT.

[5]Resource: http://www.plyojump.com/classes/mainframe_era.html.
[6]Resource: http://www.edwardsamuels.com/illustratedstory/isc4.htm.

Altair 8800 (1975) was the first commercially successful home computer. However, it was not sold as a “whole”, the customer received the components and instructions and assembled the computer himself. The next revolution in home computing was the *Apple II*. Also important for subsequent developments was the *IBM PC 5150* (1981), which came with the MS-DOS operating system. In 1982, the predecessor of portable computers (notebooks), the *GRiD Compass 1100* (weighing 5 kg, equipped with a modem and a flat screen, but no storage), appeared.

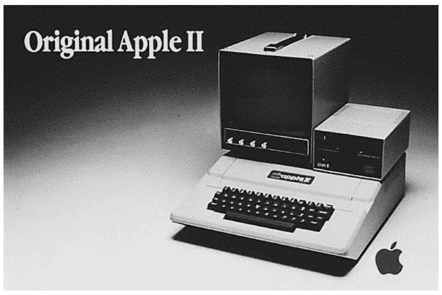


Figure 1.6: Apple II[6]

Additional information

https://www.tomshardware.com/picturestory/508-mainframe-computer-history.html

Chapter 2

Computer Architectures and BIOS

Quick preview: In this chapter we will discuss the (abstract) structure of a computer. We discuss several models in turn – the Von Neumann scheme, the Harvard scheme, and Data-flow, including their relationship to current computing architectures. Another topic discussed is the BIOS and its successor, UEFI.

Keywords: Von Neumann scheme, Harvard architecture, Data-flow architecture, Flynn taxonomy, SISD, SIMD, MISD, MIMD, firmware, BIOS, UEFI, pre-boot applications, secure boot, real-time clock (RTC).

Objectives: After studying this chapter, you will understand the basic structure of the computer and the flow of data in the system – we build on this knowledge in later chapters. Next, you will learn about the BIOS/UEFI, the motherboard firmware, its features and functions.

2.1 Basic Types of Computer Architectures

2.1.1 Von Neumann Scheme

John von Neumann, full name John Ludwig von Neumann, or János Neumann, was an American mathematician born in Hungary. From childhood he showed signs of genius, having an excellent memory, to which he added a very good ability for logical thinking.

At the age of 17 he published his first scientific paper, and at the age of 22 he was already an associate professor at the University of Berlin.

What, for example, made von Neumann important for computing:

- co-creator of *Game Theory* (application: economics and any other fields with chaos behavior),
- he has been involved in the design of some of the most famous computers (such as ENIAC),
- the creator of the *operational theory of quantum mechanics* (Von Neumann algebra),
- nuclear physics, co-designed the first nuclear bomb,
- in 1949 he created the *von Neumann mathematical rules for the design of robots*, which will improve and reproduce themselves; the possibilities of use are only now emerging, NASA wants

to use them to design robots for space.

In 1945, von Neumann designed a scheme of a self-driving computer, called *von Neumann scheme* after him, a sketch of which can be seen in figure 2.1.

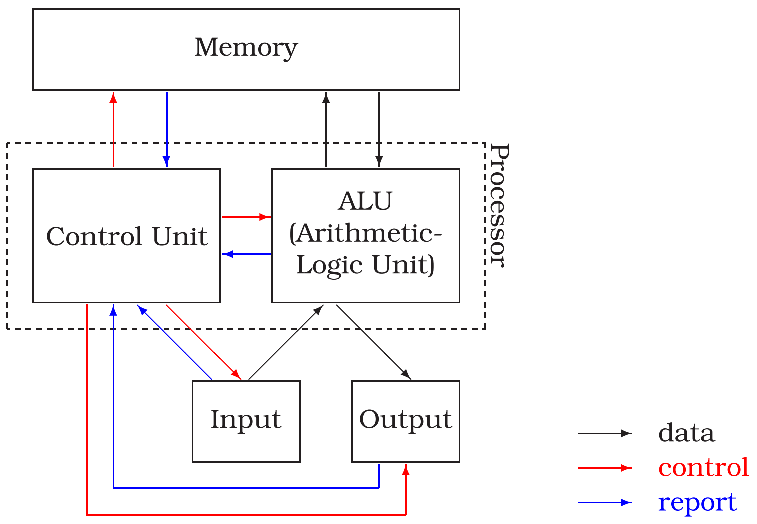


Figure 2.1: Von Neumann Scheme

✎ For each part of the scheme:

1. *Memory* – placing program, data, intermediate results, final results,
2. *Control Unit* – controls the operation of all parts of the computer using control signals,
3. *ALU* (arithmetic-logic unit) – performs all arithmetic calculations and logical operations,
4. *Input* – input devices (e.g. storage, keyboard, mouse, network) for data and instructions,
5. *Output* – output devices (e.g. storage, display, network) for data (and instructions when compiling programs).

✎ The main idea of von Neumann, which is still used today, is *placing the program and data in the same memory.* A program is run, then a process is created based on that program, and the entire process operates in memory (loading its program code, global variables, possibly dynamic libraries, etc.). When this process opens a file, this file is transferred (mapped) again to the memory.

In older computing devices (before this scheme) this feature was not fulfilled, we usually had the program on some external storage medium, from which it was also executed, and the data on another medium (e.g. punched labels).

Other important features are the following:

- there is a computer instruction set in which the program is written, this instruction set is of course known to programmers,
- because the program is placed in memory, it can be modified or otherwise used at runtime, the instructions can be treated as data (the idea was to allow "self-programming" of computers, commonly used in debugging programs or compilers),
- instructions are processed *sequentially*,
- only one program is processed at a time.

The last two points are commonly violated today, as we will see in the chapter on processors.

Remark

This scheme and some of its parts will be referred to in the following chapters, especially in the chapter on the processor. The processor has its own controller, controlling the operation of the whole processor. In the computational cores we have "workmen" – ALUs – performing arithmetic and logical operations. In the processor we also have own memory (registers, cache) in addition to common memory. This scheme is essentially the same in today's computing devices, except for small details like parallel processing of programs and instructions within a single program.

Assembler as a low-level programming language works according to the von Neumann architecture, data can be freely mixed with instructions, although this is not recommended.

Differences in modern computer systems:

- there may be more than one processor in a single computer, or a single processor may have multiple cores,
- there can be multiple programs running at the same time (multitasking, multiprocessing),
- there are I/O devices that are input-output the both (touch screens, multifunction devices = printers+scanners, etc.),
- it is not necessary to have the whole program in memory, it is possible to load only the necessary part,
- virtual memory.

2.1.2 Harvard Architecture

Another architecture that is related to today's common computer architectures is the Harvard Schema. The name derives from the computer *Harvard MARK 1*, whose the full name is the IBM Automatic Sequence Controlled Calculator (ASCC), a zero-generation relay computer at Harvard University. MARK 1 was receiving the program on punched tape and the data on electromechanical media.

Von Neumann Architecture	**Harvard Architecture**
The same parts of memory are used for instructions and data, the both.	There are separate memory parts for instructions and data (different address spaces).
There is common bus to transfer data and instructions between input/output and memory.	There are separate buses used to transfer data and instructions.
CPU accesses either instruction or data for an instruction in the same time, not the both (there is the common bus).	CPU can access instruction and its data at the same time.
It is cheaper.	It is more expensive.
It is used in computers.	It is used in microcontrollers.

Table 2.1: Differences between von Neumann and Harvard Architecture[1]

The Harvard concept brings certain advantages. In addition to greater security (instructions cannot be so easily modified on the fly), it also means that the processor can fetch data and instructions at the same time, because it uses different communication channels for them.

[1]Based on https://www.geeksforgeeks.org/difference-between-von-neumann-and-harvard-architecture/

The Harvard architecture is in fact also found in common computers: the first-level cache (L1 cache) in a processor is divided into an instruction cache (L1i) and a data cache (L1d). Higher levels are not divided in this way.

We derive the *Modified Harvard Architecture* from the Harvard scheme. Compared to the original one, we add the possibility to treat instructions as data, i.e. to treat instruction memory as data memory.

Some special hardware architectures partially implement the concept of the modified Harvard architecture, e.g. MIPS or PowerPC. A bit closer implementation is in Atmel AVR and some microcontrollers such as PICs (Programmable Interface Controllers), DSPs (Digital Signal Processors), etc.

2.2 Instruction and data streams – Flynn's taxonomy

Instruction stream is a stream of instructions that is received by the computing unit (processor). *Data stream* is a stream of data being processed.

In the context of computer architectures, computers can be divided into four groups according to the number of instruction and data streams. This division originally referred to computers, but nowaday a processor typically has multiple instruction sets, each of which may be of a different type, these are more likely to be types of instruction sets.

1. *SISD* (Single Instruction, Single Data) – simple instruction set, each instruction processes a predetermined amount of data (rather small amount), for example instructions for addition, subtraction,...
2. *SIMD* (Single Instruction, Multiple Data) – one instruction processes a sequence of data (vector, with various length), instruction sets for multimedia processing and encryption are usually of the SIMD type,
3. *MISD* (Multiple Instruction, Single Data) – multiple instructions are gradually applied to the same data (instruction chaining or instruction pipilining),
4. *MIMD* (Multiple Instruction, Multiple Data) – combination of SIMD and MISD: multiple instructions are gradually applied to a vector of data.

2.3 BIOS

Firmware is the basic software of (almost) every device (motherboard, expansion cards, hard disk, printer, etc.). The device is equipped with firmware from the manufacturer, usually on a memory chip. In some types of devices, the firmware is called BIOS.

The firmware is used to identify the device and allows it to be used correctly. It is necessary to determine the correct drivers to install, and hardware diagnostic tools also work with firmware.

The *BIOS* (Basic Input/Output System) of a motherboard (i.e. the motherboard firmware) is the system that provides the basic functions of the computer, usually in relation to communication with peripherals. It is the most important firmware in the entire computer, a simple low-level software interface to the computer hardware.

The firmware of other components is also sometimes called BIOS, for example the BIOS of a graphics card.

BIOS is stored in memory (on chip) of ROM, EEPROM or flash PROM type. If it is the BIOS of the motherboard, the corresponding chip can be found on the motherboard. It works with the configuration of the motherboard and its directly connected components stored mostly in battery-powered CMOS RAM (the chip and battery are on the motherboard). We will get to know the memory types in more detail in a later chapter.

Remark

So there are actually two chips on the motherboard related to the BIOS – one of them (usually flash PROM type) stores the BIOS executable code, the other one (usually battery powered CMOS type) contains the configuration of the motherboard and the components connected to it.

The most known BIOS vendors are the following:

- AMI (American Megatrends) – AMI BIOS,
- Award-Phoenix – Award BIOS and Phoenix BIOS,
- others – Insyde, General Software,
- Open BIOS – freely available including the source code (under GNU GPL).

✎ *BIOS Setup* is a special program for setting options related to BIOS functions that can be run in the short time between testing hardware components after the computer is turned on and the operating system is started. At this point, a prompt will appear at the bottom of the screen ("Press DEL to enter Setup", or another key – F2, F1, CTRL+ALT+ESC, CTRL+ESC, etc.). We work in text, pseudographic mode, using the English keyboard layout.

The options are different for various BIOSes, especially:

- date and time settings, display information about connected hardware,
- setting the boot sequence, setting some device properties including the processor,
- ACPI mode (the menu is most commonly called "Power Management Setup"),
- enable or disable communication ports (COM, LPT, USB), RAID support, IR, password setting,
- BIOSes of more expensive boards have various options for CPU and memory overclocking.

✎ *SMBIOS* (System Management BIOS) is a specification for data structures related to BIOS data and access methods to these structures. The purpose of SMBIOS is to allow the operating system and applications to access BIOS-related information.

Remark

While BIOS Setup is the BIOS user interface, SMBIOS can be thought of as the BIOS programming interface.

✎ Right after the computer is turned on, the first part of the BIOS code is executed: the *POST* (Power-on Self Test), which checks the underlying hardware (processor, memory, buses, I/O devices, storage, etc., but also itself).

After the POST, control is passed to the *bootstrap loader*, which determines where the operating systems are installed and ensures that one of them is loaded (passing control to the operating system loader).

2.4 EFI and UEFI

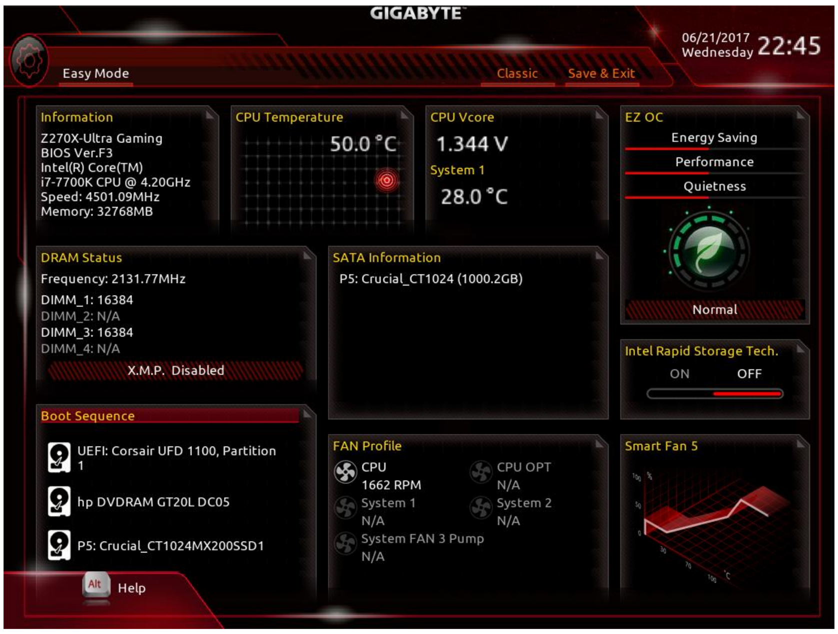
The BIOS was developed several decades ago and was designed according to the common hardware requirements of the time, and hasn't changed much since then. While new items have been added to the BIOS Setup over time, the limits on stored data structures have remained.

✎ *EFI (Extensible Firmware Interface)* is the successor of the open source BIOS. The purpose was to make the system simple, easy to configure. Among the most important features:

- support for 64-bit systems (address lengths, also limits for data structures increased),
- own concept of drivers (i.e. it is already possible to use some devices including mouse and network interface in EFI, no need to start the operating system for this),
- the newer specification includes support for encryption and network authentication,
- is usually configured in graphical mode instead of the text-based (pseudographic) BIOS Setup environment.

EFI is written in C, so the EFI code is longer than the BIOS, and the motherboard must have a flash chip with a larger capacity.

✎ There are several variants of EFI, but the best known and most widely used is *UEFI* (Unified Extensible Firmware Interface) – a consortium of companies (Intel, AMD, Apple, ARM, Phoenix, AMI, etc.) collaborated on it. The following text referc to UEFI because the terms EFI and UEFI are currently more or less interchangeable.


GIGABYTE
06/21/2017 Wednesday 22:45
Easy Mode   Classic   Save & Exit

Information
Z270X-Ultra Gaming
BIOS Ver.F3
Intel(R) Core(TM)
i7-7700K CPU @ 4.20GHz
Speed: 4501.09MHz
Memory: 32768MB

CPU Temperature
50.0 °C

CPU Vcore
1.344 V
System 1
28.0 °C

EZ OC
Energy Saving
Performance
Quietness
Normal

DRAM Status
Frequency: 2131.77MHz
DIMM_1: 16384
DIMM_2: N/A
DIMM_3: 16384
DIMM_4: N/A
X.M.P. Disabled

SATA Information
P5: Crucial_CT1024 (1000.2GB)

Intel Rapid Storage Tech.
ON OFF

Boot Sequence
UEFI: Corsair UFD 1100, Partition 1
hp DVDRAM GT20L DC05
P5: Crucial_CT1024MX200SSD1

FAN Profile
CPU 1662 RPM
CPU OPT N/A
System 1 N/A
System 2 N/A
System FAN 3 Pump N/A

Smart Fan 5

Alt Help

Figure 2.2: UEFI for Gigabyte motherboards[2]

2.4.1 Devices

Device Drivers may already be in UEFI, resulting in better access to the device without having to start the operating system (UEFI starts faster than the operating system). Device drivers can be

[2]Resource: https://www.anandtech.com/show/11618/asus-prime-z270-a-gigabyte-z270x-ultra-gaming-motherboards-review/5.

written in higher-level programming languages such as C (not just Assembler), which makes the job of driver programmers much easier.

It is also possible to build in UEFI *remote device management* (no need to remotely boot the operating system for device configuration interventions, backups, updates and some other activities), this is useful for corporate computing administrators.

2.4.2 Interface and pre-boot applications

UEFI may or may not have a Windows-like GUI, buttons, mouse controls. The network interface is already available within the UEFI, so it is possible to perform updates over the network directly from the interface.

Many operations can be performed in the *UEFI shell* – including working with files and directories, accessing the network, using .NSH scripts.

✎ *Pre-boot applications* are small applications that can be run directly inside the UEFI, installed on the motherboard or on a hidden partition of the hard disk. These include diagnostic programs, backups, firmware configuration programs, *booting of operating system*, and some manufacturers also include smaller multimedia programs (viewing images, playing music or video, working with emails, web browser, etc.).

The purpose of most pre-boot applications is to avoid lengthy booting of the operating system when the computer just wants some "simple" task, others are related to a specific operating system (compared to the BIOS, the concept of the bootloader – the bootloader of the operating system), so the bootloader of systems such as Windows, Linux, MacOS, Solaris, etc., is actually a pre-boot application in UEFI.

2.4.3 Secure Boot

✎ *Secure Boot* is a feature found in most UEFI implementations. Its purpose is to protect the computer from tampering with the kernel – in particular, it enforces the use of certificates for installed and running operating systems, as well as for any modules that are loaded into the operating system kernel.

The installed or running operating system (bootloader and kernel) and each kernel module must be digitally signed with a trusted certificate. Not every certificate can be trusted. If a system or module does not have a trusted signature, installation/execution/reading is prevented.

In practice, this means that you cannot install an operating system that does not have a trusted certificate on a UEFI computer with Secure Boot enabled. As far as Microsoft products are concerned, Windows from version 8 onwards is certified, and many UNIX systems, including common Linux distributions, are also certified. But it depends on what certificates are supported on a particular device (this is determined by the manufacturer of the motherboard or computer).

If we want to install the system without a certificate, we need to disable Secure Boot. Unfortunately, this is not possible with some motherboards, but with others it is, depending on the manufacturer. It is similar to BIOS – the manufacturer determines what the interface will look like and what options will be available (and under what names).

Additional information

https://uefi.org/

2.5 BIOS Upgrade

It may happen that the BIOS (generally firmware) needs to be upgraded. The reasons may be various – we want to replace the processor with a newer one, improve the system communication with the corresponding piece of hardware (for example, add new features to the printer, add support for another operating system (or a new version) or speed up communication with the device), in the worst case we need to fix firmware bugs. We may encounter devices that already physically support a new feature, but because the manufacturer is not yet sure of the actual impact of the feature or has not yet included it in the device firmware, the user cannot access the feature. Therefore, a firmware upgrade can sometimes mean making new features available to the device – the feature is physically implemented in the device, the upgrade just opens access to it.

Some firmware bugs can annoy, others can even cause major problems. Notoriously, bugs appear in the firmware of network devices (such as VDSL routers). We plug in and configure the device and then find that it doesn't communicate with our operating system configuration, that the firewall reacts in some cases completely unexpectedly and cannot be changed, that the router makes a very annoying sound or gets very hot, that only WEP encryption can be used and WPA or even WPA2 is not available, etc. With a printer with two paper trays, for example, we may find that one of them works only when the other one is not empty, it may also happen that when the printer runs out of paper in the tray, the printer still refuses to print after refilling the paper even after restarting the printer (here a restart of the whole computer usually helps), etc.

A dangerous bug appeared, for example, in the firmware of Toyota Prius hybrid cars (in 2004-05, because modern cars also have a beating computer heart). When running on a petrol engine, it sometimes happened that the engine would suddenly stop at speeds of something between 50 and 100 km/h. In the BMW 745i (some examples), the car would come to a "violent" stop if there was less than a third of the fuel capacity in the tank.

In early 2009, Seagate recommended that its customers (Seagate Barracuda 7200.11 hard drive) upgrade the firmware on this drive. Unfortunately, this upgrade contained a bug that caused the drive to permanently lock up access. This error was quite costly for Seagate (it paid compensation to all affected users).

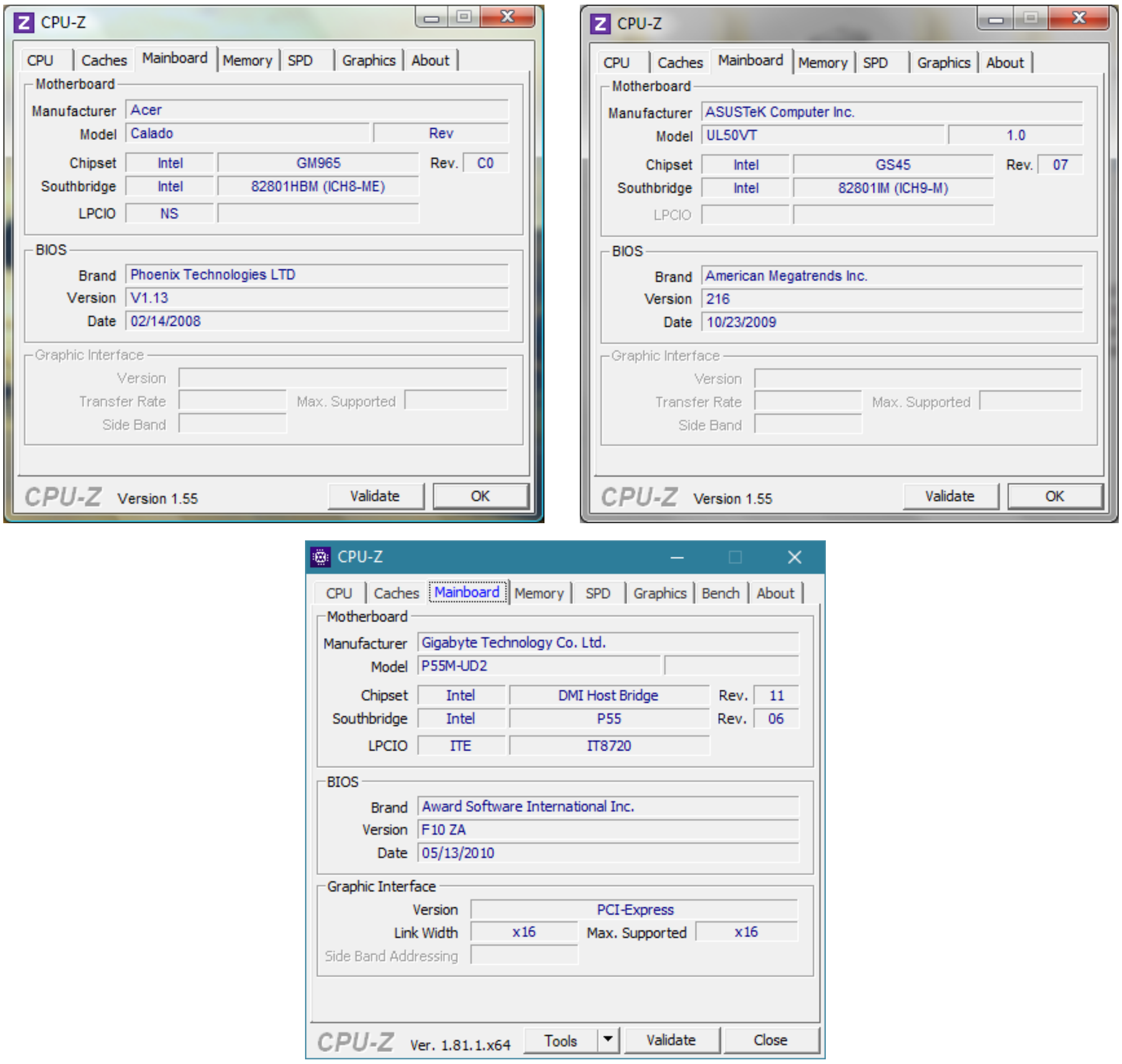
Sometimes it is necessary to upgrade the firmware (BIOS), but we should know that this operation carries certain risks. The upgrade can damage data, the operating system, but in the worst case it can also damage the device (as we read a paragraph above). So the procedure is as follows (we assume a motherboard BIOS upgrade):

- We back up everything we hold dear (especially data),
- find out the exact type of motherboard (ideally with a tool that can get good details about our hardware, like CPU-Z by CPUID[3] (we see 2.3 in the picture),

[3]CPU-Z can be obtained for example at http://www.cpuid.com/. Be careful during installation, you will be persuaded to redirect the browser home page and to install an additional browser bar – refuse both (there are 3 checkboxes in total)

- here we find the motherboard manufacturer's site and enter the type of motherboard found there,
- on the manufacturer's website we can find the exact upgrade procedure, this procedure may vary from manufacturer to manufacturer (and also from motherboard to motherboard), it is a good idea to note down the procedure somewhere or better print it out (this procedure is also usually found in the motherboard documentation),
- it is usually followed by downloading the update file (often with the extension .ROM), saving it to a USB flash drive and then installing it according to the manufacturer's instructions.

Actually the procedure can be quite different, the motherboard manufacturer usually provides instructions on their website.



Motherboard
Manufacturer: Acer
Model: Calado
Rev
Chipset: Intel GM965 Rev. C0
Southbridge: Intel 82801HBM (ICH8-ME)
LPCIO: NS
BIOS
Brand: Phoenix Technologies LTD
Version: V1.13
Date: 02/14/2008
Graphic Interface
Version
Transfer Rate
Max. Supported
Side Band
CPU-Z Version 1.55

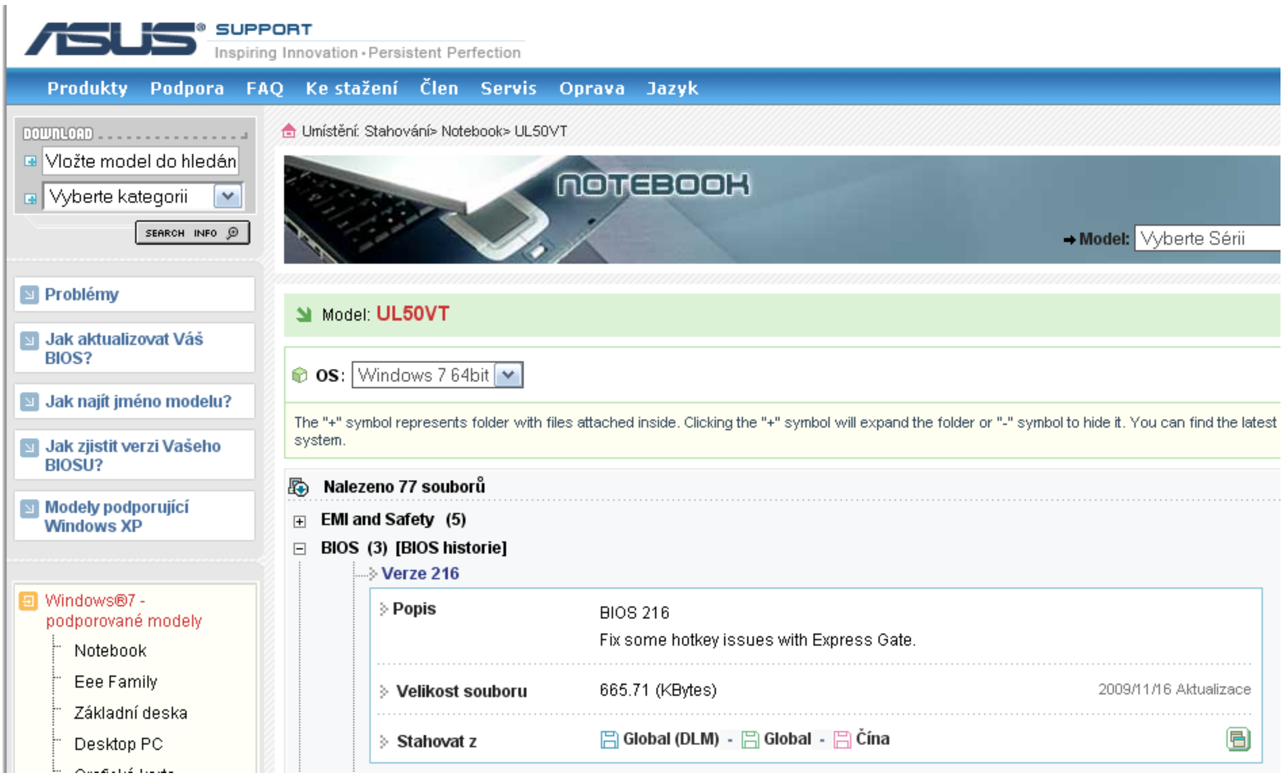
Motherboard
Manufacturer: ASUSTeK Computer Inc.
Model: UL50VT 1.0
Chipset: Intel GS45 Rev. 07
Southbridge: Intel 82801IM (ICH9-M)
LPCIO
BIOS
Brand: American Megatrends Inc.
Version: 216
Date: 10/23/2009
Graphic Interface
Version
Transfer Rate
Max. Supported
Side Band
CPU-Z Version 1.55

Motherboard
Manufacturer: Gigabyte Technology Co. Ltd.
Model: P55M-UD2
Chipset: Intel DMI Host Bridge Rev. 11
Southbridge: Intel P55 Rev. 06
LPCIO: ITE IT8720
BIOS
Brand: Award Software International Inc.
Version: F10 ZA
Date: 05/13/2010
Graphic Interface
Version: PCI-Express
Link Width: x16 Max. Supported: x16
Side Band Addressing
CPU-Z Ver. 1.81.1.x64

Figure 2.3: Detecting the motherboard type in CPU-Z, three different machines

Example

In the image 2.3 we can see the motherboard and BIOS data on three different computers with different versions of Windows. In the first case it is Phoenix BIOS, in the second case it is AMI BIOS, in the third case it is UEFI from Award.


SUPPORT
Inspiring Innovation · Persistent Perfection
Produkty Podpora FAQ Ke stažení Člen Servis Oprava Jazyk
DOWNLOAD
Vložte model do hledán
Vyberte kategorii
SEARCH INFO
Problémy
Jak aktualizovat Váš BIOS?
Jak najít jméno modelu?
Jak zjistit verzi Vašeho BIOSU?
Modely podporující Windows XP
Windows®7 - podporované modely
Notebook
Eee Family
Základní deska
Desktop PC
Umístění: Stahování> Notebook> UL50VT
NOTEBOOK
Model: Vyberte Sérii

Model: UL50VT

OS: Windows 7 64bit

The "+" symbol represents folder with files attached inside. Clicking the "+" symbol will expand the folder or "-" symbol to hide it. You can find the latest system.

Nalezeno 77 souborů

EMI and Safety (5)

BIOS (3) [BIOS historie]

Verze 216

Popis: BIOS 216
Fix some hotkey issues with Express Gate.

Velikost souboru: 665.71 (KBytes) 2009/11/16 Aktualizace

Stahovat z: Global (DLM) - Global - Čína

Figure 2.4: Downloading BIOS upgrade files for an Asus laptop

In the first case it is a laptop from Acer, in the second from Asustek (brand Asus), the third is a computer with a motherboard from Gigabyte. Let's take a look at the BIOS upgrade for the Asus laptop. On http://support.asus.com we enter the type of motherboard we found in the search box (UL50VT, but we can also proceed by selecting it in the drop-down menus). We are asked for the operating system (we enter, for example, Windows 7 64bit), then we get a list of all possible upgrades for this model of motherboard. We select what we need, download. In the image 2.4 we see a page with the current upgrade.

Note *How to update your BIOS?* – there we find all the necessary instructions. It's possible that we won't find a manual for our motherboard among the manuals. Then all we have to do is look for the documentation that came with the motherboard (or computer), we can usually find the procedure there too.

2.6 Computer Time

Real Time Clock (RTC) is a circuit located on the motherboard or integrated into the south bridge of the chipset that keeps real time. Unlike a hardware timer, which controls the timing of the hardware, a real-time clock counts in "human" units.

Real time clocks can be found not only in classical computers, but also in virtually any electronics, including "living room", some devices without RTC cannot even work at all (GPS navigation). In some architectures, on the other hand, they are not standard and you need to buy an RTC chip if you want real-time tracking from these architectures, either for displaying it on the screen or for time stamps in log files (for example in Arduino).

If it is a standalone chip, it is a CMOS chip that requires constant power, so it must be *connected to the battery* on the motherboard just like a CMOS chip with a BIOS configuration.

✎ There are several other circuits in the computer that are related to timekeeping. One more is important for us at the moment – the timer in the processor, which generates a signal at regular intervals and thus determines the speed of the processor and the connected buses. Digital devices need such "flag waving" for synchronization, they cannot work continuously.

☎ **Additional information**

http://www.electronicrepairguide.com/how-to-test-crystal.html ☎

Chapter 3

Interfaces

Quick preview: In this chapter, we'll look at the interfaces you might encounter on a computer. After explaining the basic concepts, we'll look at the universal USB and Thunderbolt interfaces, followed by sections on data interfaces for storage media, connectors for transferring graphics, and interfaces for network communication.

Keywords: Connector, cable ending, plug, socket, cable, interface, hardware interface, signal interface, serial communication, parallel communication, driver, hot-plug, plug-and-play, USB, USB hub, OTG, Power Delivery, USB-AV, USB mass storage, cache, Thunderbolt, COM, RS-232, PATA, SATA, PIO, DMA, ATAPI, eSATA, mSATA, M.2, SATA Express (SATAe), SCSI, SAS, D-SUB, VGA, DVI, HDMI, DisplayPort, RJ-45 (8p8c), Bluetooth, UPnP, DLNA, Wireless HDMI.

Objectives: This chapter gives you an overview of the interfaces through which components or other devices are connected, both outside and inside the computer.

3.1 Several Terms to Start

In order to connect different devices or components inside a device, we usually use a *connector* (plug) of a given type (e.g. USB, HDMI, SATA, . . .). The connector is usually at the end of the *cable*.

The connector is inserted into a *socket* (jack) of the same type, so for example a USB connector is inserted into a USB socket. The socket is usually on a device (e.g. the USB socket is available on a computer, tablet, etc.), but it can also be on the end of the cable (e.g. sometimes you need to interconnect multiple cables).

Definition (Interface)

An Interface (more accurately, a communication interface) is generally a component or functionality that provides communication between devices, components, programs, or anything else that needs to communicate. It is a general term that can be understood in two ways:

- *hardware interface* – specifies what the connector, socket and cable look like, for example how many wires should be in the cable, how they should be arranged in the connector and plug, what the overall shape and size of the connector is, etc.,

- *signal interface (protocol)* – determines how the wires on the communication path are specifically used, for example, what is transmitted by which wire, how logical 0 and logical 1 are represented, signal encoding, etc.

Hardware and signal interfaces are two different things and can be combined in various ways. For example, for the same USB 3.0 signal interface, there are several different-looking types of connectors and plugs (i.e., hardware interfaces) – Type A, Type B, mini-USB, micro-USB, Type C. Conversely, a USB Type C connector/plug may implement a USB version 3.1 signal interface (which most users automatically expect), but it may also be an older version 3.0 or even 2.0.

✎ The interfaces can be *serial* or *parallel* – the serial interface sends data belonging to each other sequentially (one bit at a time), the parallel interface always sends a certain block of data of a fixed size ("word") at a time (depending on the width of the communication path). Serial interfaces are more versatile, easier to extend and easy to program, which is why many areas are moving from parallel to serial interfaces. For parallel interfaces, the need for synchronization and clear separation of "data batches" and more compicated signal encoding makes it too problematic to move to higher frequencies.

Remark

But this does not mean that serial interfaces work with a single wire – usually there is a power supply and a ground, then on some wires, or better a pair of wires, data is transmitted, while on others service information, synchronization signal, etc. More than one pair of wires can be used for data, but in this case unrelated data (not belonging to the same Byte) is transmitted over various wire pairs.

✎ Interfaces can be either *universal* or *specialized* – they are designed for a specific purpose. For example, the USB interface is universal (we use it to connect various types of devices – storage devices, monitors, cameras, smartphones, we also use it to charge devices, etc.).

In contrast, HDMI is intended for multimedia data transmission and is not suitable for other purposes. The SATA interface, through which we usually connect hard drives, is not universal as well, but is intended for connecting storage devices.

Definition (Controller)

A controller is generally a component that controls the operation of one or more devices. Usually this means that when communicating with a device, we communicate with its controller, which then mediates our requests further.

The controller is usually a chip or part of a chip – an integrated circuit, with a low-level program (basically firmware). For example, the processor has its own controller (it's part of the processor chip), the hard drive (usually on the printed circuit board directly on the drive case), the memory is controlled by the memory controller (today it's usually part of the processor, in older computers it was part of the north bridge of the motherboard chipset), the printer, etc.

Remark

Controllers can be simple or complex, depending on the device they control. For example, if we compare the controllers of a regular hard disk drive and an SSD (both are storage devices, and both are treated more or less the same), the hard disk drive controller tends to be simpler, whereas the SSD controller is more complex and performs more functions, and the quality of the controller has a big impact on the quality of the SSD, including write speed and lifetime. SSD controllers today are typically ARM architecture chips.

Definition (Hot-plug and plug-and-play)

An interface has the *hot-plug* property (also hot swapping), if we can connect devices to it on the fly (i.e. it is not necessary to connect before turning on the computer). For example, USB is a hot-plug interface.

The interface has the *plug-and-play* property (Plug&Play) if it supports simplified software installation of the device, i.e. it is not necessary to configure hardware jumpers, set communication addresses, etc. Today, the vast majority of devices are plug-and-play.

Plug&Play technology (authors: Intel, Microsoft, Compaq) presents the possibility of simplified device recognition and configuration *when the device is plugged in for the first time.* The manufacturer adds circuitry to the device controller that performs automatic configuration and cooperates with the operating system to select the appropriate resources (IRQ, DMA, I/O ports, BIOS addresses, discussed later in the next chapter).

The purpose is to make it easier for the user to install the new device and to reduce physical intervention in the hardware (switches, jumpers, assigning IRQ numbers, etc.), any user configuration (if necessary) is done in software and as simple as possible. However, a plug-and-play device may also require a driver to be installed; the driver has nothing to do with this feature.

Hot-plug is something a little different – we can connect a hot-plug-enabled device and have it correctly detect *while the system is running.* This feature is used every time we use it, not just the first time. In practice, this means that, for example, a hot-plug monitor does not have to be turned on before the computer is turned on, but can be turned on at any time afterwards, and similarly, a hot-plug USB flash drive can be connected at any time while the system is running.

Such a non-hot-plug device must be plugged only into a powered-off computer or the computer must be restarted after plugging it in – this is typical of most expansion cards or PS/2 keyboards, for example.

Expansion cards designed for the PCI Express bus are usually plug-and-play, but they are not usually hot-plug. For example, if we want to plug a new graphics card into this bus, we have to turn off the computer, plug the card in, and turn the computer back on again; we can't do this on the fly. In fact, there is a way to make PCI Express work hot-plug, but it has to "learn" both hardware and software, typically we need it on some servers that cannot be simply turned off. This issue is discussed for example here:

https://electronics.stackexchange.com/questions/208767/does-pcie-hotplug-actually-work-in-practice

3.2 USB

USB (Universal Serial Bus) is a universal *serial* interface designed to connect external components to the motherboard. It has both hot-plug and plug-and-play features.

✎ We usually have a USB controller in one of the chips on the motherboard, but we often see that there are multiple controllers (in different chips) on the motherboard. So the manufacturer can add the controller of the version that is missing in the base chipset.

Topology (i.e. the way what is connected or what communicates with what) is based on *USB hubs*, which also work as repeaters (repeater – amplifies the signal). Overall, they form a sort of *tree structure* – think of the controller as the root of a tree. Each branching is provided by a USB hub – it splits one original branch into two "subbranches" (see logo).

There are usually multiple USB hubs in a computer today (connected in a tree structure) so that multiple USB sockets can be brought out. USB hubs are either on the motherboard, or used to bring interfaces out of the case, or are part of monitors, keyboards or other peripherals.

3.2.1 Properties and Versions

The properties of the USB interface are the following:

+ very low price of controllers,
- the dependence of the transfer rate on the speed and load of the processor,
+ high compatibility (usually not a problem with USB device drivers),
+ capacity of up to 127 devices (including hubs).

The speed depends not only on the limits of the technology, but also on the number of hubs in the path, the length of the cable, and the design of the device itself, including the quality of the controller. Table 3.1 lists both the theoretical throughput and the actual speed for each version. Note the units – I'm sure everyone knows the difference between a bit (b) and a byte (B) – that 1 B = 8 b.

	Version 1.1		**Version 2.0**		**Version 3.0**		**Version 3.1**	
	throughput	speed	throughput	speed	throughput	speed	throughput	speed
Low Speed	1.5 Mbit/s (0.2 MB/s)	0.18 MB/s	1.5 Mbit/s (0.2 MB/s)	0.18 MB/s				
Full Speed	12 Mbit/s (1.5 MB/s)	1.1 MB/s	12 Mbit/s (1.5 MB/s)	1.1 MB/s				
High Speed			480 Mbit/s (60 MB/s)	30 MB/s				
Super Speed					4.8 Gbit/s (600 MB/s)	300 MB/s		
Super Speed+							10 Gbit/s (1200 MB/s)	800 MB/s

Table 3.1: USB versions – throughput and speed

The speed is affected by several factors, for example:

- transfer is controlled by the processor, and if the processor is busy with other tasks, the transfer speed decreases,
- the more devices are connected via USB, the lower the speed,
- the more hubs are on the way, the lower the speed,

- the transfer speed also depends on the performance of the controller of the connected device (how fast it can transmit or receive).

In other words – the data in the speed column may actually be even lower.

	Version 3.0	**Version 3.1**	**Version 3.2**	**Version 4**
= after launching 3.1	3.1 Gen 1	3.1 Gen 2	–	–
= after launching 3.2	3.2 Gen 1	3.2 Gen 2	3.2 Gen 2×2	–
Marking after launching 3.2	SuperSpeed USB	SuperSpeed USB 10 Gb/s	SuperSpeed USB 20 Gb/s	SuperSpeed USB 40 Gb/s
Throughput	5 Gb/s 600 MB/s	10 Gb/s 1200 MB/s	20 Gb/s 2400 MB/s	40 Gb/s 4800 MB/s

Table 3.2: USB versions by latest designation

3.2.2 Hardware Interface

There are several different *standards* for USB connectors and sockets, see Figure 3.1. We can also see proprietary solutions that are specific and incompatible with the others – probably an attempt by manufacturers not to let go of the part of the market they previously gained.

The first "row" of the figure shows a USB plug and a jack type A, on the left for USB 2.0, on the right for USB 3.0. They differ in the number of pins (metal strips connected to the wires inside the cable) – there are 4 pins in type A for USB 2.0, and 8 in type A for USB 3.0 – so we can tell the versions apart by the number of pins. Additionally, the jack and connector for USB 3.0 is usually blue inside (usually, but not always). Type A is fully backwards compatible, i.e. we plug the 2.0 connector into the 3.0 socket and vice versa (with lower speed of course). Type A is mainly used for storage media (external drives, USB flash drives, etc.) or for cases for which other types are not intended, including various reductions to other interfaces.

The second "row" of the figure shows the type B USB jack and connector. The number of pins is the same as for type A, but they are arranged in pairs on top of each other in a smaller width in version 2.0 (hence the different shape), in version 3.0 this arrangement was technically impossible and it was necessary to add a "superstructure" for the excess pins. Because of this, backward compatibility is limited – the older connector will plug into the newer socket, but the reverse is not possible – the version 3.0 type B connector will not plug into the version 2.0 type B socket because it simply won't fit. Type B is typically used to connect printers and scanners.

Below we see the Micro-B variant. These connectors are used for small mobile devices. Because of the small size, backward compatibility is limited – the connector and jack for version 3.0 are wider, as we can see.

There is also a Mini-B variant, but it is only found in old mobile devices, it is gradually being replaced by the Micro- variant (that's why there is no Mini-B version 3.0, it is only up to version 2.0). Mini-B is also used on some network devices (some manufacturers use Mini-B as a console port, or

[1]From: http://www.l-com.com/what-is-a-usb-cable

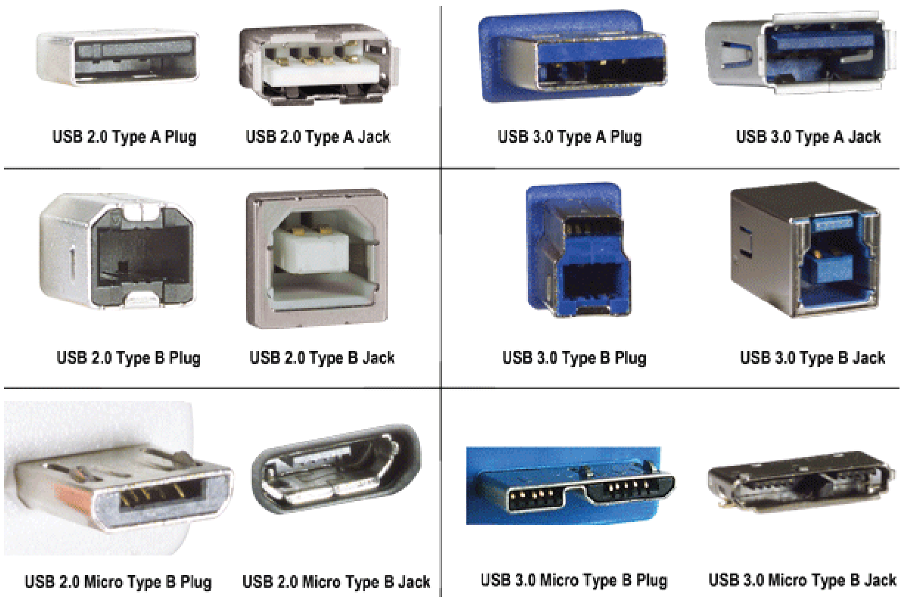


Figure 3.1: USB type A and B connectors and plugs[1]

to connect storage media for example to update firmware), and also on some prototyping boards like Arduino.

	Connector							
	2.0				3.0			
Plug	Std A	Std B	Micro A	Micro B	Std A	Std B	Micro A	Micro B
2.0 Std A	✓				✓			
2.0 Std B		✓						
2.0 Micro AB			✓	✓				
2.0 Micro B				✓				
3.0 Std A	✓				✓			
3.0 Std B		✓				✓		
3.0 Micro AB			✓	✓			✓	✓
3.0 Micro B				✓				✓

Table 3.3: USB types and versions compatibility

Table 3.3 indicates the mutual compatibility of different sockets and connectors. The Micro-AB socket is used to plug in Micro-A and Micro-B connectors.

The newest hardware interface for USB is currently *USB type C*. It is slightly smaller than type A, but at the same time slightly more robust than type Micro-B, so on the one hand it is usable for small mobile devices and on the other hand it is less prone to breakage. It has a narrow oval shape.

The advantage is symmetry (it's reversible) – we can insert the connector into the jack facing "up" or "down", it doesn't matter. The reason is that all the pins are twice in the jack, there is a whole set of pins on each of the two longer edges (of course this is true if we are only communicating at USB 3.1 speed, not for USB 3.2).

[2]From: http://www.golem.de/news/ueberblick-wir-entwirren-den-kabelsalat-um-usb-typ-c-1601-118667.html

In Figure 3.2 we can see the USB type C jack. The hardware interface is the same in all cases, but (as mentioned above) the hardware and the signal interface are two different things. The same USB Type-C can carry a USB version 3.1 signal (which most users expect), but it can also be just USB 2.0, which is typical especially (but not only) for cheaper small mobile devices. The data transfer rate depends on which signal interface is used, which is also indicated in the figure.

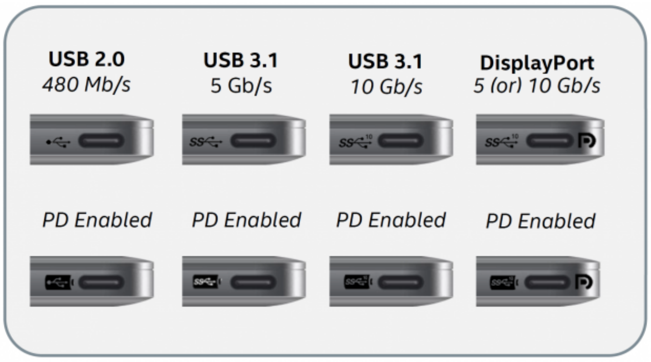


Figure 3.2: USB type C, various kinds[2]

Remark

How do we know how fast data is flowing through USB Type-C? This connector is too small to rely on color resolution, so icons are used, as seen in Figure 3.2.

- If only the USB logo is on the icon, then expect only USB 2.0.
- If the logo has the abbreviation “SS” (SuperSpeed) and no number next to it, it is USB 3.0.
- If the logo has the abbreviation “SS” and the number 10, it is USB 3.1.
- In addition, the letter “D” (abbreviation of DisplayPort) means that it is possible to connect an external monitor that “uses” USB Type-C and the DisplayPort standard as the signal interface.
- If the socket has a lightning bolt symbol (not shown here), the Thunderbolt version 3 signal interface is also supported (see below).

In addition, the icons at the bottom row of sockets in the figure that look like batteries signify improved power support (Power Delivery), and we will return to this in the following text.

Remark

It’s not just the plug and socket that matters. To take advantage of the higher speed, we also need a “faster” cable. Each (actually almost every) pin must be connected to one of the wires in the cable, otherwise it could not be used. If there are more pins in the connector (version 3.0 and higher), we must also have more wires in the cable. It also depends on the cable shielding. Higher speeds usually mean higher frequencies, and for that we need a better shielded cable.

Additional information

The pinout for the USB type C connector can be found in http://vrworld.com/2014/09/22/one-cable-rule-usb-type-c-displayport-alt/.

3.2.3 Power

The USB cable can not only transfer data, but also *power* (yes, even a USB flash drive runs on electricity, it needs power). Both power and data are routed through a common cable, inside there are twisted pair wires (to reduce crosstalk).

With USB 2.0, two data wires and two power wires are used, so there are a total of four wires in the cable. For the USB 3.0 specification, the number of data wires has been increased – for backward compatibility reasons, two remain for lower speeds, and four additional wires have been added for SuperSpeed (so 6 data and 2 power, for a total of 8 wires in the cable). There may be even more pins and wires in a USB Type-C interface.

Devices with lower requirements (camera, modem, etc.) are powered directly over the USB cable, devices with higher requirements (scanner) have additional power supply. However, devices powered over the USB cable may not be correctly detected if they are completely discharged (e.g. camera) – valid for USB up to and including version 2.0. As of version 3.0, this problem should be resolved.

Versions 2.0 and 3.0 allow power to be supplied as follows:

- USB 2.0 – voltage 5 V, current up to 0.5 A = 500 mA, power 2.5 W,
- USB 3.0 – voltage 5 V, current up to 0.9 A = 900 mA, power 4.5 W.

Version 3.1 brought new features in the area of power delivery – the *Power Delivery* function. The Power Delivery standard specifies 5 different profiles – see table 3.4. Not all of them have to be supported by the power supply.

USB 3.1 Profil	**Voltage**	**Current**	**Power**	**Dedication**
Profile 1	5 V	2.0 A	10 W	Smartphones
Profile 2	5 V 12 V	2.0 A 1.5 A	10 W 18 W	Tablets
Profile 3	5 V 12 V	2.0 A 3 A	10 W 36 W	Ultrabooks
Profile 4	5 V 12 V 20 V	2.0 A 3.0 A 3.0 A	10 W 36 W 60 W	Laptops, hub
Profile 5	5 V 12 V 20 V	2.0 A 5.0 A 5.0 A	10 W 60 W 100 W	Computers, docking stations

Table 3.4: USB 3.1 Power Delivery profiles

Support for Power Delivery (and thus the ability to power even more demanding devices) can be recognized by the icon next to the USB socket – in Figure 3.2 on page 25, this is the bottom row, the icon is colored by the battery symbol.

Compared to USB 3.0, there are also various *power saving modes* in the USB 3.1 specification. The controllers check if the device is communicating and the connected non-communicating device has almost zero power consumption, thus saving power.

Remark

If it is a charging-only cable, there are usually only wires for the power supply (e.g. only two wires for version 2.0). We can tell this by the fact that we can only see two pins in the cable connector, instead of the others there are only empty positions.

3.2.4 Comparison of Versions

As mentioned above, after the introduction of USB 3.1 there was a relabeling of USB version 3.0 and we encounter the following:

- USB 3.1 Gen 1 is actually the original USB 3.0,
- USB 3.1 Gen 2 is "real" USB 3.1 (i.e. with approximately twice the throughput of Gen 1).

And to add to the confusion, the following relabeling occurred after the introduction of USB 3.2:

- USB 3.2 Gen 1 is actually the original USB 3.0, or 3.1 Gen 1,
- USB 3.2 Gen 2 is USB 3.1 Gen 2,
- USB 3.2 Gen 2×2 is "real" and full-featured USB 3.2.

Remark

If the manufacturer labels the product as USB 3.1 Gen 1, expect lower speeds than Gen 2. This is an uncharacteristic and illogical move, and it only confuses users. Note that in Figure 3.2 on page 25 we have both middle columns labeled USB 3.1, but apparently one is Gen 1 and the other is Gen 2 (we can tell by the number 10 next to the logo, it means a theoretical throughput of 10 Gbit/s).

Similarly, if all you see is USB 3.2 with no generation designation, it's not sure what the difference in throughput is 5 Gb/s vs. 10 Gb/s vs. 20 Gb/s.

First, a comparison of versions 2.0 and 3.0. We already know of the difference in transfer speeds (table 3.1, page 22), but in reality there are more differences. Higher speeds have been achieved by increasing the number of wires, by changes in the signal encoding (different – more economical – representation of the transmitted data) and by changing the transmission frequency, which has the unpleasant consequence of shortening the range: while the signal for USB 2.0 can be transmitted up to 5 m, for USB 3.0 it is only 3 m. So we cannot use as long cables for USB 3.0 as for USB 2.0.

This does not mean that we can use a five-meter cable for USB 2.0 or a 3m cable for USB 3.0. The 5 m distance for signal transmission includes not only the cable, but the entire path from the main controller on the motherboard to the interface on the connected device. If the computer manufacturer has created an unnecessarily complex and long USB hub structure inside the computer, they may have cut a significant portion of this path.

On the other hand – if we use active cable couplers (which can amplify the signal), the distance achieved may be higher.

Another difference is in the way of communication: in USB 2.0 the data is transmitted in *half duplex* (i.e. communication can be bidirectional, but only one direction can be used at a time, the components must alternate), in USB 3.0 it is already in *full duplex* (it is possible to communicate in both directions simultaneously). This is possible because the wires for SuperSpeed are dedicated separately for both directions.

Now for version 3.1. The throughput is doubled due to the increase in transmission frequency from 5 GHz to 10 GHz, and the physical layer encoding is also changed. The number of wires in the cable remains the same, but the cable design requirements are more stringent (attenuation, shielding, etc.), yet signal quality is only guaranteed up to 1 m (without repeater), and only if the cable is certified for USB 3.1 Gen 2.

The *USB-AV standard* introduced in USB 3.1 defines the transfer of multimedia data (music, video), which was a logical step because the throughput of the device is close to HDMI 1.4.

✎ What's new in version 3.2? Apart from the fact that the previous two versions have been renamed (so when you buy a device or cable "USB 3.2", check the generation), the highest guaranteed theoretical throughput (20 Gbps) will only be over USB-C with two sets of pins (i.e. it really has to be USB-C plugs/jacks with both sets of pins actively connected to the wires, ideally a cable certified for USB 3.2 Gen 2×2). This means that the doubling of speed is mainly due to the fact that twice as many wires are used. If we use a different type of plug/jack, or a USB-C cable that does not have the "correct" number of wires, the throughput drops to the level of some older generation.

✎ In the case of USB version 4 comes compatibility with Thunderbolt (see below), which corresponds to a theoretical throughput of 40 Gb/s. The connector remains USB-C.

3.2.5 Related Technologies

✎ **USB mass storage** is a standard for USB-attached storage devices defined by the USB Implementers Forum. In an operating system that supports this standard, a USB device can be attached as a disk drive (partition). These are usually USB flash drives, external USB drives or some digital cameras. Realistically, it is a set of communication protocols that are supported by the operating system.

The issue of USB mass storage is also related to the possibility of *bootability of USB devices* (i.e. we have a USB flash drive with an operating system and we want the OS to boot from this flash drive instead of the hard disk in the computer). In order for a USB device to be used in this way, the corresponding standard must also be supported by the BIOS and motherboard. We can easily see the support in the BIOS Setup, by the list of possible storage devices for booting the system. The problem can be especially with USB 3.0 and higher, over which some (older) motherboards BIOS/UEFI does not support booting.

✎ **USB OTG** (USB On-The-Go) is a term that has been mentioned a lot lately, especially for mobile devices. Officially, it is an addition to the USB standard adding peer-to-peer communication between peripheral devices over USB without the need for computer or other computing system intervention (Fire-Wire allows this in the base, but USB does not have it directly in the specification). In fact, it is not directly peer-to-peer communication, but it allows the device in question to act as Master or Slave as needed (otherwise the Master is usually a computer with a full-fledged USB controller) – the Master initiates and controls the communication, has a master control role and e.g. also provides power.

What is the benefit of OTG for the average user, why is it good to require this feature? Common small mobile devices (smartphones, tablets, MP3 players, navigation devices, etc.) are usually equipped with a cheap small USB microcontroller instead of a full-fledged one (which we have in larger devices), and microcontrollers without OTG cannot work in Master mode, they can only be Slaves. In practice, this means that while it is not a problem to connect a smartphone with a computer (mobile is Slave, computer is Master), it is a problem to connect an external drive (possibly with a Mini USB or type C connector reduction) to a smartphone – the external drive is always Slave, and if the smartphone can only be Slave, who should be Master?

Therefore, if we have a small device with OTG support, we can connect peripherals (which are always Slaves) to it. For example, we can connect an external drive to an OTG-enabled smartphone or tablet, connect a camera to a printer, etc.

Additional information

http://www.maximintegrated.com/en/app-notes/index.mvp/id/1822

3.2.6 Higher-level Protocols

Any universal interface, not just USB, transfers different types of data, and it is useful to somehow distinguish what type of data is "inside". In the case of USB, we distinguish different *classes*, and the membership of a USB device in a particular class determines what type of data will be transferred, and thus what higher-level protocol it will communicate with.

Example

When we connect a USB device, the device first sends a signal with the class information. This is a number, for example:

- the number 08h (hexadecimal, like the following) means a Mass Storage device, this usually includes digital cameras with direct memory access,
- the number 07h means a printer or a CNC machine,
- the number 03h represents a Human Interface Device (HID) class device for simple human communication, meaning a keyboard, mouse, touchpad, etc.,
- the number 0Bh represents an SD card reader,
- the number 0Dh is used by fingerprint readers, etc.

While the USB 2.0/3.1/3.2/etc. standard specifies the physical parameters of the signal (what's on which pin, frequency, encoding etc.), the class specifies what the data sent should look like in terms of its internal structure. It communicates somehow with a printer and differently with a keyboard or a storage.

In case we want to connect a display device (typically an external monitor) via USB-C and transmit an audiovisual (AV) signal, we should be interested in *alternate modes*. Alt mode is actually an extension of USB-C that allows to transmit "foreign" signal originally intended for another type of connector. The most common Alt modes are:

- DisplayPort and Thunderbolt are indicated by the letter D or a lightning bolt beside the USB socket,
- DP: DisplayPort and HDMI signal support,
- MHL,
- VirtualLink – for virtual reality devices.

For example, if we want to connect an external monitor with the HDMI interface to the laptop via USB-C, we first need the USB-C interface supporting the DP Alt mode on the laptop, then the cable must have a USB-C connector on one side and an HDMI connector on the other.

Figure 3.3: USB-C to HDMI Adapter cable with DP Alt mode[3]

[3]From: https://www.amazon.com/

3.3 FireWire

FireWire (I.Link, Lynx, IEEE 1394) is a universal serial interface designed to connect external components to the motherboard, a competitor to USB. It originated from Apple, probably that's why this interface waited for a very long time for support in Windows, that's probably one of the reasons for its less widespread use.

It meets both hot-plug and plug-and-play features. Up to 63 devices can be connected, with a maximum cable length of 4.5 m and up to 100 m in the latest specifications. Components for FireWire implementation are more expensive than components for USB implementation, but on the other hand it is considered more reliable.

Other properties are the following:

- FireWire uses different signalling than USB, the ratio between theoretical throughput and real speed is better,
- transmission is not as heavily "batched" as USB, the data flow is more stable,
- it is possible to connect multiple devices to one controller in a tree structure (similar to USB) or as a daisy-chain (chained one by one),
- the possibility to power the connected device.

FireWire uses several types of *connectors*, which differ primarily in the number of signal circuits (4, 6 or 9) and then in size. Figure 3.4 shows the first two types and also a part of the expansion card with two 6-circuit connectors.

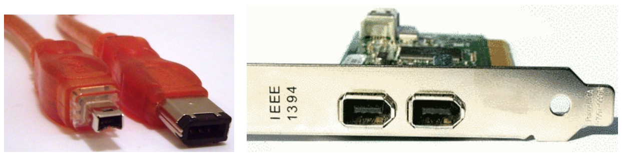


Figure 3.4: FireWire connectors and plugs[4]

The FireWire interface was selected by the High Definition Audio-Video Network Alliance (HANA) as the standard for AV transfers, and is still used today in some professional camcorders and other devices for moving mainly multimedia data, because it is reliable, with a continuous data stream and fast enough, not as dependent on CPU load as USB. FireWire also has prevailed as a system bus in many military devices (F-22 Raptor and F-35 Lightning II fighters) and in the automotive industry (Customer Convenience Port). Implementations of FireWire also appear in some networking equipment.

3.3.1 Thunderbolt

In February 2011, Intel introduced a new universal serial interface called *Thunderbolt* (TB). This interface was developed under the original name *Light Peak* (introduced in 2008, in experimental form) as an optical interface, but Thunderbolt is currently mostly encountered in the form for metallic (copper) cables. Optics are more expensive, but the cables can be longer. As with USB, there is a cost advantage with this interface because Intel does not require payment of licensing fees that would make implementation more expensive.

The latest version is 3. While version 1 had a theoretical throughput of 10 Gbit/s per channel, using two independent channels (i.e. 20 Gbit/s total), version 2 has a throughput per channel of 1 Gbit/s with up to 20 channels possible, i.e. 20 Gbit/s total as well (only it can be scaled better).

[4]From: http://en.wikipedia.org/wiki/FireWire

Version 3 doubled the theoretical throughput to 40 Gbit/s, which outperforms USB up to and including version 3.2. The real speed is about a third lower than the theoretical speed.

Devices can be daisy-chained using cables (actually in the same way as FireWire, daisy-chain or tree), up to a maximum of 7 devices, communicating in full duplex.

The metal cable can also power the device (up to 10W, which is more than USB 3.0 and less than FireWire or USB 3.1 Power Delivery in Profile 2 or higher).

The signal and hardware interface is an interesting hybrid. The signal is encoded in the same way as DisplayPort (i.e. the DisplayPort protocol is used, we'll learn more about this interface later in this chapter), it is transmitted over the PCI Express bus (more about this in the next chapter, the PCI Express bus is also used for graphics card signal transmission), and the jack/connector is one of the following two types:

- Thunderbolt 2: the Mini DisplayPort jack/plug is used, recognizable from the Mini DisplayPort with the original meaning by the lightning icon ,
- Thunderbolt 3: the USB Type-C connector, and it must have the lightning icon .

Both versions are backwards compatible, you just need to use an adapter (TB3 to TB adapter) because the interfaces are physically different.

Other Thunderbolt version 3 features are related to the USB-C connector, including possible PowerDelivery support (if the manufacturer uses it, of course) and relevant power profiles.

You can buy either active or passive cables for Thunderbolt. Passive cables are cheaper, but the declared throughput is guaranteed only for a short distance (usually 50 cm), longer passive cables (up to about 2 m) allow to work at half the speed. Active metallic TB cables can usually be found in lengths up to 2 m (but can be longer), but even with lengths over half a meter they guarantee a throughput of 40 Gbps.

It is also possible to find optical TB cables up to 60 m long, but for several times to several orders of magnitude higher price at the same throughput, and the connected device cannot be powered through them, which must be handled separately.

When we buy a cable for Thunderbolt (version 3 of course), it is not enough to look for a cable with USB-C connectors, it really must be a Thunderbolt certified cable, otherwise we "fall" to the level of USB (in the worst case USB 2.0). Only cables directly designed for Thunderbolt are guaranteed to support Alternate modes (see chap. 3.2.6), the reverse is not valid (i.e. if the cable is not directly designed for Thunderbolt, it *may* support *some* Alternate modes). However, we will pay extra for a certified TB cable.

Additional information

- https://www.intel.com/content/www/us/en/architecture-and-technology/thunderbolt/overview.html
- https://blog.startech.com/post/thunderbolt-3-the-basics/

3.3.2 Comparison

Table 3.5 contains a comparison of different technologies. Note that each technology is optimised for a certain application and certain requirements.

Interface	**Theoretical throughput**	**Max. number of devices**	**Max. transmission path length**
USB 2.0	up to 60 MB/s	127	5 m
USB 3.0	up to 600 MB/s	127	3 m
USB 3.1	up to 1200 MB/s	127	1 m
USB 3.2	up to 2400 MB/s	127	1 m
FireWire 1600	up to 200 MB/s	63	4,5 m
FireWire 3200	up to 400 MB/s	63	4,5 m
Thunderbolt	up to 1200 MB/s	7	3 m
Thunderbolt 2	up to 2400 MB/s	7	3 m
Thunderbolt 3	up to 4800 MB/s	7	3 m
SATA II	up to 375 MB/s	1	1 m
SATA 3.0	up to 750 MB/s	1	1 m
Gbit Ethernet	up to 125 MB/s	1	100 m

Table 3.5: Comparison of some parameters of universal and other interfaces

While with USB it is the quantity (the number of devices connected to one main controller) and speed that is important, whereas the cable length is usually not so important, with Gigabit Ethernet (last row) it is exactly the opposite. FireWire is optimized for smooth data flow, has a better ratio between theoretical throughput and real speed, and moreover a more stable power mechanism, so again it is somewhat harder to compare directly. For SATA, the throughput figures look pretty bad, but they are sufficient for connecting classic hard drives because the hard drive itself can't run faster, and the cable length is also sufficient. Thunderbolt is the most progressive at the moment, its disadvantage is the small spread so far.

3.4 Parallel and Serial Port, PS/2

The LPT and RS-232 ports were originally the only representatives of parallel and serial interfaces, so they were simply called that.

✎ **the LPT port** (parallel port, Centronics) is almost not used anymore and is often not even found in newer computers. It was originally designed for parallel unidirectional communication with printers, later other options were added including bi-directional communication. The parallel LPT port is still used to connect many old printers.

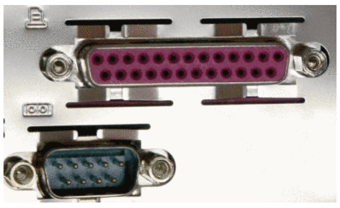


Figure 3.5: Parallel and serial ports

✎ **The RS-232 port** (serial port, COM, V.24, service port) is available in two sizes (the smaller one has 9 pins, the larger one has 25 pins just like the parallel one). The smaller one is more common, which is also referred to as D-SUB DB-9 (the larger one is D-SUB DB-25), and there are also inaccurate UART designations, especially for Arduino.

There are various variants of this port, namely RS-422 (for reliable transmissions even over longer distances in busy environments) and RS-485 (in industrial applications).

Previously, RS-232 was used to connect a mouse and a modem, but nowadays it is mainly used in industry and commerce (label and barcode printers, measuring systems, control mechanisms), then in some network devices, mainly because it is very easy to program. More recently, RS-232 has been used in various prototyping boards, such as various modules for Arduino, also because of the ease of data flow control.

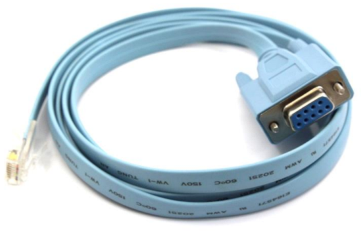


Figure 3.6: Console cable RS-232 to RJ-45

For network devices or servers, it is used as a service interface. These devices usually do not have a keyboard and screen, but they do have a 9-pin RS-232 port or other, but with the RS-232 signal interface. So we need a proper cable and a laptop with a serial interface, or we can use an RS-232 to USB reduction, because laptops nowadays usually don't have an RS-232 port (but then we need to have serial interface emulation enabled on the laptop).

For Cisco network devices: on a network device, there is a hardware port in the shape of a regular network port (RJ-45) labeled *console*, but in terms of signaling it works like RS-232. So we need to have a special cable with RS-232 on one end (computer) and RJ-45 on the other end (network device) – see Figure 3.6, or again use a reduction from RS-232 to USB.

Remark

The 25-pin serial interface is very similar to parallel. We can tell them apart by their orientation – on the computer side, the serial interface has "pins" on the computer side, whereas the parallel interface has "holes", as seen in Figure 3.5. Another possibility of confusion is with the VGA interface (we'll see next).

The PS/2 interface was originally designed for IBM computers, but has spread to other manufacturers. It is still sometimes used in computers to connect a mouse and keyboard. For computers without a PS/2 port, where we want to use PS/2 devices, we can get PS/2⇒USB adapters (see Figure 3.7).

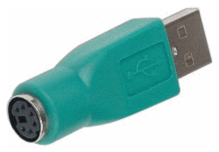


Figure 3.7: Adapter PS/2 to USB

The interface is of the Mini-DIN type (DIN interface is rounded with pins), has 6 pins and is very easy to find on the desktop – the keyboard interface (purple) and mouse interface (green) are colour coded, or the jack, which is half purple and half green.

3.5 Storage Media Data Interface

3.5.1 PATA and SATA

ATA (abbreviated from AT Attachment, where AT stands for Advanced Technology – AT for PC computers) is a long-standing standard for connecting parallel transfer storage devices. After the publication of the SATA (Serial ATA) standard, the original ATA standard was renamed PATA (Parallel ATA, cca 2003).

The old standard *ATA-1* (1986) was called *IDE* (Integrated Drive Electronics) interface, which is a somewhat inaccurate name, it actually refers to the type of cables used.

In the case of PATA, there can be a maximum of two drives on a single cable, and hardware configuration is required (i.e. it is not a plug-and-play solution): on each drive connected to the same cable, the *connectors* must be set so that one is *master* and the other *slave*. If only one disk is connected, it must be marked as *single* by the jumpers (however, on many disks this condition did not exist, the disk was set as master). Figure 3.8 shows how to connect multiple disks.

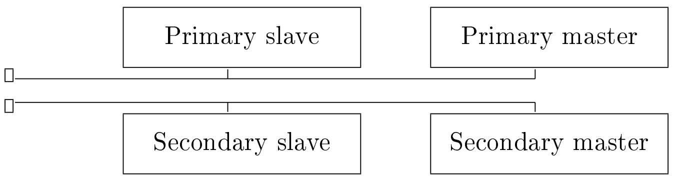


Figure 3.8: Connection of PATA storage media

Definition (Communication mode – PIO and DMA)

Devices connected via PATA or SATA communicate in either *PIO* (Programmed Input/Output) or *DMA* (Direct Memory Access) mode.

In PIO mode, the processor controls all communication. In DMA mode, the processor is used only at the beginning, it issues the command to work with the data, and the rest of the communication (the actual transfer) is handled by the hard disk controller. We will discuss DMA channels in the next chapter.

ATA is not just about how a signal is to be shaped, it includes the definition of a set of low-level ATA commands that can be used to communicate with the connected component. These are commands such as device identification, reading from or writing to a sector on the disk using PIO or DMA, device reset, buffer handling, sectors checking, etc.

Subsequent generations of the ATA standard always add some feature and commands (for example, ATA-3 introduces the S.M.A.R.T. monitoring technology, ATA/ATAPI-4 comes with the ATAPI standard).

Definition (ATAPI signal interface)

The *ATAPI* (ATA Packet Interface) standard is a higher-level interface to ATA; it runs on top of the original low-level ATA interface, adds new features, and makes devices easier to access (BIOS, operating system, and processes can easily detect attached drives and their properties via ATAPI). ATAPI brought support for optical drives to work with storage media (previously this was handled by emulating optical drives as hard disks), and also standardization of logical transfer methods (DMA, Ultra DMA), S.M.A.R.T., cards (compact flash).

SATA (Serial ATA) is a serial interface to storage devices (hard drives, optical drives and other mass storage devices). Unlike PATA, it allows only one device to be connected to one interface (we don't connect devices to the "bus"), so we can have multiple SATA interfaces in a computer and there is no need to distinguish single/master/slave drives.

Currently there are multiple versions of SATA:

- of the original specification (SATA 1) means a data throughput of almost 1.5 Gb/s (watch out, bits) with a data transfer rate of about 150 MB/s,
- SATA II should have twice the throughput (3 Gb/s, data rate of about 300 MB/s),
- next generation (SATA 3.0) means throughput up to 6 Gb/s, data transfer rate 600 MB/s,

- SATA 3.1 has brought some functional improvements such as mSATA (see below),
- SATA 3.2 added M.2 (see below) and came with SATA Express enhancements (acceleration by connecting to PCI Express, just like Thunderbolt), throughput up to 16 Gbit/s (real almost 2 Gbit/s),
- SATA 3.3 (2016) and SATA 3.4 (2018) add other new features such as SMR drive support (see below) and better monitoring and power management of connected devices.

A maximum cable length of 1 m is assumed and the interface supports hot-plugging. It communicates via PIO or DMA (or faster Ultra DMA).

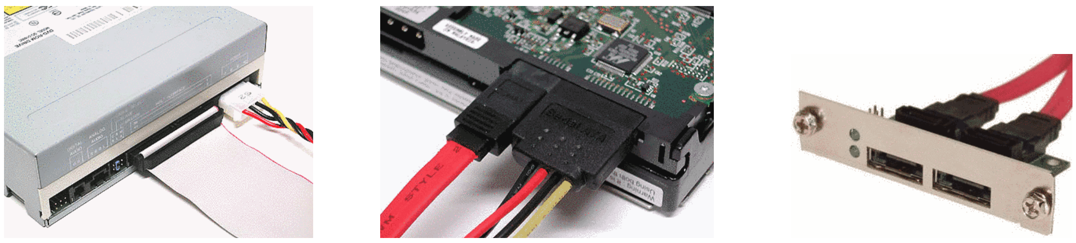


Figure 3.9: PATA, SATA and eSATA[5]

The SATA cable and connector is distinguishable from PATA at first glance. It is narrower (because communication is not parallel but serial), contains only 7 wires (4 of which are data wires, transmission is full duplex – bidirectional), is better insulated (thicker insulation layer) and the connectors on the cable are only on the ends (one cable = one device). We can see the difference in the picture 3.9: on the left is the connected PATA drive (the wide white cable is the PATA/IDE data cable, next to it is the power cable with colored wires), in the middle is the connected SATA drive (the red cable is the data cable, next to it is the power cable again). The SATA cable is usually red or blue, but it can be different, the color is not standardized.

The *eSATA* is a special version of SATA designed for connecting external drives (i.e. it takes the SATA interface out of the computer case), the eSATA connector and jack are more robust than SATA, they are designed for frequent disconnections, as we can see on the right in Figure 3.10.

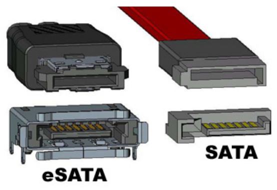


Figure 3.10: SATA and eSATA[6]

Another interface based on SATA is *mSATA* (mini-SATA), a standardized version of SATA 3.1. mSATA is characterized primarily by high throughput (speed), so it can be used by SSD devices. It looks similar to the Mini PCI Express bus slot, but uses completely different signaling, so mSATA devices really need to be plugged into a slot labeled as mSATA. It is encountered in some laptops, but is gradually being displaced by the M.2 interface.

The *M.2* interface is newer than mSATA (standardized in SATA 3.2), but is more common nowadays. Visually it is very similar (except for the keying – the placement of notches and plastic bridges), functionally and other features as well, it is connected to PCI Express. M.2 is mostly used for faster SSDs, but also for things like Wi-fi modules and other components that require fast data transfer.

Figure 3.11 shows several SSDs – one with mSATA interface and three different sizes with M.2 interface.

[5]From: http://www.cubeternet.com
[6]From: http://www.homestead.co.uk

Remark

The keying is also different for different M.2 devices. Usually we encounter one notch (M-key) or two notches (M+B-key), there is also a B-key. The M-key and B-key types communicate in different ways, they are not interchangeable, so it is usually easiest to buy a component (like an SSD or Wi-fi module) with their M+B combination with two notches that can communicate both ways.

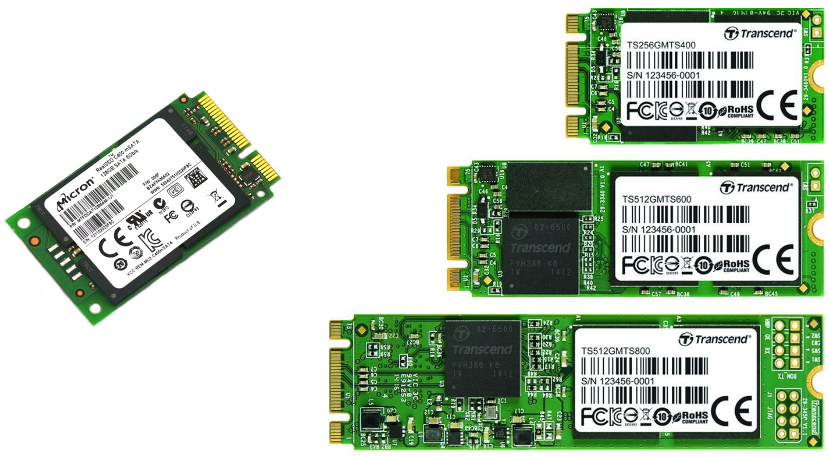


Figure 3.11: mSATA and M.2 (three various sizes)[7]

3.5.2 SCSI and SAS

SCSI (Small Computer System Interface) is another hard disk interface (1986), but it is also used by other types of devices (for example, SCSI scanners). It is therefore more universal than ATA.

Up to 8 or up to 16 devices can be interconnected, one of which is the controller (each device has its unique numeric address in the range 0–7 or 0–15, the highest address is the controller). The devices can be both internal and external (usually one bus leads in and one bus leads out from the controller). The bus must be terminated by *terminators* (terminating resistors) on "interface" devices, as shown in Figure 3.12.

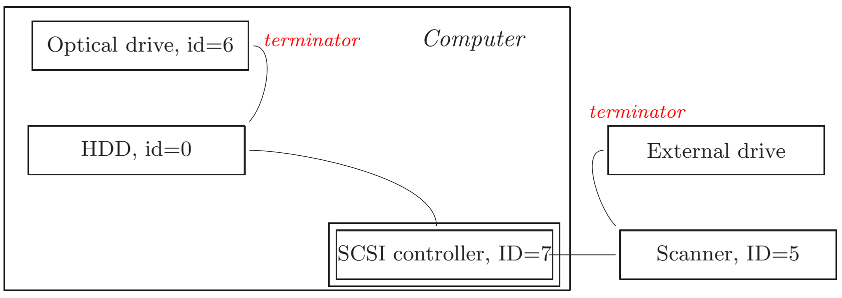


Figure 3.12: The SCSI network

Connecting multiple devices will not affect the speed. The SCSI communication set (which is used to communicate with the controller) is much more extensive than the ATA commands, including error reporting commands, and commands can be transported over a computer network – iSCSI technology.

Some (server) SATA hard drives implement a SCSI communication set in addition to the ATA set.

[7]From: http://www.svethardware.cz/art_doc-02CB0B602D89AD31C1257A460060978F.html, http://german.alibaba.com/product-free-img/msata-to-sata-adapter-109775658.html

SAS (Serial Attached SCSI) is a point-to-point serial interface, it is the successor of the original parallel SCSI. It uses the same communication set as SCSI (SCSI commands). Throughput increases with versions:

- SAS-1: 3 Gb/s
- SAS-2: 6 Gb/s
- SAS-3: 12 Gb/s (2013)
- SAS-4: 22.5 Gb/s (2017)

Higher voltages are used for signals than SATA, so cables can be longer (up to 10 m).

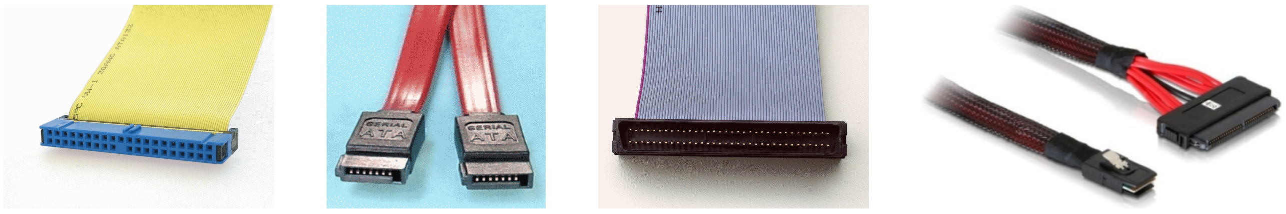


Figure 3.13: PATA, SATA, SCSI, Mini SAS and SAS (internal) connectors

SAS can be found in some server hard drives because it is considered to be about as fast as SATA (depending on the version, of course) but more reliable. Another important advantage is the possibility to "export" the interface out, using the network (iSCSI protocol), and to ensure "high availability".

Interface	**Teor. throughput**	**Speed**	**Cable**
Thunderbolt 3	40 Gbit/s		3 m (coper), 60 m (optics)
SAS-4	22,5 Gbit/s		10 m
Thunderbolt 2	20 Gbit/s	2.44 GB/s	3 m (coper), 60 m (optics)
USB 3.2	20 Gbit/s	2.42 GB/s	1 m
SATA 3.2	16 Gbit/s	1.97 GB/s	1 m
SAS-3	12 Gbit/s	1.2 GB/s	10 m
Thunderbolt 1	10 Gbit/s	1.22 GB/s	3 m (coper), 60 m (optics)
USB 3.1	10 Gbit/s	1.21 GB/s	1 m
SAS-2	6 Gbit/s	600 MB/s	10 m
SATA 3.0	6 Gbit/s	600 MB/s	1 m
USB 3.0	5 Gbit/s	400 MB/s	3 m
FireWire 3200	3.144 Gbit/s	393 MB/s	100 m+
SAS 1.0	3 Gbit/s	300 MB/s	10 m
SATA 2.0	3 Gbit/s	300 MB/s	1 m
USB 2.0	480 Mbit/s	35 MB/s	5 m

Table 3.6: Comparison of mass storage interfaces

Table 3.6 shows a comparison of different technologies for data transmission, the technologies are sorted according to theoretical throughput. Again, note that the data are relative, for example, compare the real speed for USB 3.1, Thunderbolt 1 and SAS-3. Other differences (and very significant ones at that) may be due to the implementation (programming) of the controller of a particular device, the particular application (large file transfers or sequences of small files, transfer length, etc.), the actual length of the transfer path, etc.

3.6 Graphics Cards Connectors

3.6.1 VGA

VGA (or D-SUB) is fully *analog*. The "D" in the name comes from the typical "D" connector shape.

The VGA interface can be seen in Figure 3.14. Its size and layout may seem interchangeable with the RS-232 serial interface, but as we can see, the VGA on the computer side has "holes", whereas RS-232 has "pins" – see Figure 3.5 on page 32 (on the cable side it is of course the other way around to allow the cable to be connected). Another difference is the pin/hole arrangement – while RS-232 has two rows, VGA has three.

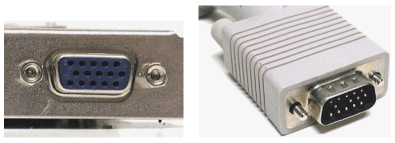


Figure 3.14: VGA interface

For display devices that process digital images, typically LCD monitors, it is usually possible to use analog VGA, but with the following D/A/D conversion:

digital (graphics card) → analog (VGA) → digital (LCD)

The consequence is a slightly worse image than can be achieved with a digital interface (some information is lost in the conversion), and the response may be degraded. For office applications it doesn't matter, the consequences can be seen for example when playing faster games.

3.6.2 DVI

✎ DVI (Digital Visual Interface) is an interface understood more as a digital interface, but in fact some types are combined (they can transmit an analogue signal without conversion, i.e. a combination of digital and analogue interfaces) and even a fully analogue variant has been proposed.

✎ There are several types of DVI connectors, as we can also see in Figure 3.15:

- DVI-A – analog only, for CRT monitors and some TV cards, D/A to analog output conversion, almost never seen in practice,
- DVI-D – uses only digital pins, is the most common, exists in two versions:
  1. dual (uses all digital pins),
  2. single (some pins are not used),
- DVI-I (Integrated D/A) – both digital and analogue pins are present and functional, there are also single and dual variants.

Dual DVI is also referred to as DVI-DL. Its bandwidth is higher (Single has 4.9 Gbps, Dual 9.9 Gbps) and is intended for high resolution and frequency monitors (for these it is required, including the cable), for office use it brings no advantages.

If we want to connect a 4k monitor, we cannot use DVI at all (or yes, but at a lower resolution).

The DVI interface transmits only the image, so if we have speakers on the monitor, we must also connect it to the sound card.

☎ **Additional information**

- http://www.hardwarebook.info/Digital_Visual_Interface_(DVI)
- https://www.datapro.net/techinfo/dvi_info.html

☎

[8] From: http://www.deltapage.com/

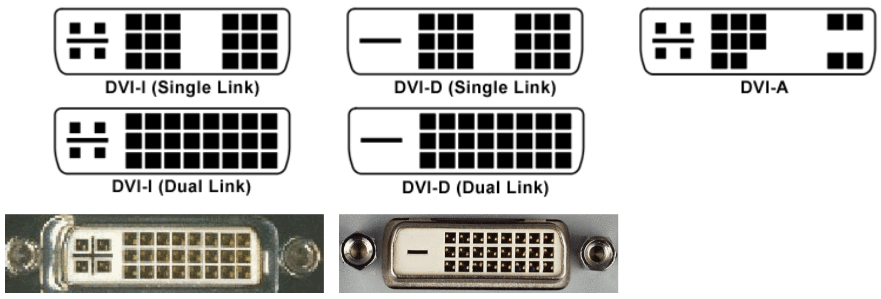


Figure 3.15: DVI interface[8]

3.6.3 HDMI, DisplayPort

HDMI and DisplayPort are fully digital interfaces, unlike the previous ones.

HDMI (High-Definition Multimedia Interface) is an interface for the transmission of uncompressed digital video and audio (as opposed to DVI, which transmits only both). The advantage is that HDMI uses the same specification for the video signal as DVI-D (it just adds audio to it), so the reductions between the two interfaces can be very simple, passive.

Remark

Why is it important that HDMI (and DisplayPort) delivers an *uncompressed* multimedia signal? Because on the one hand, compression reduces the amount of data that has to be transferred over the interface, but on the other hand it is unnecessarily time consuming. These interfaces therefore provide high throughput at which compression is not necessary, resulting in sufficient speeds (the transmitted video and audio can be reproduced in real time).

There are also different *versions* of HDMI. Today, the most common version is 1.4 from 2009, which is easily sufficient for Full HD resolutions up to 120 Hz: the theoretical throughput is 10.2 Gbit/s (real throughput is about a fifth lower). But if we wanted to transmit 4K video, it would only work at a lower frequency (30 Hz) – any faster movement in the video or game would be jerky.

Only the version of HDMI 2.0 from 2013, which has roughly twice the bitrate of version 1.4, can handle 4K resolution at a sufficient frequency (60 Hz). The version 2.1 from 2017 should handle resolutions up to 8K at 120 Hz.

It is recommended to use a maximum of 5 meters cable, but longer cables are also usable in certain circumstances (high quality strong signal transmitters in the device interfaces). For cable over 10 meters, there are active amplifiers in the terminals. There are also HDMI fibre optic cables that can be much longer.

DisplayPort is a digital interface designed for LCD monitors and other display peripherals. It transmits uncompressed digital content with support for protection with 128-bit AES encryption, and 8-channel audio. It differs from HDMI by, among other things, a more liberal license (and, of course, the shape of the port).

DisplayPort also exists in different *versions*. Version 1.0 (2006) has a theoretical throughput of 10.8 Gb/s, which is slightly more than dual-link DVI and slightly more than HDMI version 1.4.

[9]From: http://pcworld.cz/hardware/displayport-zabijak-dvi-3290

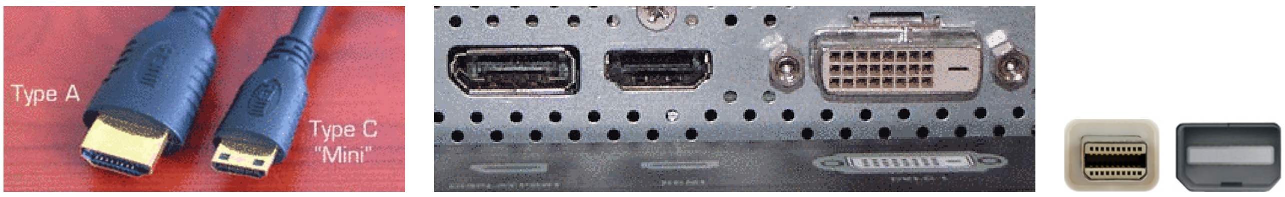


Figure 3.16: HDMI connectors A and C, then DisplayPort and its comparison with HDMI and DVI, on the right DisplayPort and Mini DisplayPort[9]

DisplayPort version 1.2 (2009) has twice the throughput (competes with the newer HDMI 2.0, 4K resolution can be transmitted at 60 Hz) and versions 1.3 and 1.4 even triple it (32.4 Gb/s). There is no direct difference in speed between 1.3 and 1.4, but improvements have been made, for example in sampling of transmitted audio, error correction, etc.

The cable can be up to 15 meters long.

The DisplayPort has a standard connector about the same size as HDMI, but seems a bit more solid and durable. In Figure 3.16 in the middle (page 40) we can see the sockets of three interfaces – DisplayPort, HDMI and DVI-D in turn. There is also a smaller version there – Mini DisplayPort; among other things, because it's used as the physical interface for Thunderbolt version 2.0 (see page 30). Figure 3.16 on the right shows the connector and socket for the Mini DisplayPort.

Remark

DisplayPort can handle the signal differently than DVI and HDMI (VGA, DVI and HDMI carry separate base colours in different wires, sync information in other wires, etc.). The transmission is more similar to transmissions in computer networks, it is called *micro-packet transmission*. This means that everything is transmitted in small data units (the color is not divided into wires according to the primary colors) without the need for a separate wire for the clock signal for synchronization (the timestamp is part of the packet). However, DisplayPort can emit DVI and HDMI signals.

There are DisplayPort reducers for DVI and HDMI (but the reverse direction is highly problematic), and despite the different signal transmission method, backward compatibility in one direction exists: some devices may be marked as *Dual-mode* (see logo on the right). These are devices capable of communicating via DisplayPort with both DisplayPort and HDMI or DVI signals (i.e. they can send a DVI or HDMI signal via DisplayPort). When plugged in, such a device (for example, a graphics card) detects the interface used on the other side (via a DisplayPort/HDMI or DisplayPort/DVI reducer) and adjusts the transmitted signal accordingly.

3.6.4 MHL

MHL (Mobile High-Definition Link) is a *signal interface* that uses HDMI at a lower level. It is mainly used to connect small mobile devices to TVs or projectors. There are several versions, of which MHL 3.0 can also transmit 4K video (but only at a lower frequency – 30 Hz), superMHL transmits up to 8K video at 120 Hz (depending on the type of cable).

A cable is used that has an HDMI connector on one side (TV/projector side) and a micro USB or USB-C (smartphone, etc.) on the other. The cable can be either passive (only if the mobile device "equips" MHL) or active with an MHL adapter (used if the mobile device does not include MHL functionality).

Today, the most common use of USB-C with Alternate mode MHL for USB 3.1 and higher, such a cable has USB-C on one end (mobile device) and HDMI on the other (TV).

3.7 Network Interfaces

In order for a device to communicate on a computer network, it must have a component that allows it to do so – a network card or an integrated version of one. This component has an interface through which it connects to the network.

✎ **Ethernet a Wi-fi:** If we connect to the network via a cable, it is usually an Ethernet network with an RJ-45 connector/socket (the more correct term is 8p8c). If it is a wireless network (Wi-fi, mobile networks), then the physical interface is the antenna, which is often (especially in mobile devices) integrated. Furthermore, if we are connecting an ADSL/VDSL modem, we need a cable with RJ-11 (telephone) connectors. The signal interface in all cases is provided by *network protocols*.

✎ **Bluetooth:** It is a well-known interface for wirelessly connecting devices over short distances. There are several power classes – the higher the power, the greater the range (class 3 means up to 1 m range, class 1 up to 100 m). Throughput is not staggering, typically in the units of Mbit/s, depending on distance and power (class), and version (version 4.0 up to 24 Mbit/s). In any case, Bluetooth is not designed to transmit large quanta of data. This interface is mainly encountered in small mobile devices and some controllers.

Compared to IrDA (infrared transmission), Bluetooth has the advantage of omnidirectionality (we don't have to aim directly at the target) and also of widespread use.

There is also a variant designed for the Internet of Things: Bluetooth LE (BT Smart) with reduced power consumption.

✎ **UPnP** (Universal Plug-and-Play) is a technology and set of protocols designed to make the existence of devices on a network as simple as possible. The idea is that a device should be able to communicate as soon as possible after connecting: to get an address, to orient itself on the network (to find out what other devices are on the network and how they can be used), to present itself on the network, to detect and respond to network events. All this regardless of what kind of network it is (Ethernet, Wi-fi, etc.). UPnP is the basis for the functioning of other services, especially in the context of multimedia, audio-video services.

UPnP is mainly used in households, but there is also a bit of a problem with the fact that devices are practically unauthenticated and the service is relatively easy to abuse. It is recommended to disable UPnP support on the router for security reasons, as it can be exploited for unauthorized access to the network from the outside (using the UPnP IGD protocol).

✎ **DLNA** (Digital Living Network Alliance) is a technology that enables sharing of data from multimedia devices (based on UPnP), and is primarily backed by Sony. It is found in laptops, small mobile devices, TVs, media players, game consoles, printers and other devices, such a device should be marked as "DLNA Certified".

We refer to the source of multimedia data as a DLNA server (e.g. NAS server, smartphone, game console), the target is a DLNA client (e.g. a TV playing video from a smartphone). The DLNA servers and clients must be running UPnP, but they do not have to be running on the router at all. An important prerequisite is that the multimedia content must be "understandable" to the DLNA client, the client must be able to handle the format, codecs etc., the DLNA server only sends the data. So, for example, if we buy a DLNA-enabled TV, we should also be interested in what it can actually play. While a DLNA server can be configured to convert the data to another format before sending it, it doesn't have to (and it doesn't have to), the DLNA specification doesn't require it.

Miracast, Chromecast and AirPlay are competitors to DLNA. Miracast is designed to mirror the screen content of the transmitting device (i.e. the computational load is more on the transmitter) and works only over a wireless network, so we have to take into account a slight delay in the image (but on the other hand there is no problem with displaying subtitles for video). Google's Chromecast can both mirror screen content and stream multimedia content (video, games, depending on the support in apps). AirPlay comes from Apple and serves similar purposes to Chromecast.

Chapter 4

Case and Motherboard

Quick preview: This chapter is devoted to the most important part of the computer, the motherboard (mainboard). We will discuss the physical structure of the motherboard, the most important components, interfaces, integrated devices, cooling and problems related to the layout of components on the motherboard. At the beginning of the chapter, we will look at what the motherboard and most of the components actually reside in.

Keywords: Case, chassi, tower, rack, node, blade server, mainframe, supercomputer, HTPC, PCB, Form Factor, front panel, back panel, bus, PCI, PCI Express (PCIe), AGP, chipset, North-South Bridge Design, One-Chip Design, FSB, DMI, HyperTransport (HT), QuickPath Interconnect (QPI), SoC, jumper.

Objectives: After studying this chapter, you will know what the motherboard is used for, what you can find on it, what the buses on the motherboard are and how they work, how the chipset looks and works, how the chipset affects the features of the motherboard and the whole computer.

4.1 Computer Case

A computer case (chassi; the plural is chassis) is not just an aesthetic matter. The material it is made of, its size, shape, and internal layout determine the functionality of the computer (its ability to hold including air circulation). If the material is too thin or too flexible, it may transmit vibrations (from fans, spinning hard disk or optical drive) and thus contribute to the amplification of the sound produced by the computer.

There are the following types of computer cases:

- desktop, tower, minitower, big tower, small form factor – for desktop computers,
- laptop chassi,
- server cases:
  - tower – usually cheaper machines,
  - server intended to rack, "rack-mount" – usually multiple server units with a unified size are placed in the rack, see below,

  - blade server,
  - mainframe (hall) for controlling complex operations, much intertwined with the following category,
  - supercomputer – a high-performance computer in the form of a cluster (i.e. many high-performance computers connected by a high-speed network, usually in a single hall or building, interconnected by hardware and software so that they function virtually as a single machine),
- enclosures for HTPC (Home Theatre PC, small home servers usually used as storage for movies, photos and other family files) – small size, quiet operation is assumed, i.e. with passive cooling,
- embedded (devices that don't seem like computers themselves but have a computer embedded in them, such as DVD players or network devices),
- smaller formats (smartphones, tablets,. . .).

✎ *Rack* contains so called *nodes* (servers, switches, routers, hardware firewalls, etc.) with height in multiples of 1.75 in = 4.45 cm (referred to as 1U, 2U, . . . , depending on the multiple of the specified height), the maximum height of the entire rack is 42U (but usually we encounter rather smaller cabinets).

Each node contains the motherboard, processors, memory, etc. The rack interconnects the nodes, providing power management, cooling and communication interfaces. This means that when we install a node in a rack, we connect it to the rack's power and cooling systems and, of course, to the network interfaces. In particular, to make the cooling efficient enough, we should place the nodes in such a way that the air can flow well if it is enclosed.

Some towers can also be mounted in a rack.

✎ *Blade System* represents the next level of integration. The whole system is integrated on a single (base) board enclosed in a case, there are no cables inside, multiple such cases are usually placed in a single *blade chassi*, which is often a rack node.

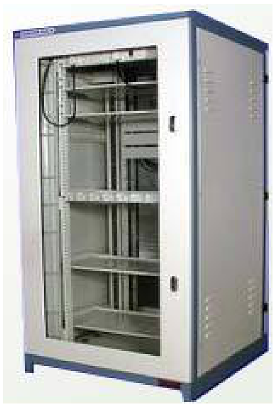


Figure 4.1: Rack[1]

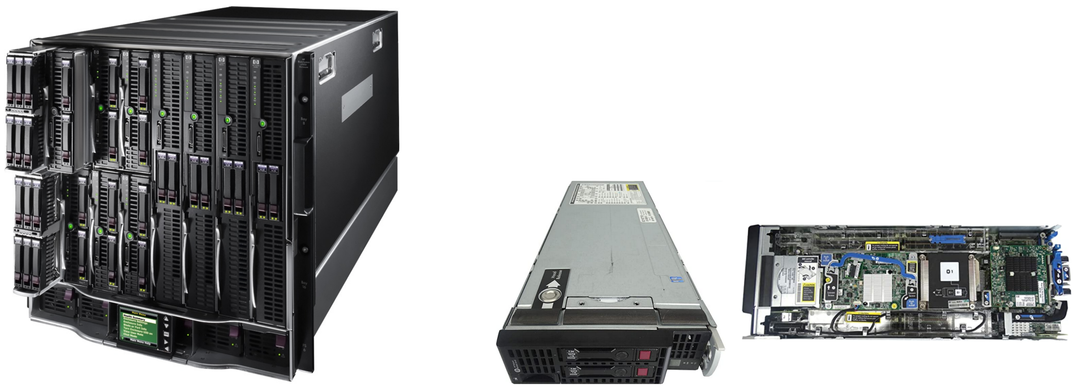


Figure 4.2: HP Blade chassi c7000 10U and one of the blade servers[2]

[1]From: http://www.shaktigroupindia.net/etsi-telecom-server-racks.html
[2]From: https://tadviser.com/index.php/Product:HP_BladeSystem_c7000, https://www.amazon.com/

A typical feature of blades is incompatibility: if we have a blade chassi from a particular manufacturer, we have to buy blades from the same manufacturer. On the other hand, the advantage is a certain compactness and also usually easier deployment and management, lower power consumption.

4.2 Motherboard

Motherboard (mainboard) is the basic component of a computer, laptop and in fact any electronics. It provides universal data and power interfaces and physical storage for individual components. It is physically implemented using a multi-layer printed circuit board equipped with many electronic circuits and connectors.

4.2.1 PCB

PCB (Printed Circuit Board) is an essential part of every motherboard. Various electronic components and connectors are placed on the PCB. Expansion cards (adapters), such as graphics cards, have a similar structure.

The usual PCB structure is outlined in Figure 4.3. The PCB is divided into layers, some of which appear more than once in the structure. The thickness of the board depends on the number of layers, the order of which can be seen in the figure. The meaning of each layer is as follows:

- Surface protective layer (top and bottom) – protects the board from external influences, protective film, spray,
- *Signal Layer* – provides logical communication, consists of a thin layer of conductive material, usually copper,
- *Insulating layer* – separates signal and ground layer, usually made of Preprog (glass-carbon reinforcement impregnated with polymer matrix),
- *Power-ground layer* – provides grounding, power and heat dissipation, mostly copper,
- *Core* – provides the mechanical properties of the board.

Surface protective layer
Signal Layer 1
Insulating layer
Power-ground layer 1
Core
Power-ground layer 2
Insulating layer
Signal Layer 2
Surface protective layer

Figure 4.3: Structure of a motherboard PCB

4.2.2 Form Factor

Many different types of motherboards are used, the differences are mainly in the following features:

- size of the board,
- size, mounting method and type of power connectors,

- manufacturer,
- the interfaces, sockets, etc., i.e. what processors are supported, used chipset, how many cards we can connect, etc.,
- overclocking options (also related to the motherboard BIOS/UEFI) and the way the components are being cooled.

The most common motherboard types (*Form Factor*) can be found in the 4.1 table. They differ mainly in size, cooling method, and also in the arrangement, size and mounting method of power connectors, sockets, chipset, etc. Manufacturers of smaller devices (but also laptops) often "personalize" the shape of the motherboard in terms of size, shape and layout of components – sometimes such a board does not even remotely resemble a rectangle.

Standard	**Size (mm)**	**Standard**	**Size (mm)**
ATX (Intel)	305×244	mini-ITX (VIA)	170×170
micro-ATX	244×244	nano-ITX	120×120
flex-ATX	229×191	pico-ITX	100×72
mini-ATX	150×150	mobile-ITX	75×45
BTX (Intel)	325×266	DTX (AMD)	244×203
mini-STX	140×147	mini-DTX	203×170

Table 4.1: The most commonly used motherboard Form Factors

The most commonly used formats are the following:

- *ATX* (Intel, 1995, since 1999) – popular for desktops, 7 slots with 20mm spacing, the variant *micro-ATX* is reduced by about 25 %, has fewer slots for expansion cards,
- *BTX* (Intel, 2004) – differently cooled components (changing the way air circulates), didn't catch on,
- *ITX* (VIA, 2001) – smaller, components in portrait orientation, this format is not directly encountered today, the variant *mini-ITX* is often used in small laptops and small desktops, the processor is (not always) soldered to the motherboard,
- the motherboard for the Raspberry Pi, for example, has dimensions between pico-ITX and mobile-ITX,
- *STX* is the newest base format, *mini-STX* is used, occupying about 3/4 of the size of a mini-ITX; on these motherboards we find common processor sockets, but for example the memory modules are shorter (laptop modules).

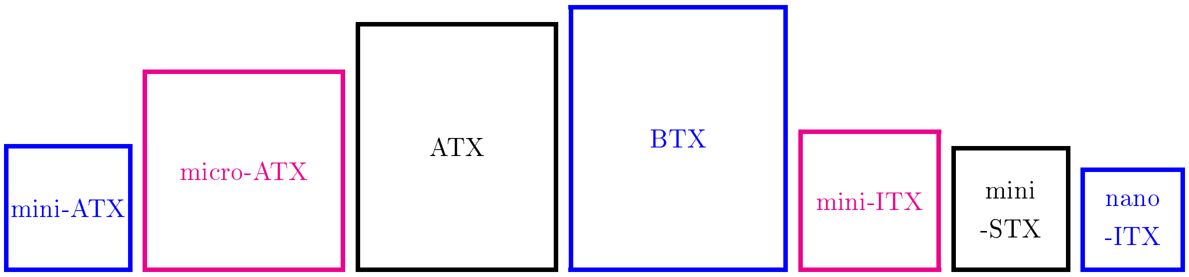


Figure 4.4: Comparison of selected motherboard form factors

Servers mainly use EATX (Extended ATX – 305×330) for rack servers or WTX (355.6×425.4) for tower servers. They include more CPU sockets, SATA ports and memory slots.

For *embedded devices* (set-top boxes, various consumer electronics, routers and other network devices, ATMs, computers in cars, etc.) and for small mobile devices, either some of the above mentioned boards are used (e.g. derived from ITX – more for larger embedded), or form factors specially designed for these purposes – ETX (95×114), COM Express (several variants, the Compact variant is 95×95), PCI/104 (again several variants) and others.

Remark

In fact, the term "form factor" is used in other meanings as well. For example, for hard drives, this is used to distinguish different sizes – a hard drive with a Form Factor of 3.5", etc.

Additional information

- https://smallformfactor.net/articles/editorial/stx-changed-view-sff/
- http://en.wikipedia.org/wiki/Computer_form_factor
- https://www.miraclepwb.com/pcb-blog/6-layers-pcb-board/
- https://www.protoexpress.com/blog/trace-current-capacity-pcb-design/

4.2.3 Interfaces

Interfaces are intended to extend the functionality and/or performance of motherboards with additional options. The purpose is to clearly define the data transfer and power supply method between the motherboard and the expansion card. As part of the interfaces we can also consider the I/O back panel or front panel.

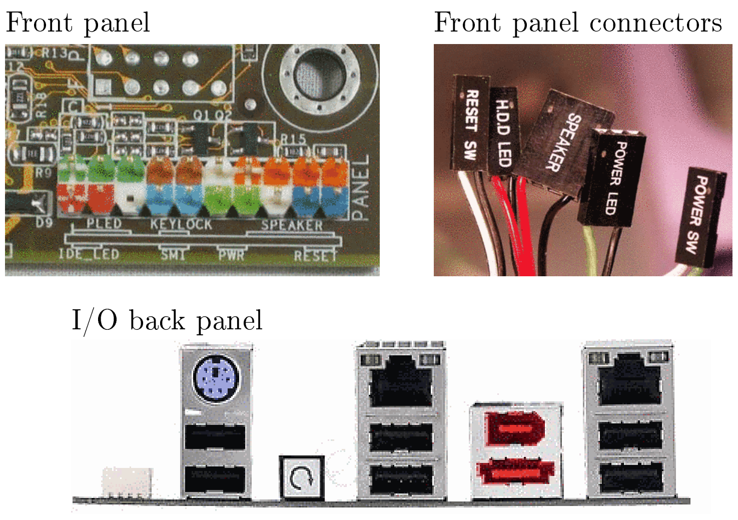


Figure 4.5: Front panel and I/O back panel[3]

Front panel is a part of the motherboard that usually adjoins the inside of the front of the computer (or is in some front corner) and its purpose is to connect the motherboard to everything

[3]From: http://infoarchena.blogspot.com/2011/04/front-panel-ii.html

that is to be brought out of the computer case on the front. In Figure 4.5 we can see it on the top left. Next to it is a set of connectors to plug in (for various indicator LEDs – for example, to indicate hard drive activity, but also to the power or reset button).

Back panel (also I/O panel, in Figure 4.5 below) faces the back of the computer and there are sockets leading from integrated components – USB jacks, sockets for VGA, DVI, HDMI (if we use integrated graphics), audio, network, etc.

4.3 Bus

Buses are an important part of the motherboard. Buses are *communication lines*, physically realized by printed circuits (in fact, we can talk about e.g. USB bus, it is also a communication line). Expansion cards or external devices are connected to the buses via sockets, other buses are used for communication between the processor and the chipset, memory (memory bus), etc., the buses are also inside the processor.

A bus is a communication path that needs to be "somehow" accessed. In the previous chapter, we were introduced to USB, SATA, and other interfaces for which connectors (plug) and sockets (jack) were defined. The term bus generalizes these interfaces (even USB is really a bus, as mentioned above), and as far as buses on a motherboard are concerned, sockets are usually called slot or socket. *Slot* has pins arranged in one row, *socket* in matrix form.

Communication over the bus takes place at regular intervals, which are determined by a timer. A *timer* is a circuit that generates a pulse at regular intervals that controls the operation of a component or its communication over the bus. The rate of pulse generation is referred to as the frequency – higher frequency means a faster timer and shorter time intervals.

Over the bus we can communicate either *parallel* or *serial* – we already know these terms from the previous section. Most of the current buses work in serial (PCI Express, USB, SATA, SAS, FireWire, Thunderbolt, RS-232, Lightening), parallel buses are hardly used anymore (PCI, ISA, EISA, PATA, SCSI, LPT).

Buses can be also divided into

- *synchronous* – communication between components connected to the bus is controlled by a common timer, all operating at the same frequency,
- *asynchronous* – the components have their own timers, but in order for the communication to work they have to "agree" (within the communication initialization the communication parameters including timing are agreed upon).

The parallel bus is always synchronous. Serial communication can be synchronous or asynchronous, for example USB is a serial bus that can operate the both synchronously or asynchronously.

4.3.1 ISA, EISA

ISA (Industry Standard Architecture, from IBM) is a rather historical thing. It was used to connect internal expansion cards between 1981 and 1993. It's an 8 or 16-bit bus (it's all about its width – how many bits can be transferred at a time) with a refresh rate of 8 MHz and a parallel data transfer method. The transfer rate is 8 or 16 MB/s depending on the bus width.

EISA (Extended ISA) is a descendant of the ISA bus, now also historical. It is a 32-bit bus that allowed transfers at a theoretical speed of 33 MB/s, but the real transfer rate was closer to 20 MB/s.

4.3.2 PCI

PCI (Peripheral Component Interconnect) is an interface introduced in 1993 that has survived in a modified form to this day. There is also a smaller version for portable computers (Mini PCI).

The connectors on the cards are physically modified to limit the possibility of damaging the expansion cards if the wrong slot is used (*keying*). PCI was also the first bus to come with support for plug-and-play technology.

Features:

- 32/64bit bus width, 33/64 Mhz refresh rate with parallel data transfer method,
- theoretical transfer rate depending on version 133 (32bit, 33 Mhz), or 266 (64bit, 33 Mhz), or 533 (64bit, 66 Mhz) MB/s,
- supply voltage ranges 3.3–5 V, higher PCI versions work with lower voltages; the voltage used should not be confused because cards requiring 3.3 V have a notch (keying) in a different place underneath than cards requiring 5 V (or if the notch is in both places, the card can handle both).

The PCI bus can still be found on some motherboards today (it's cheap), so if it fits on the motherboard, we can get some sound and network cards to fit into this slot.

4.3.3 AGP

Before we get to the most widely used bus type today, PCI Express (PCIe), let's take a brief look at its predecessor, the AGP bus.

The AGP (Advanced Graphics Port or Accelerated Graphics Port) bus was created in 1997 as an interface to replace the shortcomings of PCI for graphics cards (a specialized modification of PCI). At that time, only ISA, EISA and PCI buses were available (PCIe bus did not exist yet), so there was no place to plug in more powerful graphics cards.

So the AGP bus solved the problem of connecting graphics cards. However, the problem was the existence of several variants of AGP slots, which differed, among other things, in keying (i.e. there were notches on the cards in different places for different variants). A user buying a new card always had to know which type of AGP slot he wanted to use it in.

Nowadays you can find AGP on many old motherboards, on newer ones it has already been displaced by the PCIe bus.

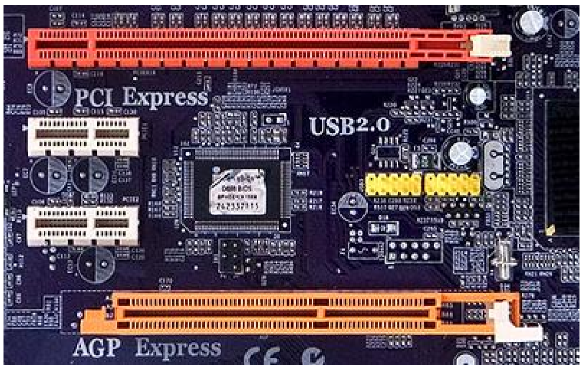


Figure 4.6: Comparison of PCIe and AGP bus slots[4]

The AGP bus slot is a bit reminiscent of the PCIe×16 slots. At first glance, however, you can recognize them by their location on the motherboard – the AGP slot is always more offset from the back of the board. In the picture 4.6 we can see a comparison of AGP and PCIe×16 slots. The top red one is PCIe, the bottom orange one is AGP, and the left picture is the back of the motherboard. The AGP slot starts from the left a few centimeters farther than PCIe.

[4]From: http://ixbtlabs.com/articles3/video/guide-p3.html

4.3.4 PCI Express

PCI Express (PCI-E, PCIe; 2004) is currently the most widely used bus interface for expansion cards. The basic idea is to move from whole-word parallel transfer (as in PCI) to serial communication.

The communication bandwidth is divided into several channels, which we call *links* (PCI Express Lines). A physical interface can have 1, 4, 8, 16, 32 lines, we talk about interface speed ×1, ×4, etc. The most common ones are ×1 for sound and network cards, and ×16 for graphics cards.

There are a certain number of lines (typically 16, 32 or 64) available on the motherboard, and these lines are then divided into individual *slots* on the motherboard. In the image 4.7 we can see four PCIe slots (and one PCI), and the lines are divided into the given four slots.

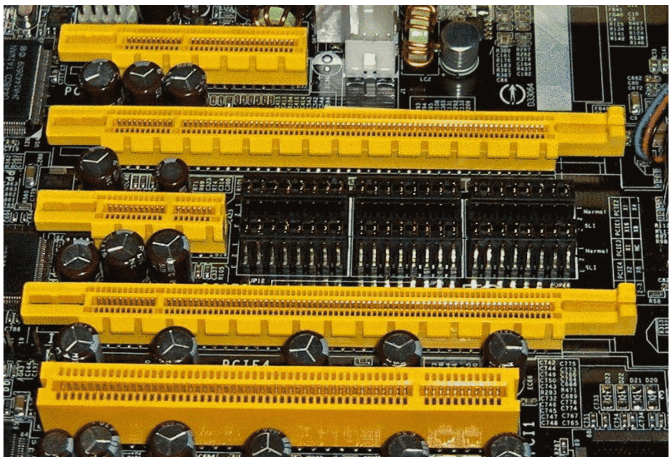


Figure 4.7: Bus slots, from top to bottom: PCIe×4, PCIe×16, PCIe×1, PCIe×16, PCI[5]

One channel (line) consists of a total of four conductors, two for each direction. Within a channel/line, only serial communication is carried out; if there are multiple channels, each of them carries independent communication (the transmitted word is not shared between multiple channels and does not need to be synchronized at the interface level).

PCIe slots with different number of lines have *different length* (we can see it in Figure 4.7), we usually put the card in the slot with the length we need. However, we can also put the card in a "longer" slot, then only the lines that the card uses are activated.

Theoretically, it could also work the other way around (plug the longer card into the shorter slot) – but this is only possible if the short slot is modified (it has a groove at the end to allow the longer card to be physically inserted, then only the part of the pins that is connected in the slot is activated.

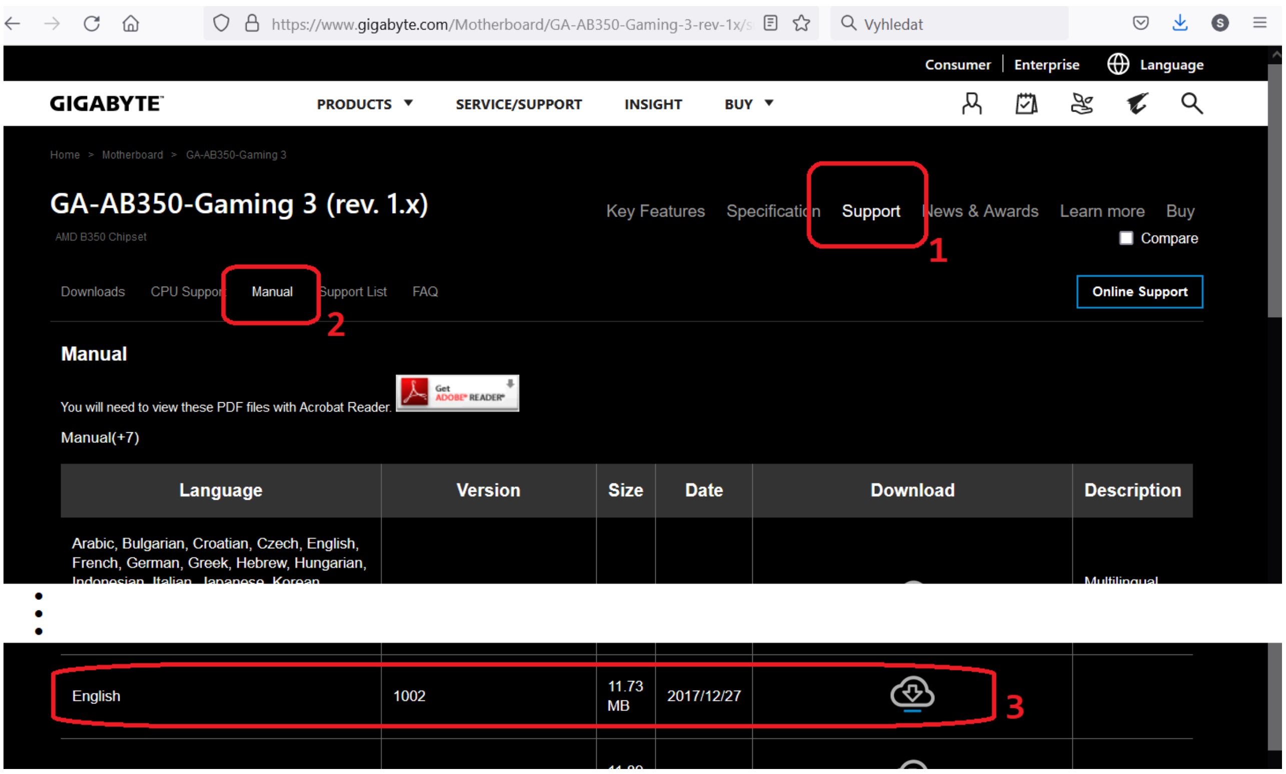
This principle also works inside the slot: some ×16 slots actually have only 8 active lines instead of 16, so if we insert a PCIe×16 card into such a slot (which is of course possible), it only communicates over 8 lines and the card works slower than in a full-fledged PCIe×16 slot (by the way, some cards that look like ×16 only use 8 lines anyway).

Example

Let's take a look at one very common motherboard, Gigabyte GA-AB350-GAMING 3. The manual of this motherboard can be obtained (as with others) on the manufacturer's website, just ask Google (enter the name of the board). In our case, the motherboard details are at https://www.gigabyte.com/Motherboard/GA-AB350-Gaming-3-rev-1x

In the menu select *Support*, below *Manual* (each manufacturer has it on their website a little differently), in the table below we can find the manual in English.

[5] From: https://en.wikipedia.org/wiki/PCI_Express


https://www.gigabyte.com/Motherboard/GA-AB350-Gaming-3-rev-1x/s

Vyhledat

Consumer | Enterprise | Language

GIGABYTE PRODUCTS SERVICE/SUPPORT INSIGHT BUY

Home > Motherboard > GA-AB350-Gaming 3

GA-AB350-Gaming 3 (rev. 1.x)

AMD B350 Chipset

Key Features Specification Support News & Awards Learn more Buy

1

Compare

Downloads CPU Support Manual Support List FAQ

2

Online Support

Manual

You will need to view these PDF files with Acrobat Reader. Get ADOBE® READER®

Manual(+7)

Language	Version	Size	Date	Download	Description
Arabic, Bulgarian, Croatian, Czech, English, French, German, Greek, Hebrew, Hungarian, Indonesian, Italian, Japanese, Korean					Multilingual
English	1002	11.73 MB	2017/12/27		

3

Figure 4.8: Gigabyte GA-AB350-GAMING 3 motherboard information website

Part of the picture in Figure 4.9 is from the mentioned manual. From the naming of the PCIe slots we can deduce how many active lines we actually have in each slot.

The short slots PCIEX1_1 and PCIEX1_2 can be expected to have only one line active for each direction, but what about the long slots? So PCIEX16 has 16 active lines for each direction, though PCEX4 looks like x16 but has only 4 active lines, and the lower PCIEX1_3 looks like x16 but has only one active line for each direction.

So it definitely makes sense to download and browse the motherboard manual.

Thus, if we want to connect a graphics card, we use the slot that has the most active lines. This is usually the slot closest to the processor.

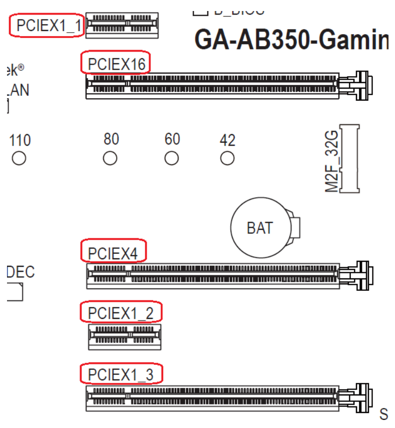


Figure 4.9: PCIe slots on the Gigabyte motherboard

Throughput depends on both the number of lines and the version – there are various *versions* for PCIe as well:

- Version 1.x: throughput of 250 MB/s per link and per direction, i.e. on a motherboard with 32 active links this would be 8 GB/s for each direction shared by all PCIe connected components. If the graphics card is plugged into a slot with 16 lines (×16), it has 4 GB/s available.
- Version 2.x: 500 MB/s throughput per link, double,
- Version 3.x: throughput of 985 MB/s per line, again double the previous version,
- Version 4.x: throughput of 2 GB/s (1969 MB).

Some components use higher speed in one direction (e.g. a graphics card), others in both directions (e.g. an SSD communicating via PCIe).

Remark

Considering that the higher version doubles the throughput, this means that 8 lines of the higher version will provide roughly the same throughput as 16 lines of the lower version.

The cards are usually also *powered* through the buses. Cheaper cards manage on power directly from the PCIe slot, but typically higher-level graphics cards require additional power.

Remark

In the chapter on interfaces, it states that the interfaces to mSATA and M.2 storage media connect to PCI Express ×4, and Thunderbolt actually does the same thing. So the PCI Express bus is not necessarily just for connecting expansion cards, it can be part of the communication path of other interfaces.

4.4 Chipset

Chipset is a set of integrated circuits that provide communication between the individual components (processor, memory, expansion cards, BIOS, various interfaces and connectors), it is actually a kind of communication center. It also contains some controllers, including memory module controller, USB controllers, SATA controller.

The chipset determines how the individual components will be logically and physically connected to each other. The quality of the chipset largely depends on the quality of the motherboard itself, as this is where the main communication paths of the components lead.

The most famous chipset manufacturers are Intel, AMD, VIA, SiS, nVidia (nForce chipsets). These chipsets are then used as they are by various motherboard manufacturers.

There are two basic concepts in use today – North-South Bridge Design and One Chip Design. The difference between the two is the number of chips (generally parts) that make up the chipset. There are also modifications of these, for example, the One Chip Design concept has evolved into the most widely used chips for mobile devices today – SoCs.

4.4.1 North-South Bridge Design

The *North-South Bridge Design* concept is used in most computers, laptops and servers today. The chipset is divided into two main chips, called North Bridge and South Bridge, and the motherboard manufacturer can add some additional chips to enrich the chipset.

- *North Bridge* handles high-speed communication with the graphics card, RAM and CPU, it is closer to the CPU, *South Bridge*
- *South Bridge* handles communication with other components: expansion cards, disks, external interface controllers and various drives, and mediates BIOS services. It determines the maximum transfer rate of disks, USB, etc. It is usually located closer to the expansion card slots.

The term *I/O Controller Hub* is also used for the south bridge.

General chipset diagram with FSB bus is on Figure 4.10, on Figure 4.11 we can see a specific chipset from Intel for Core 2 Extreme processor, Figure 4.12 shows a chipset diagram for motherboards with processors for the AM3 socket (by AMD).

These diagrams can usually be found on the internet, just type e.g. “chipset block diagram” (or instead of “chipset” directly the name of the specific chipset) and choose image search.

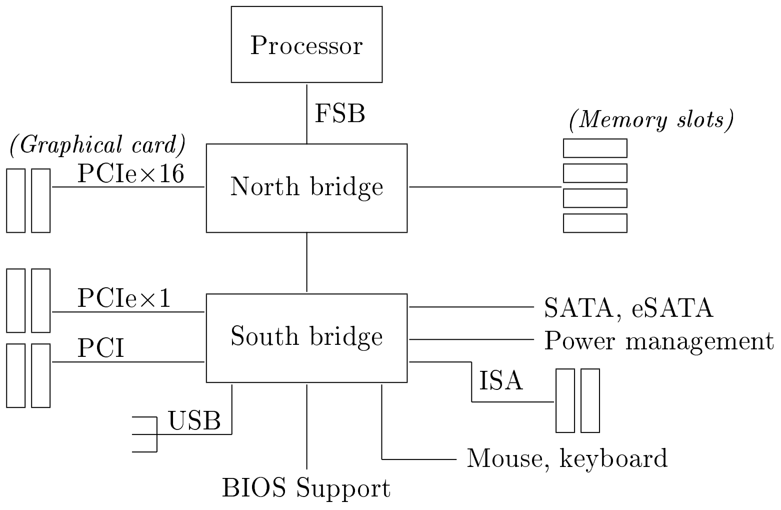


Figure 4.10: General chipset diagram with FSB bus

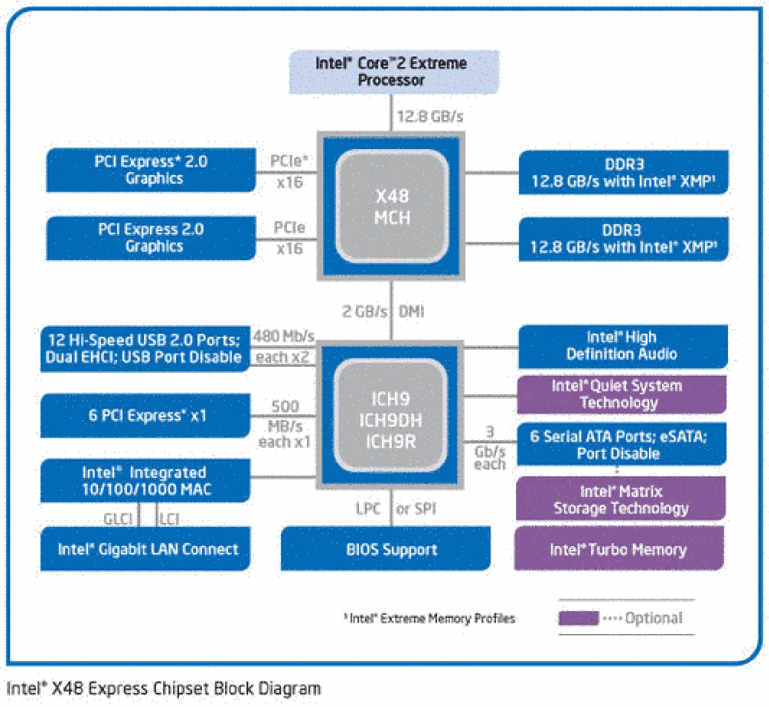

Intel® Core™2 Extreme Processor
12.8 GB/s
PCI Express* 2.0 Graphics
PCIe* x16
PCI Express 2.0 Graphics
PCIe x16
X48 MCH
DDR3 12.8 GB/s with Intel® XMP¹
DDR3 12.8 GB/s with Intel® XMP¹
2 GB/s DMI
12 Hi-Speed USB 2.0 Ports; Dual EHCI; USB Port Disable
480 Mb/s each x2
6 PCI Express* x1
500 MB/s each x1
Intel® Integrated 10/100/1000 MAC
GLCI
LCI
Intel® Gigabit LAN Connect
ICH9 ICH9DH ICH9R
LPC or SPI
BIOS Support
Intel® High Definition Audio
Intel® Quiet System Technology
3 Gb/s each
6 Serial ATA Ports; eSATA; Port Disable
Intel® Matrix Storage Technology
Intel® Turbo Memory
¹Intel® Extreme Memory Profiles
Optional
Intel® X48 Express Chipset Block Diagram

Figure 4.11: Chipset X48 Express diagram for the Intel Core 2 Extreme processor[6]

Recently, there have been tendencies to integrate some of the functions of the north bridge, or even the entire north bridge, into the CPU chip, mainly to speed up communication with RAM, save

[6]From: http://ixbtlabs.com/articles3/mainboard/nvidia-nforce-790i-intel-x48-chipsets-p2.html

space on the motherboard, and make cooling easier.

The function of the *memory controller* (i.e. the component that mediates the communication between the processor and the RAM modules), which used to reside in the north bridge, has moved to the processor case. AMD started this trend with 64-bit processors, and after a few years Intel joined the trend.

So the memory controller was the first "cover". The next stage is to move the entire north bridge into the processor case – the module acting as the north bridge in the processor is called *System Agent* (at Intel).

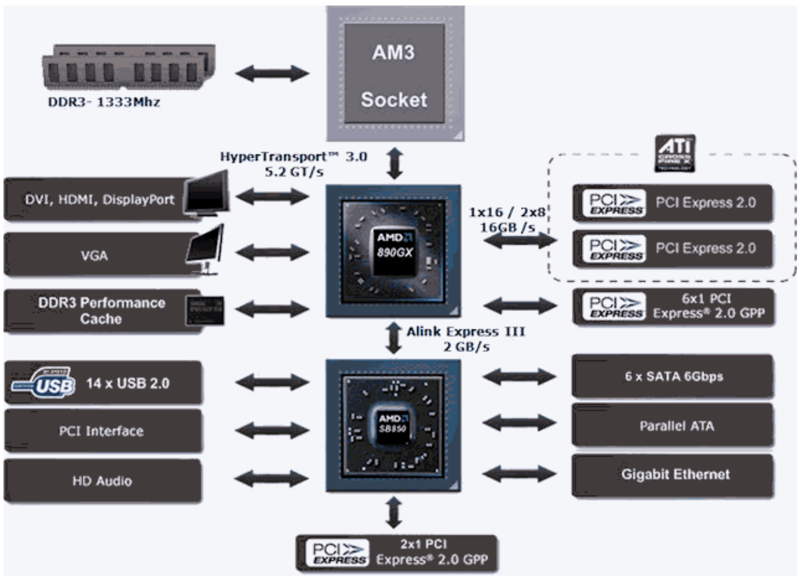


Figure 4.12: AMD chipset schema for AM3 processors[7]

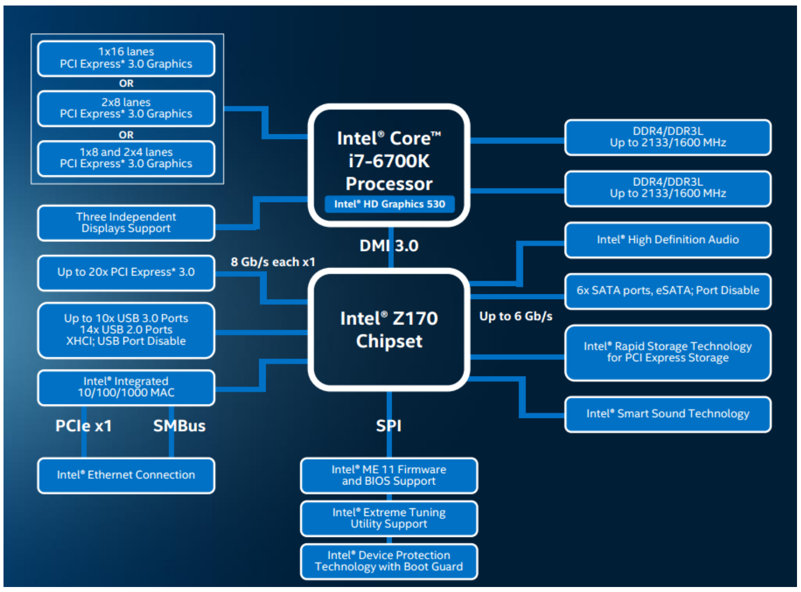

1x16 lanes PCI Express* 3.0 Graphics
OR
2x8 lanes PCI Express* 3.0 Graphics
OR
1x8 and 2x4 lanes PCI Express* 3.0 Graphics
Three Independent Displays Support
Up to 20x PCI Express* 3.0
Up to 10x USB 3.0 Ports 14x USB 2.0 Ports XHCI; USB Port Disable
Intel® Integrated 10/100/1000 MAC
PCIe x1
SMBus
Intel® Ethernet Connection
Intel® Core™ i7-6700K Processor
Intel® HD Graphics 530
DMI 3.0
8 Gb/s each x1
Intel® Z170 Chipset
Up to 6 Gb/s
SPI
Intel® ME 11 Firmware and BIOS Support
Intel® Extreme Tuning Utility Support
Intel® Device Protection Technology with Boot Guard
DDR4/DDR3L Up to 2133/1600 MHz
DDR4/DDR3L Up to 2133/1600 MHz
Intel® High Definition Audio
6x SATA ports, eSATA; Port Disable
Intel® Rapid Storage Technology for PCI Express Storage
Intel® Smart Sound Technology

Figure 4.13: Chipset diagram for Intel Skylake processors[8]

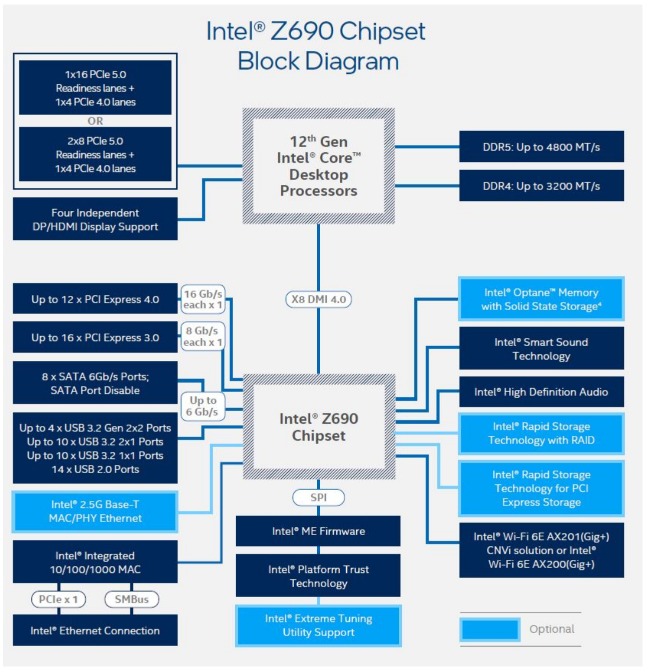
[7]From: http://www.abclinuxu.cz/blog/virtualizace/2010/8/hardware-3-vyber-chipsetu-a-zakladni-desky

[8]From: http://www.anandtech.com/show/9483/intel-skylake-review-6700k-6600k-ddr4-ddr3-ipc-6th-generation/5

When comparing Figures 4.11, 4.12, and 4.13, we find the following:

- Figure 4.11 fully corresponds to the basic scheme from Figure 4.10 (i.e. in the center of the scheme there are three large chips – above the processor, below it the north bridge MCH X48 and even lower the south bridge ICH9, the processor is directly connected only to the north bridge, to the north bridge are connected from the right modules of RAM – DDR3, from the left slots for graphics card – PCI Express 2.0),
- Figure 4.12 still assumes that we have a chipset in two large chips outside the processor, but the memories (also DDR3) are already connected to the processor, not to the north bridge (so part of the north bridge – the memory controller – moves to the processor),
- in Figure 4.13, the entire north bridge is already integrated into the CPU case, so everything that was connected to the north bridge in the 4.11 image is now connected to the CPU.

Figure 4.14 shows a block diagram of the chipset where the entire north bridge is integrated into the processor. So if we haven't inserted the processor into the socket yet, then this board doesn't even have a north bridge yet.



Intel® Z690 Chipset Block Diagram
1x16 PCIe 5.0 Readiness lanes + 1x4 PCIe 4.0 lanes
OR
2x8 PCIe 5.0 Readiness lanes + 1x4 PCIe 4.0 lanes
Four Independent DP/HDMI Display Support
12th Gen Intel® Core™ Desktop Processors
DDR5: Up to 4800 MT/s
DDR4: Up to 3200 MT/s
X8 DMI 4.0
Up to 12 x PCI Express 4.0
16 Gb/s each x 1
Up to 16 x PCI Express 3.0
8 Gb/s each x 1
8 x SATA 6Gb/s Ports; SATA Port Disable
Up to 6 Gb/s
Up to 4 x USB 3.2 Gen 2x2 Ports Up to 10 x USB 3.2 2x1 Ports Up to 10 x USB 3.2 1x1 Ports 14 x USB 2.0 Ports
Intel® 2.5G Base-T MAC/PHY Ethernet
Intel® Integrated 10/100/1000 MAC
PCIe x 1
SMBus
Intel® Ethernet Connection
Intel® Z690 Chipset
SPI
Intel® ME Firmware
Intel® Platform Trust Technology
Intel® Extreme Tuning Utility Support
Intel® Optane™ Memory with Solid State Storage⁴
Intel® Smart Sound Technology
Intel® High Definition Audio
Intel® Rapid Storage Technology with RAID
Intel® Rapid Storage Technology for PCI Express Storage
Intel® Wi-Fi 6E AX201(Gig+) CNVi solution or Intel® Wi-Fi 6E AX200(Gig+)
Optional

Figure 4.14: Intel Z690 chipset block diagram[9]

4.4.2 Chipset Buses

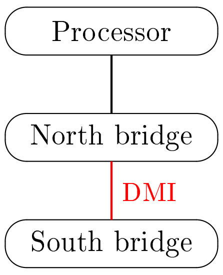


Figure 4.15: DMI bus

In the above pictures of the chipsets, we will have noticed what is connected to what and how these connections are labeled. The processor and chipset and the individual parts of the chipset communicate with each other using special buses.

There is a higher speed connection between the north and south bridges. On Intel architectures this bus is called *DMI* (Direct Media Interface) and is a modification of the PCIe×4 bus, or we can find PCIe×4 directly here. On AMD motherboards it is ALink Express (based on PCIe×4) or directly PCIe×4.

[9]From: https://www.servethehome.com/intel-12th-gen-core-alder-lake-details-at-intel-innovation-2021/intel-z690-chipset-block-diagram/

The bus that provides the communication *between the north bridge and the processor* is very important. Its speed has a major impact on the speed of the whole system. Between the chipset and the processor we can encounter mainly the following buses:

- *FSB* (Front Side Bus) is the original designation of this bus in its classic form, where the entire north bridge is in a separate chip,
- *HT* (HyperTransport) – a bus on AMD architectures (from K8 architecture onwards), the memory controller has been moved from the north bridge to the processor, which means increasing the throughput of this bus (and thus the speed) simply by transferring less data over it, plus the processor can communicate with the RAM very quickly when it doesn't have to be via another chip, *QPI* (QuickPath Interconnect, older name CSI) is an Intel bus for more powerful builds, the memory controller has been transferred to the processor (i.e. similar to AMD HT),
- *DMI* – if the north bridge is integrated into the processor.

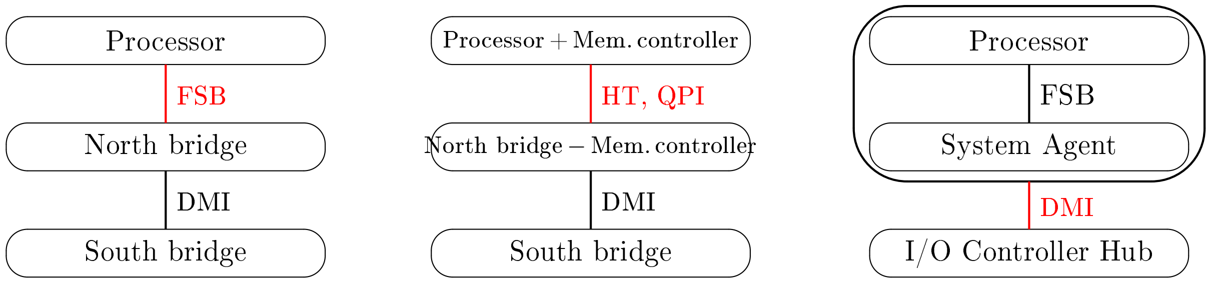


Figure 4.16: Bus between the north bridge (or chipset) and the processor

Figure 4.16 illustrates the development of this bus (including the shifting of relevant functionalities).

4.4.3 One Chip Design

In the *One Chip Design* concept, both bridges are integrated into one chip. It is a solution, that seeks to minimise the cost of manufacturing the device or the amount of space taken up and was originally designed for low-cost or smaller devices.

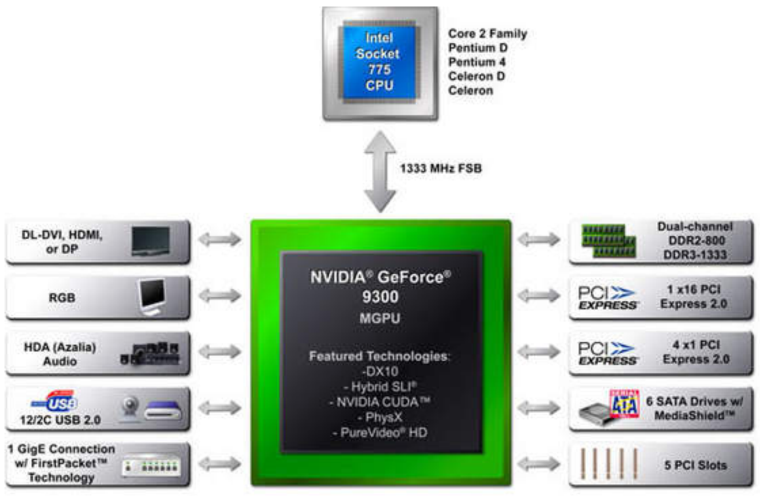


Figure 4.17: NVidia GForce 9300 chipset – OneChip design[10]

[10]From: http://www.tomshardware.com/reviews/g45-geforce-9400,2263-4.html

Advantages:

- relatively small size, saves space on the motherboard (one chip takes less space than two),
- cost more than half the cost of the North-South bridge concept,
- significantly lower power consumption, etc.

Disadvantages: limited modularity (different north and south bridges cannot be combined) and usually lower performance.

Limited modularity is not such a problem anymore and the advantage of small size, lower price and power consumption outweighs this disadvantage, especially in small devices, so we encounter the concept of One Chip Design in small mobile devices (smartphones, tablets), as well as in embedded devices (especially in living room electronics) or HTPC (multimedia centers). An example of a diagram can be found in Figure 4.18.

Remark

Do not confuse One Chip Design with the newer form of the North-South Bridge concept, in which the North Bridge is integrated into the processor. In both cases, there are two large chips on the motherboard, one of which is the processor, but in North-South Bridge Design the second chip is the south bridge (i.e. only part of the chipset), whereas in One Chip Design the entire chipset is in the second chip.

4.4.4 SoC Chips

SoC (System on Chip) is actually a special case of One Chip Design, where the following changes have been made compared to the parent concept:

- the entire chipset (which was originally in one separate chip on the motherboard) is moved into the processor case, i.e. the chipset is integrated into the processor (or the processor into the chipset, depending on the point of view),
- everything that is technically possible is further integrated into this (now single) big chip.

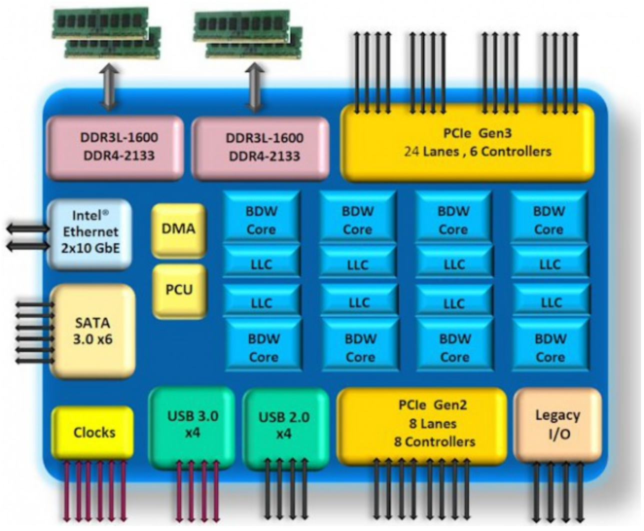


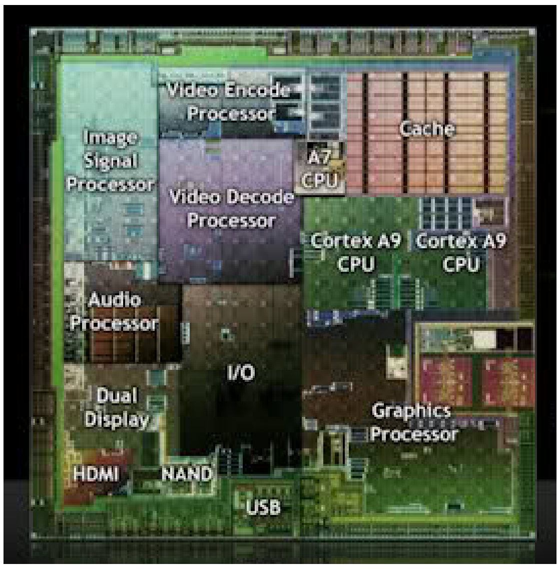
Figure 4.18: Diagram of the SoC chip Xeon D introduced in 2015[11]

In one chip case there is a processor, chipset, timing pulse generator, timers, communication interface controllers (USB, Ethernet, as needed), graphics, etc., i.e. both the original chipset and other modules that we would otherwise find in separate chips on the motherboard.

This trend first emerged in embedded devices, but has gradually made its way to cheaper mainstream devices. Today, virtually all small mobile devices and most Internet of Things (IoT) devices use an SoC chip.

[11]From: https://www.pcper.com/category/tags/xeon-d

SoC chipsets/processors used to be produced mainly by Qualcomm (Snapdragon), Apple (for its mobile devices), Broadcom, Samsung, TransMeta, nVidia (Tegra processors), Atmel, and nowadays there are also SoC processors from AMD (AMD Geode) and Intel.



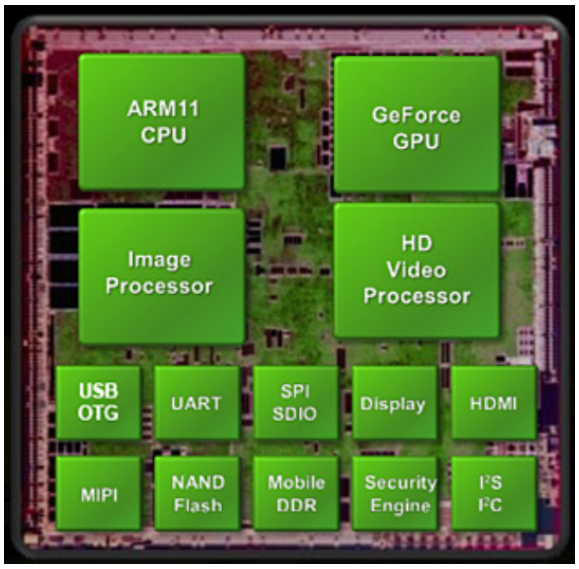


Figure 4.19: SoC processor ARM Cortex A9 and SoC procesor NVidia Tegra with the ARM 11 processor[12]

SoC chips are not necessarily designed for low-cost devices. In Figure 4.20, we see a block diagram of Ampere SoC chip (ARM type, 2018) designed for servers, manufactured on a 16nm manufacturing process. It has 32 compute cores, supports up to 1TB of RAM, and offers 42 PCIe version 3 lanes.

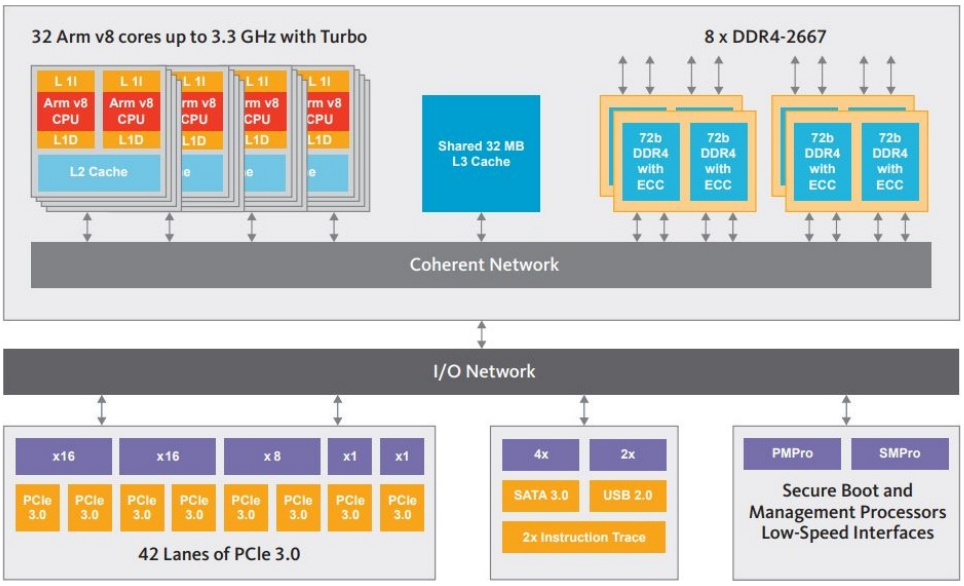


Figure 4.20: SoC processor Ampere dedicated for servers[13]

[12]From: http://i48.tinypic.com/o7t3y9.jpg, http://www.xbitlabs.com/news/mobile/display/20080603141353_Nvidia_Unleashes_Tegra_System_on_Chip_for_Handheld_Devices.html

[13]From: https://www.servethehome.com/ampere-32-core-64-bit-arm-chip-x-gene-3-ip/

Remark

Most SoC chips are ARM type (indeed almost all of the above). What does this mean? ARM Holdings does not directly manufacture these processors, it only develops them and then licenses its partners the specifications to manufacture them. So, for example, Qualcomm Snapdragon, Apple mobile, Samsung Exynos, nVidia Tegra, etc. are all built on the ARM architecture.

4.5 Switch and Jumper

Switches have two positions – on and off. They are used to configure in hardware such features that have two states. The switch comes in several different forms – one of the forms is in Figure 4.21 on the right.

Jumpers, unlike switches, can have several different states. Usually it is a block of pins arranged in a matrix on which a jumper cap with a conductive bridge can be slipped. The cap is usually on two adjacent pins, and depending on which two pins of the block are so connected, one particular option is turned on.

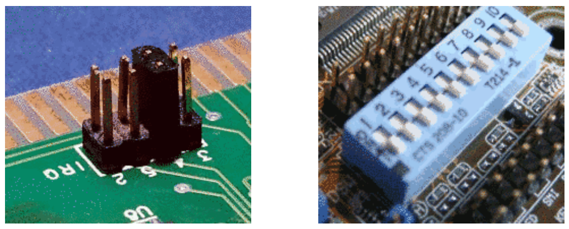


Figure 4.21: Switches and jumpers[14]

Switches and jumpers on motherboards are used for important settings that are usually not changed during computer operation. Today, in the age of Plug & Play, they have been replaced by much more convenient auto-detection algorithms.

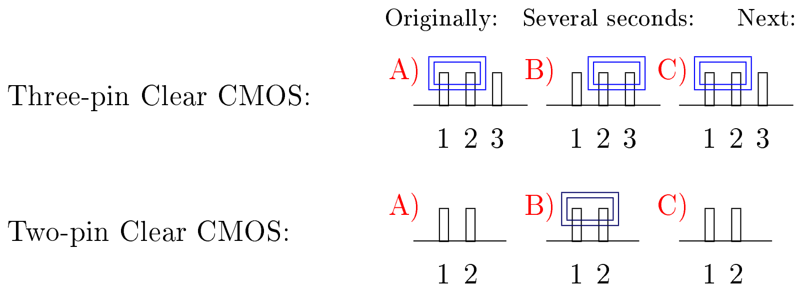


Figure 4.22: Clear CMOS using jumpers

The switch or jumper can also be a dedicated *Clear CMOS*. It is designed to clear user BIOS settings and bring it to factory defaults (BIOS reset), which is the same effect as removing the battery maintaining the BIOS settings. On most motherboards this is a three-pin jumper, which can be seen in Figure 4.22. By default the jumper is set to pins 1,2. After connecting to 2,3 (when the computer

[14]From: http://www.infopackets.com/news/hardware/2004/20041228_install_sound_card_to_computer_with_onboard_integrated_sound.htm

is switched off) the CMOS will be cleared, then we should not forget to return the jumper to pins 1,2. The procedure may vary for individual motherboards, the specific procedure can be found in the motherboard documentation.

In addition to using pins or removing the battery, some motherboards offer other options, such as a Clear CMOS button (so we can just press the button).

Chapter 5

Processors

Quick preview: In this chapter, we will look at the brain of the computer, the processor. We will focus on processor structure, features, techniques for increasing processor throughput and performance, and last but not least, we will review current processors.

Keywords: Processor, CPU, APU, IGP, MCU, DSP, FPGA, litography, Moore's Law, ALU, FPU, register, instruction set, AVX, AES, CISC, RISC, cache, pipelining, scalar architecture, Out/of/Order Execution, Speculative Execution, dependences, hyperthreading, SMT, TDP, Turbo Boost, Turbo Core, IRQ, DMA, I/O ports, I/O addresses, socket, Intel, AMD, ARM.

Objectives: The goal of this chapter is to learn about processor architecture and functions, and to become familiar with current processors.

5.1 Basic Processor Functions

The *Processor* is a complex integrated circuit (millions or billions of transistors), it is a silicon wafer (actually a mixture of silicon and other elements) into which the circuit connections are etched in several layers by a photoelectric (litographic) process.

If a graphics processing unit (GPU) is also integrated into the microprocessor, it is referred to as *APU* (Accelerated Processing Unit, for AMD processors) or *IGP* (Integrated Processing Unit, for Intel).

There are three additional types of processors:

- *MCU* (Micro Controller Unit) – they are made for a specific activity (specialized), but they can also be universal, low cost, small size, low power consumption, less performance, small expandability.

  These simple inexpensive processors are now used in simple electrical devices (microwaves, washing machines, alarms, traffic lights, etc.), work as controllers (drives, printers, keyboards, monitors, control fans, etc.).
- *DSP* (Digital Signal Processor) – similar to MCUs (also specialized for certain types of tasks), but with higher performance, optimized for processing large amounts of data of the same type.

They are mainly good at processing mathematical functions on large volumes of data, and also include digital-to-analog and analog-to-digital converters. In computers they are used as auxiliary processors, for example in sound cards.

- FPGAs (Field Programmable Gate Array) are highly customizable chips. While other types of chips are "programmed" from the factory (they have an instruction set specified by the manufacturer), FPGAs can be configured, i.e., they can specify what a particular chip should be able to do. These chips are more likely to be found in robotic devices where chip programmability is virtually essential.

Remark

In a computer, we actually usually have multiple processors – a main microprocessor and then certainly a few DSPs or MCUs (a sound card, possibly a graphics card and others, in the keyboard, then various controllers).

The physical structure of the processors, and indeed of each chip, is similar to the physical structure of the motherboards we described above. The base is a silicon *wafer*, which is made by cutting silicon ingots into very thin wafers that must be polished perfectly. Current processors typically have several dozens of layers. The circuits are then deposited by a *photoelectric process* (we also talk about lithography):

- for each layer is prepared *photolithographic mask* as a template of circuit diagram, masks are created in specialized CAD systems,
- the necessary conductive and insulating materials are applied to the wafer,
- the mask is applied and the assembly is exposed to UV light, softening the areas not covered by the mask,
- a chemical solvent is used to etch only the softened areas,
- the same is done for the other layers of circuitry.

5.1.1 Manufacturing Process

The manufacturing process means what the transistors look like, how they are arranged, connected and powered, what materials are used, what wavelength of UV light is used in the lithography process, how the UV light is directed, and from that, how big the transistors are. The specific wavelength of UV depends on what materials can be used, how thick the mask can be and therefore how much the circuits can be squeezed, but also, for example, how high a frequency the resulting processor can operate at.

Simplistically, it is referred to as xxx-nanometer fabrication technology (often "litography" in English), as this is related to the UV wavelength, material and shape of the transistor.

Example

For example, at Intel we see the following data:

- "prehistoric" i4004 processor was made with 10$mu$m technology (micrometers),
- the Core 2 is 65 or 45 nanometers,

- Core i7, i5 and i3 first and second generation processors are made with 32nm or 45nm technology,
- third and fourth generation Core i7, i5 and i3 processors are manufactured at 22nm,
- Core i7, i5 and i3 fifth and sixth generation processors are manufactured using 14nm technology,
- seventh generation is manufactured with 14+ lithography, eighth and ninth 14++,
- the tenth generation is partly produced with 14++ lithography, partly with the newer 10nm process.

To give you an idea: for processors manufactured with 45nm technology, the masks are illuminated with UV radiation at 193 nm, while for processors manufactured with 22nm and newer technology, a wavelength of 13.5 nm is used.

AMD manufactures its third-generation Ryzen processors using 7nm lithography (in fact, TSMC manufactures those processors for AMD). In the case of ARM it's more complicated, each manufacturer has it differently. Samsung is the furthest along so far, roughly on par with AMD.

Additional information

- Processor manufacturing by Intel:
  - Intel Factory Tour – 32nm Manufacturing Technique:
    https://www.youtube.com/watch?v=SeGqCl3YAaQ
  - Intel: The Making of a Chip with 22nm/3D Transistors | Intel:
    https://www.youtube.com/watch?v=d9SWNLZvA8g
- Processor manufacturing by AMD (video, animations):
  http://www.youtube.com/watch?v=qLGAoGhoOhU

5.1.2 Transistors

A transistor is a tiny semiconductor device that allows a small change in voltage at the input to cause a large change in voltage or current at the output.

In processors we find unipolar transistors (also called FET – Field Effect Transistor), where the current flow is controlled by an electrode called *gate*. The gate either passes electrons or not, depending on the voltage of the electric field driving the gate (hence the name FET).

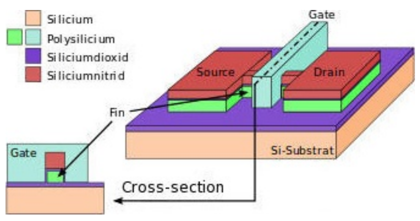


Figure 5.1: FinFET transistor[1]

The current passes in the direction from the electrode "source" (S, emitter) and "drain" (D, collector), and may or may not be passed through the gate (G, also gate). All this is on the surface (substrate) of the semiconductor. The electrode G is also referred to as the input electrode, because the current flow is influenced through it. So there is a channel for electrons between electrodes S and D, which can be interrupted by electrode G. Figure 5.1 shows a scheme of the transistor with these parts labeled.

So what about the shape of the transistor? Previously, *planar transistors* (Planar FETs) were used, in which the gate is placed above the channel between S and D, and the voltage of its electric

[1]From: https://www.computerhope.com/jargon/f/finfet.htm

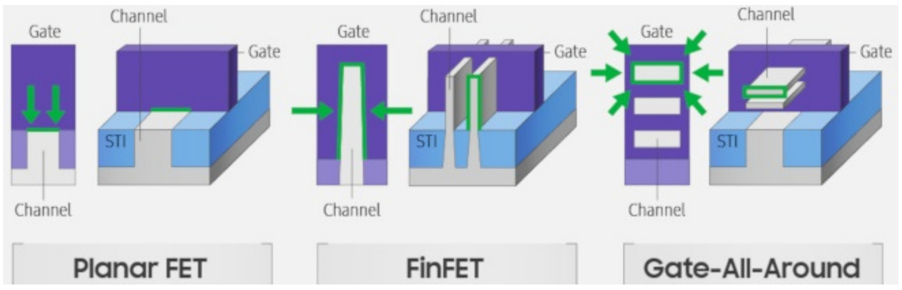


Figure 5.2: Various types of transistors: planar, FinFET, GAA-FET[2]

field affects the throughput of the channel through the substrate (base). The planar transistor can be seen in the left third of the figure 5.2.

However, as the manufacturers got closer to 20nm technology, the *tunnel effect* started to manifest itself, because with smaller transistors S and D get closer together and the channel is getting shorter and shorter, so the gate is getting thinner and thinner. The gate became so thin that even in the closed state, electrons could pass through it.

Then the manufacturing technology switched to *FinFET transistors* (because the channel shape really resembles a fin). The channel is raised above the substrate and the gate surrounds it on three sides, so the risk of tunneling is reduced (but again only up to a point). Because of the lifting above the substrate and thus the transition to "another dimension" we also talk about 3D transistors.

In addition, the channel can be divided into two or three streams (actually fins) to maximize the influence of the gate. The FinFET transistor is shown in Figure 5.2 in the middle. The smaller figures indicate the direction of the gate's electric field. The limit of applicability of FinFET is before the 3nm process.

This limit should be surpassed by transistors with the *GAA-FET* (Gate-all-around FET) structure, or in Samsung's case called *MBCFET* (seen in Figure 5.2 on the right), where the gate surrounds the entire channel (or all fins) from all sides.

✎ In addition to the structure of the transistor, the material is of course also important. It is not only that the material has the appropriate "utility" properties to work as a transistor or integrated circuit, but also that the fabrication of circuits using the photoelectric process is feasible.

In electronics, including processors, we can encounter, for example, MOSFET (Metal Oxide Semiconductor Field Effect Transistor) technology, where the gate electrode is made of metal (Metal) and coated with an insulating layer of silicon dioxide (that's the Oxide in the name) that separates the metal from the semiconductor (substrate). Aluminum used to be the metal, later it was modified silicon – polycrystalline silicon (although it didn't fit with "M" anymore, it was still called MOSFET), but gradually it switched to other metals.

MOSFETs have become quite widespread, besides processors and memory chips they are also used in power supply implementation, in various sensors, consumer electronics,. . .

MOSFET technology is not just one either. For example, a type of MOSFET called CMOS is used a lot in chips. Today, CMOS is used not only for memory chip processors (for example, the BIOS configuration is stored in the CMOS chip), but also for making analog circuits (amplifiers, image sensors, etc.).

[2]From: https://www.cdrinfo.com/d7/content/samsung-electronics-announced-mbcfet-transistor-structure-advanced-foundry-nodes

We talked about 22nm transistors, 14nm, 10nm, 7nm transistors. The "nice little number" doesn't correspond to the size of the transistor anymore, it's usually the size of some very small part of the transistor, like the width of the gate. For example, in Intel's 10nm processors (Ice Lake) the distance between the gate and the metal conductors to which the drain and source electrodes are connected is 36 nm (i.e. this distance from the gate to both sides), in the case of TSMC's 7nm process (AMD processors) it is 40 nm. The fin width for 10nm Intel is 34 nm, for 7nm TSMC manufacturing process it is 30 nm, for 7nm Samsung process it is 27 nm. Marketing knows what to (not) point out.

5.1.3 Moore's Law

Processors (like many other electronic components) consist primarily of transistors (semiconductor components) in integrated circuits. For decades, manufacturers have been trying to miniaturize – they want processors to be smaller, or the same size but more powerful (i.e., more transistors in the same space). The way to do this is to downsize the transistors.

Theorem (Moore's Law)

Every two years, the number of transistors in processors doubles.

Gordon E. Moore (co-founder of Intel), 1965, revision 1975

The original version before the revision talked about doubling the number of transistors every 18 months (so the revision meant slowing down). A graphical representation of Moore's law can be seen in Figure 5.3.

In practice, we can see the application of Moore's Law in that the manufacturing process is constantly improving in terms of transistor size (now in nanometers).

Remark

Moore's law doesn't say anything about computing power – it doesn't say that doubling the number of transistors doubles the power. In fact, performance increases faster:

- Intel i80286 (year 1982, 134 thousands of transistors, frequency 6–12 MHz), performance cca 2 MIPS (milions instructions per second) at higher frequency,
- Intel Core i7-2600K (year 2011, 995 millions of transistors, frequency 3,4 GHz per core), performance 128 300 MIPS

(if performance doubled every two years, there would be processors with performance $2^{15} = 32\,768$ MIPS in 2011).

Manufacturers have a growing problem with shrinking transistors. Transistors are already so small that further miniaturization runs into material limits – if an electron has a boundary just a few atoms wide in front of it, then there are bound to be a lot of holes in that boundary through which the electron will go somewhere else than it should. By the way, the width of a silicon atom is just under half a nanometer.

[3] From: https://managementmania.com/cs/mooruv-zakon

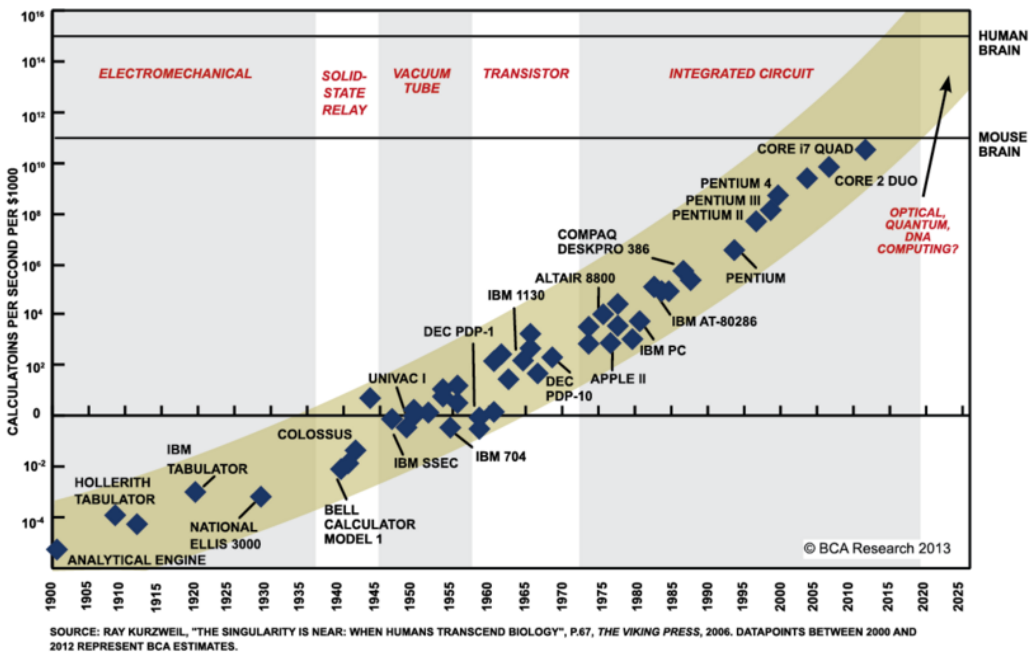


Figure 5.3: Moore's Law[3]

5.2 Structure of Processor

5.2.1 Logical Structure

The processor consists of the following components:

- *controller* – processor operation control,
- *decode unit* – converts numeric code to instructions which the processor understands, fetches instruction operands,
- *ALU* (arithmetic-logic unit working with integers) – usually one for each core,
- *FPU* (mathematical coprocessor, unit for floating point operations) – one or more,
- *internal bus* – processor components communicate with each other,
- *registers* – small fast memories for instruction operands and intermediate results,
- *MMU* (Memory Management Unit) – converts a virtual address to a physical address in memory,
- other components, a memory controller, a graphics core, etc.

Nowadays, most processors are multi-core. The *core* of a processor is a relatively separate part of the processor, the instructions executed are allocated to individual cores. The structure of the core can be different in various processors (especially if they are from different manufacturers), usually inside core we find its own ALU, FPU (or common FPU, but multi-threaded and each core has its own threads), registers, decode unit, etc. However, a core can also be less independent, for example in older processors from AMD we find depleted cores where a pair of cores share a decode unit, FPU and some other components.

The *Controller* is a sequential circuit that controls (almost) all parts of the processor and thus indirectly the computer, using control signals (sends) and status messages (receives), as we saw in the von Neumann schematic drawing. It also coordinates the activity of other controllers in the computer (memory controller, disk controller, etc.).

Arithmetic-Logic Unit (ALU) performs arithmetic and logical operations on data. It has the following parts:

- part for arithmetic operations (ADD, SUB, MUL, DIV), i.e. it can add, subtract, multiply, divide,
- part for logical operations (OR, AND, XOR, etc.),
- barrel-shifter – bit shifts right and left (SHR, SHL).

Example

What is bit shift? Let's take a binary number, such as `00100101` (we have eight bits, eight binary digits). A bit shift to the left means that we shift all these binary digits one place to the left (i.e. to a more significant position in the number), and add a zero to the vacant position at the end of the number:

SHL: `00100101` ⇒ `01001010`

If it were a number in the decimal system, then by shifting and adding a zero to the end, we would actually multiply the original number by ten. But since we're in binary, a bit shift to the left corresponds to multiplying by two. To better illustrate this, let's convert these binary numbers to the decimal system:

SHL: `37` ⇒ `74`

We actually erased the first digit of the original (binary) number – it doesn't matter here, we lost a zero in the highest order, no information was lost. But if it was a one, and the data type didn't allow us to "extend" the number to more bits, then we lost some information.

The bit shift to the right is exactly the opposite:

SHR: `00100101` ⇒ `00010010`

If it were a decimal system, then erasing the last digit would mean integer division by ten. In binary, it's integer division by two.

We erased the last digit. So while there is no risk of losing information when moving to a higher row (when the data type is not sufficient), on the other hand, the lowest order information is lost. As mentioned above, this is an *integer* division by two, that is, the remainder after the division is "forgotten".

Bus is, as we know, a communication path. It has to communicate inside the processor, so inside the processor we have an internal bus, but also the processor has to communicate with its environment (the north bridge, or if integrated, the south bridge). In the following, we will write about the FSB for simplicity.

FSB speed determines how fast the processor communicates with its environment. The higher the frequency, the faster the processor communicates (note that this is not the same as the speed of the processor itself). The FSB speed of newer processors and boards is not given in Hz units (i.e. pure frequency), but in T/s (Transfers) units, usually GT/s (giga-). This is because this bus is multipath (it can be four-way, eight-way, etc.), with a certain width (number of bits transferred at a time, e.g. 64-bit, 128-bit, etc.). A higher number is better, of course, but this value is hard to compare across different processors. In addition to the frequency, it takes into account how wide the bus is (i.e. how much data it can transfer in one cycle) and, on the other hand, the number of processor cores (more cores means higher bus utilization).

Multiplier (multiplier factor) is the value by which the FSB frequency is multiplied, the result is the speed of the processor, more precisely its cores. So the multiplier actually determines the multiplicity of the processor's speed (cores) relative to its bus speed. The multiplier typically takes values in units or tens, for example 26 (but not necessarily an integer, it can be 5.5 for example). In current processors, it is not permanently set to the same value, but dynamically varies over a fixed range.

Example

Assume that the processor multiplier is set to 24, the bus frequency is (for simplicity) 100 MHz. The processor then operates at the speed of

$$\boxed{\text{processor frequency}} = \boxed{\text{multiplier value}} \times \boxed{\text{FSB frequency}}$$

In our case 24×100 MHz $= 2400$ MHz $= 2{,}4$ GHz

The processor frequency can be raised in two ways – either by increasing the multiplier value or by increasing the FSB frequency (or its newer variants).

The memory bus is between the memory controller and the memory modules. Since the memory controller is nowadays mostly part of the processor, it runs between processor and memories. The faster the memory bus, the faster the processor can, for example, retrieve the data it has to process from memory. Just as the FSB is overclocked, the memory bus can also be overclocked.

Buses are not just about the highest possible speed. First of all, excessive speeds can cause system instability, and also – the speeds of the different buses (and the processor) must be synchronized. If, for example, the processor and memory bus frequencies are not synchronous, there can be situations where one component has to wait for the other before their cycles "meet". Moreover, in AMD processors, the memory bus is extended into the processor and is part of the processor's internal bus, so here the synchronization of the DMI and memory bus is even more important than in Intel.

5.2.2 Instruction Set

Instruction set is simply a set of instructions of a given type.

Every processor has one or more (currently more) instruction set(s) – one main instruction set (arithmetic operations, data movements, etc.) and some additional sets, e.g. for multimedia processing, encryption, sets for compatibility with older processors,. . .

Instructions often work with registers. A register is a small piece of memory situated inside the processor, typically in computing cores. Registers are addressed with their names, not with numeric addresses. The names may be R1, R2,. . . , or D1, D2,. . . , AX, BX, EAX,. . . , etc. according to the appropriate architecture.

Example

An instruction for addition may look like this:

```
ADD RT, RA, RB
```

adds the contents of the registers RA and RB, the result is placed into RT, so RT=RA+RB, typical for ARM processors,

```
ADD BX, AX
```

adds the contents of the registers BX and AX, the result is in BX too, so BX+=AX, typical for Intel and AMD processors.

Instruction is a number composed of the following information:

- identification of the instruction (opcode, operational code),
- type of operands (specifying their length),
- other information, such as which default register to use,
- the actual operands to be processed by the instruction.

Example

For example, for the `MOV AX, 0x1234` assembly instruction from the Intel instruction set, the corresponding machine instruction on a 32-bit system is `0xB8 0x34 0x12`, in binary `1011 1000 0011 0100 0001 0010`. More information is encoded in this number – for example, that it is a `MOV` instruction (moving, here storing, a number into a register), the register `AX` is to be used, and the hexadecimal number `0x1234` is to be stored in it.

The instruction set actually defines not just the instructions themselves, but the entire related ecosystem – data types, registers, IRQ interrupt handling rules, etc.

Machine code is a sequence of instructions in numeric form. The machine code is specific to the processor (hardware architecture), but if we have an executable file in the operating system (such as .EXE in Windows), it is not purely machine code – there is a lot of information there for the operating system. The operating system actually acts as an intermediary here, reading the data from the executable, and sending machine code to the processor.

On top of that, some code "more" is actually executed while the process code is being processed, for example, if a process makes a system call (wants to open a file, render something to the screen, asks for another piece of RAM, etc.), the operating system runs its own code in the context of the process to execute this system call. If the process is running in a runtime environment or subsystem (which is common, including most system processes), then some of the executing machine code is part of the runtime environment.

Instruction sets in use. First of all, each processor has its "main" instruction set covering common tasks (64-bit instruction set in today's processors), then for backward compatibility it can support older instruction sets to run programs compiled for older architectures. Older instruction sets can only be emulated, so they tend to be slower. But that's not all – we also have *additional instruction sets* in the processor, containing for example multimedia instructions or instructions for faster encryption. For example:

- Older multimedia SIMD sets: Intel MMX, AMD 3DNow!, SSE, SSE2,. . . , SSE5.
- AVX (Advanced Vector Extensions, creation 2008/09, deployment 2010–2011), AVX 2.0 – second version, next version is AVX-512.

- AES – hardware support for AES encryption, there may be other similar instruction sets related to cryptography, e.g., SHA set or instructions to support encryption with Galois fields.
- We can find instruction sets for speeding up neural networks and for artificial intelligence in general, e.g. DL Boost (Deep Learning Boost) by Intel.

Multimedia instruction sets (currently mainly AVX in various versions) are used not only for processing multimedia data (image, sound, video), but also for speeding up complex mathematical calculations (if we generalize, multimedia data processing is actually hard mathematics) or encryption (for example, OpenSSL uses AVX and AVX2 to speed up encryption).

ARM processors also have several instruction sets. In addition to encryption support, there is also, for example, the SVE (Scalable Vector Extension) multimedia set.

Multimedia and encryption instruction sets are typically *SIMD* (Single Instruction Multiple Data, vector sets) – see Flynn's Taxonomy, page 11, which means that one (the same) instruction is used for the whole vector of data (we do the same thing with different parts of the image).

Each multimedia set needs its own registers. Because they usually handle large rational numbers (floating point), these registers are quite large (for example, AVX in the first version uses 16 YMM registers, each of which can contain either eight 32-bit rational numbers or four 64-bit numbers.

5.2.3 Cache

Cache is a buffer. It generally serves in communication between two differently fast components, it ballances differencies in speed of these components. In the case of a processor, cache is between a (fast) processor and a (slow) memory. The cache is preloaded with data and instructions from memory that are expected to be needed by the processor in subsequent steps.

Processors often execute some instructions repeatedly – there are loops, repeatedly called functions, etc. If a code has already been loaded into the processor and is likely to reload soon, why to reload it repeatedly if it is already in the processor?

In a processor, we distinguish between L1 (level 1), L2 (level 2) and L3 (level 3) caches, then we talk about level 1, 2 or 3 caches.

The *L1 cache* is located directly in the processor, in multi-core processors each core has its own L1 cache. It is the fastest and has the smallest capacity of the three cache levels (usually in the tens of KB on desktops). The first level of cache is divided into two parts – the data cache (D-cache, L1d) and the instruction cache (I-cache, L1i), and then the capacity of both parts is also kept separate.

The *L2 cache* can be common to all cores or each core can have its own, or it can be common to a module of two cores. It is usually larger than L1, typically in the hundreds to thousands of kB (in units of MB if the L3 cache is not used), and is clocked at the frequency of the bus used to communicate with memory).

L3 cache is common to all cores (often understood as a common communication environment for cores), or common to a group of cores, with a capacity in units or tens of MB. In general, only one processor (core) can write to L3 at a time, but multiple processors can read at the same time. In processors with integrated graphics, it can also be connected to the graphics core (then the graphics core can communicate with the regular computing cores via L3).

A *cache block* (cache line) is a set of contiguous adress locations from the primary memory, the whole block is transmitted into cache. The usual length of this block is 32 or 64 Bytes. Current DDR memories contain commands in their protocol that can perform burst read, burst write, the length of which transmitted data corresponds to the size of the cache block.

Each block is stored in cache at least with the following additional information:

- block of data/instructions, transmitted contents,
- *block index* calculated from the source address of a block in memory,
- *tag* (metadata), e.g. dirty bit (see below).

A *mapping function* is function determining correspondence between the primary memory blocks and their cached copy.

A *replacement algorithm* is collection of rules used in case that the cache is full and alongside some new data or instructions need to be cached, this algorithm determines a "victim" to be removed from cache to make free space for new data or instructions.

If a processor needs to work with a part of main memory (perform a read or write operation), its request is captured by the cache controller, we say that a *read or write hit* has occurred.

Reading from cached memory. If a processor needs to load data or an instruction from main memory, two possibilities may occur:

- the requested content is in cache,
- the requested content is not in cache.

The request (read hit) is captured by a cache controller, which finds out the corresponding possibility. The first possibility means that the requested content is simply read from cache.

To handle the second possibility, the new block has to be transferred into cache (we call this situation by *read miss*). If cache is full, the replacement algorithm is used, and then the transfer can be performed.

A *dirty bit* is a part of metainformation used in the second situation – we write to cache only. This flag bit is set in case that the processor makes a write hit. if the write hit for the same address is preformed repeatedly, this bit remains set. The main memory is updated later, and the repeated writes into the main memory are not carried out unnecessarily often.

Writing to cached memory. If it is necessary to write (store) data or instructions, these two situations may occur:

- copy of the corresponding block is in cache,
- copy of the corresponding block is not in cache (we call this situation by *write miss*).

These situations are solved similarly as for reading operation. These two methods are used in the both situations, so if it is necessary to update something in the main memory:

- write-through: the both the cache and main memory, is updated withing this write operation,
- write-back: only the cache is updated, by a dirty bit for the given glock is set.

The write-through algorithm is quite simple, but it is not optimal in case that the same piece of data has to be modified very often.

Remark

The size of the highest level cache used (for example, Intel calls it *Smart Cache*) has a quite big impact on the speed of the whole system. Especially with power-efficient processors, we should pay attention to the size of the L2 (or L3) cache.

Figure 5.4 shows the so-called *memory pyramid*, where we can see that the more expensive the memory, the less it is (and the closer it is to the CPU cores).

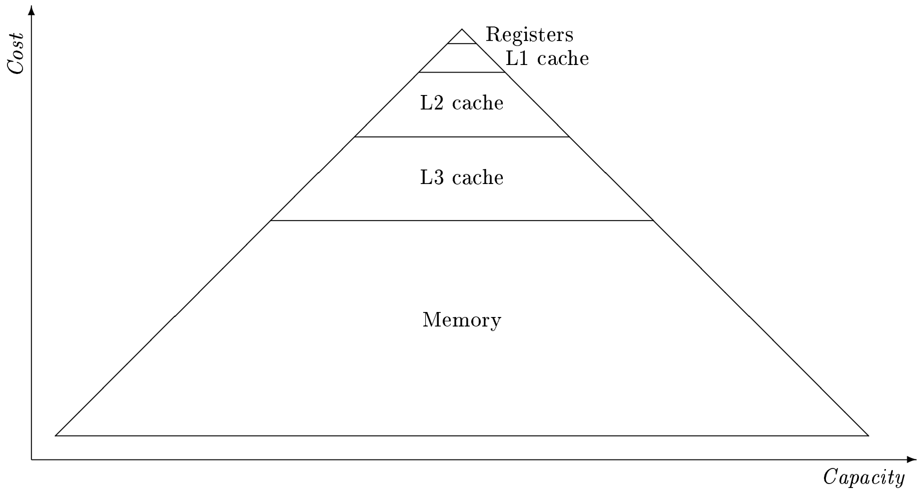


Figure 5.4: Memory pyramid – more expensive memory types have smaller capacity

In fact, we have somewhat more memory circuits in the processor than just registers and L1/L2/L3 caches. For example, the Memory Management Unit (MMU), which we encountered in the list of logical components in the processor, uses its own cache called the Translation Lookaside Buffer (TLB) to speed up the translation of a virtual address to a physical address. If it has translated an address in the immediate past, the result is stored in the TLB, and there is no need to translate again when the same memory page is accessed again.

When translating a virtual address, the (translated) entry is first looked up in the TLB, and if it is not there, it sets "TLB miss" and the translation must be done by the MMU.

5.2.4 Processor Modes

Current Intel and AMD processors can operate in the following two modes:

1. *real mode* – the processor behaves like the 1978 i8086, uses only 20 bits on the address bus (addresses a maximum of 1 MB of RAM), memory protection and thus multitasking cannot be used, processes have access to everything,
2. *privileged mode* – processor uses the entire address bus range, uses support circuitry for this mode including additional registers (accessible only in protected mode), memory protection implementation, multitasking assumption, allows switching between kernel mode (privileged) and user mode.

Current operating systems do not use real mode. We can meet it for example in BIOS, in some diagnostic boot programs, etc.

Privileged mode means mainly the possibility of hardware memory protection. It differs from the real one by adding some components (some registers and flags are added) and different address handling.

On Intel and AMD processors there are four *rings* (ring) for protected mode – Ring 0, Ring 1, Ring 2 and Ring 3. The first of these (Ring 0) is the most secure, and is usually where the operating system kernel runs,

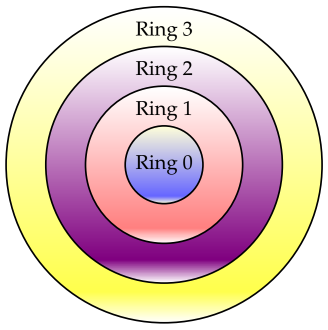


Figure 5.5: Protected mode on Intel processors

whereas processes running in other rings do not have direct access to this ring. Typically, everything but the kernel (i.e., normal processes) runs in Ring 3.

The higher the circuit number, the further away from the CPU and other hardware (in terms of accessibility). What runs in Ring 0 has direct access to the hardware, but higher numbered circuits access the hardware essentially through the API of what's in "internal" circuits.

Ring 1 and Ring 2 are usually not used. Nevertheless, they can be used to further scale access permissions and for example Ring 1 is used in some virtualization techniques, especially on servers. In fact, there are also "negative" circuits – this code is directly part of the processor.

✎ ARM processors (used in mobile phones, tablets, networking devices, consumer electronics, etc.) are more complicated. The basic mode for normal processes is *User mode*, in which system resources cannot be accessed, similar to Intel's circuit 3. Code running in other modes already has at least partial access to system resources, for example (this list is not exhaustive):

- *supervisor mode* – used when handling software interrupts, including system calls initiated by normal processes,
- *system mode* – this mode runs special privileged applications that need access to system resources (not every ARM architecture provides this mode),
- *IRQ mode*, *FIQ mode* – modes for handling hardware interrupts (IRQ),
- *hypervisor mode* – only on processors with extended hardware virtualization support (multiple operating systems can run fully on the device at the same time).

Whatever the processor type, there are always several bits in some control register that tell the processor what mode to use. If these bits indicate that it is currently in user mode (or Ring3) and an instruction comes in that has no business being in that mode (because it accesses system resources), the processor will refuse to process it and raise an exception. The typical consequence is an immediate termination of the corresponding process.

✎ On newer processors, protection against code execution on data memory pages has appeared. On AMD processors it is *NX bit* (Non-Executive), on Intel *XD bit* (Execute Disable). If the page has this bit set to 1, it is considered to be a data page, and an exception is raised when the page attempts to treat the contents of the page as code (to execute instructions stored there), which usually results in the termination of the process that attempted to do so. The purpose is to prevent memory overflow attacks.

5.2.5 Registers

Registry are small very fast memory locations located in the processor, usually used to store addresses, intermediate calculation results and instruction operands. The size of registers is usually 16, 32, 64, 128, 256 bits, etc. (some powers of 2).

✎ Division of registers by their purpose:

- *data registers* – for data, for example a number to be multiplied, rather generic at Intel,
- *address registers* – for addresses of variables from memory (like pointers) or array indexes,
- *control registers* – they contain status information, for example, they can indicate that the instruction just completed has a negative number as a result (the number itself will be stored in some data register).

In the literature, we also see the DMR (Data Memory Register) and AMR (Address Memory Register) designations for data and address registers, respectively.

✎ Register sets are usually implemented as a memory block in the processor (core) called *register file*, which is basically an array of registers. SRAM (static RAM) with multiple inputs/outputs (ports) is usually used for these circuits. If the processor supports *register renaming* (which almost all processors do nowadays), then one *register name* can correspond to several physical registers in the register file, the specific physical location is determined by the instruction currently being executed. This makes it possible to parallelize computation even within a single core.

There may be more than one register file in the processor core, for example, these blocks may be separate for additional SIMD sets including a set for floating point operations.

5.2.6 Intel and AMD processor Registers

To get a better idea, let's see what registers exist on Intel and AMD platforms.

✎ *General registers* of Intel processors are

- RAX, EAX, AX, AH, AL – used mostly for multiplication and division, and also for I/O operations (accumulator),
- RBX, EBX, BX, BH, BL – used for indirect memory addressing to variables (base),
- RCX, ECX, CX, CH, CL – used as a counter for cycles, shifts and rotations (counter),
- RDX, EDX, DX, DH, DL – used for indirect I/O addressing (data).

These can be thought of as data registers, but some can be used for other purposes including addressing, so we rather call them generic.

The meaning of labeling is shown in the figure 5.6 (except for the widest forms starting with the letter R). The earliest Intel processors had only registers with two-letter designations. The last letter of the designation is either L (Lower, lower byte), H (Higher, higher byte) for 8-bit parts, or X (includes the lower byte on the right and the higher byte on the left). The second to last letter (A, B, C, D) indicates the specific register type.

However, 16 bits were soon no longer sufficient, so the registers were expanded first to double (32 bits, three-character designation, beginning E), then again to double (64 bits, beginning R), with the proviso that the original designation could continue to be used to access lower areas of the same register.

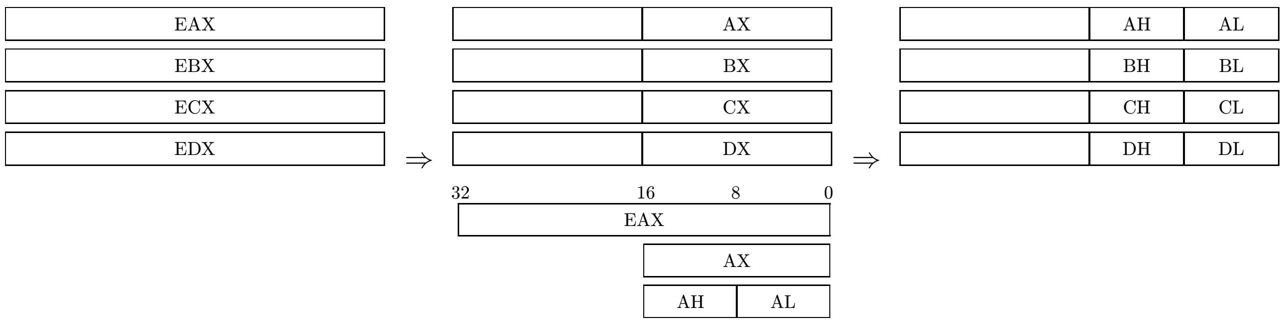


Figure 5.6: Relationship among Intel registers of different widths

✎ The *Address registers* of Intel processors are used to store addresses. These addresses can hold variables, but they can also be addresses of instructions or special data structures (typically a stack).

For several decades now, *segments* have been used in memory management, although on 64-bit architectures only for backward compatibility reasons. Each process has several memory segments, each of them for a specific purpose. Usually there is one segment for program code, one or two segments for global data, one or two segments for stack (where local variables and anything related to functions being run are stored).

The advantage is that when the address of the beginning of a segment is known, we can use a shorter local (relative) address (called offset) for the address of a specific location in the segment (for example, the address of the instruction currently being processed, which is definitely in the code segment and nowhere else) – these addresses start with the number 0 at the beginning of the segment and require fewer bits to store them than if we were to store them as absolute addresses (starting at the very beginning of the entire memory space).

For each segment it is necessary to remember the address of its beginning, for this purpose *segment registers* are used. When running in protected mode, the segment registers nowadays do not contain directly the address of the beginning of the segment, but a selector that can be used to find this address. 16-bit segment registers:

- CS – segment with program code (Code Segment),
- DS – first segment for data (Data Segment),
- ES – Extended Data Segment,
- SS – program stack segment,
- FS, GS – newer segment registers, their use is determined by the operating system.

The other address registers (in order of 16-bit, 32-bit and 64-bit) are as follows:

- IP, EIP, RIP – Instruction Pointer, points to the instruction in the code that is currently being processed,
- BP, EBP, RBP – base register (local variables),
- SP, ESP, RSP – Stack Pointer,
- SI, ESI, RSI – index in the field used as data source (Source Index),
- DI, EDI, RDI – index in the field used as the destination of the data (Destination Index).

Example

Let's look at some examples for the content and use of registers. These examples are generally applicable to Assembler programming.

- CS:IP – the full address of the instruction in the code that is currently being executed (specify the address of the code segment and then the instruction counter, which is the local address in the code segment),
- SS:SP – the full stack top address (the address of the stack segment and the local stack top address within it),
- ES:[$15A0] – some address inside the ES segment,
- DS:[BX][$20D8] – some address inside the DS segment, the relative adress is the contents of the BX register plus the specified number,
- DS:SI – this absolute address is used when using the SI register in string instructions, here we are working in the DS data segment,
- ES:DI – this absolute address is used when using the DI register in string instructions, here we are working in the ES data segment,

- MOV AX, [BX][SI] – we want to work with array items, we have previously loaded the start address of the array into BX (for example using MOV), now we are loading the given array item (whose index we have previously stored in the SI register) into the AX register.

The *eFlags register* is a special type of register in which we have (indirect) access to its individual bits (called *flags*). Each meaning bit has its own label. The flags usually indicate the completion status of the instruction just processed. For example, after executing an arithmetic instruction (addition, multiplication, etc.) we can see whether or not the result is zero, whether the result is a positive or negative number, etc.

The flags are mainly used in value comparison and subsequent code branching. Processor instructions for comparison usually work by subtracting the two numbers being compared (or character indices in an ASCII table) from each other, and depending on whether the result of the subtraction is zero, positive, or negative, it is possible to tell which value was greater or if they were equal.

In the protected mode, the following flags are available in the eFlags register and other flag registers:

- ZF (Zero Flag) – set to 1 if the result of the operation is zero,
- SF (Sign Flag) – set to 1 if the result is negative,
- OF (Overflow Flag) – set to 1 if the result overflows,
- CF (Carry Flag) – set to 1 if the operation just completed has carried 1 to the next higher order,
- IF (Interrupt Flag) – the processor is allowed to execute interrupts,
- DF (Direction Flag) – controls the direction of processing for string operations,
- IOPL (IO Priority Level) – a two-bit field, specifies the priority required to perform I/O. If the program priority is higher (i.e., with a lower number) than the contents of this field, the I/O operation is not performed and a GPF (General Protection Fault) exception is raised,

and others.
Newer processors have even more flag registers:

- control 32-bit flag register CR0 – important flags:
  - PG – set to 1 when paging is enabled,
  - PE – set to 1 when protected mode is enabled,
- registers CR2 and CR3 to manage paging,
- debug registers DR0, DR1, DR2, DR3, DR6, DR7 are used when debugging programs,
- descriptor table registers GDTR (global descriptor table address), LDTR (local), IDTR (interrupt descriptors),
- TR task register,
- registers for testing TR3, TR4, TR5, TR6, TR7.

Remark

If you think that Assembler is not used anymore, you are mistaken. Assembler is still used, for example, to optimize code that is supposed to run very “quickly”, many libraries are optimized this way. Usually you don’t write the whole code in Assembler, but only “critical” parts of it, it is inserted into the code of a higher programming language.

Example

The following is a short Assembler code embedded in a C source file:

```
...
__asm {
  mov ax,[x1]     // load the value of the variable x1 into the AX register
  mov bx,[x2]     // load the value of the variable x2 into the BX register
  add bx,255      // add the number 255 to what is in the BX register
  cmp ax,bx       // compare the contents of registers AX and BX, the result is in eFlags
  jnz @not_zero   // we work with the ZF flag, Jump if not zero, the target is the given label
  jmp @is_zero    // when they are not equal, jump to  label @is_zero
  ...
@not_zero:        // we jumped on this label if AX is not equal to BX
  sub bx,ax       // BX = BX - AX
  mov ax,[val]    // load the value of the variable val into the AX register
  div bx          // AX = AX / BX
  mul ax          // AX = AX * AX
  jo @is_overflow // if the available memory space was overflowed in the previous operation,
                   jump to @is_overflow label (jump if overflow, OF flag)
  ...
}
```

5.2.7 ARM processor registers

ARM processors use a different type of instruction sets than Intel, hence the different marking and use of registers. In the case of 32-bit processors, the basic general register designation is simply R0, R1, R2,..., R15, so we have 16 general registers. In addition to these designations, there are also some substitution names, for example:

- A1–A4 are the registers used for instruction arguments (colloquially "scratch registers", something like working registers), this is another name for registers R0–R3,
- SP (Stack Pointer, same as R13) points to the stack currently used by the process,
- LR (Link Register, same as R14) is used when we call a subroutine (something like a function) in the assembler and the address of the instruction following the subroutine call is stored here (i.e. where to return to when the subroutine is finished),
- PC (Program Counter, same as R15) points to the instruction that is "planned", i.e. to be processed in the next step.

Furthermore, the CPSR (Current Program Status Register) is used, which is similar to the eFlags register at Intel and AMD, i.e. from individual bits we can learn, for example, whether the result of the previous instruction is zero, a negative number, whether a transfer to a higher order occurred, etc.

Example

As mentioned above, some registers physically overlap. For example, if we exit a subroutine (function) in the code and return to the original code, the return point is stored in the LR register (or R14) and needs to be accessed in the PC register (or R15). There are two ways to do this:

`MOV PC,LR` or `MOV R15,R14`

In fact, 32-bit ARM processors have 48 "physical" registers (some of them overlapping, so physically a bit less), and the specific register used depends on what mode the processor is currently running in (user, system, IRQ, ...). For example, the SP register mentioned above has a variant SP_usr for user mode, SP_svc for software interrupt handling mode, SP_irq for IRQ handling mode, etc.

The registers of 64-bit ARM processors are named slightly differently: the general registers are X0–X30 (so 31 general registers). All of them are 64-bit, and if you only need to work with the lower halves of them (there are both 64-bit and 32-bit instructions on these processors, the 32-bit ones just use the lower halves of the general registers), then the names W0–W30 are used. The registers with the same number refer to the same memory location, only the letters "X" and "W" specify a different number of bits to be used.

Name overlap is not as extensive as in the 32-bit architecture. The SP register does not overlap with any general register and again has several physical variants for different processor modes. Other specific registers, including PC, have also become independent. However, the LR register overlaps with the X30.

In ARM, we can see the name *register bank*. If a register has potentially different contents in different processor modes, we refer to it as "banked" register. Each mode has its own register bank, which contains the physical locations of these registers. For example, in the 32-bit ARM architecture, registers R0–R7 are the same (and accessible) in all modes, i.e. they are never banked. Registers R8–R12 are the same in almost all modes except FIQ mode, i.e. this mode has its own instances of these registers in its bank.

Registries R13 (SP) and R14 (LR) are banked in all privileged modes (i.e. all except user mode, in other words – each mode has its own instance, user mode has "original"). The R15 (PC) and CPSR (flags) registries are not banked, but there is limited access to them in user mode.

#n	Mode	Example	Meaning	Side effect
1	Immediate	`MOV R1, #25`	`R1=25`	none
2	Direct	`LOAD R1, Variable`	`R1=Variable`	none
3	Register direct	`ADD R1, R2, R3`	`R1=R2+R3`	none
4	Indirect	`LDR R1, [Variable]` or `LDR R1, (Variable)`	`R1=*Variable`	none
5	Register indirect	`LDR R1, [R2]` or `LDR R1, (R2)`	`R1=*R2`	none
6	Index addressing	`LDR R1, 24[R2]` or `LDR R1, 24(R2)`	`R1=*(R2+24)`	none

7	Base register	`LDR R1, [R2, R3]` or `LDR R1, (R2, R3)`
		`LDR R1, [R2, #12]` or `LDR R1, (R2, #12)`
			`R1=*(R2+R3)`
			`R1=*(R2+12)`
				none

8	PC-relative	`LDR R1, [PC, #-8]` or `LDR R1, (PC, #-8)`	`R1=*(PC-8)`	none
	Pre-index mode with base updated	`LDR R1, [R2, #8]!` or `LDR R1, (R2, #8)!`	`R1=*(R2+8)`	`R2=R2+8`
	Post-index mode with base updated	`LDR R1, R2, #8`	`R1=*(R2)`	`R2=R2+8`

Table 5.1: Addressing modes for ARM processors

The table 5.1 contains an overview of address modes (i.e. instruction data access modes) for ARM processors, with an example instruction, meaning and secondary effect for each. For example, the first mode listed means that we are working directly with a numeric constant, the second with a variable (in memory), the third with the contents of a register, the fourth with a register containing the address of a variable from memory (i.e., the register serves as a pointer), etc.

Additional information

- https://wiki.cdot.senecacollege.ca/wiki/Aarch64_Register_and_Instruction_Quick_Start
- http://simplemachines.it/doc/arm_inst.pdf

5.3 Instruction Formats

Instructions are simply numbers providing information on what the processor should carry out and with what operands. The *instruction set architecture* (ISA) describes the features of an instruction set.

The length of the whole instruction and its opcode depends on the processor architecture. There are two main types of processor architectures:

- *CISC* (Complete Instruction Set Computing) – there are many instructions, for each complex operation there is an own instruction,
- *RISC* (Reduced Instruction Set Computing) – there are only "basic" instructions, all complex operations are composed of the simple instructions.

CISC instruction sets consist of lots of instructions and there are more addressing modes, so they need more bits to represent the opcode, the instructions are longer. Instructions are represented by *microprograms* stored in a special memory in the processor.

RISC instruction sets consist of less various instructions and there are less addressing modes, so their opcode is usually shorter. We don't need any special memory for instructions, they are represented by circuits. The consequence is that it usually takes one processor cycle to process an instruction (this is related to the frequency at which the processor operates).

The instruction sets for ARM processors are of the type RISC. The instruction sets for Intel and AMD processors are somewhere between RISC and CISC – they are CISC with some RISC features. The instruction sets for most server and supercomputer architectures are close to RISC.

Another typical feature of RISC sets is the non-destructiveness of operations – while CISC instructions often load the result of an operation into one of the operands, RISC often uses another operand for the result.

Example

The following operation is typical for CISC sets (the contents of the `EAX` register and the contents of the `val` variable are added, the result is stored in the `EAX` register):

```
ADD EAX, val
```

Thus, the original contents of the `EAX` register have been overwritten, "destroyed". In contrast, the following is typical for RISC sets:

```
ADD val1, val2, res
```

The instruction adds the contents of the variables `val1` and `val2`, and stores the result in the variable `res`. The original contents of the first two operands are preserved.

CISC	**RISC**
many instructions, each common operation has its own instruction	only basic instructions (few), more complex operations consist of multiple short instructions
instructions are long (and of different lengths) and have more operands, evaluation – different number of cycles for each, depending on complexity	instructions are short (usually all the same length) and simple, evaluation – (almost) one CPU cycle each
because each instruction may take different amounts of time to process, we do not know in advance how long it will take to process a particular sequence of instructions	(almost) all instructions are processed in one processor clock, so it is known in advance how long it will take to process a sequence of instructions
each instruction is represented by a microprogram, so we need space for microprograms in the processor (they take up a large part of the processor)	instructions are represented in circuits (hardwired), circuits with instructions take up little space, do not consume much CPU space
instruction fetching is slow, high overhead	instruction fetching is fast, does not delay
the decode unit is more complex, instructions have different structures with different flags	the decode unit is simple, it just works with circuits
problems when increasing the frequency – the processor does not keep up at higher frequencies	easier to increase frequency – working mostly with circuits
upgrading the processor is easier, especially reprogramming microprograms or storing new ones in memory	the processor upgrade is hardware-based, it is necessary to interfere with the circuitry

Table 5.2: Comparison of CISC and RISC architectures

RISC sets appeared in the PowerPC series processors, then in some server platforms, today they are mainly pure RISC processors in embedded and small mobile devices (ARM processors).

"Pure" CISC sets were used in older Intel processors (for example old Pentia), but nowadays Intel and AMD use CISC with RISC features (so a kind of combination) as basic instruction sets. Simple instructions (such as common arithmetic operations) are implemented in "RISC" circuits (to save space in the processor), complex instructions are in the form of microprograms. If older instruction sets (which are mostly CISC) need to be implemented in a newer processor, this is done by emulation (i.e. microprogramming).

RISC processors are therefore mainly ARM processors. Among others, the open-source RISC-V suite, as well as IBM's PowerPC or MIPS, both of which can be found today in embedded systems, the PowerPC processor is used, for example, in the Mars rover Curiosity. There is also the open specification OpenRISC.

5.4 Techniques for Increasing Processor Throughput and Performance

5.4.1 Pipelining and Scalar Architectures

✎ Pipelining is a concept that takes advantage of the fact that execution of an instruction can be divided into multiple phases (stages), and the computing processor core can execute multiple instructions at the same time when each is in a different stage.

Example

Pipelining can look like this (two possibilities, the first is a two position queue, the second is a five position queue):

1. instruction fetch and decode
2. executing the instruction and saving the result

1. instruction fetch
2. instruction decode and loading operators
3. execution
4. memory access
5. working with registers

For example, in current processors, the instruction processing cycle can be divided into the following steps (stages)

- *fetch* – fetch instruction from memory (or L1i cache),
- *decode* – decode the instruction, determine what needs to be done to execute the instruction, indirect – getting the data needed to execute the instruction, including memory access, may be included in the decode phase,
- *execute* – the actual execution of the instruction,
- *memory access* – if necessary, the results are written to memory,
- *write back* – performing register changes, including the register with the address of the next instruction to execute.

Example

Consider the simpliest case where all instructions take the same time, they all are computed in five clock cycles. Computation of each instruction consists of five stages. Figure 5.7 shows a simple pipeline with five "slots" (stages).

	clock$_i$	**clock**$_{i+1}$	**clock**$_{i+2}$	**clock**$_{i+3}$	**clock**$_{i+4}$	**clock**$_{i+5}$	**clock**$_{i+6}$	**clock**$_{i+7}$	**clock**$_{i+8}$
I$_1$	**Fetch**	**Decode**	**Execution**	**Memory Access**	**Write Back**				
I$_2$		**Fetch**	**Decode**	**Execution**	**Memory Access**	**Write Back**			
I$_3$			**Fetch**	**Decode**	**Execution**	**Memory Access**	**Write Back**		
I$_4$				**Fetch**	**Decode**	**Execution**	**Memory Access**	**Write Back**	
I$_5$					**Fetch**	**Decode**	**Execution**	**Memory Access**	**Write Back**

Figure 5.7: Example of a 5-stage pipeline with five instructions inside

It is possible to have multiple pipelines too, either working symmetrically (all of the same length, all for any type of instruction) ar asymmetrically (different length, specialized for some type of instructions).

In the real world, there are different kinds of instructions (integer, floating-point, vector multimedia, branch instructions,...) that can be of different lengths (number of cycles per computation).

Pipelines can be either common for different types of instructions (symmetric pipelines), or specialized for a particular type of instruction.

The following terms are used:

- Scalar architecture (pipelining) – one instruction is processed sequentially by different parts of the processor, which are chained together (queued), several instructions are in parallel in different stages of processing, Superscalar Architecture – there is more than one such concatenation (multiple queues),
- *Hyperscalar architecture* – extending queues (even several tens of steps).

Almost all current processors are superscalar, many are also hyperscalar.

For example, Intel processors support pipelining with a pipeline length between 14 and 20, often two pipelines, depending on microarchitecture. ARM processors have multiple pipelines with very different lengths (1 or 2 stages for integer instructions, much more for other instructions).

Example

Figure 5.8 shows pipelines used at ARM Cortex A55 (mainstream processor). There are multiple pipelines there (inside cores), for different types of instructions.

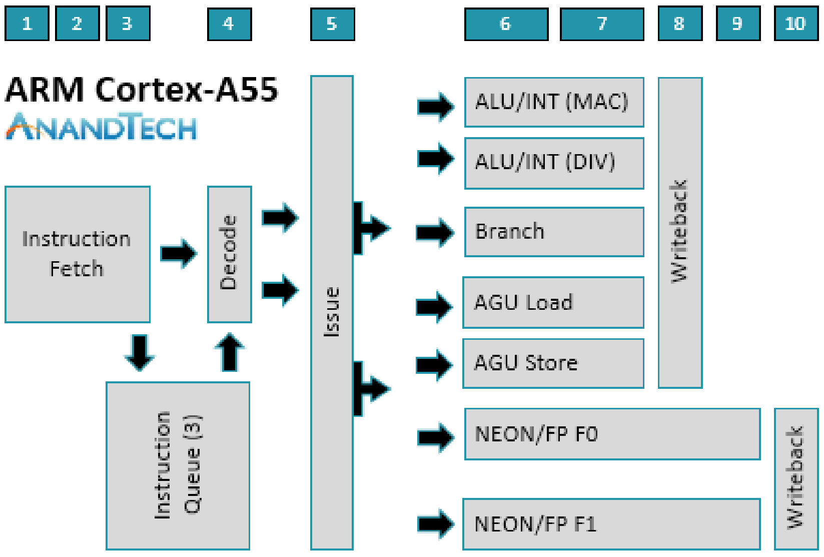


Figure 5.8: Pipelines at ARM Cortex A55[4]

The stages Fetch (3 ticks), Decode (1 tick) and Issue (1 tick) are in all pipelines, the Writeback stage (1 tick) as well. The Issue stage performs register file reads.

Two pipelines are intended for integer instructions; they both provide simple instructions (addition, shifting, etc.), but only the first one provides multiplication (MAC means Multiply Accumulate), and only the second one provides division (DIV). The last two pipelines (NEON/FP) are intended for floating-point operations.

AGUs (Address Generation Units) execute the `LOAD` and `STORE` instructions (one pipeline for `LOAD`, one pipeline for `STORE`).

[4]From: https://www.anandtech.com/show/11441/dynamiq-and-arms-new-cpus-cortex-a75-a55/4

Additional information

- https://www.anandtech.com/show/11441/dynamiq-and-arms-new-cpus-cortex-a75-a55 (five chapters)
- https://www.arm.com/products/silicon-ip-cpu (AMD processors overview)

5.4.2 HyperThreading, Multiple Cores, Multiprocessing

Multiprocessing means multiple processors in one system. Multiple processors on a single motherboard are more common on server platforms (there must be multiple sockets on the motherboard), on desktops and mobile devices we meet with a single multi-core processor. Multiple processors (and actually cores) should be supported by the operating system in order to be used at all:

- *monoprocessor* (monoprocessor) – Windows with DOS kernel (version 9x, ME),
- *multiprocessor* (multiprocessor) – UNIX systems including Linux, Windows with NT kernel (NT, 2000, XP, Vista, 7, 8, 10), can schedule at least some tasks so that they can be processed on multiple processors simultaneously.

There are two types of multiprocessing, depending on whether or not the individual processors are used universally by any process:

- asymmetric (ASMP) – one processor is dedicated to system processes and user processes run on the other processors,
- symmetric (SMP) – any process can run on any processor.

All current operating systems support SMP.

Multiple cores. The processor core consists of the ALU, controller, registers, decode unit and other important support components. A multi-core processor is a processor that contains multiple (physical) cores integrated on a single chip, as well as logic designed to interconnect them.

It takes advantage of the fact that a few simple cores on a single silicon wafer have higher computational performance at a lower cost than one complex and super-powerful one on the same area, or even several physical processors (which would take up more area).

Current processors are usually multi-core. Usually the number of cores is a power of 2 (2, 4, 8, etc.), but we can also see odd numbers, for example 3. Typically, some core is defective, deactivated, but the others are fine, so why throw away such a processor.

Hyperthreading. A single processor (or core) is capable of executing the code of several computational threads (threads, for example, the code of several processes). There are several different implementations of hyperthreading.

Intel, AMD and ARM processors use a variant of SMT (Simultaneous MultiThreading, Intel calls it simply HT, HyperThreading, AMD directly by the abbreviation SMT): one physical core is reported to the system as two logical cores with shared resources, respectively if there is more than one physical core in the processor, the number of logical cores is doubled (so, for example, the operating system " sees" two physical cores as four logical ones).

The processor must be superscalar (i.e. multiple pipelines), two pipelines working in parallel. Each pipeline may process a computational thread of another process (or, in the case of a multi-threaded process, its different threads). This feature must be enabled in the BIOS (it usually is). The performance increase is quoted especially for multi-threaded applications in tens of percent.

It is important to note that if two (or more) different computational threads, even from different processes, are to be executed in parallel on the same processor core, their activity must be completely separated – for example, they use separate address spaces, i.e. under the same address (as a number) each thread "sees" something else from its own address space, each thread has its own register content that cannot be overwritten by another thread, etc. So whenever the operating system determines that the pipeline should switch to another computational thread (e.g., another process "receives" a processor/core), the so-called *context switching* takes place – the register contents, including the address ones from the original thread, are stored and the register contents of the next thread are loaded (this happens even without multithreading, of course, but only in aggregate for the entire physical core; with multithreading, separately for logical cores).

interleaved hyperthreading works a bit differently, which (unlike SMT) makes sense even if there is only one pipeline in the kernel. There are two basic forms:

- Fine-grained multithreading (with fine granularity) – switches between multiple compute threads within the pipeline, even at each tick (but depending on the architecture, this can be in units to tens of ticks). Each compute thread whose instructions are processed must have a place to store its state, i.e. its own instance of the register file. This type of hyperthreading has appeared in some server architectures, e.g. Sun Ultra-SPARC T1 (4 threads in each core at the same time, switching between threads at each tick), in later generations it was switched to SMT.
- *coarse-grained multithreading* (with coarse granularity) – similar to the previous one, but only switches between threads during "hardware events", e.g. cache miss, TLB miss, synchronization events, memory page fault, etc. This type of hyperthreading was implemented, for example, in some Intel Itanium (Montecito) server processors, but these are long out of production.

5.4.3 Data Dependencies

While pipelining helps to make the best use of CPU capacity, it also brings problems because successive instructions can be data-dependent: for example, the next instruction needs the result of the previous instruction, but when their computation overlaps, the next instruction must wait.

	clock$_i$	**clock**$_{i+1}$	**clock**$_{i+2}$	**clock**$_{i+3}$	**clock**$_{i+4}$	**clock**$_{i+5}$	**clock**$_{i+6}$	**clock**$_{i+7}$	**clock**$_{i+8}$
$\mathbf{I}_m$	**Fetch**	**Decode**	**Execution**	**Memory Access**	**Write Back**				
$\mathbf{I}_{m+1}$		**Fetch**	**Stall**	**Stall**	**Stall**	**Decode**	**Execution**	**Memory Access**	**Write Back**

Figure 5.9: Instructions with a data dependency on the previous one

We can see the situation in Figure 5.9 (the instruction $\mathbf{I}_{m+1}$ is data dependent on the previous instruction). We call such a situation *hazard*.

- *data hazard* – the necessary data is not really accessible at the right time,
- *structure hazard* – hardware resource conflict, for example a hardware component does not allow a certain combination of the following consecutive instructions:

```
MUL R1, R2, R3        // means R1 = R2*R3
MUL R4, R5, R6        // means R4 = R5*R6
```

Seemingly everything is ok. But the multiplier in this architecture needs two steps to multiply two numbers, meanwhile it cannot multiply other numbers, so the second multiply instruction has to wait one step,

- *control hazard* – caused by a conditional branching problem (for example, in the source code we had an if, switch, while loop or something similar) where there are at least two different paths in the code that can be taken, and the processor gets uncertain when guessing the correct path.

The "data hazard" situation is just the result of data dependency. For example, the above situation could be caused by a sequence of these instructions (the code is for an ARM processor):

```
ADD R2, R4, #150        // means R2 = R4+150
ADD R3, R2, #84         // means R3 = R2+84
```

The first instruction stores the result in R2, the next instruction needs it for its calculation. However, the Decode phase of the second instruction (which retrieves the value from R2) can only do this after the Write Back phase of the previous instruction.

A partial solution is to use the *interstage buffer* mechanism, which are memory locations (similar to registers) used at the beginning or end of each phase, used for "early transfer" of values between instructions. For example, according to the figure 5.9, at the end of the Execution phase of the first instruction, the contents of register R2 would be transferred via the interstage buffer to the next instruction, reducing the number of empty steps from three to one.

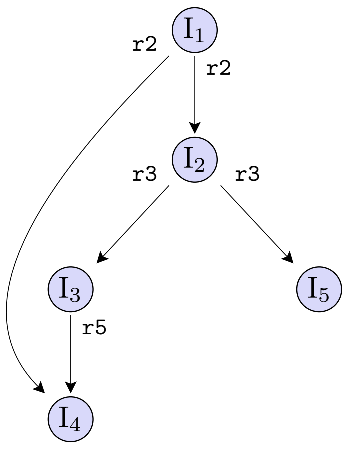
Dependency graph (Data-flow graph) is a graph showing data dependencies between instructions, as we can see in the following example. The nodes of the graph are instructions, the edges show dependencies.

The number of edges in the graph (as well as the captured instructions) depends on the length and structure of a given pipeline, the set of edges dynamically changes as instructions pass through the pipeline. An instruction remains dependent as long as its parameters are in *inconsistent state* (i.e., waiting to be supplied by another instruction), after which at least one edge leads to it. When the dependency ends, the edge is removed, and if no edges lead to the instruction (i.e., all parameters are in *consistent state*), the instruction can proceed in its pipeline.

Example

Consider the following sequence of ARM instructions:

```
I1:  LDR R2, [R4]
I2:  ADD R3, R2, #150
I3:  SUB R5, R3, #80
I4:  ADD R4, R5, R2
I5:  LDR R3, =280
```



The first instruction changes the value of the R2 register, and then the I_2 and I_4 instructions use this value. So the two instructions are dependent on the I_1 instruction.

There are similar relationships between other instructions, all of which can be seen in the graph on the right.

Data dependencies are handled as follows (we don't want the pipeline to remain unnecessarily empty):

- stalling – we add "empty" intervals (NOP operations, also called "bubble"), which is not an ideal solution, either done by the processor or using a software solution,
- using interstage buffers – described above before the example,
- operand forwarding – bypassing registers and buffers including interstage buffers, values are passed between instructions inside the working component (e.g. ALU, FPU), it is faster and within the current step (tick),
- instructions reordering – often (not always) it is possible to reorder pending instructions so that interdependent instructions do not follow each other too quickly,
- software solution – similar to the previous one, the reordering (optimization) is done by a compiler.

Example

As mentioned in the first option, the insertion of empty loops can also be solved in software during compilation into executable code, using the "empty" NOP (Not-operation) instruction.

Original instructions:

```
ADD R2, R4, #150
ADD R3, R2, #84
```

Modified instructions:

```
ADD R2, R4, #150
NOP
NOP
NOP
ADD R3, R2, #84
```

Example

The implementation of the third option (operand forwarding, register bypassing) depends on the specific architecture. Figure 5.10 shows two options, the first one is obviously more optimal and is applicable if the ALU (which is performing the calculation) can calculate the result of the first instruction and use it to prepare the parameters of the second instruction the both in a single step.

Data dependency types. The example on page 85 shows the dependency graph. This graph (suitably implemented) can be used for dependency detection. The goal is to have as few edges in the graph as possible, and at the same time as few bubbles in the pipeline as possible. Instruction reordering procedures vary depending on what type of "data hazard" problem is involved (ARM instructions follow):

- *RaW* (Read after Write, True Data Dependence, Flow Dependence) – example:

  I_1: `LDR R2, [R4]`
  I_2: `ADD R3, R2, #150`

  if these two instructions are not in the correct order or the I_2 instruction is executed too fast, the result of the calculation is incorrect,
- *WaR* (Write after Read, Anti-dependence) – the sequence of operations is reversed from the previous example:

  I_1: `ADD R4, R2, R3`
  I_2: `SUB R2, R5, #50`

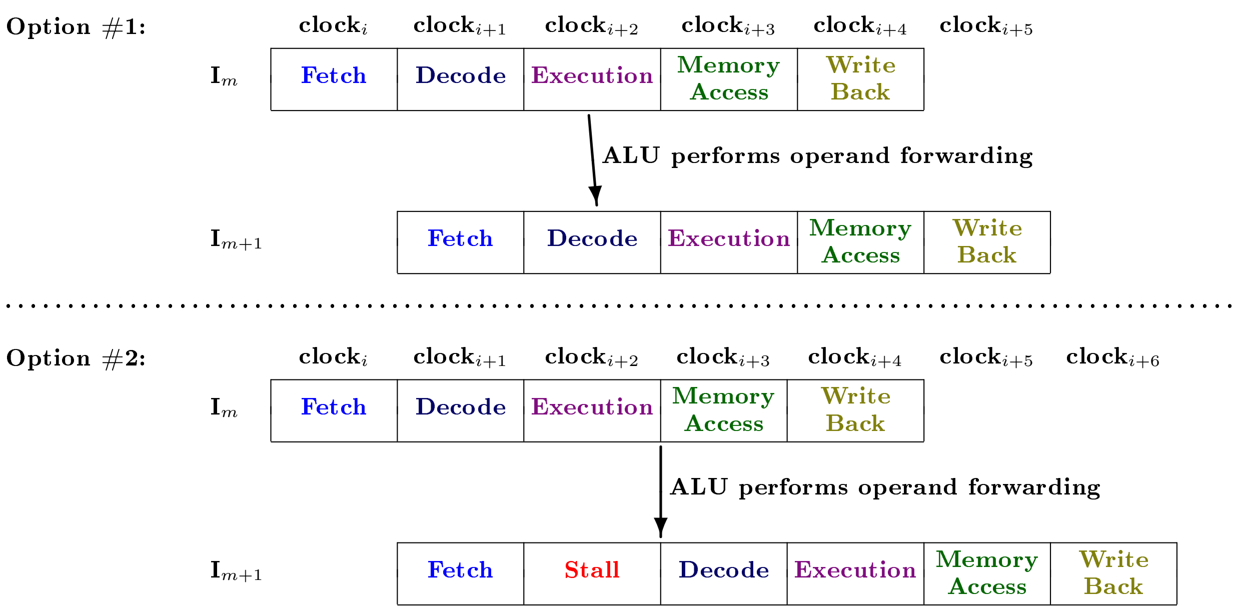


Figure 5.10: Operand forwarding

the `SUB` (subtract) operation must be performed after I$_1$ (first `R2` is read by the I$_1$ instruction, then it is overwritten by the I$_2$ instruction), otherwise we get the wrong result,

- *WaW* (Write after Write, Output Dependence) – both instructions are written to the same location, for example

  I$_1$: `ADD R3, R2, #10`
  I$_2$: `ADD R3, R5, R2`

  this sequence may look different, in practical terms more likely as follows:

  I$_1$: `ADD R3, R2, #10`
  I$_2$: `ADD R3, R3, R2`

  in the second case we even have two types of dependencies at the same time: WaW a RaW.

When solving the "data hazard" problem by rearranging instructions, it should not happen that after removing one type of dependency another type of dependency is created.

5.4.4 Unconditional and Conditional Jumps

Unconditional Jumps. In programs there is a lot of branching, loops, calls of exit/break/continue instructions, jumps to other parts of code (including function execution and return after function termination), etc., which makes pipelining a bit more difficult. Unconditional jumps in code are created either by using `break` inside a loop or branching, or by calling a function or, conversely, returning after a function has been executed.

Example

The following two code fragments are similar, one in ARM, the other in Intel. First is the add instruction, second is the jump instruction (which is supposed to take us to a label somewhere further down in the code), then in the sequence we have the instruction that we should correctly skip.

We have the following code for ARM:

```
        ADD R2, R3, #50
        B someLabel
otherL: SUB R4, R5, [R6]
        ...
someLabel:
```

We have the following code for Intel:

```
        ADD EAX, 50
        JMP someLabel
otherL: SUB EBX, [EDI]
        ...
someLabel:
```

When executing the second instruction in the sequence (`B`/`JMP`), the jump target is known only after several steps in the pipeline (first fetch, then decode, and only in the third stage of the first instruction do we find out the target). But the next instruction goes into the pipeline only one step later (because "some" instruction has to go there), and so does the third one in the next step (the jump is supposed to send us somewhere else, but we don't know where yet, so we do `SUB`). Because of the delay in execution, the next two instructions in the pipeline are delayed/rejected.

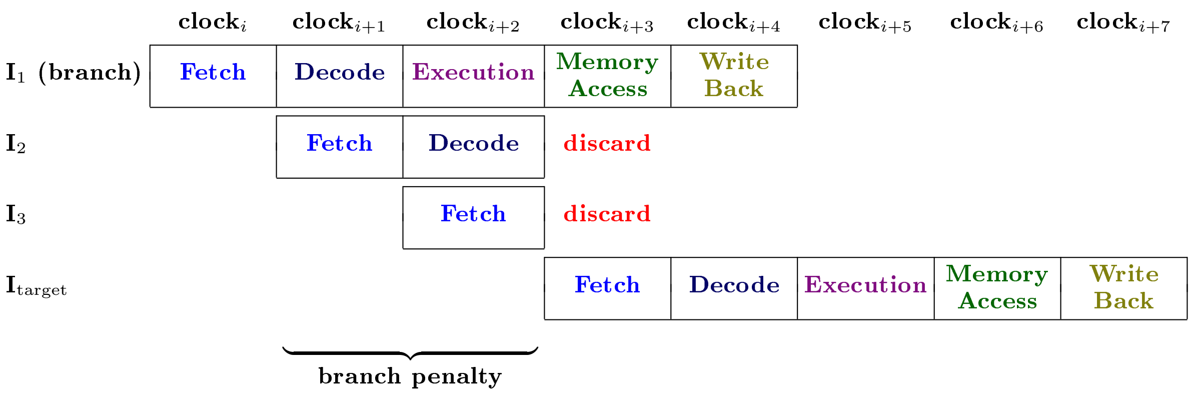


Figure 5.11: Branch penalty, the address of the jump target is passed to the PC/IP register after the Execution phase

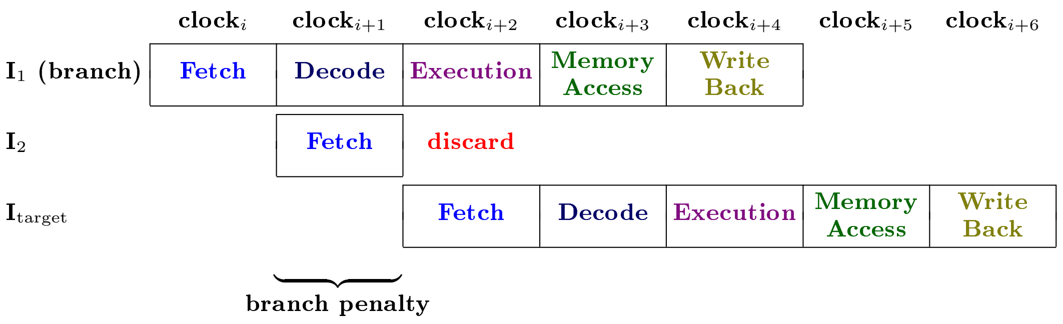


Figure 5.12: Branch penalty, the address of the jump destination is passed to the PC/IP register during the Decode phase

The number of steps taken that will not actually be used is called *branch penalty*. Too high a branch penalty degrades CPU performance, so it is a good idea to address this parameter. There are two possibilities:

- calculating the jump target at an earlier step in the pipeline and transferring it within the component,
- reordering the instructions so that the instruction before the jump (in the example `ADD`) is moved after the jump instruction, i.e. we evaluate the jump target first, start the addition one step

further (we will execute the instruction for sure, we will not throw it away), in the meantime the jump target will be detected, so we will know where to jump.

The first option assumes that the PC or IP register (depending on the architecture – ARM or Intel) is updated for the jump instruction at the decoding stage. Compare Figures 5.11 and 5.12. The first image corresponds to an architecture where the PC/IP register (which should contain the address of the next instruction to be executed) is updated after the `B` or `JMP` instruction execute phase, respectively, while the second image shows a "faster" architecture where the register is updated at the end of the decode phase.

If we also use the second solution (swapping the order of the jump instruction and the original previous instruction), we won't even have to throw anything away in the "faster" architecture. Of course, the condition is that the swapped instructions must not be mutually dependent.

✎ If instruction flow control depends on the result of a jump instruction that has not yet left the pipeline, we call this situation *control hazard*.

✎ **Conditional code branching** (conditional jumps) is the result of either simple branching (machine code created by compiling `if`, `switch`, etc.) or loop implementation (`while`,...).

Example

Again, this is followed by a section of code for Intel processors (it would be similar for ARM, except for the registers and number format). The first instruction multiplies the register contents by 25, the second instruction compares the contents of the two registers and stores the result in the flag register. The third instruction checks the flag register to see if the first value compared is greater than or equal to the second (JGE – Jump if Greater or Equal), and if so, jumps to the specified flag.

```
           SUB ebx, 25
           CMP eax, edx
           JGE someLABEL
           ADD eax, 32
           ...
someLABEL: MOV edx, eax
```

Obviously, a comparative instruction must first be carried out `CMP` and then `JGE` (Jump if greater or equal), because we have a dependency. The compare instruction stores the value in a register, which the jump instruction then reads from. But again, writing to the flag register must be done before the jump instruction reads from it. Unlike an unconditional jump, here we cannot swap the two instructions.

But there is still the `SUB` instruction before testing. If we can't swap `CMP` and `JGE`, then we insert `SUB` between these bound instructions and we have a sequence of `CMP`, `SUB`, `JGE`. The `SUB` instruction also writes to the flag register, but at a later step than `JGE` reads the result of the comparison from this register. So again we have saved at least one stage.

If there was an instruction before `CMP` that could not be swapped, we would have to put an empty Stall step after `CMP` instead, or a NOP instruction in the software solution.

It remains to solve the same problem as with the unconditional jump, i.e. "collision" of write and read accesses for the PC/IP register with the address of the next instruction to execute. The following `ADD` instruction will either be skipped (if testing succeeds) or executed (if testing ends up false). So we have about a 50% chance that if we start executing this instruction, we won't have to discard it later. A 50% chance is still better than unnecessarily inserting empty intervals into the pipeline.

So if the testing ends up false, the branch delay is zero, and if the testing ends up true, we discard one or more steps and continue with the `MOV` instruction at the jump target.

Branch Prediction. The previous example suggests that conditional jumps are likely to work with probabilities. The probability that the jump does not come is about 50 percent, but it moves up or down according to certain rules:

- once we're inside the loop (for example, `while`), we usually do it more than once, even more than a thousand times,
- it happens, however, that the condition is not met even on the first test, so we don't get inside the loop at all,
- or if the loops are nested, the previous two factors are combined.

A simpler algorithm for predicting the result of a branch prediction is that the instruction uses a bit (so two values) taking either the value "likely to jump" or the value "likely not to jump". At first it is in the "will not" state (the probability is less than 50 %), but once inside the loop the bit takes on the value "will" as the probability of staying inside increases above 50 %.

Our system switches between two states:

- if the jump probability is greater than 50 %, i.e. continuing in the current low branch, we skip the loop (LNT – branch Likely Not Taken),
- jump probability less than 50 %, so we probably stay in the branch we are in and go inside the loop (LT – branch Likely Taken).

Transitions between states can be captured as arrows, the arrow is labeled with a label specifying the cause of the transition. For example, by the time we get to the loop, we are in the LNT state (probably skipping the loop). If the first time the condition was evaluated as true and we need to go in, we are likely to stay inside the loop for some time, i.e. move to the LT state.

We plot the resulting scheme as a graph, which we call a finite automaton (because it has a finite number of states) and we can see it in the figure 5.13.

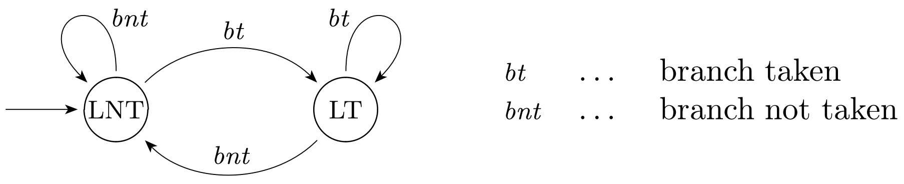


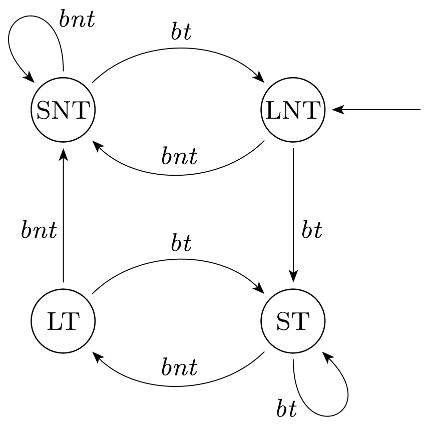
Figure 5.13: Finite automaton for a simple branching prediction algorithm

The more advanced branch prediction algorithm is more complex, it uses loop nesting and instead of two states (less than 50 %, greater than 50 %) we have more states.

If, according to the previous simple algorithm, we repeatedly come to the same loop (because we repeatedly execute the "supervised" loop), then we actually repeatedly "jump in probabilities". So instead of two states, we will have four for different probabilities of staying in the current branch (code sequence), as we can see in the figure 5.14.

- LT – branch is likely to be taken (something above 50 %),
- ST – branch is strongly likely to be taken (well over 50 %),
- LNT – branch is likely not to be taken (just under 50 %),
- SNT – strongly likely not to be taken (well below 50 %).

In the implementation we need two bits instead of one bit, which gives four states.



bt	...	branch taken
bnt	...	branch not taken

Figure 5.14: Finite automaton for a more complex branching prediction algorithm

Well, but we need to have that one or two bits stored somewhere, and separately for different conditional branching instructions, because for such an instruction we need to know if we have been there before or if we are coming to it for the first time (possibly in combination with information about the parent loop). Such a place is called *branch target buffer* (i.e., the branching target buffer), and for each branching instruction we have recently passed through or otherwise discovered, we have this information here:

- address of the branching instruction,
- one or two bits for the algorithm described above,
- the address of the jump target so that if we have to jump, we can react immediately.

So when such an instruction is executed, in phase 1 (fetch) of the pipeline, it is checked if there is an entry in the branch target buffer for the instruction. If not, it is created during the execution of the instruction (the jump target is added last) and the initial LNT state is set. When an entry already exists, it is used according to the algorithm. However, the buffer is not bottomless, the records are continuously flushed.

Additional information

- https://slideplayer.com/slide/5019089/
- http://www2.cs.siu.edu/ cs401/Textbook/ch3.pdf
- https://en.wikibooks.org/wiki/X86_Disassembly/Branches

Chapter 6

Memory System

Quick preview: In this chapter, we will discuss the memory system – types and properties of memory chips and storage media. We'll focus on the mechanism they work on, different designs we can encounter, how to test them, and how to affect their properties.

Keywords: Memory, memory access time, memory cycle time, RAM, SRAM, DRAM, SDRAM, CMOS, DDRx, GDDRx, volatile, non-volatile, cache, address decoder, word line, bit line, latency, bandwidth, module, DIMM, SO-DIMM, bank, multi-channel, SPD chip, XMP profile, parity checking, ECC, registered, RDIMM, UDIMM, ROM, PROM, EEPROM, NOR Flash, NAND Flash, SD, eMMC, SSD, SLC MLC, TLC, QLC, TRIM, cache, storage, disk drive, spindle, track, sector, cluster, advanced format, LBA, parking, PMR, SMR, AHCI, Interleaving, S.M.A.R.T., RAID, optical drive.

Objectives: After studying this chapter, you will know how the memory semiconductor chips basically work, get an overview of the different types and their properties and uses, and be able to test memory. In addition, you will gain an orientation in the issue of storage media.

6.1 Semiconductor Memory Properties

6.1.1 Semiconductor Memory Types

When performing transfer of a word of data from/into a memory, a *memory access time* is the time between the initialization and the completion of this operation. A *memory cycle time* is the minimum time delay between the initiation of two successive memory operations.

There are the following types of memory chips:

- *RAM memories* – volatile (energy-dependent) memories
  - static (SRAM)
    - CMOS (subtype of SRAM)
  - dynamic (DRAM), evolution:
    $\rightarrow$ Fast Page Mode (FPM) $\rightarrow$ synchronous (SDRAM) $\rightarrow$
    $\rightarrow$ double-data-rate SDRAM – versions: DDR, DDR2, DDR3, DDR4
  - graphics RAM chips – in use: GDDR5, GDDR5X, GDDR6, HBM, HBM2

- *ROM memories* – non-volatile memories
- *NAND Flash and NOR Flash chips*

A *cache memory* is a small, fast memory between two components with very different access times, its role is to reduce the access time of the slower component which is the bottleneck of the communication. These component pairs are usually (slow) main memory and a (fast) processor, or a (slow) hard disk and a main memory or processor (faster components).

A cache memory between a processor and a main memory usually holds the currently active instructions and data (that are likely to be needed in the near future).

A *virtual memory* is a method of virtual increasing the amount of physical memory, completely transparent from the point of view of running processes. The part of the address space that is currently being used, is located in the physical memory, the rest may be postponed to a storage, typically a hard disk.

Processes do not distinguish between memory blocks located in the main memory and those that are deferred on disk, it is a fully transparent access mechanism. A very important consequence of using virtual memory is also to protect the memory space of each process and the kernel because each address access is done by address translation, processes can only address the memory that belongs to them.

Remark

These two concepts – a cache memory and a virtual memory – have something in common: this memory must include, in particular, what it is expected to be used in the near future (by processor in case of the cache, by the process in case of the virtual memory).

6.2 Semiconductor RAM Memories

The term of RAM (Random Access Memory) is derived from the fact that access to any address in this memory always takes the same amount of time, so it is not necessary to go through all the addresses between the current and the desired location.

6.2.1 Internal Organization of Memory Chips

The main part of a memory chip is a *grid of memory cells* to store particular bits, each memory cell can store one bit. In the grid, each row constitutes a word, number of columns corresponds to number of bits in one word.

Other parts of a memory chip are

- *address decoder* – this component receives an address (one number, the number of bits depends on the address bus width) and determines the word to be processed,
- *sense/write circuits*, one for each column of a cell, they provide output depending on whether read or write operation is executed.

Internally in a memory chip, there are

- *word lines* – one word line is located between the address decoder and all cells of one word, each row has its own word line, a signal on a word line means that a given operation will be performed with this word,

- *bit lines* – they connect the memory cells of different words in the same position (= the same column of the grid, e.g. all first bits of different words, etc.) with the Sense/Write Circuit, each cell column has one or two bit lines,
- other lines leading from the Sense/Write Circuit to output read data, control lines, lines for power supply, etc.

The designation of a simple memory chip is words × bits in word, so rows × columns.

Example

Suppose a memory chip with the capacity 500 MB, the word length is 64 bits.

500 MB means $500 \cdot 1024 \cdot 1024 \cdot 8$ bits, divided by 64:

$$\frac{500 \cdot 1024 \cdot 1024 \cdot 8}{64} = 65\,536\,000 \text{ words}$$

So, the designation of this chip is $65\,536\,000 \times 64$, or $64\,000\text{K} \times 64$, or simply $62.5\text{M} \times 64$.

6.2.2 Static and Dynamic RAM

Static and dynamic RAM have differently implemented memory cells.

In a *static RAM* (SRAM) concept, each memory cell is connected to one word line and two bit lines, and it consists of two transistors and two inverters (in the standard type of implementation). Static RAM chips are more complicated than dynamic ones, more expansive, but faster. We can meet them mainly as cache memories.

A *CMOS memory* is a subtype of SRAM, but its memory cell is differently implemented – there are six transistors and no inverters in a CMOS memory cell.

CMOS chips are volatile (as common for all RAM types), but have very low power consumption, they have enough a small battery, if the main power source (power supply unit) is not available. At least one CMOS chip is in every mainboard – the CMOS chip with BIOS configuration.

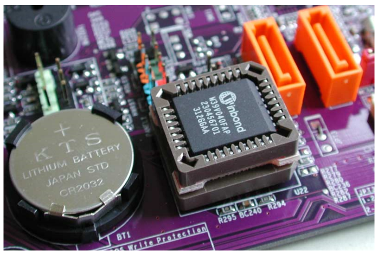


Figure 6.1: CMOS chip with BIOS configuration[1]

Remark

CMOS chips are also used for RTC (Real-Time Clock) in older devices, connected to a battery as well. RTC is a special integrated circuit keeping track of the current time in human units. The real time is important for each device – to display the time for a user correctly, but also to synchronize transmission across interfaces, including network interfaces. GPS modules also need RTC to be able to correctly determine its location.

In the current mainboards, RTC is mostly integrated in the southbridge chip (the "south" part of the mainboard chipset).

[1]From: https://sites.google.com/site/kenngo2306/home/assignment-1/bios-and-cmos

✎ In a *dynamic RAM* (DRAM) concept, each memory cell is connected to one word line and one bit line, and it consists of one transistor and one capacitor. Dynamic RAM chips are simple, cheap, but slow. The typical use is the main memory chips.

The DRAM memory controller uses a variety of signals (operations) when performing read and write operations. The main operations are RAS (Row Address Strobe) to determine the row in the memory cell grid, and CAS (Column Address Strobe) to determine the column in this grid. One row and one column defines the specific memory cell.

Capacitors used in DRAMs unfortunately do not hold the charge for long time (only tens of milliseconds). Therefore, it is necessary to refresh this charge periodically. To do this, we have another signal – RF (Refresh). Additionally, the cell content must be restored at each reading by RP (Row Precharge) – this type of memory is *destructive when reading.*

✎ When programming, we usually work with bits in the whole word, respectively, we usually use data types with a length of multiple bytes. This means that changing the row in the memory cell grid is actually much less common than changing the column. If we stay on the same line, it is not necessary to send again the RAS signal in each operation, just change the column (CAS signal). This mode of operation is called *Fast Page Mode* (FPM) and is common in current memory chips.
The basic type of DRAM with FPM is asynchronous, it means that the memory controller has read and write operations fully under control, and it also determines the timing (how long time each operation takes). The processor gets the result depending on the memory access time. The memory controller also handles the refresh.

6.2.3 Synchronous Dynamic RAM

✎ A Synchronous DRAM (SDRAM) is a variant of DRAM with FPM where all operations (signals) are synchronized with the timer, i.e. the duration of each operation is aligned with the time signal, its length is given in the number of cycles.

The refresh signal is provided internaly in a chip, not by a memory controller. When reading more memory cells in one row, the memory controller generates only one CAS signal (after RAS) to move into the first cell of the row, and the next CAS signals are generated internaly.

The main advantage is that SDRAM is able to communicate with a processor more optimally, the communication is faster.

✎ The *memory latency* is amount of time it takes to transfer the first word of a block of data. The *bandwidth* is a number of bits or bytes transferred in one second.

Each sinusoidal signal has a rising edge and falling edge. SDRAM chips communicate only at the rising edge of the signal.

✎ **Double-Data-Rate SDRAMs** (DDR SDRAM, or shortly DDR) can communicate at the both edges of the signal in the memory bus, so in comparison with the original SDRAMs they communicate twice as fast.

The older single-data-rate type of SDRAMs is sometimes denoted by SDR.

6.2.4 Forms and Generations of DDRs

Currently, the third and fourth generation are used: DDR3 and DDR4. They exist in the form of DIMM (Dual In-line Memory Module) and shortened SO-DIMM (Small Outline DIMM) for laptops and all-in-one computers.

✎ For the third and fourth generation memory modules, we may encounter a *low-voltage* variants that operate at lower voltages than conventional modules. We know them by the letter "L" at the end of the type designation, such as `DDR3L` for the third generation. Specifically, for the third generation, these modules operate at 1.35 V (the standard modules operate at 1.5 V). If there is the specification "dual voltage" in the documentation, the module can handle both voltages.

The advantage of these modules is more stable operation even at lower frequencies, but above all less heat during operation, which is an advantage for devices that are harder to cool and do not have large coolers (laptops, minicomputers). The result is lower power consumption, because more efficient active cooling also requires power.

☞ **Remark**

Some memory controllers and mainboards support both DDR3L and DDR4, or support DDR3 and DDR3L. But it does not mean that both supported types can be operated at the same time. If this is not explicitly permitted in the documentation, it is usually not possible to use modules with different operating voltages (as is the case with DDR3 vs. DDR3L), and the computer would not even start up in this case. ☜

✎ **JEDEC** (Joint Electron Devices Engineering Council) is one of the engineering trade organizations, it brings together a number of world technology companies and is involved in creating different technology standards. This organization, among other things, creates standards for memories.

Chips	JEDEC labeling	Labeling by chips	Examples

DDR	PCxxxx	DDRyyy	PC1600 =DDR200
			PC2700 =DDR266
			PC3200 =DDR400

DDR2	PC2-xxxx	DDR2-yyy
		or DDR2-yyyy
			PC2-4200 =DDR2-533
			PC2-6400 =DDR2-800
			PC2-8500 =DDR2-1066

DDR3	PC3-xxxx
	or PC3-xxxxx
		DDR3-yyy
		or DDR3-yyyy
			PC3-6400 =DDR3-800
			PC3-8500 =DDR3-1066
			PC3-10600=DDR3-1333

DDR4	PC4-xxxxx	DDR4-yyyy	PC4-12800=DDR4-1600
			PC4-19200=DDR4-2400
			PC4-25600=DDR4-3200

where xxxx is transfer rate (MBps)
and yyy is doubled frequency (bus clock, MHz)

Table 6.1: DIMM labeling

6.2.5 Properties of DDRs

✎ A *bank* is a collection of memory slots that are addressed together, for example, 4 slots can be in 2 banks. One of the banks containing a memory module on the same bus is *active*, that is, we can work with it.

Memory controllers were formerly integrated into the north bridge of the chipset, and today they are typically integrated into the processor. A memory controller can be *multichannel*, which means that it can communicate with multiple banks at once. Today, in most processors, we have a dual-channel or four-channel controller, with one channel typically representing a 64-bit communication path (so the data bus width is typically $2 \times 64 = 128$ bits in summary).

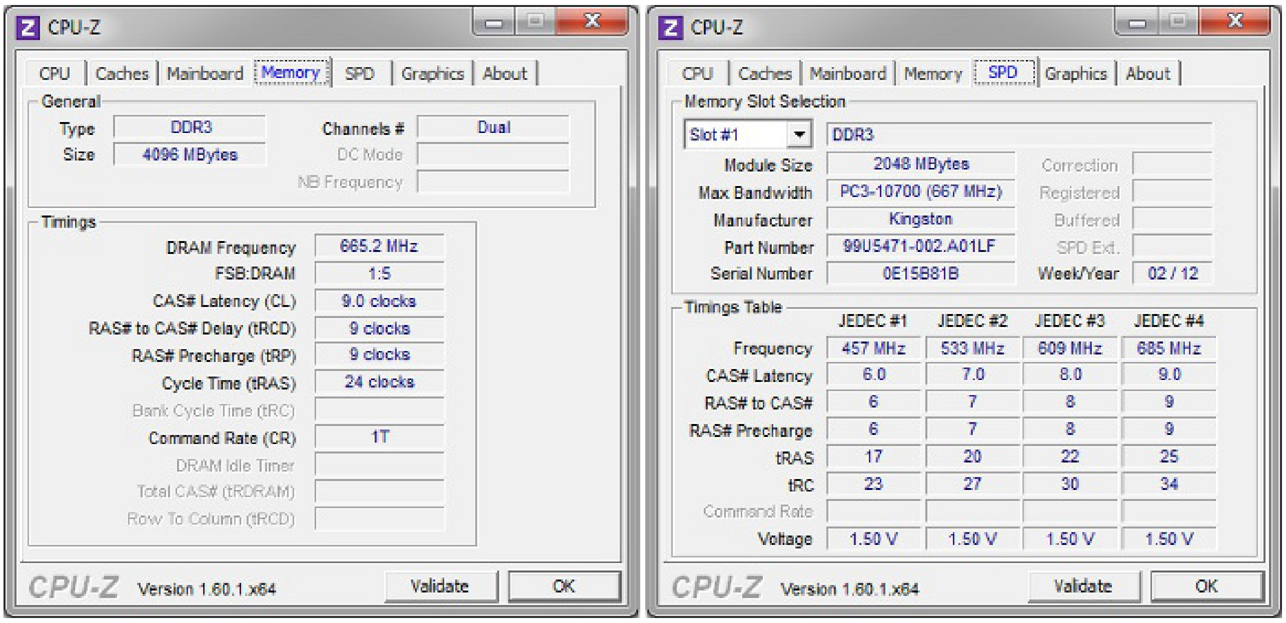
On a mainboard, it is mostly color-coded, which banks are on the same bus, but the way of marking is to be found in the documentation – some manufacturers denote the same color for the banks on the same bus, others the same color for the banks on different busses.

☞ **Remark**

Why is it important? If we want to use two DIMMs, we should plug them into slots on different busses to be in different banks. If we had both modules on the same memory bus, it would not be possible to read from both modules at the same time, the memory would have worse latency.

It is also necessary that both the modules used are of the same type (identical, including the frequency) and there must be only a couple or two pairs (if we use an odd number or otherwise do not meet the specified conditions, the feature of the dual channel will not be used). In order to make full use of the quad-channel property, of course we would have four identical and equally configured modules. ☜

✎ The *SPD chip* (Serial Presence Detect) contains memory information stored by the manufacturer (manufacturer, date of manufacture, working frequency, timing, working voltage, latency for a certain frequency of memory, etc.), of the 128-B length.


CPU-Z
CPU | Caches | Mainboard | Memory | SPD | Graphics | About
CPU-Z Version 1.60.1.x64
Validate
OK
CPU-Z
CPU | Caches | Mainboard | Memory | SPD | Graphics | About
CPU-Z Version 1.60.1.x64
Validate
OK

General

Type	DDR3	Channels #	Dual
Size	4096 MBytes	DC Mode	
		NB Frequency	

Timings

DRAM Frequency	665.2 MHz
FSB:DRAM	1:5
CAS# Latency (CL)	9.0 clocks
RAS# to CAS# Delay (tRCD)	9 clocks
RAS# Precharge (tRP)	9 clocks
Cycle Time (tRAS)	24 clocks
Bank Cycle Time (tRC)	
Command Rate (CR)	1T
DRAM Idle Timer	
Total CAS# (tRDRAM)	
Row To Column (tRCD)	

Memory Slot Selection

Slot #1 DDR3

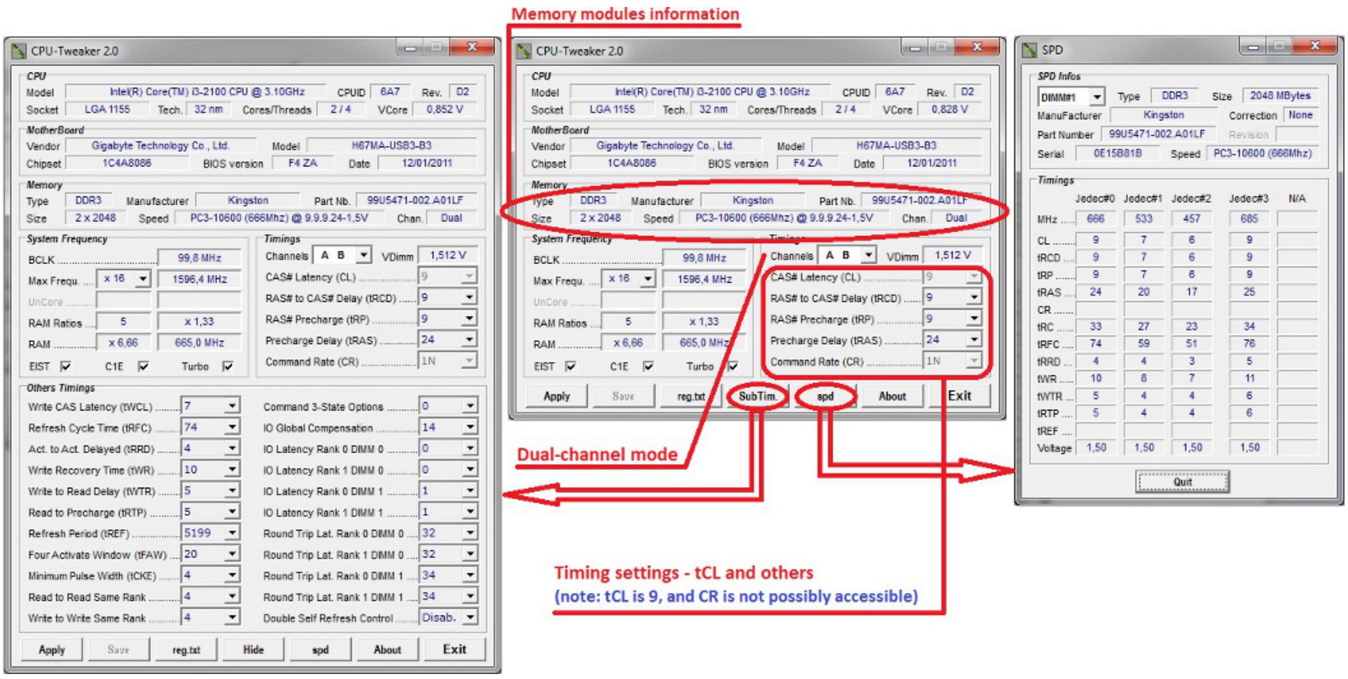
Module Size	2048 MBytes	Correction	
Max Bandwidth	PC3-10700 (667 MHz)	Registered	
Manufacturer	Kingston	Buffered	
Part Number	99U5471-002.A01LF	SPD Ext.	
Serial Number	0E15B81B	Week/Year	02 / 12

Timings Table

	JEDEC #1	JEDEC #2	JEDEC #3	JEDEC #4
Frequency	457 MHz	533 MHz	609 MHz	685 MHz
CAS# Latency	6.0	7.0	8.0	9.0
RAS# to CAS#	6	7	8	9
RAS# Precharge	6	7	8	9
tRAS	17	20	22	25
tRC	23	27	30	34
Command Rate				
Voltage	1.50 V	1.50 V	1.50 V	1.50 V

Figure 6.2: Memory information in CPU-Z

The XMP (eXtreme Memory Profile) overclocking profiles can be stored in SPD too, and we can access them in BIOS/UEFI, if our mainboard is a mainstream or high-stream product, and we can see them in some programs, e.g. CPU-Z. These profiles can simplify overclocking or underclocking memory.


Memory modules information
Dual-channel mode
Timing settings - tCL and others
(note: tCL is 9, and CR is not possibly accessible)

CPU-Tweaker 2.0

CPU
Model: Intel(R) Core(TM) i3-2100 CPU @ 3.10GHz; CPUID: 6A7; Rev.: D2
Socket: LGA 1155; Tech.: 32 nm; Cores/Threads: 2 / 4; VCore: 0,852 V

MotherBoard
Vendor: Gigabyte Technology Co., Ltd.; Model: H67MA-USB3-B3
Chipset: 1C4A8086; BIOS version: F4 ZA; Date: 12/01/2011

Memory
Type: DDR3; Manufacturer: Kingston; Part Nb.: 99U5471-002.A01LF
Size: 2 x 2048; Speed: PC3-10600 (666Mhz) @ 9.9.9.24-1,5V; Chan.: Dual

System Frequency
BCLK: 99,8 MHz
Max Frequ.: x 16; 1596,4 MHz
UnCore:
RAM Ratios: 5; x 1,33
RAM: x 6,66; 665,0 MHz
EIST ✓ C1E ✓ Turbo ✓

Timings
Channels: A B; VDimm: 1,512 V
CAS# Latency (CL): 9
RAS# to CAS# Delay (tRCD): 9
RAS# Precharge (tRP): 9
Precharge Delay (tRAS): 24
Command Rate (CR): 1N

Others Timings

Write CAS Latency (tWCL)	7	Command 3-State Options	0
Refresh Cycle Time (tRFC)	74	IO Global Compensation	14
Act. to Act. Delayed (tRRD)	4	IO Latency Rank 0 DIMM 0	0
Write Recovery Time (tWR)	10	IO Latency Rank 1 DIMM 0	0
Write to Read Delay (tWTR)	5	IO Latency Rank 0 DIMM 1	1
Read to Precharge (tRTP)	5	IO Latency Rank 1 DIMM 1	1
Refresh Period (tREF)	5199	Round Trip Lat. Rank 0 DIMM 0	32
Four Activate Window (tFAW)	20	Round Trip Lat. Rank 1 DIMM 0	32
Minimum Pulse Width (tCKE)	4	Round Trip Lat. Rank 0 DIMM 1	34
Read to Read Same Rank	4	Round Trip Lat. Rank 1 DIMM 1	34
Write to Write Same Rank	4	Double Self Refresh Control	Disab.

Apply | Save | reg.txt | Hide | spd | About | Exit

CPU-Tweaker 2.0

CPU
Model: Intel(R) Core(TM) i3-2100 CPU @ 3.10GHz; CPUID: 6A7; Rev.: D2
Socket: LGA 1155; Tech.: 32 nm; Cores/Threads: 2 / 4; VCore: 0,828 V

MotherBoard
Vendor: Gigabyte Technology Co., Ltd.; Model: H67MA-USB3-B3
Chipset: 1C4A8086; BIOS version: F4 ZA; Date: 12/01/2011

Memory
Type: DDR3; Manufacturer: Kingston; Part Nb.: 99U5471-002.A01LF
Size: 2 x 2048; Speed: PC3-10600 (666Mhz) @ 9.9.9.24-1,5V; Chan.: Dual

System Frequency
BCLK: 99,8 MHz
Max Frequ.: x 16; 1596,4 MHz
UnCore:
RAM Ratios: 5; x 1,33
RAM: x 6,66; 665,0 MHz
EIST ✓ C1E ✓ Turbo ✓

Timings
Channels: A B; VDimm: 1,512 V
CAS# Latency (CL): 9
RAS# to CAS# Delay (tRCD): 9
RAS# Precharge (tRP): 9
Precharge Delay (tRAS): 24
Command Rate (CR): 1N

Apply | Save | reg.txt | SubTim. | spd | About | Exit

SPD

SPD Infos
DIMM#1; Type: DDR3; Size: 2048 MBytes
ManuFacturer: Kingston; Correction: None
Part Number: 99U5471-002.A01LF; Revision:
Serial: 0E15B81B; Speed: PC3-10600 (666Mhz)

Timings

	Jedec#0	Jedec#1	Jedec#2	Jedec#3	N/A
MHz	666	533	457	685	
CL	9	7	6	9	
tRCD	9	7	6	9	
tRP	9	7	6	9	
tRAS	24	20	17	25	
CR					
tRC	33	27	23	34	
tRFC	74	59	51	76	
tRRD	4	4	3	5	
tWR	10	8	7	11	
tWTR	5	4	4	6	
tRTP	5	4	4	6	
tREF					
Voltage	1,50	1,50	1,50	1,50	

Quit

Figure 6.3: CPU Tweaker

One of the manufacturer independent possibilities for overclocking is the CPU Tweaker application, so it is not necessary to use BIOS.

✎ **Number of sides and ranks.** Modules can be either *single-sided* or *double-sided*. Single-sided modules have chips only on one side of the PCB, while the two sides have chips on both sides of the PCB. Double-sided modules have the advantage of higher capacity but, on the other hand, there are more likely to be compatibility problems with some (mainly older) mainboards, and because they are a little wider, air flows worse around them (i.e. the cooling problem).

This is partly (but not fully) related to the term *single-rank* (1R×4), *dual-rank* (2R×4), *quad-rank* (2R×8) modules. The quad-rank modules are a domain of servers, whereas the single- and double-rank variants we can meet at desktops and laptops.

Simply said, a dual-rank module typically has twice as many chips as a single-rank module. One chip on the dual-rank module is similar in terms of access to two chips on a single-rank module, so from a memory controller point of view, a dual-rank module is actually similar to two single-rank modules. Generally, multi-rank modules have more chips than single-ranked modules.

✄ How can we know if the choosen module is a dual-rank type? If we find a full module designation (several numbers separated by dashes, at the end is the slash and behind it 1R or 2R is written), just 1R or 2R means either single-rank or dual-rank (CM_RATE). Unfortunately, this tag is not always traceable.

✎ **Timings for memory operations:**

- tRAS – time to search the row,
- tRCD (RAS to CAS Delay) – delay between RAS and CAS operations,

- **tCL** (Column Address Strobe Latency, CAS Latency) – time to search the column,
- tRP – time for Row Precharge,
- tRFC (Row Refresh Cycle Time) – during Refresh, the module must be labeled as inactive and marked as active when it is restored; tRFC is the time from tREF till module activation,
- tREF – time to refresh a module.

The information tCL is usually entered as the number of time cycles needed to find the column. If we want the information in nanoseconds, we have to calculate it from the length of the time cycle, which depends on the memory frequency.

The full information is of the form tCL-tRCD-tRP-tRAS/CM_RATE.

Example

For example, the following strings of memory speed information can be found:

tCL–tRCD–tRP–tRAS	/	CM_RATE
3 – 3 – 2 – 6	/	1T
5 – 5 – 5 – 16	/	2T
9 – 7 – 6 – 19	/	2T
16 – 18 – 18 – 38	/	2T

The newer types od DIMMs generaly have higher *nominal* latencies (in number of cycles) than older types. Typical values of tCL are the following:

- DDR: 2–3
- DDR2: 3–6
- DDR3: 6–11
- DDR4: 12–20

Remark

In the latency definition, we see that it is the "number of time cycles. . . ", but we know that the working frequency of modules and memory bus is increasing, that is, the time cycles actually shorten.

If we want to determine the latency in nanoseconds as the product of the number of cycles and the length of one cycle, then the value of this real latency is around 15 ns for DDR2, between 13 and 14 ns for DDR3, about 14 ns for DDR4. This means that the actual latency is rather stagnant.

6.2.6 Protection

Parity checking. The modules with this property have one one additional chip for parity (for example – on a memory module there are 8 standard memory chips and 1 parity chip). The used method is odd parity, as we can see in Table 6.2. Each row contains odd number of "1" bits (or: sum of the bits is odd).

When even sum of bits on the same location over all memory chips (including the parity chip) is detected, the parity error occured.

Remark

Modules with parity checking are more protected, but slower. This property is necessary to be supported by the both – the module and the mainboard (memory controller).

Chip 1	Chip 2	Chip 3	Chip 4	Chip 5	Chip 6	Chip 7	Chip 8	Parity chip
...	...	...	...	...	...	...	...	...
0	0	0	1	0	0	0	0	0
1	0	1	1	0	1	0	0	1
1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	1
0	1	0	1	0	1	0	0	0
...	...	...	...	...	...	...	...	...

Table 6.2: Odd parity example (binary digits, simplified)

Moreover, when parity-checking error occurs, the computer is *turned off*! Nowadays, this method is not used, but it is the base of the next method.

Error checking. The modules with the property *ECC* (Error-Correcting Code) use parity bits too. The module includes only 8 (or 4 or 16) "big" memory chips and one or two smaller parity chips. Eeach group of 8 bits in memory chips receives 1-bit additional information (odd parity) in the parity chip.

The single-bit errors on data can be detected. When parity error becomes, the memory controller ensures the repeated sending of damaged data, our PC (or server) keeps working, it is not turned off.

The modules, that are not ECC, are marked as "non-ECC".

Remark

ECC modules are determined for server mainboards. They are a bit slower, but safer. This property must be supported by the module and by the mainboard (and by the memory controller) too.

If our mainboard does not support ECC, we use non-ECC modules.

There are some subsequent techniques for more expansive server memory modules: Multiple-bit error correction, Chipkill, Memory scrubbing, Intel Single Device Data Correction (SDDC). To use them, we must have a compatible mainboard.

Registered. The memory modules marked as "Registered" contain one additional chip, the register chip. This chip acts as a buffer for memory operations.

The buffer can be imagined as a printer queue, but for memory operations. The purpose is to prevent a situation where the flow of requests would be so great that the module could not handle it (this would mean loss or damage to the data being transferred). It is in such moments that the operations instead of the burst on the circuitry of the module begin to accumulate in a buffer, from which it is gradually being executed at a stable speed. A secondary result is a better overall response, because the modules can communicate at higher frequencies.

These modules are labeled "Registered" or "Buffered"; however, modules without this property either do not have any of these strings in specification or they are "Unbuffered". Alternatively we can find this marking:

- RDIMM means Registered,
- UDIMM are not Registered, so they are Unbuffered.

A lot of server memory modules meet the both properties ECC and Registered. We can also meet with "Registered with Parity" modules, but it is the same (ECC and Registered).

Remark

It depends on a mainboard if it is able to work with the both RDIMM and UDIMM modules, or with only one of these types. However, when it understands both types, we can not mix them in any way (e.g. if we want to use two RDIMMs and two UDIMMs, it will not work). Similarly, with ECC support – it depends on the mainboard, and when the both ECC and non-ECC can be used, we should not mix such modules (in fact, it is the same as modules working under different operating voltages).

On the contrary, we can mix the modules that differ in frequency or latency (timing), but the result may be that the memory controller refuses to operate in dual-channel mode and goes into single-channel mode (but it depends on the controller and the mainboard).

Figure 6.4: DIMM ECC Registered DDR4 memory modules (Crucial CT4K32G4RFD4266 DDR4 128 GB (4 x 32 GB) DIMM 288-Pin 2666 MHz/PC4-21300 CL19 1.2 V Registered ECC Memory Kit)[2]

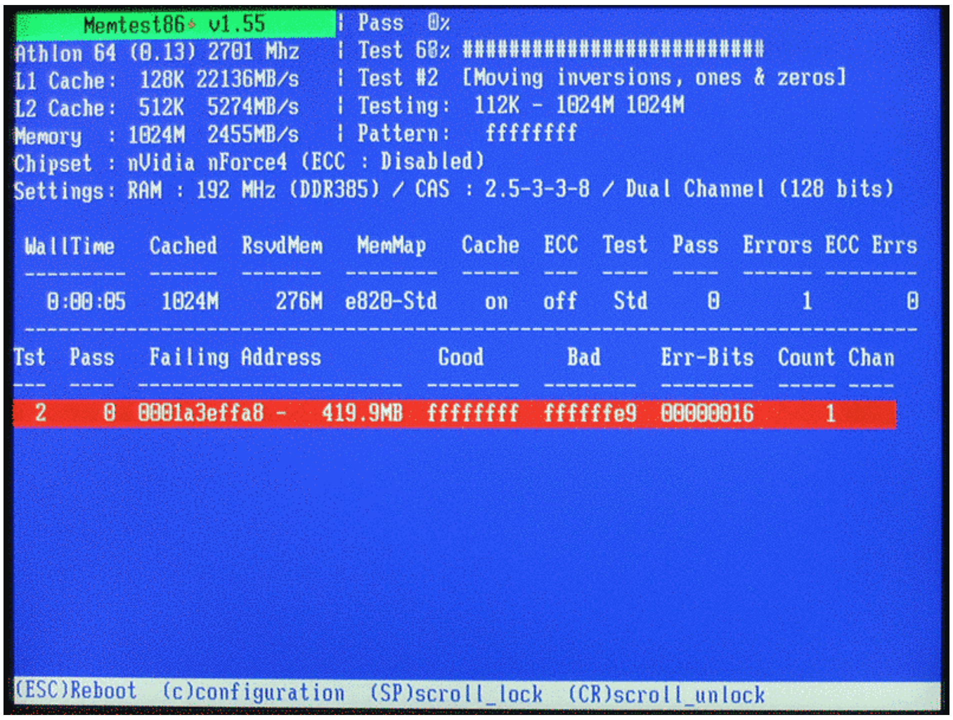
6.2.7 Memory Testing

If we just want to find out what state our memory modules are in in terms of speed, latency and other properties, we can use the tools supplied with the operating system in the simplest case (if we just need information about the memory size), but of course there are much better tools. Some of the best known are

- *SiSoftware Sandra Lite* (http://www.sisoftware.net/, there exists a freely available variant),
- *Everest Ultimate* (http://www.lavalys.com),
- CPU-Z (http://www.cpuid.org,
- HWInfo (http://www.hwinfo.com),. . .

MemTest86+ (freeware, http://www.memtest86.com) is the most widely used tool for integrity testing.

[2]From: https://www.amazon.co.uk



```
      Memtest86+ v1.55       | Pass  0%
Athlon 64 (0.13) 2701 Mhz    | Test 68% ##########################
L1 Cache:  128K 22136MB/s    | Test #2  [Moving inversions, ones & zeros]
L2 Cache:  512K  5274MB/s    | Testing:  112K - 1024M 1024M
Memory  : 1024M  2455MB/s    | Pattern:   fffffff
Chipset : nVidia nForce4 (ECC : Disabled)
Settings: RAM : 192 MHz (DDR385) / CAS : 2.5-3-3-8 / Dual Channel (128 bits)

 WallTime   Cached  RsvdMem   MemMap   Cache  ECC  Test  Pass  Errors  ECC Errs
 ---------  ------  -------  --------  -----  ---  ----  ----  ------  --------
   0:00:05   1024M     276M  e820-Std    on   off   Std     0       1         0
 -----------------------------------------------------------------------------
Tst  Pass   Failing Address          Good      Bad     Err-Bits  Count  Chan
---  ----  -----------------------  --------  --------  --------  -----  ----
  2     0  0001a3effa8 -   419.9MB  ffffffff  ffffffe9  00000016      1

(ESC)Reboot  (c)configuration  (SP)scroll_lock  (CR)scroll_unlock
```

Figure 6.5: Memtest86+

Testing method. We can test memory for memory errors either directly in the operating system (running an application in Windows), but this procedure is not very useful because the data goes through the operating system and we cannot test the entire memory. A better option is to boot from removable storage media (CD or USB flash) where we have a test program installed (download and burn the ISO/save the flash drive and boot. For example, MemTest is available in two forms – the lightweight version runs in Windows, the MemTest86+ version is bootable.

Some operating systems make it easier for us to do this; in addition to the individual operating systems, the MemTest86+ (or similar) item usually appears in the system start menu, and when selected, the memory test is performed without starting the operating system.

If any of the above programs report an error, we should find out what specifically is causing the error. It could be a faulty hardware (memory module, slot, memory controller, or motherboard or processor), but it could also be a bad memory timing setting causing instability or insufficient power supply. So if an error is reported and we have more than one memory module, we swap modules with each other, or we can try another slot, and run the test again.

6.3 Graphics RAM

The graphical interface needs memory to store video information – the current display content (which is transmitted into display via HDMI/DP/DVI/VGA). The common name for this memory is VRAM (Video RAM). Currently, the dynamic RAM technology is used as well, but the requirements for this memory are a bit different, graphics RAM is specially designed for graphics processing units.

This type of memories is used by graphics cards, game consoles and in high-performance computation.

In modern devices, we can meet two types of graphics RAMs, the both are SDRAMs (they are also denoted by SGRAM):

- *GDDR* (Graphics DDR SDRAM) – used by Nvidia and Hynix graphics chips, the current versions are GDDR5X, GDDR6 and GDDR6X,
- *HBM* (High Bandwidth Memory) – used by AMD graphics chips, the current version is HBM2.

The both GDDR and HBM memories are standardized by JEDEC.

Graphical SGRAMs can be found only on more expensive graphics cards. On cheaper graphics cards, we often have only DDR3 or DDR4 memory which is not optimized for this way of using.

Additional information

- http://www.cs.unc.edu/ lin/COMP089H/LEC/zach.pptx
- http://blog.logicalincrements.com/2017/02/types-vram-explained-hbm-vs-gddr5-vs-gddr5x/
- https://graphicscardhub.com/gddr5-vs-gddr5x-vs-hbm-vs-hbm2/

6.4 Read-only Memory

The more precise designation for the following discussed memories is "non-volatile", because in fact, most of them allow for not only read, but also write. Of-course, writing is not as simple as in RAM memories.

ROM (Read-only Memory) holds information written ones by a manufacturer. The cell structure is very simple, we need only one bit line, one word line and one transistor (dis)connected to ground. If the transistor is connected to ground, a logic value "0" is stored here, otherwise (the connection is disrupted) "1" is stored.

PROM (Programmable ROM) can be once programmed (written). Before programming, all cells contain "0". Programming means disruption of some connections, similarly as in ROM, but it can be made by a programmer.

EPROM (Erasable PROM) is repeatedly writable. The structure is similar to ROM and PROM, but the connection to ground is made using the second special transistor. This transistor is normaly turned-off (logical "1"), a programmer can some transistors switch on (make logical "0"). Erasing of the entire chip is done by exposing to UV light, EPROM chips have a "window" to make it possible.

EEPROM (Electrically Erasable PROM) chips are erased by an electrical impulse, so it is not necessary to remove the chip from mainboard and store it to erasing device.

NOR Flash is special type of EEPROM divided into blocks. We move and write blocks, there is no need to delete and write the whole chip.

NOR flash chips are mainly used to store BIOS/UEFI code at mainboards, because reading of this type of memory is quite fast and it can be easily updated – we can "flash BIOS" (do BIOS upgrade). Most other components with own BIOS usually have the code of their BIOS stored in NOR flash as well.

[3]From: https://www.aliexpress.com, https://www.aliexpress.com

Figure 6.6: EPROM chip and EPROM eraser[3]

6.5 NAND Flash

6.5.1 NOR Flash vs. NAND Flash

There are two types of flash memories, depending on their construction. The both types are non-volatile (permanent) and they both use memory cells made from floating gate transistors, their memory cells have two main connections:

- *NOR flash* – the first connection of a memory cell leads to the “source line”, the second connection to a “bit line” resembling NOR gate,
- *NAND flash* – a group of memory cells is connected together in a sequence, resembling the NAND gate, the first cell in the sequence is connected to bit line, the last cell is connected to a source line.

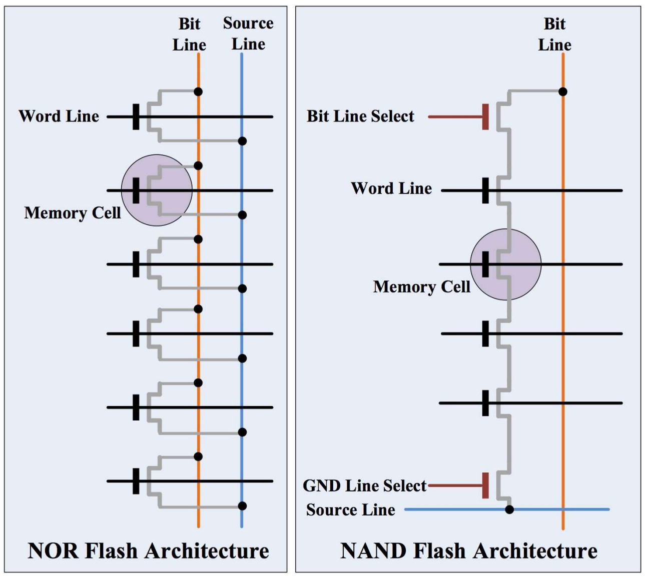
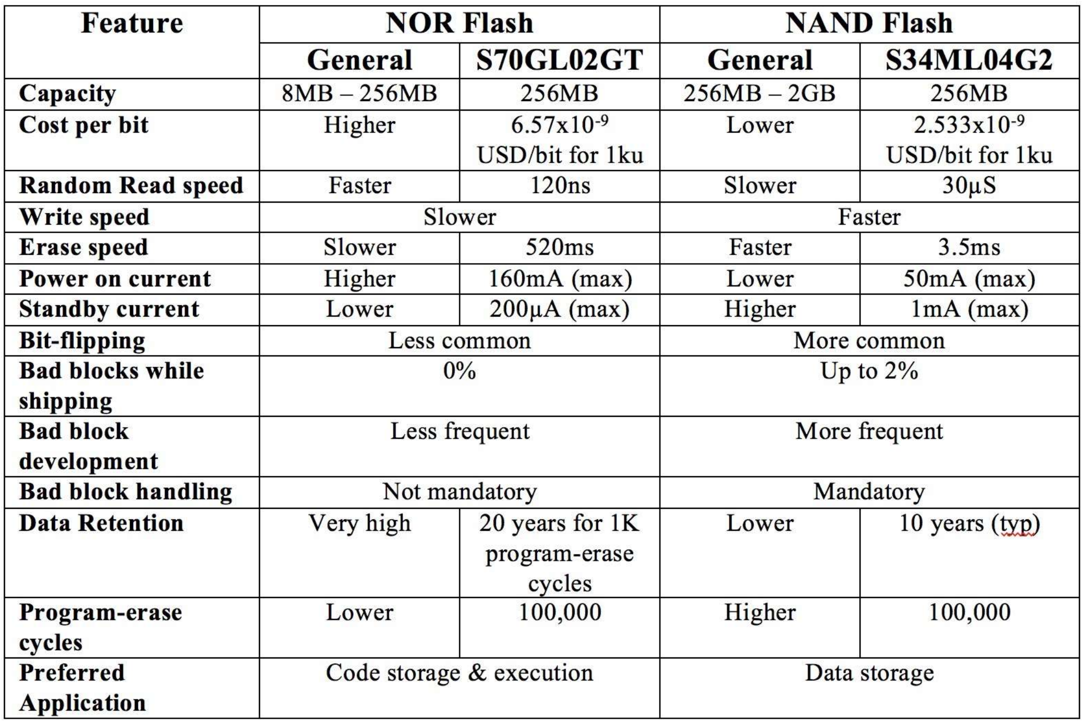


Figure 6.7: Comparison of NOR and NAND flash architecture[3]

The cells of the both types are connected to the word line too. Their construction is different, so they differ in some properties.



Feature	NOR Flash		NAND Flash	
	General	S70GL02GT	General	S34ML04G2
Capacity	8MB – 256MB	256MB	256MB – 2GB	256MB
Cost per bit	Higher	6.57×10^{-9} USD/bit for 1ku	Lower	2.533×10^{-9} USD/bit for 1ku
Random Read speed	Faster	120ns	Slower	30µS
Write speed	Slower		Faster	
Erase speed	Slower	520ms	Faster	3.5ms
Power on current	Higher	160mA (max)	Lower	50mA (max)
Standby current	Lower	200µA (max)	Higher	1mA (max)
Bit-flipping	Less common		More common	
Bad blocks while shipping	0%		Up to 2%	
Bad block development	Less frequent		More frequent	
Bad block handling	Not mandatory		Mandatory	
Data Retention	Very high	20 years for 1K program-erase cycles	Lower	10 years (typ)
Program-erase cycles	Lower	100,000	Higher	100,000
Preferred Application	Code storage & execution		Data storage	

Figure 6.8: Comparison of NOR and NAND flash features[3]

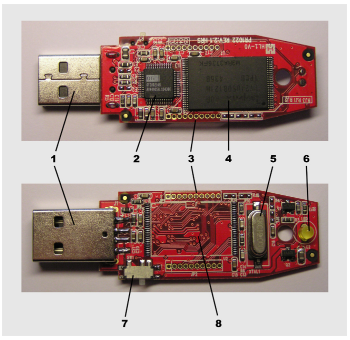
As we can see in Figure 6.8, NAND flash memories commonly have higher capacity, they are cheaper, with faster writing, but the probability of errors (bad blocks) is higher.

NOR flash circuits are used mainly as internal memory intended for code (e.g. BIOS), NAND flash circuits are used mainly as storage, to store data.

The both types of flash memories provide limited number of write operations for each memory cell. Before writing, the cell has to be erased. New chips are erased, but during their usage, more and more memory cells must be erased before writing, which means worse latency.

6.5.2 USB Flash Drive

USB flash drives are the most common NAND flash devices. Their structure is simple, as we can see in Figure 6.9, and they are quite cheap, but they can not be considered as reliable long-time storage. Their memory controller is simple, compared to e.g. SSDs.



1. USB connector
2. Mass storage controller
3. Test points
4. NAND flash chip
5. Crystal ocscillator
6. LED
7. Write-protect switch
8. Space to put the second flash

Figure 6.9: USB flash drive structure[4]

However, we can also meet very sophisticated USB flash drives with hardware keypad that enable data protection (hardware encryption). One of the protected USB flash drives is in Figure 6.10.

[3]From: https://www.embedded.com/design/prototyping-and-development/4460910/Flash-101–NAND-Flash-vs-NOR-Flash

[4]From: https://howflashdriveworks.wordpress.com/what-is-a-flash-drive/

[5]From: https://www.kingston.com

Figure 6.10: Protected USB flash drive[5]

These devices serve as a storage, so we need a filesystem inside. The most common filesystems for USB flash drives are

- FAT32 – widely supported (in Windows, Linux, MacOS, other UNIX systems,...), but there is a 2GB file length limit,
- NTFS – supported in Windows and Linux, with problems in MacOS, long files allowed, but metadata structures are unfortunately more extensive and this file system too often performs write operations (which damages flash memory, reduces its lifetime),
- exFAT – widely supported (Windows, Linux, MacOS,...), less writing operations than NTFS.

6.5.3 Flash Cards

The first memory cards (PCMCIA) used NOR circuits, the current cards are already NAND flash. The flash memory is enclosed in a thin plastic case (a NAND flash chip and controller are inside). Contacts (pins) are not covered, which negatively affects their life (contacts are gradually wearing out and exposed to external influences).

Memory cards are mostly used on portable devices, but we can buy a desktop card reader.

✎ **SD (Secure Digital)** has a capacity originally limited to 2 GB. The newer form, *SDHC* (also SD 2.0), has more capacity (up to 32 GB). Currently, we can meet the next generation – SDXC (Secure Digital eXtended Capacity) has an upper limit of 2 TB. Again, we need a reader that supports this format.

There are also smaller versions of microSD (often used to expand smartphone memory) and miniSD.

✎ SD and SDHC cards communicate via the *SPI* or *QSPI* bus (SPI transmits data in one bit, QSPI in four bits). For speeds, there are several Class Ratings for SDHC (write operation):

- Class 2 – minimal write speed on 2 MBps,
- Class 4 – minimal write speed on 4 MBps,
- Class 6 – minimal write speed on 6 MBps,
- Class 10 – minimal write speed on 10 MBps.

The speed class should be stated on card under the logo.

✎ The mentioned speeds are no longer enough (especially when we want to record video at higher resolution), so the SDHC and SDXC cards use the new **UHS** (Ultra High Speed) bus with additional speed classes:

- UHS-I supports Classes U1 and U3,
- UHS-II supports Class U3.

Class U1 provides the speed at least 10 MBps (as Class 10), Class U3 provides the speed at least 30 MBps.

[6]From: http://electronicdesign.com/memory/whats-difference-between-sd-and-uhs-ii-memory-cards

While the UHS-I specification changes the communication bus in particular (the pins on board remain), the UHS-II specification adds new pins that are placed on the card in the second row under the original ones – see Figure 6.11.

Compatibility: reverse compatibility is ensured, but with limitations. An older card reader will work with a newer card, but at "older" speed. Similarly, in a newer reader, an older card will work, but again at "older" speed. For the UHS-II card, the second row of pins will not be used in an older reader.

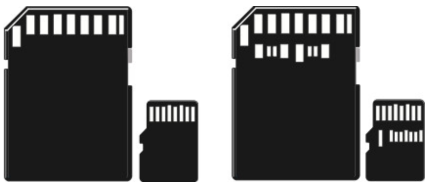


Figure 6.11: Memory cards for the UHS-I bus (left) and for the UHS-II bus (right)[6]

Remark

If you need to store FullHD video on the card, then just the speed 10 MBps is sufficient (Class 10, or better U1). Higher resolution requires Class U3.

It is important to know how to discover the speed class on the card. The original speed classes are marked with a number enclosed in the letter "C", the UHS classes are indicated by the number inside the letter "U" and we can find the specification ("I" or "II"), and the both types of classes can be stated (i.e. if the card is inserted into a UHS-capable reader, the UHS class is used, but when we put it in an older UHS reader without UHS support, the "older" class is used).

Figure 6.12 shows a 64GB SDXC memory card with the both class designations (U1 and C10), with the UHS-I specification. The both classes specify a minimum write speed of 10 MBps, but this does not mean that the card could not communicate faster (the bus allows it, it also depends on the controller). Under speed classes it is even written that the card can read (R) up to 60 MBps, write (W) up to 35 MBps, in case if it is inside a UHS-I reader.

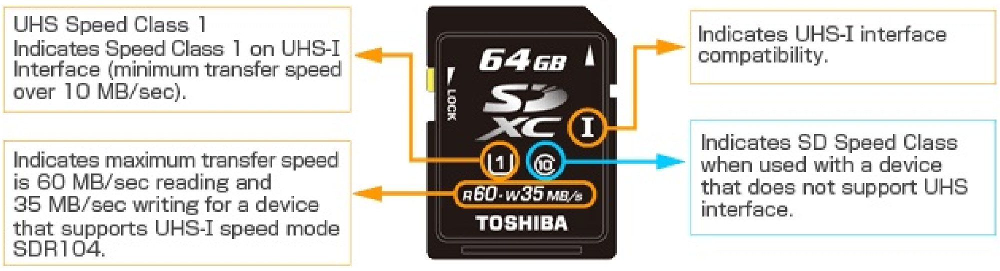

UHS Speed Class 1
Indicates Speed Class 1 on UHS-I Interface (minimum transfer speed over 10 MB/sec).
Indicates maximum transfer speed is 60 MB/sec reading and 35 MB/sec writing for a device that supports UHS-I speed mode SDR104.
64GB
SD XC
I
1
10
R60·W35 MB/s
TOSHIBA
LOCK
Indicates UHS-I interface compatibility.
Indicates SD Speed Class when used with a device that does not support UHS interface.

Figure 6.12: Speed class rating on a memory card[7]

Additional information

Detailed comparison of various memory cards:
http://en.wikipedia.org/wiki/Comparison_of_memory_cards

[7]From: http://www.sdxc2.com/

6.5.4 eMMC and SSD

✎ The both eMMC (embedded MultiMediaCard) and SSD (Solid State Drive) are storage devices based on NAND flash. The eMMC storage is cheaper, with simplier memory controller, whereas SSDs have larger sizes, better performance and more sophisticated memory controller.

The following text will refer to the SSD.

SSDs are available in these forms:

- in a 2.5” package – the same form factor as for laptop hard disks, communication via SATA/AHCI signal interface, cheaper, but slower,
- in a SATA Express package (SATAe), this interface is not common in mainboards,
- in a package with U.2 interface (SFF-8639) – similar to SATAe, but the particular pins are used differently, not common as well,
- narrow card intended to a mSATA in a mainboard, communication via SATA/AHCI, in some laptop mainboards, but it is not common,
- narrow card intended to a M.2 slot in a mainboard, common in newer mainboards,
- expansion card intended to a PCI Express slot, communication via NVMe interface,
- external (communication mostly via USB of some newer version, or Thunderbolt).

✎ *AHCI* (Advanced Host Controller Interface) is the signal interface of the SATA communication interface. It is optimized for (old) hard disks, not for SSDs, so it is a bottleneck for most SSDs. Different versions of SATA allow various speeds. For SSDs via AHCI, SATA 3.0 or higher is recommended, i.e. 6 Gbps. *NVMe* (Non-Volatile Memory Host Controller Interface Express) is the newer signal interface used by storage devices connected via PCI express lines.

The NVMe signal interface is optimized for high-bandwidth hardware interfaces (SATA does not make much sense, unlike at PCIe interfaces with higher versions). It can handle multiple threads (use a multi-core processor), supports multiple queues of commands (i.e. it performs parallelization of operations much better), the execution of operations itself is optimized (e.g. it is not necessary to access the registers so often). So SSD communicates via NVMe much faster than via AHCI.

The M.2, SATAe and U.2 interfaces can use the both AHCI and NVMe signal interfaces, but some M.2 slots support only one of these interfaces, similarly SSDs support either one or both of these interfaces.

Figure 6.13 shows possible configurations of a M.2 slot and a M.2 card (e.g. SSD). There are two types (“B” key for SATA/AHCI, “M” key for NVMe), and their combination. We usually encounter implementations of either M-key (NVMe only) or B+M-key (both). In some cases, it is necessarry to configure this property in BIOS/UEFI.

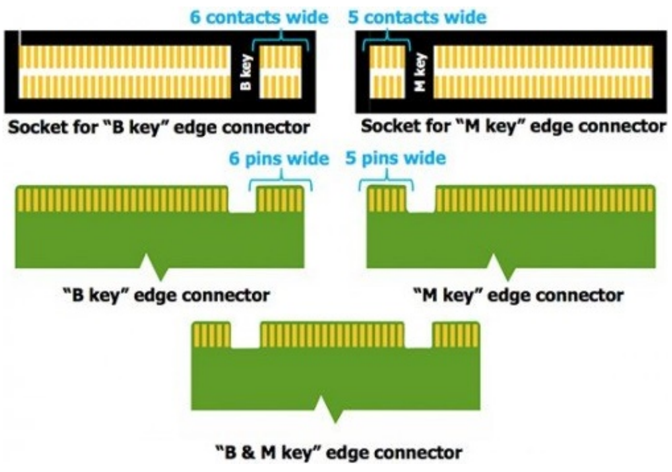


Figure 6.13: M.2 keying[8]

[8]From: https://rog.asus.com/articles/maximus-motherboards/buying-an-m-2-ssd-how-to-tell-which-is-which/

[9]From: https://www.atpinc.com/blog/what-is-m.2-M-B-BM-key-socket-3

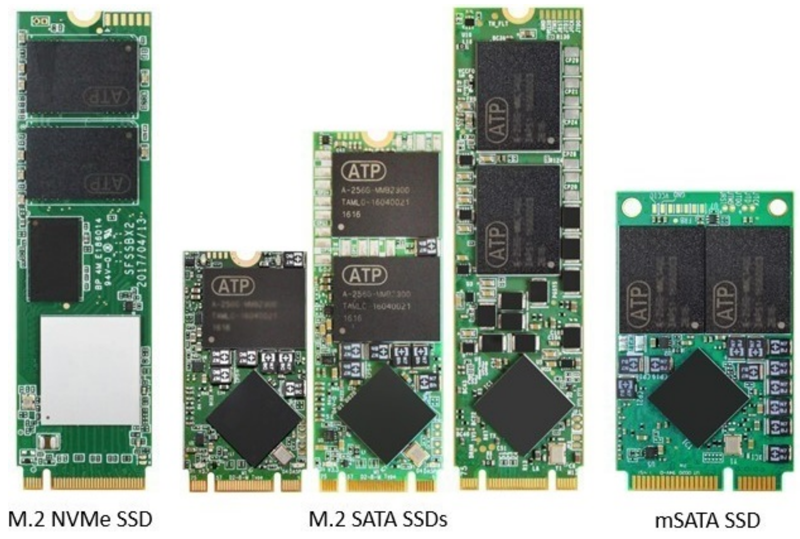


Figure 6.14: M.2 SSDs with different keying (AHCI and NVMe types)[9]

SSD memory controllers are much more complicated and "intelligent" than hard disk memory controllers. Depending on their quality, (reading and writing) the transfer speeds and the storage cell life depend too much. All the procedures described below are provided by the controller. Most SSD memory controllers are ARM chips in real.

SSD manufacturers either make their own controllers, such as Samsung (MDX Controllers) or OCZ (Indilinx Controllers). Others modify the existing ones (modified Marvell controllers found in SSD from Toshiba or the Crucial brand), others use a simple, low-cost SandForce controller.

Additional information

https://rog.asus.com/articles/hands-on/easy-guide-to-ssds-sata-msata-m-2-and-u-2/

6.5.5 Memory Cells in SSD

Memory cells in SSD are:

- *SLC* (Single Level Cell) – distinguishes two levels of voltage:
  - 0 % is interpreted as 1,
  - 100 % is interpreted as 0,

  one SLC cell can store one bit,
- *MLC* (Multi Level Cell) – distinguishes four levels of voltage:
  - 0 % is interpreted as 11 (in binary),
  - 33 % is interpreted as 10,
  - 66 % is interpreted as 01,
  - 100 % is interpreted as 00,

  one MLC cell can store two bits,
- *TLC* (Triple Level Cell) – eight levels (2^3, i.e. 0, 14, 28, 42, 56, 70, 84 and 100 %), this memory cell can store three bits,
- *QLC* (Quad Level Cell) – sixteen levels, we can store four bits into this type of cell.

The more bits are stored in one cell, the slower the write to that cell.

Cells can be used as a "better" type, e.g. a TLC cell can be accessed as MLC or SLC cell, if a controller needs it. In some SSDs, we have a *SLC cache* when some cells are treated as SLC: data is first written in the SLC style (faster), and later when the write queue is empty, the data is moved to cells operating in MLC/TLC/QLC mode (slower).

Memory cells are associated in *pages*. Due to the compatibility of SSD and HDDs, the page size is typically 4 KiB (i.e. 4096 B). These pages are further organized in *blocks*. Typically, when using SLC cells, 64 pages are in one block (256 KiB per block), MLC cells double – 128 pages per block (512 KiB), and TLC cells three times – 192 pages per block (768 KiB).

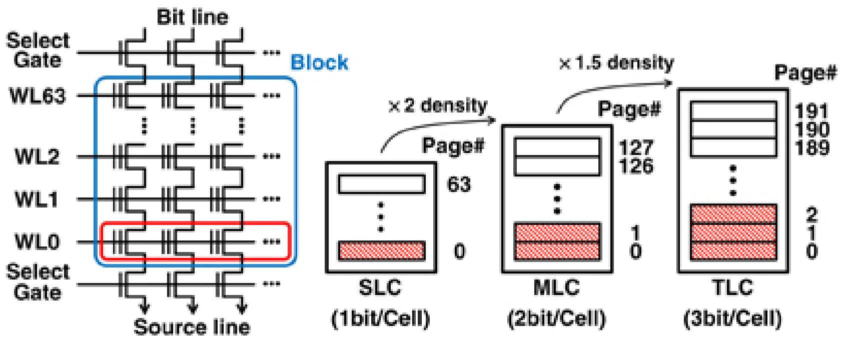


Figure 6.15: Structure of flash memories – pages and blocks[10]

In fact, the hierarchy goes even further, but the higher leveling does not have any direct effect on write operations. Blocks are organized into *planes*. In one plane, there are usually 1024 blocks, and on a single plate (*die*, as with processors) there is always a certain number of planes. A manufacturer can place a number of such dice in one flash chip.

6.5.6 Writing and Reading

For the sake of simplicity, we assume the use of SLC cells.

Writing and reading at cell level. The SLC cells may be in one of two states – either *discharged* (voltage level 0, the cell stores "1"), or *charged* (voltage level 1, the cell stores "0").

The default cell state is "discharged" (contains 1), and if we want to write a value of 0 in the cell, it must be programmed (raised the voltage). Conversely, when we want to write a value of 1 to a cell that is "charged" (contains 0), we must discharge it.

Reading and writing is controlled by the *top gate*. Charging and discharging refers to the state of the *floating gate*.

During charging, the electrons, which remain in the gate even after a power disconnection (permanent memory), enter the floating gate through the insulating layer (which is below it). The charged floating gate (containing electrons, i.e. negatively charged) then forms an electric field that prevents the passage of the electrical current during reading (the current does not pass ⇒ we read 0, current goes through ⇒ we read 1).

Writing: if we want to write a value of 1 to a cell, it is necessary to discharge the floating gate by applying the opposite voltage, which "pushes" electrons from the gate.

[10]From: http://m.iopscience.iop.org/1347-4065/53/4S/04EE04/article
[11]From: https://blog.superuser.com/2011/02/10/the-modern-marvel-of-the-ssd/

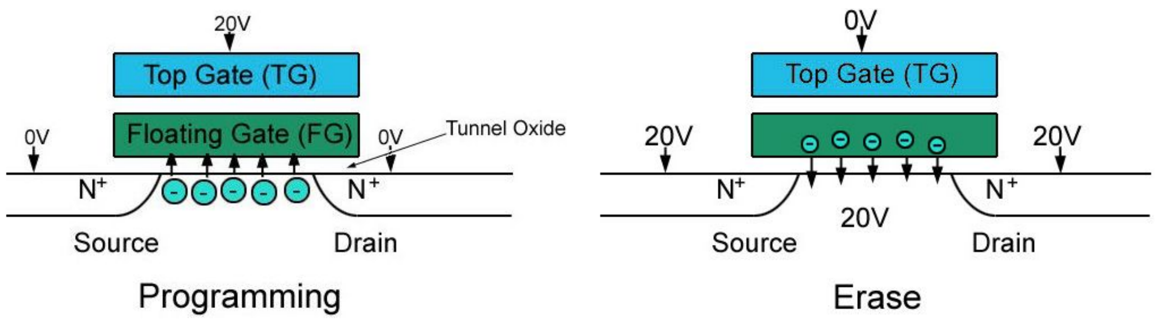


Figure 6.16: SLC memory cell – programming and erasing[11]

Reading: low voltage (which does not directly affect electrons) is put to the floating gate via the top gate, and the voltage is measured.

✎ **Reading and writing page.** The operating system works with pages, which is fully respected when reading from the flash chip, one page is read at a time. However, writing is more complicated. Any flash chips (NOR and NAND) are block memories, so the entire block is always deleted and written. We will look at this process in more detail.

If the content of a block is to be changed (even a single page in this block, but the entire block is always written at once), it is not possible to simply "rewrite" it (as at hard drives), the block has to be erased first. It is proceeded as follows:

- A controller uses buffer which is faster but not permanent.
- If the block has not yet been written or is deleted (all memory cells are discharged, they have a binary value of 1, 11, 111 or 1111 by cell type), the situation is simple – the data can be written directly into the page.
- However, if the page is not marked as deleted, it is necessary to delete (whole block) and then write:
  - the entire block with the page to be modified is copied into the buffer,
  - the cells of the block are erased, after erasing these cells contain the maximal possible binary value (1, 11, 111, 1111, by cell type),
  - in the buffer, the completion of the block's final state is performed, i.e. we do modifications in the relevant page,
  - at this point we have the block in the buffer in the state in which we want it in the flash memory,
  - the buffer contents (one block) is stored into the relevant place in the flash memory.

The slowest operation is erasing.

If it was necessary to perform the block deletion every time it was written, the writing operation would be very slow. New flash chips are deleted already at production, so writing to "new" pages is very fast, but writing to the pages with some old content is slow. To write new data, the controller tries to select pages, which are already deleted, thus positively affecting the writing speed. If existing data is to be overwritten, the entry does not necessarily have to be made on the same page as the original data – the page is marked as unused and determined to delete (can be deleted later) and another (previously deleted) page is selected.

Thus, for more used flash chips, the situation can come where no free unerased pages are available, so the controller can not avoid block erasing. Therefore, such SSDs may progressively increase the writing time. However, a high-quality controller can handle this, and flexibly erases the blocks that have enough pages labeled to erase, and prepares for faster writing operation.

6.5.7 TRIM Operation

In modern filesystems (NTFS, FAT, ext4,...), file intended to delete is only marked as deleted, but its content will remain on disk/SSD. If this is done with SSD, the blocks on which the file was located are not erased. They are not blocked from writing another file (because they are associated with the logical address to which the file system may want to save a different file), but when trying to write this block, the memory controller must erase it first – the memory controller cannot see inside filesystems.

To avoid necessity of erasing before writing, the operating system (or a process with higher privileges) can send the TRIM request. The TRIM specification is a part of the ATA standard (SATA), but it can be used for NVMe SSDs as well as SATA SSDs. To use this operation, TRIM must be supported in the memory controller of the SSD, and in the operating system as well.

When sending the TRIM signal (the parameter is a list of files to erase their blocks), the corresponding blocks are erased, and they are prepared for fast writing another files.

✄ In *Windows*, TRIM is supported in Windows 7 and newer. But Windows 7 can use it only at SATA SSDs, not at NVMe SSDs. Windows 8 and later can use TRIM at NVMe SSDs too. This signal is sent whenever a file is deleted.

MacOS uses TRIM for Apple-branded SSDs only by default and most third-party tools for other SSDs stopped working after Yosemite update. From update 10.10.4, there exists a `trimforce` utility enabling TRIM for non-Apple SSDs.

Linux implemented TRIM support as first. But using TRIM in the same way as in Windows and MacOS has been found to be unnecessarily slow (file deleting is slower), so its auto-sending is disabled by default. There are two possibilities to use TRIM in Linux:

- to activate automatic TRIM for a particular partition, it means to add one word – "discard" – into one file,
- to use the `fstrim` command, which analyzes the filesystem, finds all deleted but not SSD-erased files, and transmits their list into the SSD memory controller, this command can be automatically executed in times with low load.

☎ **Additional information**

- https://www.intel.com/content/www/us/en/support/articles/000016148/memory-and-storage.html?wapkw=trim
- https://www.howtogeek.com/176978/ubuntu-doesnt-trim-ssds-by-default-why-not-and-how-to-enable-it-yourself/
- https://en.wikipedia.org/wiki/Trim_(computing)

☎

6.5.8 Lifetime of SSDs

✎ **The effect of erasing operation on the lifetime of SSDs.** Whenever a memory cell is erased, the *insulating layer* that the electrons pass through when erasing and writing wears out. When it is too damaged, the controller denotes it as faulty and the cell is not used (in fact, the entire block). In particular, delete operations are the worst because they are always performed for the whole block at a time (additionally, writing to a page is usually associated with erasing the entire block in which the page is located).

Another complication is that some electrons are stuck in the insulating layer over time, which in turn (apart from the gradual destruction of the insulation layer itself) also aggravates the possibility of correctly reading the contents of the cell, the captured electrons effect reading. The controller attempts to compensate for this by increasing the reading current and its application for a longer period of time, which can further increase the wear rate of the insulating layer and extend the write time. If the length of write time exceeds the specified limit, the block also stops being used.

The controller at each block records the number of erasing cycles performed. This value is then an important indicator in load balancing algorithms (we determine which blocks will be used for write preferentially – rather those that have been erased a few times).

✎ **Increasing of lifetime of memory cells.** The number of write cycles differs for various types of flash memory cells: cca 100 000 for SLC cells, cca 3000 for MLC cells, cca 1000 for TLC cells. It looks terrible, but not every day we overwrite all the flash content.

✂ Some possibilities have been described above, especially the more efficient setting pages for writing. Other possibilities:

- In addition to the declared amount of memory, the controller has extra (often unpublished) memory (usually about one third additional). This memory is used to replace (bridge) memory areas whose lifetime has expired.
- Frequent files (log files, etc.) are sometimes moved to pages where memory cells are not too worn. This equalization of wearing is called *static Wear Leveling*.
- The controller has implemented *dynamic Wear Leveling* algorithms that are used every time we write. In general, this means that pages with a lower wear rate are preferred for writing, so that writing and subsequent reading times are as low as possible for a long time.
- In the operating system, we should disable certain features that are not needed and unnecessarily reduce the life of the flash memory, ideally it should be done by the operating system itself (if it detects flash memory) – disable disk defragmentation, restrict the use of log files, temporary files or cookies, in Windows also Prefetch and Superfetch functions, Ready Boost, Volume Shadow Copy, etc.

Considering how today flash memory controllers work, manufacturers offer a lifetime guarantee of 3 to 5 years, longer for better quality devices. If we do not write a quantum of data on SSD everyday, lifetime is quite sufficient and matches the normal lifetime of a computer.

If we assume that our SSD is nearing the end of its life, we can use it for data that we rarely change and rather read (for example, the movie store). Reading the memory cells does not destroy them anywhere.

✎ **Litography, manufacturing technology.** Another parameter that affects the memory cell lifetime is manufacturing technology. Manufacturers try to switch to a "smaller" manufacturing process (as small transistors as possible), which today means a 14nm planar process (planar because the transistors are in one plane). An advantage is more efficient operation (less energy) and lower production cost of one TLC or MLC cell, but the problem is a further decrease in lifetime because fewer cells get worse while writing and the gate is destroyed faster.

An interesting solution is the 3D NAND manufacturing technology (also V-NAND, different manufacturers use various names), where the larger transistors remain (36–40 nm), but the memory cells are multilayer. The place is also saved but in a different way, and the advantages of larger transistors (especially in terms of longer lifetime and easier production) remain. Samsung came with 3D NAND technology first, later Micron joined.

What if power fails? The memory controller uses a special buffer (RAM) when writing data, it uses it not only for the data itself but also for the *mapping table*. This table is primarily stored in the flash memory, but because the flash is too slow, the copyof this table is transferred to the buffer at startup and is used there. In this table, there is logical (LBA) mapping information on physical addresses, so if this table is not up to date or even damaged, data on the SSD will not be lost, but we can not trace them (we do not know what address to look for).

The mapping table is saved from the buffer back into the flash chip when the system shuts down. If a power outage occurs, this transfer is not performed (or is performed partially, it is much worse) and the flash chip has an out-of-date or damaged mapping table. The problem is not only with those files that we created or removed (there are missing or overwritten records in the table), but also those files that have changed (moved in length) or moved into a different block – for example when applying static Wear Leveling. All these files will not be reachable in the normal way, the mapping table will not be consistent. This damaged SSD can usually be restored, but it is time consuming and costly.

If there is an SSD in a laptop with a functional battery, there is no problem, the outage will survive. For devices without a battery, there is a solution used by some SSD manufacturers: if high quality ceramic capacitors are added to the SSD, the firmware will still have enough power to store the mapping table on the flash chip when outage is detected. But for low-cost reasons many manufacturers do not do it, and the risk remains... Then the only defense is making backups.

Additional information

- http://www.storage-switzerland.com/Articles/Entries/2013/5/16_The_Unknown_Risk_of_SSD_Mapping_Tables.html
- http://codecapsule.com/2014/02/12/coding-for-ssds-part-3-pages-blocks-and-the-flash-translation-layer/

6.6 Memory Hierarchy

Various uses of memory in a computing device require different features of this memory. In the case of primary memory, we need more capacity, lower energy consumption and low cost; DRAM memory meets these requirements. For cache, we need faster memory, which may be less amount, so it does not matter a slightly higher cost; SRAM memory meets these requirements. Similarly, we can continue with flash memory, CMOS, processor registers, etc.

Indirect proportion is primarily between the price per Byte and the size of a given type of memory, from which the indirect proportion between speed and size is derived. Figure 6.17 shows the memory pyramid. As we can see, storage (secondary and ternary memory) is the slowest with worse latency, but has the largest capacity and lowest cost per Byte. Primary memory in DIMM slots is faster with

[12]From: http://www.moorinsightsstrategy.com/research-paper-datacenter-memory-and-storage/

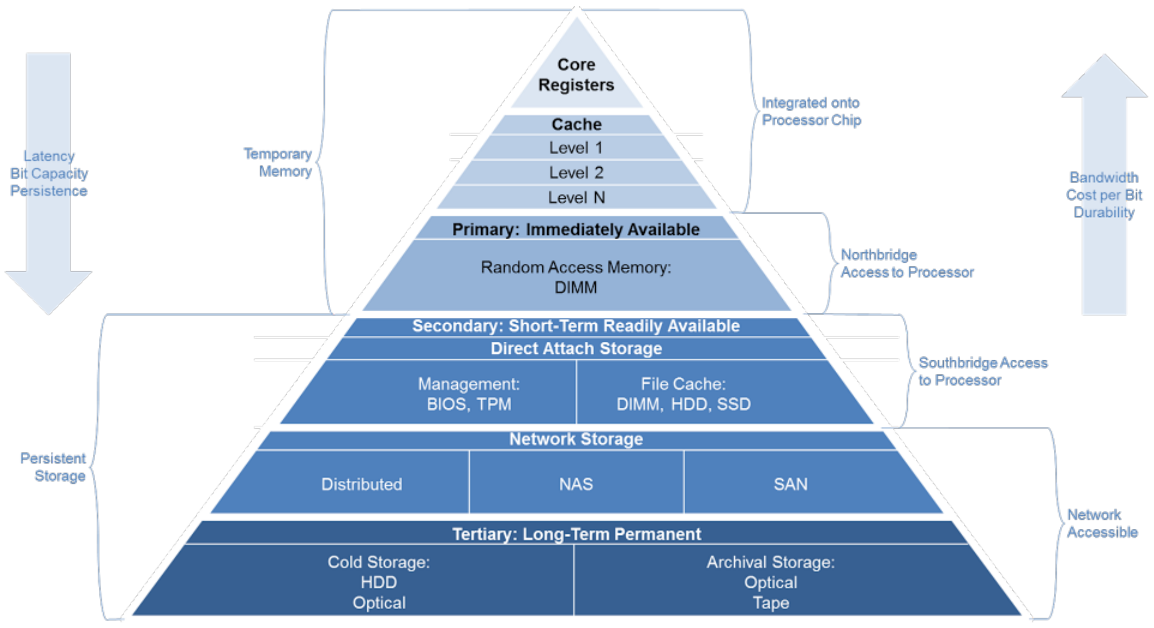

Latency
Bit Capacity
Persistence
Temporary Memory
Core Registers
Cache
Level 1
Level 2
Level N
Integrated onto Processor Chip
Bandwidth
Cost per Bit
Durability
Primary: Immediately Available
Random Access Memory: DIMM
Northbridge Access to Processor
Secondary: Short-Term Readily Available
Direct Attach Storage
Management: BIOS, TPM
File Cache: DIMM, HDD, SSD
Southbridge Access to Processor
Network Storage
Distributed
NAS
SAN
Persistent Storage
Network Accessible
Tertiary: Long-Term Permanent
Cold Storage: HDD Optical
Archival Storage: Optical Tape

Figure 6.17: Memory pyramid[12]

better latency, less capacity, higher cost per Byte. We could continue through the cache memories of various levels to the registers.

6.7 Cache Memory

Main role of a cache memory is to ballance differencies in speed in communication between two components. In case of a processor cache, this memory ballances differencies in speed of a processor and a primary memory, and this cache performs multiple tasks.

The main two parameters to load a piece of a code into cache are the following:

- *temporal* property (recently executed instructions or data are likely to be executed again soon): Processors often execute some instructions repeatedly – there are loops, repeatedly called functions, etc. If a code has already been loaded into the processor and is likely to reload soon, why to reload it repeatedly if it is already in the processor?
- *spatial* property (instructions close to recently executed instructions, or data close to recently processed data, are likely to be executed soon).

The above stated properties are also called *locality of reference*, or principle of locality, of temporal/spatial locality.

A *cache block* (cache line) is a set of contiguous adress locations from the primary memory, the whole block is transmitted into cache. The usual length of this block is 32 or 64 Bytes. Current DDR memories contain commands in their protocol that can perform burst read, burst write, the length of which transmitted data corresponds to the size of the cache block.

Each block is stored in cache at least with the following additional information:

- block of data/instructions, transmitted contents,
- *block index* calculated from the source address of a block in DRAM,
- *tag* (metadata), e.g. dirty bit (see below).

A *mapping function* is function determining correspondence between the primary memory blocks and their cached copy.

A *replacement algorithm* is collection of rules used in case that the cache is full and alongside some new data or instructions need to be cached, this algorithm determines a "victim" to be removed from cache to make free space for new data or instructions.

If a processor needs to work with a part of main memory (perform a read or write operation), its request is captured by the cache controller, we say that a *read or write hit* has occurred.

Reading from cached memory. If a processor needs to load data or an instruction from main memory, two possibilities may occur:

- the requested content is in cache,
- the requested content is not in cache.

The request (read hit) is captured by a cache controller, which finds out the corresponding possibility. The first possibility means that the requested content is simply read from cache.

To handle the second possibility, the new block has to be transferred into cache (we call this situation by *read miss*). If cache is full, the replacement algorithm is used, and then the transfer can be performed.

A *dirty bit* is a part of metainformation used in the second situation – we write to cache only. This flag bit is set in case that the processor makes a write hit. if the write hit for the same address is preformed repeatedly, this bit remains set. The main memory is updated later, and the repeated writes into the main memory are not carried out unnecessarily often.

Writing to cached memory. If it is necessary to write (store) data or instructions, these two situations may occur:

- copy of the corresponding block is in cache,
- copy of the corresponding block is not in cache (we call this situation by *write miss*).

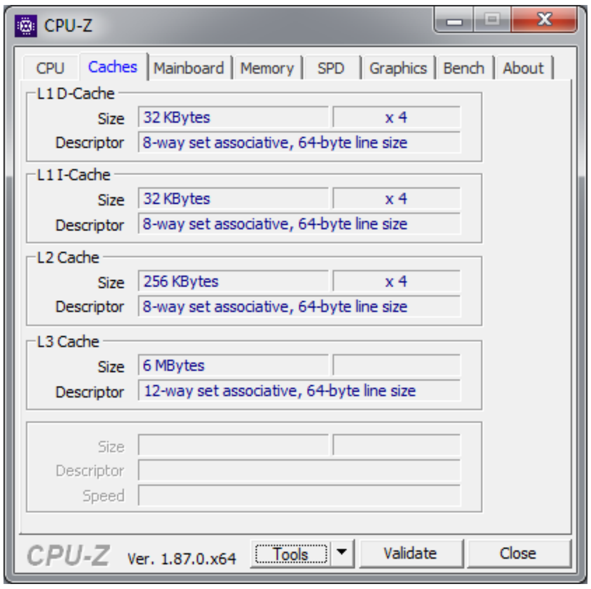
These situations are solved similarly as for reading operation. These two methods are used in the both situations, so if it is necessary to update something in the main memory:

- write-through: the both the cache and main memory, is updated withing this write operation,
- write-back: only the cache is updated, by a dirty bit for the given glock is set.

The write-through algorithm is quite simple, but it is not optimal in case that the same piece of data has to be modified very often.

There are several mapping algorithms (to specify the location of a block from the main memory while transfering to cache). Two "extreme" forms are

- *direct mapped cache* – the address of a block in cache is directly depended on the source address of a block in DRAM, the source address can be determined directly from the block index in the tag (a block can be placed only in one particular address in cache); this algorithm is very fast, but there is a relatively high risk of cache miss (some blocks can not be cached at the same time because their cache address and index would be the same),
- *fully associative cache* – there is no limit for block mapping, there is no dependency between the address in cache and source address in DRAM (a block can be placed anywhere in cache); this algorithm is slow, but the cache miss risk is low.



L1 D-Cache
Size 32 KBytes x 4
Descriptor 8-way set associative, 64-byte line size

L1 I-Cache
Size 32 KBytes x 4
Descriptor 8-way set associative, 64-byte line size

L2 Cache
Size 256 KBytes x 4
Descriptor 8-way set associative, 64-byte line size

L3 Cache
Size 6 MBytes
Descriptor 12-way set associative, 64-byte line size

Size
Descriptor
Speed

CPU-Z Ver. 1.87.0.x64

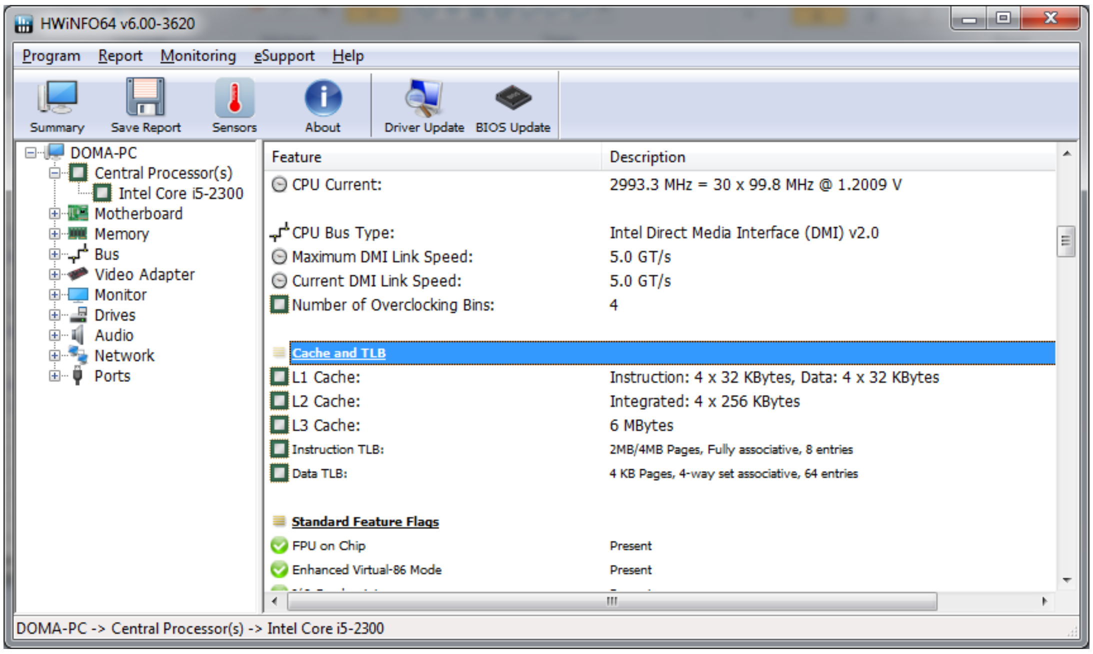
Figure 6.18: Cache information in CPU-Z

In fact, algorithms are used that have properties somewhere between these basic algorithms.

The *n-way set associative cache*: a block from DRAM can be put in "n" different locations (e.g. in 4-way set associative cache in four different addresses), so there is a relationship between the source address od a block in DRAM and the destination address in cache, but not as limited as the direct mapped cache prescribes.

Additional information

- http://csillustrated.berkeley.edu/PDFs/handouts/cache-3-associativity-handout.pdf
- https://www.sciencedirect.com/topics/computer-science/set-associative-cache



Feature	Description
CPU Current:	2993.3 MHz = 30 x 99.8 MHz @ 1.2009 V
CPU Bus Type:	Intel Direct Media Interface (DMI) v2.0
Maximum DMI Link Speed:	5.0 GT/s
Current DMI Link Speed:	5.0 GT/s
Number of Overclocking Bins:	4
Cache and TLB	
L1 Cache:	Instruction: 4 x 32 KBytes, Data: 4 x 32 KBytes
L2 Cache:	Integrated: 4 x 256 KBytes
L3 Cache:	6 MBytes
Instruction TLB:	2MB/4MB Pages, Fully associative, 8 entries
Data TLB:	4 KB Pages, 4-way set associative, 64 entries
Standard Feature Flags	
FPU on Chip	Present
Enhanced Virtual-86 Mode	Present

DOMA-PC -> Central Processor(s) -> Intel Core i5-2300

Figure 6.19: Cache information in HWInfo

Current processors (made by Intel, AMD) have two or three levels of cache: L1, L2, L3. The fist level is closest to the processor and is usually divided in two parts: L1D (L1 cache for data) and L1I (L1 cache for instructions. The remaining levels are not divided.

The cache information can be obtained in various programs, we can see the output of two such tools: CPU-Z (Figure 6.18) and HWInfo64 (Figure 6.19).

6.8 Magnetic Media

6.8.1 Structure and Disk Geometry

Magnetic disk media usually have one of the dimensions (*Form Factor*) found in Table **??**. These values are fairly universal and often apply to other components as well, e.g. optical drives for desktop computers usually have the same dimensions as the $5\frac{1}{4}$" drive cases.

Dimension in inches	**mm**	**Usage**
8	203	for very old hard drives and very old floppy disks
$5\frac{1}{4}$	133	likewise
$3\frac{1}{2}$	88	nowadays for hard disks, formerly also floppy disks, ZIP, etc.
$2\frac{1}{2}$		some hard drives, especially for laptops, also cheaper SSDs

Table 6.3: Typical dimensions (form factor) of magnetic disk media

A magnetic disk consists of one or more *disk platters* mounted on a common *spindle*, each platter has two *surfaces* (in some disks, the surfaces entirely at the edge of the platters column are not used), each used surface has its own *read/write head* (all heads are on the common arm). Then there is the active (electromechanical) part of this device – a disk drive consisting of two *motors* (one to spin the spindle of the disk, and one to move read/write heads). The next part of a magnetic disk is a *disk controller*.

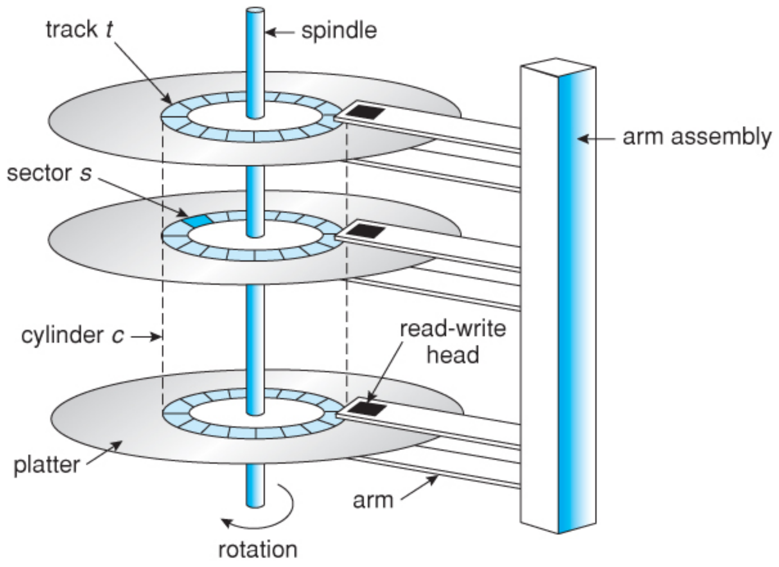


Figure 6.20: A magnetic disk structure[13]

[13]From: https://www.cs.uic.edu/ jbell/CourseNotes/OperatingSystems/10_MassStorage.html

Each surface is devided into concentric *tracks*, and all tracks with the same diameter across all surfaces form a *cylinder*. Tracks are devided into *sectors*. One sector carries 512 B (=1/2 KiB) of data in the common disk. If a disk is in *Advanced format*, each sector carries 4 KiB of data. The "Advanced Format" disk type can only be used on a computer with an operating system that supports this type. The main purpose is to increase the amount of data that can be addressed on the disk.

Remark

A *disk controller* can always work with whole sectors, whereas an *operating System* can always work with whole clusters (blocks in Linux). A cluster/block is usually 8, 16 or 32 sectors (powers of 2)

Advanced Format disk controllers operate in one of two possible modes – 512e emulation or native 4K; the former is supported in most current operating systems (Windows from Vista onwards, Linux, MacOS X, other UNIX systems), the latter emulation consists in "forming" a long 4KB sector as eight short 512B sectors. This is not a detriment to functionality, since operating systems do not access sectors individually anyway, but usually 8 or 16 sectors as a whole, depending on the filesystem settings.

A magnetic disk rotates at a *speed*:

- 7 200 rpm (Rotations per Minute) – common disks for desktop computers,
- 5 400 rpm – laptop disks,
- more (10 000 rpm, 12 000 rpm,...) – mostly server disks.

Addressing methods specifies how the physical location of specific data on disk is determined.

Cylinder–Head–Sector (CHS) is an older addressing method used in old IDE disks. It has the advantage of taking the geometry of the disk into account. The location of the data is determined by three pieces of information (i.e. it is a non-linear, multi-level addressing), namely the cylinder (cylinder), the head (which determines the surface on which to write, the intersection with the specified cylinder is the track) and the sector. The the address of the outermost cylinder (and therefore the track) is 0.

Logical Block Addressing (LBA) is the second addressing method that is still used today. LBA was originally intended for SCSI drives, but was later adopted by the ATAPI standard in more recent versions. LBA is linear addressing at the logical level, it does not take disk geometry into account. Sectors are numbered from 0, starting on the first (outer) cylinder, through all surfaces, then the second cylinder, etc.

Head parking is the appropriate placement of heads to prevent damage to the disc in the event of an accidental impact. If the power to the disk is turned off or if an impact occurs, the heads will "fall" onto the disks (the air cushion will cease to exist), which can corrupt the data.

The *Auto Park* (automatic parking of heads) is based on the fact that after shutdown the disk rotates for a while by inertia, thus producing enough energy to move the heads to the parking area (the principle of recuperation is used).

For newer linear motor discs, the location of the head parking area depends on the *manufacturer's decision* (it can be a sector or a place outside the disc surfaces, some discs even park anywhere on the surface). For safety reasons (in case of failure or shaking), the platters are covered with a thin soft protective layer to mitigate the effects of an unexpected impact.

Especially with laptops, parking is done very often (even several times per minute, they are equipped with a motion sensor), which reduces the life of the disc.

6.8.2 Reading and Writing

✎ *Longitudinal Magnetic Recording* (LMR) means that bits (oppositely oriented magnetic fields) are written horizontally with the disk surface. Possible write density only up to about 150 GB/inch^2 because at higher densities, a phenomenon called paramagnetism occurs, which results in spontaneous loss of stored data (the magnetic fields for different bits interact with each other).

✎ *Perpendicular Magnetic Recording* (PMR, also CMR = Conventional) has been in commercial use since 2005 (Toshiba). The mathematical induction vectors are not oriented longitudinally with the surface, but perpendicular to it and the recording head has a different design to read magnetic fields oriented in this way.

✎ *Shingle Magnetic Recording* (SMR) is an enhancement to perpendicular recording technology. The individual tracks on the surface are moved towards each other and even overlap in the "shingle" method (first a track closer to the edge of the disc is written, then a track a little further towards the center so that the previous one partially overlaps, followed by another, etc.). This method can save up to 20 or 25 percent of disk space.

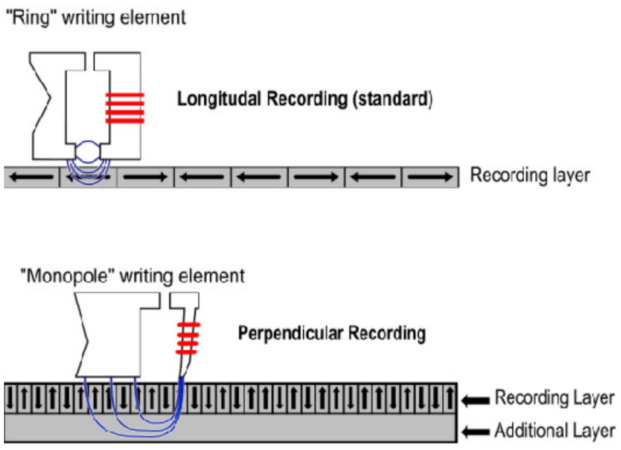


Figure 6.21: Longitudial and perpendicucular magnetic recording[14]

☞ **Remark**

Consider what it means when one track partially overlaps an adjacent track – it follows that if we want to write to a track, we necessarily overwrite the contents of the adjacent track that is closer to the center. This means that *basis* is more for sequential writing, where we write each track in a direction from the edge of the disc towards the centre. In other cases, the writing slows down. ☜

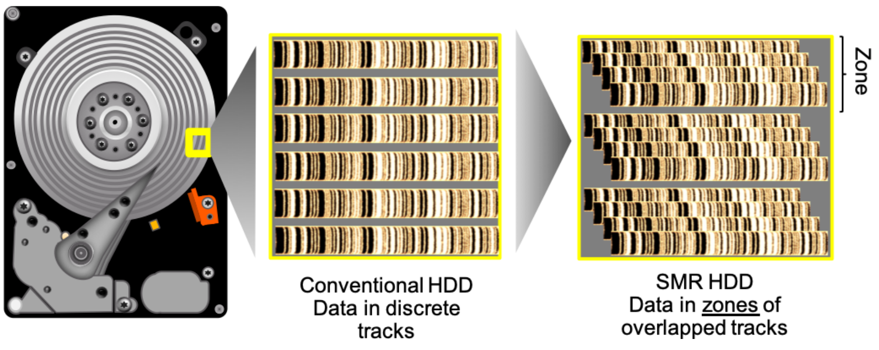


Figure 6.22: Comparison of PMR and SMR recording[15]

[14]From: https://www.semanticscholar.org/paper/Shingled-Magnetic-Recording-for-Big-Data-Suresh-Gibson/dcfaeb270c0864d1ac2d4633362a90a7c101f13c

In order to allow non-sequential writing, tracks are grouped into *zones*. Tracks within the same zone overlap (partially), tracks at the interface between zones do not overlap. This softens the requirement to use sequential notation and restricts it to notations within the same zone. A write pointer (cursor) is defined in each zone to indicate where the last write was completed and therefore from which point writes are allowed. Thus, the controller requires a bit more intelligence than PMR technology.

Other modifications: there are SMR write zones and PMR write zones on the disk, SMR zones are intended for sequential writing of large files, PMR for conventional use. Alternatively, SMR zones can be handled similarly to SSDs – the contents of the entire zone are compiled in a buffer, then copied (sequentially) from that buffer to disk.

The main purpose of this technology is to increase the overall recording density, even at the expense of the data interface throughput (i.e., it is expected that the writing will probably be slower). The typical use of these disks is where larger amounts of data are written sequentially, such as in a NAS for backup or to store camera footage. However, this technology is not very popular among customers.

☞ **Remark**

There are two types of SMR disks: *drive-managed*, which organize the data themselves, and *host-managed*, where the operating system determines what goes where on the disk. In particular, drive-managed SMR disks can start rearranging data to optimize storage under heavy or sustained load, which can take minutes or even hours. During this time, they are not responsive at all. Therefore, drive-managed SMR drives are not suitable for use in RAID arrays (they can in effect kill the RAID array) and are therefore not suitable for NAS, and we can't use them e.g. for storing camera output. ☜

6.8.3 Technology

✎ **AHCI** (Advanced Host Controller Interface) is a hardware-independent interface for communication with a SATA controller. Why enabling AHCI can be good for:

- NCQ support (see below),
- Hot Plug support (ability to connect and disconnect SATA devices on the fly),
- eSATA, mSATA, M.2 and other similar technologies,
- improved power management, ability to bundle commands into clusters to reduce command execution overhead, etc.

This specification was created by Intel and is currently supported in all newer operating systems.

✎ **NVMe** (NVM Express) is the successor to AHCI used by external storage (typically SSDs) communicating over PCIe or M.2 bus (which actually also means communicating over PCIe). It is not used for memory connected via SATA.

NVMe has much higher throughput compared to AHCI, it can also make much better use of the PCIe bus – AHCI would be a "bottleneck" when connected to a PCIe or M.2 slot.

As for operating system support, the driver was added to Windows with an update to version 7, and has been supported natively since 8.1. In other operating systems (including Linux and MacOS), NVMe is supported.

[15] From: https://zonedstorage.io/introduction/smr/

Interleaving. The disk rotates quite fast, so if the data follows one another physically, it only gets to the next one after the next rotation. This would effectively mean slowing down instead of speeding up as the RPM (revolutions per minute) increases. When interleaving is used, the data is not written to the sectors directly after each other, but to every n-th sector – then the interleaving factor is 1:n.

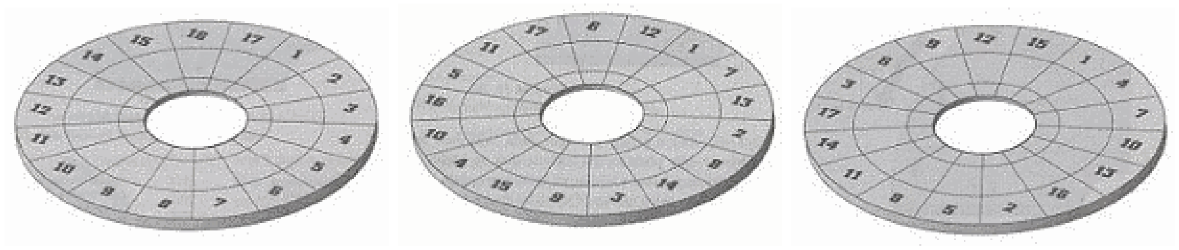


Figure 6.23: Interleaving 1:1, 1:3, 1:6[16]

NCQ (Native Command Queuing). It is a natural ordering of data read/write commands. It is an improvement on interleaving technology, where commands to work with individual regions on the disk are ordered to better match the actual path of the heads over the disk, and multiple commands can be executed in a single disk rotation.

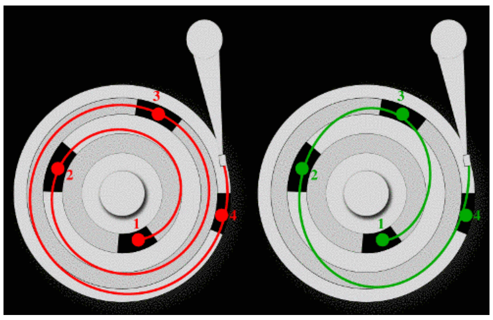


Figure 6.24: Disk without NCQ and with NCQ[17]

Zone Bit Recording (ZBR). This technology solves the problem of varying sector lengths on tracks that are different distances from the disk axis. In ZBR, the disk is divided into zones according to the distance of the cylinders from the center of the disk. There are less sectors in zones closer to the center, and more sectors in zones closer to the edge of the disc.

6.8.4 Monitoring

For disks, we talk about the *Mean Time Between Failures* (MTBF) property. This is a statistic that gives an indication of the reliability of a given disk.

Hard disk damage can occur in a variety of ways, such as wear and tear, head impact on the disk surface, frequent temperature fluctuations, seized bearings, malfunctioning motors, damaged electronics, etc., compromised integrity. Some types of damage can be repaired (compromised integrity, minor surface damage resulting in corruption of the data written to that location), others cannot, but it is usually possible to at least salvage the data (not always).

S.M.A.R.T. (Self Monitoring Analysis and Reporting Technology). It is a mechanism for monitoring, analyzing, and reporting hard disk errors independently of the operating system. This mechanism is integrated in nearly all drives (since the second half of the 1990s).

Different disks provide various amounts of information (the newer the disk, the more information). The disk controller continuously monitors the values (number of read errors, number of write errors,

[16]From: http://www.fi.muni.cz/usr/pelikan/ARCHIT/TEXTY/GEOMHD.HTML
[17]From: http://expertester.wordpress.com/2008/07/19/how-to-enable-ahci-without-reformating/

changes in the spin rate of the disk, number of remapped sectors, number of “working” hours, disk temperature, etc.) and writes them to a specified location from where a specialized program can retrieve these values and possibly “alarm” them.

Some common parameters (there are many more):

- `Raw_Read_Error_Rate` – number of read errors,
- `Spin_Up_Time` – the time it takes for the plates to spin up to the required speed, if it is not OK, it usually means a damaged motor,
- `Reallocated_Sector_Count` – when a damaged sector is found, it is remapped, replaced with a spare sector; this value shows the number of such remaps, if it grows, the disk will soon be used up (i.e. we track the frequency of changes),
- `Seek_Error_Rate` – number of erroneous header exposures, increasing errors mean that the header shift mechanism is not quite right,
- `Spin_Retry_Count` – the number of failed motor starts to spin up the platters, in case of errors we should back up and get a new disk,
- `Temperature_Celsius` – disk temperature, this is more of an informational figure (usually it should not exceed about 60 ℃).

In addition to the specific value (*raw value*), we get the corresponding “risk index” – *value*, which is a value between 0–100 (0 is usually the maximum risk, 100 is OK), plus a *thresh* value (also 0–100) representing the boundary between risk and good condition. So the lower the value, the worse situation!

In the respective tools, we can usually get all these values in a table whose row might look like this:

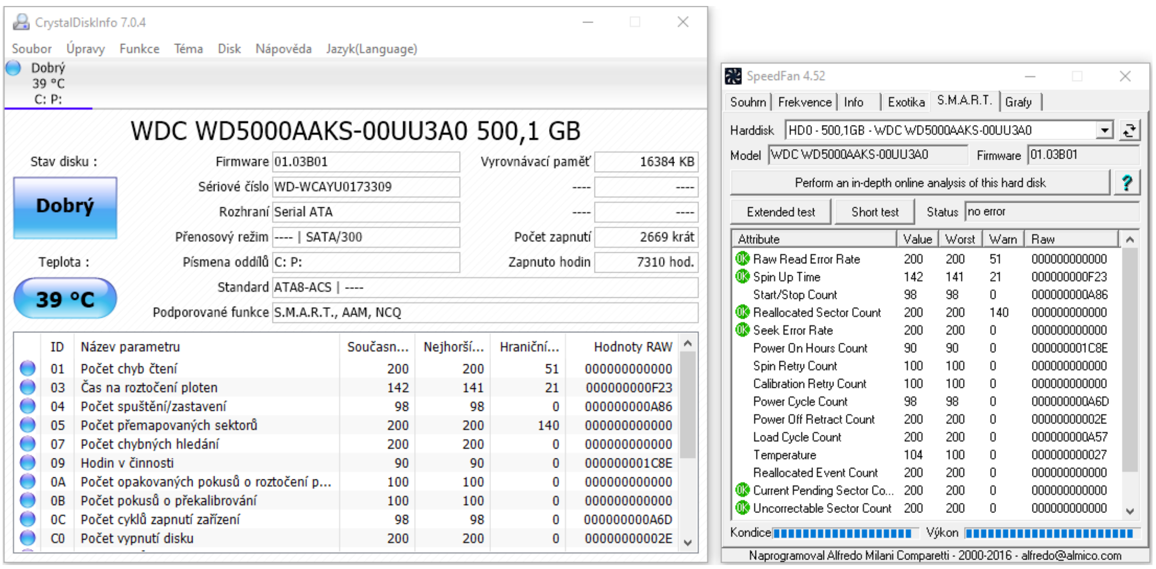
ID#	Attribute	Flag	Value	Worst	Thresh	Type	Updated	When_failed	Raw_Value
3	Spin_Up_Time	0x0007	079	079	011	Pre-fail	Always	-	6260
7	Seek_Error_Rate	0x000f	061	058	051	Pre-fail	Always	-	7863642
194	Temperature_Celsius	0x0022	074	067	000	Old_age	Always	-	28 (Lifetime)

Table 6.4: Sample table with S.M.A.R.T. parameters, selected three rows

In the table 6.4 we see that the tested disk has a good spin-up time so far, but the number of erroneous head exposures is increasing dangerously and is approaching (from above) the value thresh, the temperature is acceptable.

The S.M.A.R.T. settings can be partially accessed in the BIOS, but these are only basic settings (mainly enabling the mechanism itself). To be able to work with this mechanism better, we need some tool:

- for Windows: HDDScan, CrystalDiskInfo, SpeedFan, HDD Health, SiS Sandra, etc., disk manufacturers often provide their own tools
- for Linux: smartmontools package, found in all repositories, usually already installed


CrystalDiskInfo 7.0.4
Soubor Úpravy Funkce Téma Disk Nápověda Jazyk(Language)
Dobrý
39 °C
C: P:
Stav disku :
Dobrý
Teplota :
39 °C
SpeedFan 4.52
Souhrn Frekvence Info Exotika S.M.A.R.T. Grafy
Perform an in-depth online analysis of this hard disk
Extended test
Short test
Kondice
Výkon

WDC WD5000AAKS-00UU3A0 500,1 GB

Firmware	01.03B01	Vyrovnávací paměť	16384 KB
Sériové číslo	WD-WCAYU0173309	----	----
Rozhraní	Serial ATA	----	----
Přenosový režim	---- | SATA/300	Počet zapnutí	2669 krát
Písmena oddílů	C: P:	Zapnuto hodin	7310 hod.
Standard	ATA8-ACS | ----		
Podporované funkce	S.M.A.R.T., AAM, NCQ		

	ID	Název parametru	Současn...	Nejhorší...	Hraniční...	Hodnoty RAW
●	01	Počet chyb čtení	200	200	51	000000000000
●	03	Čas na roztočení ploten	142	141	21	000000000F23
●	04	Počet spuštění/zastavení	98	98	0	000000000A86
●	05	Počet přemapovaných sektorů	200	200	140	000000000000
●	07	Počet chybných hledání	200	200	0	000000000000
●	09	Hodin v činnosti	90	90	0	000000001C8E
●	0A	Počet opakovaných pokusů o roztočení p...	100	100	0	000000000000
●	0B	Počet pokusů o překalibrování	100	100	0	000000000000
●	0C	Počet cyklů zapnutí zařízení	98	98	0	000000000A6D
●	C0	Počet vypnutí disku	200	200	0	00000000002E

Harddisk HD0 - 500,1GB - WDC WD5000AAKS-00UU3A0
Model WDC WD5000AAKS-00UU3A0 Firmware 01.03B01
Status no error

	Attribute	Value	Worst	Warn	Raw
OK	Raw Read Error Rate	200	200	51	000000000000
OK	Spin Up Time	142	141	21	000000000F23
	Start/Stop Count	98	98	0	000000000A86
OK	Reallocated Sector Count	200	200	140	000000000000
OK	Seek Error Rate	200	200	0	000000000000
	Power On Hours Count	90	90	0	000000001C8E
	Spin Retry Count	100	100	0	000000000000
	Calibration Retry Count	100	100	0	000000000000
	Power Cycle Count	98	98	0	000000000A6D
	Power Off Retract Count	200	200	0	00000000002E
	Load Cycle Count	200	200	0	000000000A57
	Temperature	104	100	0	000000000027
	Reallocated Event Count	200	200	0	000000000000
OK	Current Pending Sector Co...	200	200	0	000000000000
OK	Uncorrectable Sector Count	200	200	0	000000000000

Naprogramoval Alfredo Milani Comparetti - 2000-2016 - alfredo@almico.com

Figure 6.25: S.M.A.R.T. values in Crystal Disk Info and SpeedFan (same disk and system)

Tools for maintaining and monitoring drives can often be found from their manufacturers (just search the manufacturers' websites, e.g. for Seagate, Samsung, Western Digital, Hitachi). In these cases we often use the tool in the case of a disk from that manufacturer.

Seagate and Western Digital currently have the widest product range, with several families of drives for various purposes:

- Seagate offers Barracuda drives for everyday use, IronWolf for data centers, NAS, etc., and SkyHawk for storing camera data.
- WD offers the Blue series for everyday use, Black for maximum performance, Red for data centers and NAS, Purple for camera data storage, and Green for economical operation.

6.8.5 RAID

RAID (Redundant Array of Independent Disks) means to use an array of disks. It is therefore about partitioning data across multiple disks. The purpose is to increase data safety (against loss) and possibly speed. There are several types of RAID depending on how each disk is used. There are also differences between these types in the minimum number of disks required.

RAID 0 splits data between (at least) two disks, each piece of information is stored in only one place. There are two types:

- *chaining* (not used) – when the first disk is full, the second disk is used, then the third disk, etc., we chain the disks and erase the physical boundaries between them,
- *striping* – the data is split during writing and the individual parts are stored on different disks (one file can be split on several disks), it allows parallel work of several processes (threads) with the same file (on parts stored on different physical disks).

The advantage is that we have two smaller disks, but logically one large "virtual" disk, and the speed of parallel file work is usually higher (with the second type).

The disadvantage of RAID 0 is the lack of fault tolerance, the data is not redundant, in case of a failure we always lose part (but often all). Also, there may be a problem with the eventual recovery of corrupted data.

JBOD (Just a Bunch of Disks) is similar to RAID 0 chaining. If we have multiple disks connected to a JBOD, the first one fills first, when it is full, the second one comes next, and so on.

The difference from RAID 0 chaining is that JBOD does not require all connected disks to have the same capacity – if we chain multiple disks in this way, it can fully utilize the capacity of each disk.

RAID 1 is an implementation of *mirroring*. The written data is recorded on (at least) two disks simultaneously, each information is stored on both (or all) disks at the same time.

Compared to the previous type, the main advantage is security – in case of a failure of one disk we have data backed up on the other, another advantage is the increase in reading speed (reads from both disks in parallel).

The disadvantage is the need to double the disk capacity compared to the actual need (we assume that both disks have the same capacity). Writing is slower, data needs to be written to both disks in RAID.

RAID 0 (striping)

disk A	disk B
1	2
3	4
5	6
7	8
⋮	⋮

RAID 1 (mirroring)

disk A	disk B
1	1
2	2
3	3
4	4
⋮	⋮

Figure 6.26: RAID 0 and 1

Combination of RAID 0 and 1 is a two-stage RAID. There are two types depending on how type 0 and 1 are combined:

- RAID 0+1 (also RAID 01) – we store data striped on two disks – A, B, then mirror both of them on two more disks – C, D (i.e. 4 disks A, B, C, D, when writing we first stripe the data and then mirror it),
- RAID 1+0 (also RAID 10) – the first part is stored simultaneously on disks A, B, the second on disks C, D, the third on A, B, etc. (i.e. also 4 disks, but the mirroring is done before the striping, or within the striping).

RAID 01

disk A	disk B	disk C	disk D
1	2	1	2
3	4	3	4
5	6	5	6
7	8	7	8
⋮	⋮	⋮	⋮

RAID 10

disk A	disk B	disk C	disk D
1	1	2	2
3	3	4	4
5	5	6	6
7	7	8	8
⋮	⋮	⋮	⋮

Figure 6.27: RAID 01 and 10

RAID 3 uses a parity disk. We use $N + 1$ identical disks, N of which are used to store data, and the last (parity) disk stores the XOR checksum of that data.

If the parity disk fails, the data remains intact. If one of the data disks fails, its contents can be reconstructed (using the same positions of all other disks, including the parity disk).

Data security is quite high. The “bottleneck”, however, is the parity disk – it is written to every time any other disk is written to, which slows down the writing and also reduces its lifetime. Parity disk failure is statistically significantly more likely than other disk failures.

RAID 5 is functionally similar to RAID 4, but it is an attempt to eliminate the increased load on a single parity disk. Blocks of data are striped to different disks and one of the $N+1$ disks is used as a parity disk, but for each $(N+1)$ tuple of blocks it is a different disk.

RAID 3

disk A	disk B	disk C	disk D
1	2	3	P(1–3)
4	5	6	P(4–6)
7	8	9	P(7–9)
⋮	⋮	⋮	⋮

RAID 5

disk A	disk B	disk C	disk D
1	2	3	P(1–3)
4	5	P(4–6)	6
7	P(7–9)	8	9
⋮	⋮	⋮	⋮

Figure 6.28: RAID 3 and 5

6.8.6 RAID Configuration

RAID controller (physically it is a chip on the motherboard or integrated into the chipset) must be activated in the BIOS/UEFI. The type of RAID depends on the RAID controller and possibly the operating system, it is certainly not common to be able to choose any of the above. There are usually several available from RAID 0, 1, 10, JBOD, 3 and 5.

Next, we need several hard disks (but the controller can also support RAID for SSDs). Most RAID types need disks with the same parameters, especially capacity, the exception is JBOD.

We start in *BIOS/UEFI Setup*. Activation of the RAID controller can be e.g. in *Advanced*, item *OnBoard Device Configuration* (but it can be different in each BIOS, depending on its manufacturer, it can also be *Integrated Peripherals*, item *OnBoard SATA/IDE Ctrl Mode*, etc.). We can find an entry for RAID (again, it may be called differently, depending on the manufacturer). We always need to set the mode according to the type of disks and possibly also set the *Enabled* entry.

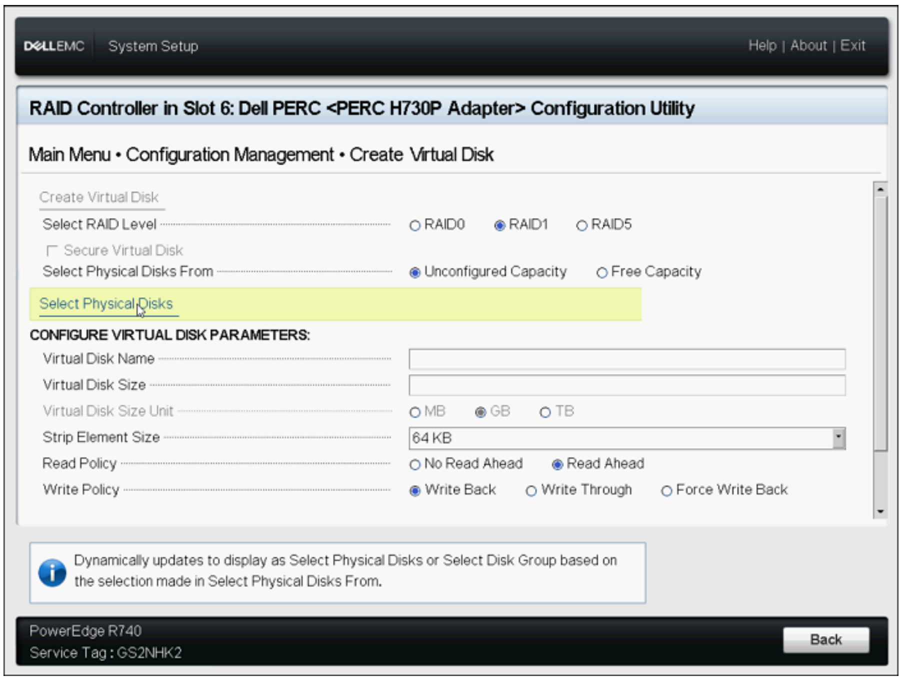
The other configuration is either also in UEFI or can be in another tool. If it is a RAID controller supplied with the motherboard (integrated in the chipset), then this information can be found in the motherboard manual.

It is necessary to determine the RAID type, other possible parameters, and to assign the drives to the array.

Remark

The disks should be empty (or their contents will probably be destroyed). Control programs offer automated array creation for that case. In some circumstances, it is possible to create a RAID 1 (mirroring) on one disk with data on it and the other empty, then the data is copied to the other disk during array creation.

Even so, creating a RAID from the data disk is somewhat risky, and data is lost with other types of arrays.


DELLEMC System Setup Help | About | Exit

RAID Controller in Slot 6: Dell PERC <PERC H730P Adapter> Configuration Utility

Main Menu • Configuration Management • Create Virtual Disk

Create Virtual Disk

Select RAID Level	RAID0 [✓ RAID1] RAID5
Secure Virtual Disk	
Select Physical Disks From	[✓ Unconfigured Capacity] Free Capacity
Select Physical Disks	

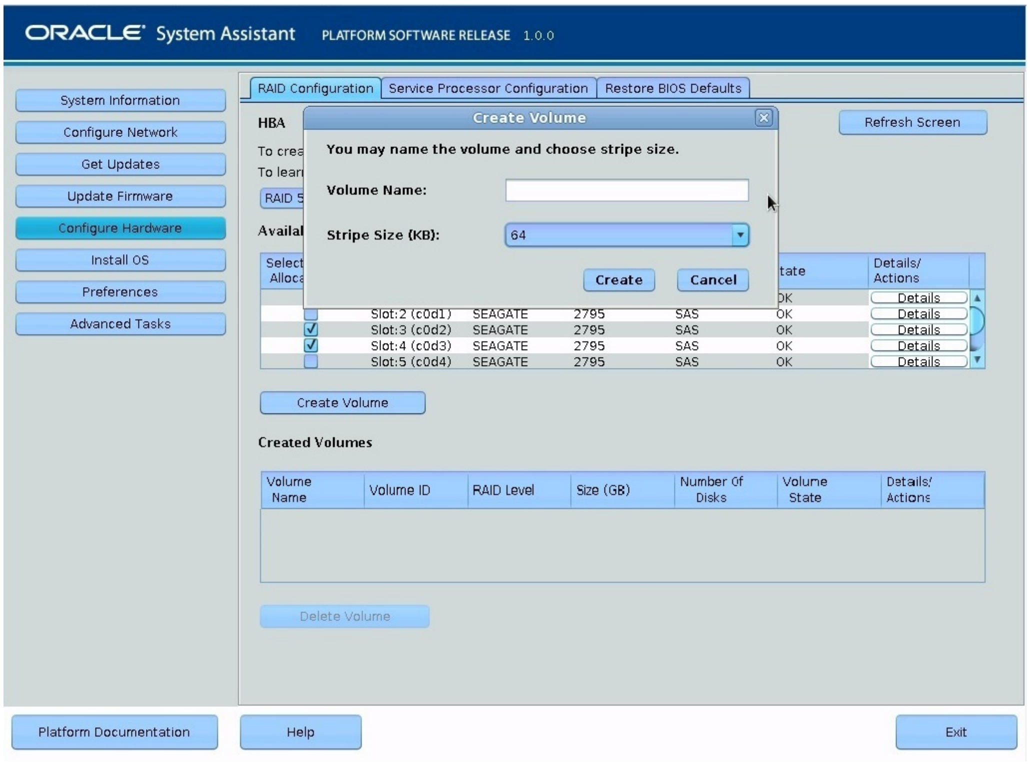
CONFIGURE VIRTUAL DISK PARAMETERS:

Virtual Disk Name	
Virtual Disk Size	
Virtual Disk Size Unit	MB [✓ GB] TB
Strip Element Size	64 KB
Read Policy	No Read Ahead [✓ Read Ahead]
Write Policy	[✓ Write Back] Write Through Force Write Back

Dynamically updates to display as Select Physical Disks or Select Disk Group based on the selection made in Select Physical Disks From.

PowerEdge R740
Service Tag : GS2NHK2
Back

Figure 6.29: RAID configuration on Dell server[18]


ORACLE System Assistant PLATFORM SOFTWARE RELEASE 1.0.0

System Information
Configure Network
Get Updates
Update Firmware
Configure Hardware
Install OS
Preferences
Advanced Tasks

RAID Configuration | Service Processor Configuration | Restore BIOS Defaults

Refresh Screen

HBA

To crea
To lear
RAID 5
Availal

Create Volume

You may name the volume and choose stripe size.

Volume Name:	
Stripe Size (KB):	64

Create Cancel

Select Alloca					tate	Details/ Actions
					OK	Details
	Slot:2 (c0d1)	SEAGATE	2795	SAS	OK	Details
✓	Slot:3 (c0d2)	SEAGATE	2795	SAS	OK	Details
✓	Slot:4 (c0d3)	SEAGATE	2795	SAS	OK	Details
	Slot:5 (c0d4)	SEAGATE	2795	SAS	OK	Details

Create Volume

Created Volumes

Volume Name	Volume ID	RAID Level	Size (GB)	Number Of Disks	Volume State	Details/ Actions

Delete Volume

Platform Documentation Help Exit

Figure 6.30: RAID configuration on Oracle system[19]

[18]From: https://www.dell.com/support/kbdoc/en-uk/000178269/dell-poweredge-how-to-create-a-virtual-disk-through-the-system-setup-on-14g-servers?lwp=rt

[19]From: https://docs.oracle.com/cd/E41059_01/html/E48314/z4000e311419867.html

6.9 Optical Media

6.9.1 CD

A CD (Compact Disc) is an optical medium (i.e. a laser is used for reading and recording), single-sided (recording on the underside of the disc). The diameter is usually 12 cm, but there is a smaller version (8 cm). The axis hole is always the same, 15 mm. The CD (as a single-layer DVD) is inserted into the drive with the printed side up.

Like a hard disk, a power cable and a data cable (nowadays usually SATA) must be connected to the CD (DVD) drive.

Unlike magnetic media, the data is stored on a spiral, not on concentric tracks (the start is at the centre of the disc, which is also different from magnetic media). The recording is done by means of different lengths of depressions (*pits*) separated by gaps (*lands*). A depression/gap or gap/depression transition is a logic 1, a long gap or depression is a logic 0, due to the coding used two 1s never follow each other.

Standards for CD: this term can be used to refer to various forms of CD media.

- *red book* – Audio CD (CD DA – Digital Audio); aimed at recording sound,
- *yellow book* – CD-ROM, read-only; unlike the red book, it focuses on data, error correction has been added,
- *orange book* – CD-R (single write) and CD-RW (repeat write),
- green book – CD-I, interactive CD, added support for interactive operations,
- white book – Video CD, usable for movies,
- blue book – Enhanced CD, CD plus and CD-G,
- beige book – Kodak PhotoCD, for recording and viewing photographs,
- scarlet book – SACD.

Transfer speeds are given as a multiple of 150 kB/s (which is typical for a red book – Audio CD), e.g. 16× means 16 times this speed.

CD-ROM (according to the yellow book) is read-only. Its physical structure is as follows:

- The base is a disk of a mixture of polycarbonate and polymethylmethacrylate, in which a spiral with depressions is extruded,
- The subsequent layer is reflective, reflecting laser light well (780 nm wavelength),
- the third layer holds label.

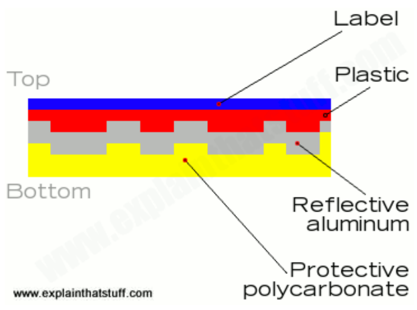


Figure 6.31: Physical structure of CD media[20]

[20]From: https://www.explainthatstuff.com/cdplayers.html

Reading is done by a light-sensitive element (photodiode, phototransistor, etc.) that detects the reflected laser beam. The reflective layer on the data medium reflects the laser beam at full intensity, but less energy is reflected at the depression in the reflective layer. The change in laser intensity is interpreted as logic 1.

CD-ROMs *are made* by pressing according to a preformed matrix (which determines the positions of the pits), the pressed disc is encased in protective plastic. This is why we cannot write to a CD-ROM in a conventional drive. The advantage of this process is the relatively low cost, especially in mass production, and the high durability of the medium.

✎ **CD-R (CD Recordable)** allows writing once and reading only afterwards. It has the same dimensions as a CD-ROM, but a slightly different internal structure, including the materials used.

There is a layer of *organic pigment* on the disc into which the recording is made, this layer is covered by a reflective layer of silver, silver alloy with another metal or gold and a cover layer of polycarbonate. In production, a spiral is molded into the disc to serve as a guide for the write head during writing.

There are *ATIP* (Absolute Time in Pregroove) marks on the spiral used to synchronize the writing of data at a constant speed. Also included on the disc from the factory is information about the disc capacity, manufacturer, writing requirements (laser power), etc. ATIP tags remain after writing, and are also used to detect the type of disk.

During *write*, the laser diode head follows the spiral with orientation according to the ATIP markers. At the point where the analogy of the depression is to be, the laser intensity and thus its temperature is increased, changing the chemical (optical) properties in the organic pigment layer. In reality, of course, nothing is burned, it is a thermochemical process.

Reading is similar to CD-ROM, but instead of pressed depressions we have light and dark places.

✎ **CD-RW (CD Re-Writeable)** allows writing repeatedly after erasing. The data layer into which the writing is performed is made of an alloy which crystallises at a certain temperature and at another temperature acquires an amorphous (non-crystalline) structure. Sites with a crystalline structure reflect more light and this is the principle behind reading data from the medium (the crystalline structure is analogous to land and the amorphous structure is analogous to pit).

Erasing of the disk is done by converting from an amorphous to a crystalline structure – the erased location is heated to a temperature below the melting point of the material, crystallization occurs. Subsequent writing is done by heating the area that is supposed to reflect less light (analogy to the depression in a CD-R) to a temperature above the melting point of the material and turning it amorphous.

The change between amorphous and crystalline structure is partially destructive, so erasing followed by writing can only be repeated about 1000×.

☞ **Remark**

The reflectivity of different types of media varies greatly. For CD-ROM it is 80 %, for CD-RW only 25 %, CD-Rs are somewhere in between. The quality of the burn depends on the quality of the media, the burning drive and also the burning speed. In general, higher speeds mean a higher probability of errors (the laser operates at higher power to heat the disc locations to the specified temperature at higher speeds). ☜

6.9.2 DVD

A DVD (Digital Versatile Disc or Digital Video Disc) has a much larger capacity than a CD. In addition, smaller pits are used and the track spacing is much smaller (all of which is the main reason for the increase in capacity), while the spiral itself is the same length and width as a CD. Further capacity increases are possible by storing data in two layers and also by using both surfaces of the media.

The dimensions of the DVD are the same as the CD for compatibility reasons (including the centre hole), but the internal structure is different. The carrier layer is half, usually actually two (to preserve the thickness of the media), with one or two data layers and a single or double-sided reflective layer in between.

In *reading*, a laser beam with a shorter wavelength is used than in CD (660 nm – red light), and the distance at which this beam is focused is different. For a comparison, see figure 6.32.

DVD standards are divided into two groups with mutual incompatibility. The first group of standards was issued by the *DVD Forum* group (minus standards). These standards have been criticised by some companies, mainly because of the unsatisfactory licensing terms and price, so the other (plus) standards were issued by the *DVD+RW Alliance*.

- DVD-ROM
- DVD-R, DVD-RW
- DVD-RAM – has trigger tracks instead of spirals, better error detection and correction
- DVD+R, DVD+RW

They differ in materials used, error rate (+ are slightly better, but have slightly less capacity), original price of the license. Current DVD drives usually support standards from both groups (usually with the exception of DVD-RAM). Dual-layer media is also indicated by the addition of the letters "DL", for example DVD+R DL is dual-layer DVD+R.

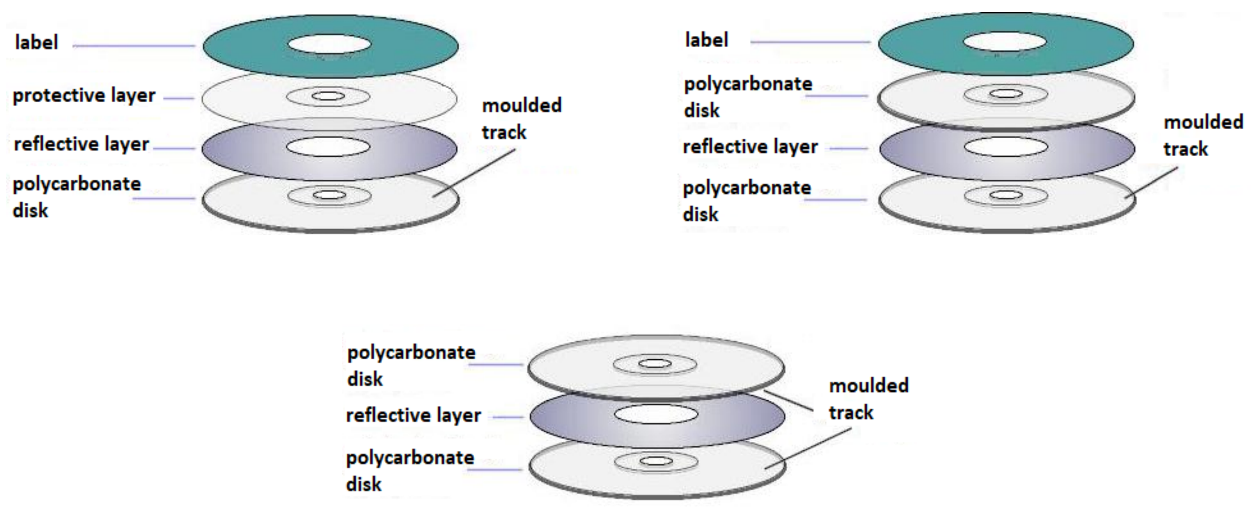


Figure 6.32: Comparison – CD, single-layer DVD, dual-layer DVD[21]

DVD+/-RW can also be used for packet writing – if we have the appropriate software, they can be treated like USB flash drives.

[21]Based on http://www.root.cz/clanky/nasledovnici-kompaktnich-disku-dvd/

Remark

So what is the difference between DVD-RAM and DVD+/-RW with packet writing? Both are rewritable, both allow you to work with optical media in a similar way to a hard disk or removable media.

For DVD-RAM, we need a special optical drive that supports this type of writing, and it writes in concentric circles (like a magnetic disk).

For DVD+/-RW with packet writing, a regular optical drive is sufficient, but we need special software, the data is written on a spiral.

DVD variants can be found in Table 6.5, including dimensions, number of sides and layers, and capacity. The most common are DVD-5 and DVD-9.

Labeling	Diameter	Sides	Layers per side	Capacity
DVD-1	8 cm	1	1	1.46 GB
DVD-2	8 cm	1	2	2.66 GB
DVD-3	8 cm	2	1	2.92 GB (2×1.46)
DVD-4	8 cm	2	2	5.32 GB (2×2.66)
DVD-5	12 cm	1	1	4.70 GB
DVD-9	12 cm	1	2	8.54 GB
DVD-10	12 cm	2	1	9.40 GB (2×4.70)
DVD-14	12 cm	2	2/1	13.24 GB
DVD-18	12 cm	2	2	17.08 GB (2×8.54)

Table 6.5: DVD variants

The features of a DVD are mainly derived from the requirements of the film industry. It is possible to use *DRM* (Digital Rights Management) to prevent duplication of the media: a copyright can be stored on the disk, which is not transferred during duplication, and illegal copies are detected accordingly.

The applicability of the DVD can be specified for one or more *regions*. There are 8 of these regions, 6 of which are for areas on Earth (e.g. Europe is in region #2, the USA and Canada have region #1), region #7 is not specified, and #8 is for mobile areas (planes, ships, etc.). DVDs designated in this way for a region can only be played on a player with that region set.

Transmission rates are, as with CDs, given as a multiple of the base rate, but this base rate is different. The 1× designation corresponds to a DVD speed of 1350 kB/s, which is 9× for a CD.

6.9.3 Structure of Rewritable Disk

The structure of a rewritable disc (including CD/DVD-R) is determined by the burning technology used. Usually there are several *zone types*:

- *internal zone* (information area) usually contains a table of contents (TOC), administrative information (such as the number of tracks and their start and end locations), test and calibration information,

- *Lead-In* – a security and administration zone for control data, identifier, etc., it is a zone with information about stored files,
- *Lead-Out* – end-of-disk warning,
- *external zone* – administration, test and calibration information.

Calibration information is used to calibrate the laser for writing.

✎ CD and DVD burning methods:

1. *DAO* (Disc-at-Once) – the whole disk is burned at once without continuously turning off the laser, used e.g. for music CDs. The disk is divided into five zones:

internal zonea	Lead-in	data zone	Lead-out	external zone

2. *TAO* (Track-at-Once) – one track is burned at a time, then the laser is turned off, other tracks can be added at any time. The disk has a similar structure to the previous one, but after Lead-out there is an empty space for writing additional tracks (each track dedicated to data contains Lead-in, data and Lead-out zones).

		internal zone	
track #1:	Lead-in	first track data	Lead-out
track #2:	Lead-in	second track data	Lead-out
etc.		⋮	
		external zone	

3. *MultiSession* – It is not necessary to burn the whole disk at once, only a single session is written to one or more tracks at a time. The Lead-in zone is only one (created on the first burn), the Lead-out zone is also only one (created on the last burn). The TOC is only created on the last burn at the same time as the Lead-out.

   At the end of a burn of a session that is not yet the last one on the disk, *closure block* is inserted after the data instead of Lead-out, at the beginning of the next session the *opening block* is inserted and then the data follows. After the data of the last burned session, the Lead-out zone is inserted instead of the Close-out block. Below we can see the structure of a disk burned in MultiSession mode (already full, including the closing sections). The written data sections can of course be of different lengths.

	internal zone	
Lead-in	data_1	closure block
opening block	data_2	closure block
opening block	data_3	closure block
	⋮	
opening block	data_n	**Lead-out**
	external zone	

4. *Incremental Recording* (for DVD) – similar to MultiSession, but without closing and opening zones. A Lead-in zone is created at the beginning of the first write, then the disk gradually fills up, and a Lead-out zone is created at the end of the last write. Until this last zone is created, the disk is not readable in drives that do not support this method; after it is created, the disk structure is similar to a DAO disk.

5. *Sequential Recording* – similar to Incremental Recording, but for better compatibility with other drives, a temporary Lead-out zone is created after each data write, and the next time the disk is written, this zone is overwritten with data (and a new temporary Lead-out zone is created).
6. *Random Recording* – this is a method very similar to hard disk methods, data can be written anytime and anywhere.

File naming and organization: When burning, we should also set a standard for naming files and directories, generally a file system.

The oldest standard used for CDs is *ISO 9660*, which specifies eight characters for the name and three characters for the file extension. Because long file names are commonly used today, as well as diacritics, this standard is not usually chosen.

Much used is *ISO 9660 Level 2* (also for CD), which extends the file name space to 30. We also encounter the *Joliet* and *Romeo* standards. Joliet stores filenames in two forms at once – a long name taken from the operating system, plus a shortened name. Romeo stores only the long name (128 characters).

When burning DVDs, different file systems are used. The most popular is *UDF* (used for DVD and Blu-Ray, but can also be used for CD). It supports long file names.

It is also possible to use the *MRW* file system on DVD. It is newer than UDF and has several new features, perhaps the most interesting of which is improved error management.

6.9.4 Blu-Ray

Blu-Ray disks are again the same size as CD/DVD for backwards compatibility. The data is stored on a spiral, the laser beam has a wavelength of 405 nm (blue light, although physically it is more like violet). This wavelength and the smaller distance of the head from the surface made it possible to increase the capacity of the disk – a single-layer Blu-ray with a diameter of 12 cm has a capacity of 25 GB, a dual-layer up to 50 GB.

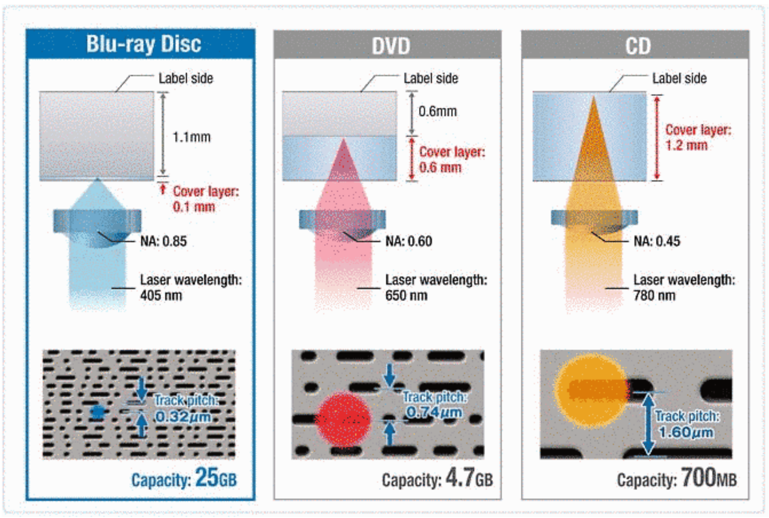


Figure 6.33: Comparison of CD, DVD, Blu-Ray[22]

[22]From: https://deepblue.lib.umich.edu/bitstream/handle/2027.42/86196/me450fall2009finalreport-project02-masklessphotolithographysystem.pdf?sequence=1

✎ Blu-Ray variants:

- BD-ROM – read-only, moulded at the factory,
- BD-R – single-bournded,
- BD-RE – rewritable,
- BD-AV – for amateur home videos,
- BDMV (Blu-Ray Disk Movie) – for professional video,
- BD-J (Java) – interactive Blu-Ray using Java,
- Mini-BD – smaller size (8 cm).

Adding "DL" indicates dual-layer media, adding BD+ (for BD-ROM) indicates the use of the hardware protection described below.

✎ Blu-Ray originally used *AACS* (Advanced Access Content System) for content protection (DRM), but this was broken. Therefore, a new hardware protection system *BD+ key* from Cryptography Research was added; it is a component that must be added to the drive by the manufacturer.

Chapter 7

Expansion Cards

Quick preview: In this chapter, we will look at the different types of expansion cards that can be found in a computer. In more detail, graphics, sound, and networking. Expansion cards (also dedicated) are plugged into a slot leading to a specific bus, most commonly PCIe today.

Keywords: Expansion card, low-profile, graphics card, graphics chip, integrated card, text mode, graphics mode, pseudographics, video memory, GPU, shader, stream processor, sound card, sampling, sampling rate, sample depth, sound synthesis.

Objectives: After studying this chapter, you will have a general overview of some of the common types of expansion cards – graphics and sound cards.

7.1 What an Expansion Card Is

Expansion card (adapter) is a module that extends the functionality of the motherboard. It is made of similar material to the motherboard and has a very similar structure. It is connected to the motherboard (and thus to the processor) by inserting it into the slots on some bus.

Also, there are *integrated solutions* (what we would otherwise have separately on a card designed for a slot is integrated in the chip on the motherboard, in the chipset, in the processor, etc.). But then it is not a card in the true sense of the word.

On the other hand, an *dedicated* expansion card is just a card designed for a slot (i.e. not integrated and not actually external).

Cards marked as low-profile can also fit into low cases. In any case, it is a good idea to check the overall dimensions of the card (also the length).

Because graphics cards in particular have a large heatsink, width can also be an issue – collisions with cards in adjacent slots, the CPU cooler, or anything else nearby.

7.2 Graphics Card

Graphics card (video card) is used to control the graphics displayed mostly on the monitor. Its task is to process and transform graphics data into a form that can be understood by the monitor or other devices connected to the card. Nowadays it is often integrated usually inside the processor.

✎ *Graphic chip* is a chip that contains (among other things) a graphics processor. Most graphics card manufacturers use graphics chips from the two major manufacturers and fit them to graphics cards under their own logo, or add “something of their own”.

Almost exclusively today we see graphics chips from AMD (usually Radeon chips) and Nvidia (usually GeForce chips), and to a lesser extent specific chips from Matrox (which has special chips for CAD, for example). AMD and nVidia chips can be found on graphics cards from MSI, Asus, Gigabyte, Sapphire, Zotac and others.

✎ Integrated graphics have the following features:

- sufficient for office use,
- cheaper,
- video memory usually “borrows” from memory chips, so the cooperation between graphics and video memory is slower.

Today, graphics cores are integrated into most processors. Generally, graphics in AMD APUs are considered better (but it depends on the specific processor type), and the quality of graphics cores in Intel processors is also graded. In general, the more powerful the processor as a whole, the better the integrated graphics core.

7.2.1 Basic Display Types and Video Memory Content

✎ **Text mode** allows to display only text (in colors, or flashing). It is usually encountered at system boot, in the old BIOS (not UEFI), bootable diagnostic tools, and text consoles (the Windows Command Prompt also works in text mode, however emulated).

The text mode can be operated in different *resolutions* (number of rows and columns per displayable characters), e.g. 80×25 means 80 columns and 25 rows, $80 \times 25 = 2000$ characters on the screen.

In text mode, the screen is divided into a *grid* of character cells arranged in rows and columns. The contents of these cells are stored in a video memory, which is a memory space for keeping the screen up-to-date; we can think of it as a 2D array. For each cell, two pieces of information are stored – the ASCII code of the character to be displayed and its attribute. Each of these pieces of information occupies one byte, so for each cell we have 2 B. Physically, the cells are stored in the video memory “by rows”, i.e. first the first row, then the second, and so on.

The attribute of a character specifies primarily the color. In the most common text mode, we have 8 bits available for the attribute, the last 4 of which are used for the color of the character itself ($2^4 = 16$ different colors), the 3 bits before them for the background color of the character ($2^3 = 8$ different colors), and the remaining bit (the first one) determines whether the character should blink.

If multiple different colors were to be displayed, we would need more bits to store the attributes, so there would need to be more than 2 B’s per grid character.

Example

For example, the following coloured text is displayed on the screen: abc
if the following information is in the video memory:

1 B	1 B	1 B	1 B	1 B	1 B	
Character	*Attribute*	*Character*	*Attribute*	*Character*	*Attribute*	
`'a'`	0 000 1111	`'b'`	0 001 1110	`'c'`	0 000 0100	...

Instead of letters, the ASCII values of these characters should correctly be here. The attribute consists of eight bits that determine the background color, the text color, and whether to blink the character. The meaning of each bit is as follows:

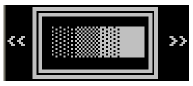
1 bit	3 bits	4 bits
blink	background color	text color

Graphics mode allows you to display a matrix of pixels (viewable points) instead of a matrix of characters. Each pixel is determined only by its color from a given color palette. So each colour has a number (usually 0 is black, the highest possible value is white, the others are something in between). Which specific colour is hidden under which number is determined in the *colour palette*.

Color Depth is the number of bits needed to store the color information. For example, a color depth of 8 bits means that we have $2^8 = 256$ colors available, a color depth of 24 bits, or 3 B, means that there are $2^{24} = 16\,777\,216$ colors available.

Pseudographics offers possibilities to work with (rough) objects in text mode. We create images or various interface elements (menus, buttons, tables, etc.) using special characters (rectangles of different sizes, the intensity is determined by the rectangle grid mask), monochrome or colored (what text mode offers).

The pseudographic interface is encountered in BIOS Setup, but also in many other applications. On the right is an example of a simple shaded button made of pseudographic characters.



7.2.2 Graphics Cards Evolution

Graphics card evolution has gone from video memory storage to 3D accelerators to today's GPUs.

As monitors with higher and higher resolutions and more colours have emerged, so have the standards, which are simplistically referred to by various abbreviations:

- MDA (Monchrome Display Adapter) – 2 colours (monochrome text),
- CGA (Color Graphics Adapter) – TV compatibility, 4 colours, multiple colours only at very low resolution,
- EGA (Enhanced Graphics Adapter) – 16 colours from a palette of 64 possible colours,
- VGA (Video Graphics Array) – 16 colors, but the much wider base palette for selection (2^{18} colors),
- SVGA (SuperVGA) – Graphics accelerators and later full-fledged graphics chips began to be used, at first only the ability to draw the most commonly used basic shapes (line segments, hardware cursor, area filling, etc.),
- XGA (eXtended Graphics Array), SXGA, UXGA, QXGA, HXGA gradually increasing resolution and colour depth.

Graphics accelerators and later full-fledged graphics chips began to be used, at first only the ability to draw the most commonly used basic shapes (line segments, hardware cursor, area filling, etc.),

Additional information

On the web, these standards are described on many pages, such as:
https://www.cknow.com/cms/ref/video-display-standards.html.

3D accelerators were graphics chipsets with their own simple graphics API (application programming interface, i.e., a set of functions that could be used to control it) and typically processed an object-based 3D representation of an image into a 2D array of pixels in video memory. It's called an accelerator because it speeds up imaging (the computation doesn't have to be done by a processor, plus the 3D accelerator is specialized for it).

Originally, the 3D accelerator was a special expansion card (a regular graphics card could do 2D, this card was for 3D and connected to the "2D" card, which sent its output to the monitor). Very soon, however, solutions containing both functionalities on one card appeared, today we can't even buy anything else.

3D acceleration could theoretically be done in software by the processor (and is done if, for example, the game uses an API that is not directly supported by our graphics chip in hardware), but it's much slower.

GPU (Graphic Processing Unit) is the next generation. It is a specialized SIMD processor designed primarily to process image data (which does not mean that it cannot process something else with a similar structure). And it's the GPU that we have in graphics chips, hence graphics cards.

There are various *graphical APIs* (programming interfaces, frameworks), which are basically sets of commands, objects, etc. for manipulating a 3D scene (e.g. in a computer game) or other mathematically described objects. The most well-known are DirectX and OpenGL, others work with these two in some way (Direct3D, OpenCL, CUDA, etc.). These are libraries that exist for various operating systems, so code written in them is more or less portable among different platforms.

7.2.3 Graphics Card Structure

Graphics memory is represented by several chips around the GPU on the card and is used to store this information:

- firmware,
- storage of objects and textures for computing 3D scenes, generally everything the GPU has to process and store in video memory for output, *videomemory* with the contents of the screen/screens and pages (*Frame Buffer*) – displayed "frames"),
- cache.

The size and speed of the video memory has a big impact on the quality of the graphics card, its throughput.

Figure shows the PCB (printed circuit board) of the Nvidia GeForce RTX 3080 graphics card from 2020. In the middle, we can see the graphics processing unit (GPU) surrounded by graphics memory chips. It's still sheathed with a heatsink, of course.

[1]From: https://www.cnews.cz/nvidia-geforce-rtx-3080-20gb-potvrzena-behem-2-3-mesicu/

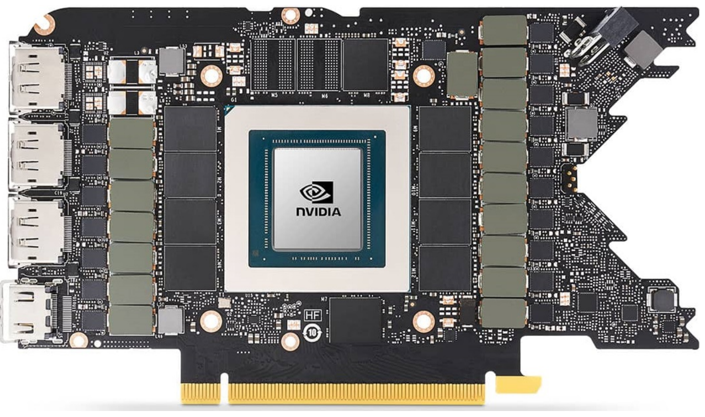


Figure 7.1: PCB of Nvidia GeForce RTX 3080[1]

✎ The form of memory chips for graphics cards can vary:

- older variants – VRAM (VideoRAM, dual port), SGRAM (Synchronous Graphic RAM) allowing block operations, WRAM (Window RAM) dual port with block operations,
- DDR2/3/4 memory – same as RAM, not optimized for video memory role, but cheaper,
- GDDR memory (Graphics Double Data Rate) – memory designed specifically for graphics cards, used on cards with Nvidia's graphics chips, today GDDR5X, next generation GDDR6 and the latest GDDR6X,
- HBM (High Bandwidth Memory) – standardized in 2013, the latest memory from AMD and Hynix (also used by Samsung), the newer generation is HBM2 from 2017, a year younger is an improvement HBM2E.

✎ **Graphical Processing Unit (GPU)** calculates all necessary image parameters sequentially and determines specific pixel values at different positions in the video memory. GPUs commonly use some of the technologies we learned about in the chapter on processors – for example, the scalar architecture, i.e., *pipeline* is called an instruction processing path, and today's GPUs have multiple pipelines working in parallel.

Thus, the GPU normally processes data in parallel. This is also possible with current processors (CPUs), but there are some differences. In a GPU, special small parallel programs called *shaders* run across the entire array of computing units, and there is also more memory.

☞ **Remark**

So what about *computing cores*? A typical CPU has several cores. A GPU has hundreds to thousands of cores. While CPU cores are general-purpose (complex), capable of performing a wide variety of tasks, GPU cores are small, simple, for a narrow set of tasks.

GPUs can be faster than the main processor for certain types of tasks, but only when it comes to the need to perform similar types of computations in parallel and repeatedly. This is used in graphics and multimedia processing, of course, but there are applications for using GPUs for scientific computing, or for encryption or, conversely, for breaking security mechanisms (typically by brute force technique), cryptocurrency mining, and other types of tasks where the same relatively simple operation is performed in parallel on a long vector of data. ☜

✎ **GPU components.** Whichever of the graphical APIs is used, a 3D scene consists of objects whose basic representation (i.e. their surface) is in the form of a network of triangles or other simple graphical elements. The points where these triangles meet are called vertices. Converting an object into a representation using triangles is called *triangulation*.

If we manipulate the vertices, we can change the shape and position of the object. The more vertices we use for an object, the more detail we can represent on the surface of the object.

✎ *Shaders* are small, simple programs that perform a specified kind of calculation on the displayed scene, according to their type. This term can also be applied to the computational units that execute these programs. Shaders, in the sense of programs or pieces of code, come in several different forms, depending on what they do:

- vertex shaders can manipulate vertexes and thus change the shape or position of an object,
- geometry shaders also work with vertices, but in a slightly different way – they can dynamically add or remove vertices to create interesting animations,
- tessellation shaders have a similar role to geometry shaders (they were added in slightly later versions of the API), but are a bit more specialized: their purpose is to dynamically refine surfaces, adding details,
- pixel shaders are used to transform a 3D scene into a 2D image, they determine which pixel of the output is transformed to which part of the object surface (including, e.g., visibility when overlapping objects),
- other more specialized types of shaders can appear.

Most shaders can be classified as 3D shaders (they work with vertexes, with a 3D scene), while pixel shaders are 2D shaders (they are used to create 2D output).

Shaders in the sense of computational units are nowadays usually used in the form of unified shaders, i.e. they can change their role between vertex, geometry, pixel,... In figure 7.2 we can see the difference between integrated solutions and dedicated cards. In all three cases they are unified shaders, but there is a very big difference in their number.

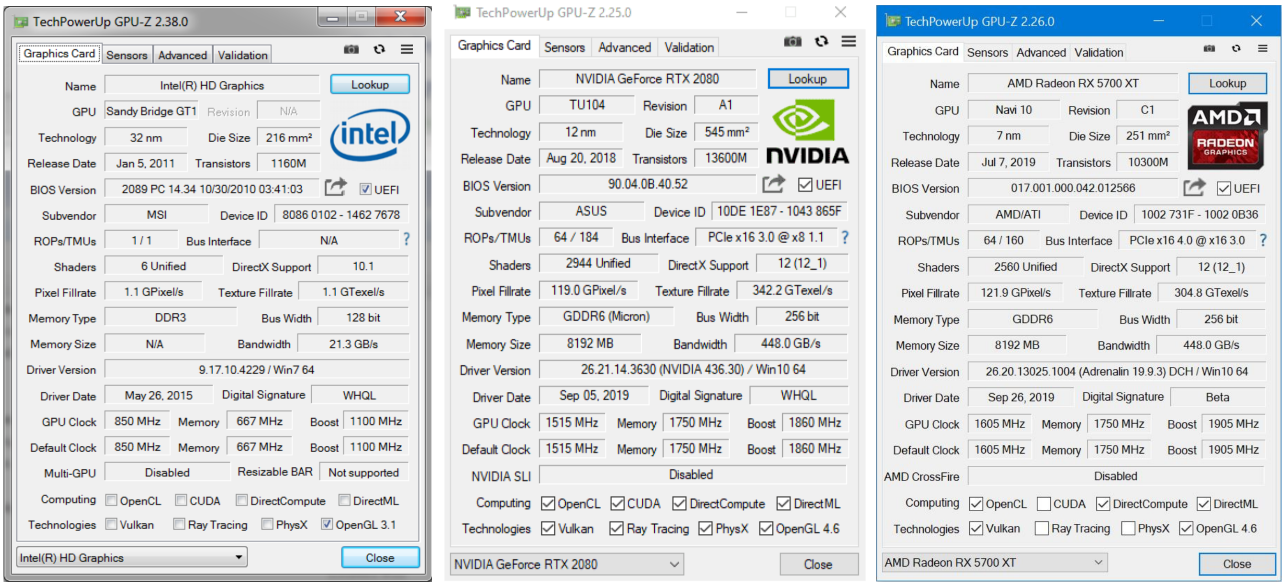
Texture is an image that determines the appearance of the surface on which the object is "wrapped". It determines, e.g. colors of individual parts of the object, reflectivity of the object material, etc.

✎ *TMUs* (Texture Mapping Units) work with textures, which are usually bitmaps/images that are to be mapped to the surface of individual objects in a scene and determine the color or texture of the surface of that object. The task of the TMU is to adjust the texture by rotation, resizing, and other mathematical operations so that it "fits" to the surface of the object.

✎ *ROP* (Render Output Units, also rasterization units) are the components of the graphics chip that are responsible for rendering, i.e. creating the final 2D raster image and storing it in the frame buffer, from where it is transferred to the output. Their job is to determine what any pixel of the resulting image will look like, i.e., the output of the pixel shaders applies textures matched by the TMUs. The ROP units also modify the image to make it look more realistic, for example smoothing the edges by anti-aliasing)

7.2.4 How to Find out the Chip

How can we find out which graphics chip is on a particular graphics card, or which graphics core is integrated into the processor? We can use the common tools we have seen on the previous pages or a specialised tool. Figure 7.2 shows the output of Techpowerup's GPU-Z tool.



TechPowerUp GPU-Z 2.38.0

Name: Intel(R) HD Graphics
GPU: Sandy Bridge GT1 | Revision: N/A
Technology: 32 nm | Die Size: 216 mm²
Release Date: Jan 5, 2011 | Transistors: 1160M
BIOS Version: 2089 PC 14.34 10/30/2010 03:41:03 | ☑ UEFI
Subvendor: MSI | Device ID: 8086 0102 - 1462 7678
ROPs/TMUs: 1 / 1 | Bus Interface: N/A
Shaders: 6 Unified | DirectX Support: 10.1
Pixel Fillrate: 1.1 GPixel/s | Texture Fillrate: 1.1 GTexel/s
Memory Type: DDR3 | Bus Width: 128 bit
Memory Size: N/A | Bandwidth: 21.3 GB/s
Driver Version: 9.17.10.4229 / Win7 64
Driver Date: May 26, 2015 | Digital Signature: WHQL
GPU Clock: 850 MHz | Memory: 667 MHz | Boost: 1100 MHz
Default Clock: 850 MHz | Memory: 667 MHz | Boost: 1100 MHz
Multi-GPU: Disabled | Resizable BAR: Not supported
Computing: ☐ OpenCL ☐ CUDA ☐ DirectCompute ☐ DirectML
Technologies: ☐ Vulkan ☐ Ray Tracing ☐ PhysX ☑ OpenGL 3.1
Intel(R) HD Graphics

TechPowerUp GPU-Z 2.25.0

Name: NVIDIA GeForce RTX 2080
GPU: TU104 | Revision: A1
Technology: 12 nm | Die Size: 545 mm²
Release Date: Aug 20, 2018 | Transistors: 13600M
BIOS Version: 90.04.0B.40.52 | ☑ UEFI
Subvendor: ASUS | Device ID: 10DE 1E87 - 1043 865F
ROPs/TMUs: 64 / 184 | Bus Interface: PCIe x16 3.0 @ x8 1.1
Shaders: 2944 Unified | DirectX Support: 12 (12_1)
Pixel Fillrate: 119.0 GPixel/s | Texture Fillrate: 342.2 GTexel/s
Memory Type: GDDR6 (Micron) | Bus Width: 256 bit
Memory Size: 8192 MB | Bandwidth: 448.0 GB/s
Driver Version: 26.21.14.3630 (NVIDIA 436.30) / Win10 64
Driver Date: Sep 05, 2019 | Digital Signature: WHQL
GPU Clock: 1515 MHz | Memory: 1750 MHz | Boost: 1860 MHz
Default Clock: 1515 MHz | Memory: 1750 MHz | Boost: 1860 MHz
NVIDIA SLI: Disabled
Computing: ☑ OpenCL ☑ CUDA ☑ DirectCompute ☑ DirectML
Technologies: ☑ Vulkan ☑ Ray Tracing ☑ PhysX ☑ OpenGL 4.6
NVIDIA GeForce RTX 2080

TechPowerUp GPU-Z 2.26.0

Name: AMD Radeon RX 5700 XT
GPU: Navi 10 | Revision: C1
Technology: 7 nm | Die Size: 251 mm²
Release Date: Jul 7, 2019 | Transistors: 10300M
BIOS Version: 017.001.000.042.012566 | ☑ UEFI
Subvendor: AMD/ATI | Device ID: 1002 731F - 1002 0B36
ROPs/TMUs: 64 / 160 | Bus Interface: PCIe x16 4.0 @ x16 3.0
Shaders: 2560 Unified | DirectX Support: 12 (12_1)
Pixel Fillrate: 121.9 GPixel/s | Texture Fillrate: 304.8 GTexel/s
Memory Type: GDDR6 | Bus Width: 256 bit
Memory Size: 8192 MB | Bandwidth: 448.0 GB/s
Driver Version: 26.20.13025.1004 (Adrenalin 19.9.3) DCH / Win10 64
Driver Date: Sep 26, 2019 | Digital Signature: Beta
GPU Clock: 1605 MHz | Memory: 1750 MHz | Boost: 1905 MHz
Default Clock: 1605 MHz | Memory: 1750 MHz | Boost: 1905 MHz
AMD CrossFire: Disabled
Computing: ☑ OpenCL ☐ CUDA ☑ DirectCompute ☑ DirectML
Technologies: ☑ Vulkan ☐ Ray Tracing ☐ PhysX ☑ OpenGL 4.6
AMD Radeon RX 5700 XT

Figure 7.2: GPU-Z (integrated, Nvidia, AMD)[2]

The first screenshot shows information about the integrated Intel HD Graphics in the Intel processor from the Sandy Bridge family (i.e. 2nd generation Core i), the graphics core is made with 32nm technology and uses DDR3 memory (it is obvious that it takes from memory modules). It supports DirectX version 10.1 and OpenGL version 3.1 technologies. It's not much, but quite adequate for older integrated graphics. If we were to click the *Lookup* button, we would get a web page with more detailed information about this graphics core.

On the second screenshot there is info about the Nvidia GeForce RTX 2080 graphics chip made with 12nm technology equipped with GDDR6 memory (on Asus graphics card), on the third screenshot there is info about the AMD Radeon RX 5700 XT chip made with 7nm technology with the same type of memory.

Shaders are programmed in the language according to the chosen graphics API. For example, GLSL is used to program shaders for OpenGL, HLSL is used to program shaders for Direct3D. These languages are supported in special development environments (usually for game development or virtual reality) such as Unity or Unreal Engine.

7.2.5 Powering Graphics Cards

Weaker cards are powered via the PCI Express bus, but more powerful cards require *additional power supply*. If this is not provided, they operate in "power saving mode", which is equivalent in performance even to integrated cards. A 4 to 8-pin PCIe power connector ("cubes") is used, or it can be two 6-pin or 8-pin connectors, or something else with a reduction.

If there is no connector leading from the power supply, a reduction from two 4-pin Molex connectors to an 8-pin PCIe power connector is used. The power supply should be more powerful (400-500 W), even more if using multiple cards, also depends on the card performance and power supply efficiency.

[2]Resources: own, https://cryptocurrency-miner.com/gpu-z/, https://www.techpowerup.com/news-tags/GPU-Z
[3]From: http://www.legitreviews.com/article/730/1/, https://www.hwcooling.net/nvidia-ampere-geforce-rtx-3000-opravdu-novy-12pinovy-napajeci-konektor-udajne-ceny/, https://www.exasoft.cz/

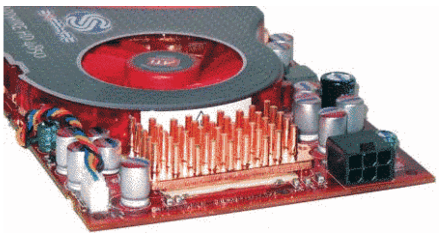
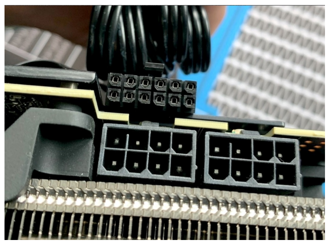
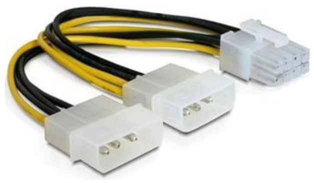




Figure 7.3: Additional power supply on graphics – 6pin, 2× 8pin, molex to 8pin reduction[3]

7.3 Sound Card

7.3.1 Principle

A Sound Card (or its integrated variant) is a device that performs the conversion between analog (microphone) and digital (CD, DVD, file, etc.) signals, produces sounds from digital data, and sends them to speakers. It may include interfaces to multimedia devices/tools, including possibly music (MIDI – Musical Instrument Digital Interface).

It can be dedicated, external or integrated (e.g. in the chipset). For dedicated ones we can meet PCI, PCIe×1, PCIe×16 bus, external ones are usually connected via USB, Thunderbolt or older ones via Firewire.

The most well-known sound card and chip manufacturers are Creative Labs, Axagon, Asus, Focusrite, Genius, Gamebird, Sweex, Orico, Realtek (usually integrated).

✎ **3D sound** is important in some games, movies and sometimes in physical modelling. It is achieved in two possible ways:

- *sorrounding* – Dolby Surround or other similar standard, usually using five or more speakers suitably positioned to surround the listener,
  - horizontal – the speakers are positioned horizontally around the room,
  - 3D – the speakers are positioned not only in plane, but also at different heights, e.g. some are screwed to the ceiling,
- *emulated* – a DSP chip on the sound card or in the speaker modifies the sound to give a sorround or 3D impression (we only have two speakers).

7.3.2 Sampling and Sound Synthesis

✎ **Sampling.** During sampling, i.e. the conversion of analogue sound (generally a signal) to digital data, some information is always lost. The principle can be seen in Figure 7.4. The horizontal axis is time, and the signal value is taken at regular intervals.

✎ *Sampling frequency* is the frequency at which samples (instantaneous values) of analog audio are detected, these values are then digital data. The higher the frequency (more frequent sampling), the more faithful the sound after digitization (for example, for audio CDs it is usually over 44 kHz, or 44,000 samples per second). It is assumed that the human ear can perceive an average range of 18 Hz to 18 kHz.

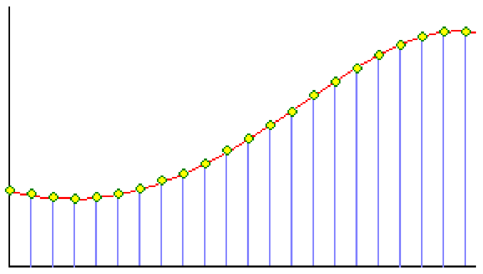


Figure 7.4: Analog sound sampling

Also related to sample rate is the concept of *sample depth* (sample precision, sample size). It is the number of bits used to store a single sample. A higher sample depth (more bits) means a finer division of the interval (in Figure 7.4, this interval would be plotted vertically). The usual sample depth is 8 or 16 bits.

Sound Synthesis is actually the opposite process to sampling. We convert digital data into sound. There are several ways to generate sound from data, differing mainly in the quality of the resulting sound and the complexity of the conversion processor.

FM synthesis is the easiest and cheapest (and also the lowest quality). Instrument sounds are generated by a special chip by composing from pre-prepared samples. Other methods use higher quality sample representations, for example MIDI synthesis uses modules for different instruments. Physical modelling synthesis performs a software emulation of the sound based on physical principles, but this method is very computationally intensive.

7.3.3 Interfaces

Like all other cards, the sound card must be connected to the motherboard and via external connectors to other devices such as speakers, radios, and musical instruments.

The internal bus for connecting the sound card is usually PCI or PCIe×1. External cards are usually connected via USB bus.

The higher quality cards that support multi-channel sound have multiple color-coded outputs. 6-channel cards have a blue Line-In, a pink Mic-In, a green for the Front Speaker Out, a grey for the Rear Speaker Out, and a yellow orange for the Center or Bass (Center/Subwoofer). The 8-channel also has black connectors for the side speakers (Side Speaker Out).

We can also encounter a digital output, the connector is labeled S/P-DIF (abbreviation of Sony/Philips Digital Interconnect Format), we can see this connector in the picture 7.5 on the far left.

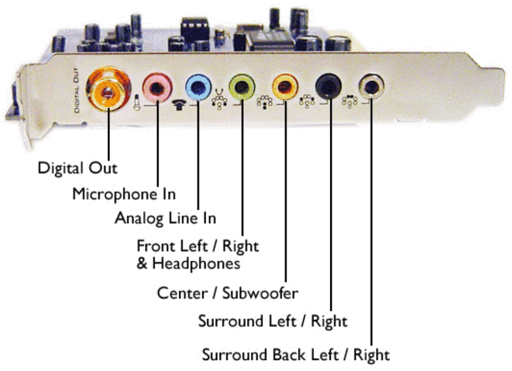


Figure 7.5: Sound card connectors[4]

Figure 7.6: External sound cards Axagon ADA-17 USB HQ mini autio and Motu 896mk Hybrid[5]

[4]From: http://koalazereeukalipt.wordpress.com/
[5]From: https://www.tsbohemia.cz, https://www.muzikant.cz

Chapter 8

I/O Devices

Quick preview: The topic of the chapter is various input and output devices, including devices handling “both directions”, i.e. input-output devices. Of the input devices, we’ll briefly focus only on some pointing devices as tablets, and touchscreens, and then we’ll deal more with devices that work with image, primarily screens.

Keywords: Character, block and special devices, pointing device, trackball, trackpoint, touchpad, tablet, touchscreen, resistive and capacitive technology, additive and subtractive model, RGB, CMYK, CIE system of colorimetry, sRGB, Adobe RGB, ProPhoto RGB, pixel, color depth, resolution, screen response time, refresh rate, input lag, CRT screen, cathode ray tube, refresh rate, LCD panel, CCFL and LED backlight, Edge LED, TFT, TN panel, IPS, VA, OLED, AMOLED.

Objectives: After studying this chapter, you will know how some types of input and output devices work, learn the principles of display devices, and become familiar with their features. You will be able to evaluate different types of display devices according to their parameters.

8.1 Input and Output

First, let’s categorize all the devices that can be included in this chapter according to the direction of data flow and the size of the batch of data being processed.

By data flow direction

- input devices – used to input data to the computer (keyboard, mouse and other pointing devices, scanner and other digitizing devices, microphone, camera, etc.),
- output devices – used to output data from the computer (screen, printer, plotter, speakers, etc.),
- input/output devices – can do both (touchscreen, multifunction device, etc.).

By batch size of processed data

- character devices – small transmission width, one or several characters (mouse, keyboard, printer, microphone),
- block devices – larger transmission width, the transmitted block of data has a certain structure or it is necessary to transmit metadata along with the data[1] (storage devices),

[1]Metadata is data about data. For example, for a file stored on disk, metadata is the file name, position in the file

- special devices – one type of data with a very small bit width (one or a few bits) without any structure, such as a timer.

8.2 Input Devices

8.2.1 Trackball, Trackpoint, Touchpad, Tablet

Trackball is a ball controller, a ball built into the board, but can be a separate device. It is a “mechanical mouse inverted”, also tends to be a near analogue of buttons. Nowadays, this device is not very common anymore. The big disadvantage is mainly the dirt from the user’s fingers clogging the ball, which causes inaccuracy (cursor jumping), the ball and the scanning rollers had to be cleaned frequently.

Trackpoint (Pointing stick, Flex point, SensePoint,. . .) is a pin on the keyboard (or other surface) on a joint, it works similarly to a joystick. Some laptop keyboards are equipped with it. The movement of the trackpoint is translated into a numerical form similar to that of a mouse.

Originally an IBM specialty (a division later sold to Lenovo), it is rarely found under other names by other manufacturers.

Touchpad is a touchpad, common in laptops. The slight movement of the finger on the pad is decoded and transmitted to the computer, the drivers also allow scrool operation – so scrooling is a matter of drivers (based on software), the touchpad itself does not do anything like that. Current touchpads use the same technology as touchscreens.

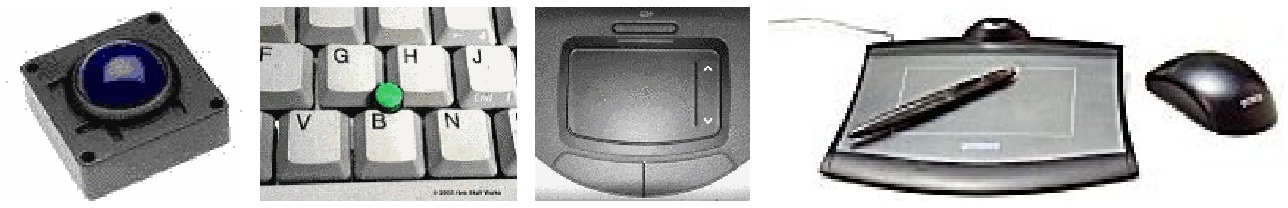


Figure 8.1: Trackball, trackpoint, touchpad, tablet

Tablet (this does not mean a separate device) consists of a solid pad and a special pen. The pen can be active or passive, the active pen requires power and is usually powered by electromagnetic induction.

In addition to coordinate changes (similar to a mouse), the tablet also distinguishes between different pressure levels. The total amount of pressure that can be resolved depends on the quality of the tablet. The response to a change in pressure can be e.g. a change in the thickness of the drawn line.

Remark

The tablet in the above sense gradually became a separate device with its own operating system.

Conversely, in recent years, so-called graphic drawing tablets, which connect to a computer and serve as a large touchscreen, have become increasingly popular. Well-known brands include Wacom and XP-PEN.

system (where it is to be stored), access permissions, etc.

Figure 8.2: Wacom Cintiq 16, the graphic drawing tablet[2]

8.2.2 Touchscreen

We will discuss displays themselves later. In the early 1970s, several different touch screen solutions appeared independently of each other, all of which were "single-touch", i.e. they could process only one touch at a time. Multi-touch did not appear until the 21st century.

There are several different touchscreen technologies. The most common are

- *resistive touch technology* (older) – two layers of electrically conductive material come together when touched, the point of contact can then be detected; can be operated by any object, high precision, low power consumption, cheaper, but transparency is worse, less resistant (if the top layer is damaged it stops working completely),
- *capacitive touch technology* (newer, more common) – we have only one conductive layer applied on an insulator (like glass), touch is registered only when we touch something conductive (even human finger is conductive); precise and durable technology, transparency is better than resistive technology, it is more durable, but can only be operated with a finger or a special pen intended for capacitive displays, supporting multi-touch.

There are other technologies – e.g. acoustic conductive technology, IR technology.

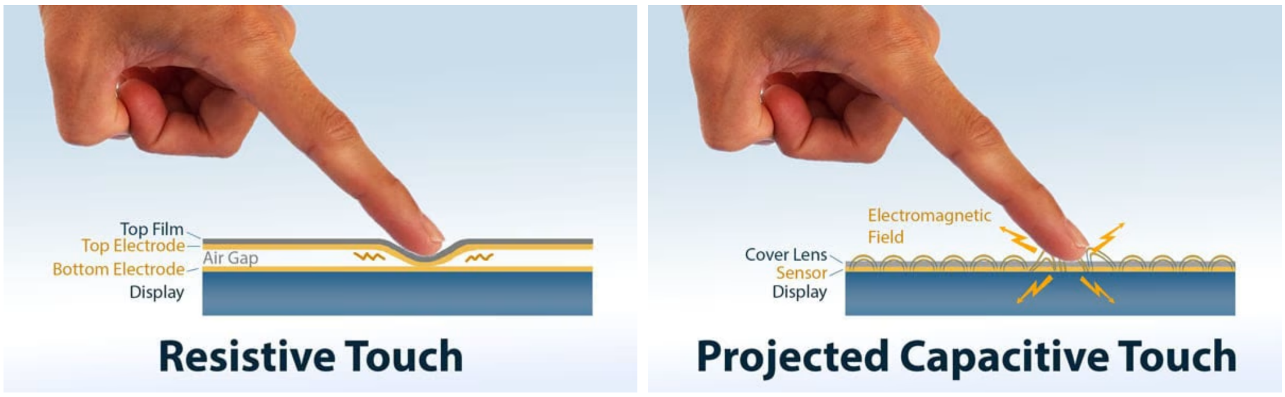


Figure 8.3: Comparison of the resistive and capacitive touch technology[3]

[2]From: https://www.amazon.co.uk/
[3]From: https://www.newvisiondisplay.com/capacitive-vs-resistive-touchscreen/

8.3 Displaying Colours

8.3.1 How Computer Processes Colours

The computer can only process numbers, so we have to convert colours to numbers. Each number has an upper limit. There are an infinite number of colours from a human point of view, but computers cannot handle infinity, so a way had to be devised to represent a wide enough range of colours using finite numbers. The method of representing colours in computer is based on the following principles:

- each colour is created by combining *basic colours*,
- specifies the following basic colours and the method of combinating them.

✎ There are two types of display devices:

1. devices that display by generating light of certain wavelengths (screens, generate their own light – the image is backlighed),
2. devices that produce "non-electronic" outputs printed on paper, film or other similar material (printers, etc.), and thus colours are perceived by the reflection of light from the surface and do not generate light themselves.

The first type of devices use an additive model of colour compositing, the second type of devices use a subtractive model.

✎ **Additive model** is intended for devices that actively display colors by emission (screen, projector). The basic colours are red, green and blue (*RGB*). Mixing all the basic colours at full intensity produces white.

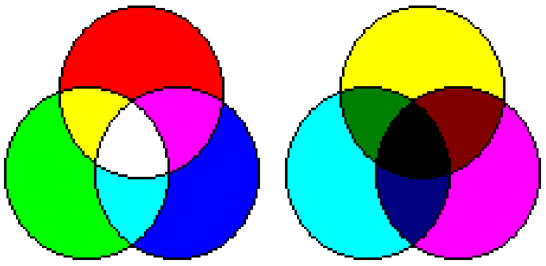


Figure 8.4: Aditive and subtractive model

✎ **Subtractive model** is intended for devices that display passively (not by emitting), such as printers. The basic colours are cyan (blue-green), magenta (red-violet) and yellow (*CMY*). Mixing all colours at full intensity produces black (grey-black).

✎ **CMYK** (Cyan-Magenta-Yellow-Key – black) is a typical example of a subtractive model commonly used in printers. It is a black enrichment of the CMY model described above. There are several reasons for this – one of them is financial (a separate color is cheaper than mixing it, when black is the most used color in printing) and also mixing CMY does not produce a completely pure black for technical reasons (we would get a "dirty" dark gray).

✎ **RGB** (Red-Green-Blue) is again a typical example of an additive model. It is used in several forms, especially in screens and projectors. In fact, there are other additive models than RGB, others are convertible to RGB and are usually worked with in larger and better graphic editors. Their purpose is to simplify the setting of certain color-related properties (such as image brightness). We will not deal with them, they are a matter of subjects dealing with graphics processing.

8.3.2 CIE

The *CIE system of colorimetry* was created to unify the RGB standards to correspond as close as possible to the CMYK system.

[4]From: http://en.wikipedia.org

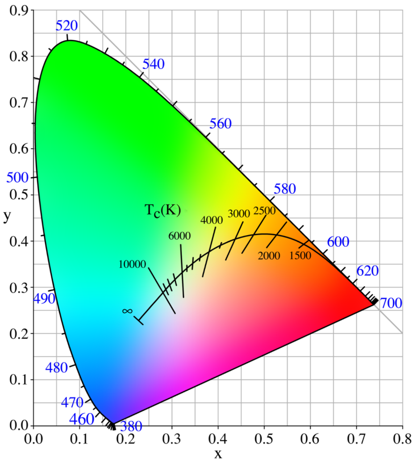


Figure 8.5: CIE chromacity diagram[4]

This was mainly due to the numerous problems with the differences between displaying an image (especially a photo) on a screen and then printing that image. It has the following characteristics:

- "horseshoe" capturing light of different wavelengths depending on red and green, we see it in Figure 8.5,
- within this horseshoe there are points for the base colours (as many points as base colours).

The convex n-gon formed by joining these points determines the colour space of the model, the set of all colors that can be displayed in the given color space. This area is called the *gamut*.

sRGB colour space model (standard RGB) is an older implementation of the CIE standardization system (1996) originally for CRT screens. This standard is found in many of today's screens, rather cheaper ones.

The red, green, blue and white coordinates were given values close to the CMY representation in the CIE system. The purpose was to make the appearance of the image on the screen closer to its printed equivalent.

Adobe RGB colour space model was created out of the need to increase the colour space, and also because some colours didn't look very natural in the sRGB colour space (especially in the green area, for example grass).

In the CIE standardization system, the triangle for Adobe RGB has more content than sRGB, especially towards the green. Nature photos have more natural colours when displayed on an Adobe RGB screen.

Adobe RGB is supported by some screen (mostly more expensive), but there are intermediate levels (for example, slightly cheaper screens that can "nearly" Adobe RGB, such as 95 %).

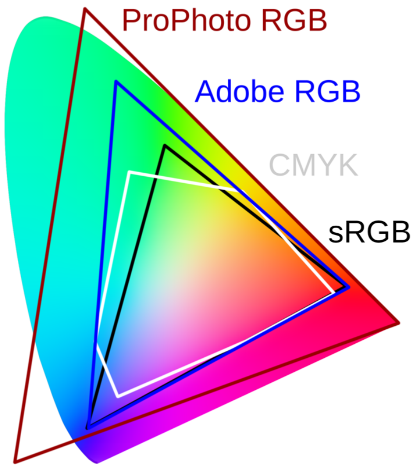


Figure 8.6: Various types of colour space models[5]

Remark

We usually use Adobe RGB when we have a good quality photo printer (or professional printer), because when printing via a photo service, it is (usually) necessary to convert to sRGB anyway.

There's no point in using an Adobe RGB-capable screen if we're creating electronic output (such as web graphics that we don't intend to print), or if we're designing an application interface – in both cases, our customers will be people with "ordinary" screens, so what looks absolutely amazing on a high-quality screen may look much less aesthetically pleasing on a regular sreen.

ProPhoto RGB developed by Kodak has very large gamut. It is intended for manipulations with photos, not for the final output, because this gamut is not widely supported. The aim is to reduce data loss during some image editing.

[5]From: https://www.milujemefotografii.cz/srgb-prophoto-rgb-a-dalsi-vyznate-se-v-barevnych-prostorech

8.3.3 Graphics Terms

✎ *Pixel* is an abbreviation of picture element, it is a single viewable point. Apart from its coordinates, it has only one property – colour.

A raster image is usually thought of as a grid of pixels. Thus, for this type of image (e.g. BMP, JPG, PNG, GIF, . . .), it is important what color values are assigned to each pixel, even if they are not stored as a list of color values (virtually always some form of compression is used to keep the image from taking up too much space).

Vector images, on the other hand, are not defined as a grid of pixels, but as a sequence of operations (e.g. operations to draw a line segment, curve, etc.); the conversion to a grid of pixels is done only when the image is output to a display device. One commonly used (and widely available) vector format is SVG.

✎ *Color Depth* is the amount of memory needed to represent one pixel (in bits), usually separately for each base color (8+8+8, 16+16+16, etc.). For display devices, part of the color depth may be reserved for the alpha channel (transparency level).

✎ *Resolution* is the number of pixels (i.e. colored dots) per unit length, usually an inch. It is usually given as the number of rows and columns of a pixel matrix, or in *dpi* (Dots per Inch). When comparing the quality of output to a screen and a printer, we can see dpi for the printer and ppi (pixels per inch) for the image on the screen (or stored on disk).

8.4 Screen

A screen is an electronic output device used to display text and graphical information. It is usually connected to the graphics card and possibly to other devices (speakers), and may be integrated into a device.

Screens come in different diagonal sizes (in inches) – e.g. 14", 15", 19", 21". The aspect ratio is usually 4:3 (traditional) or 16:9 (widescreen), or 16:10 or other, e.g. 21:9.

Very wide monitors can be curved, as shown in the following image. The degree of curvature is derived from the expected distance between the user and the screen.

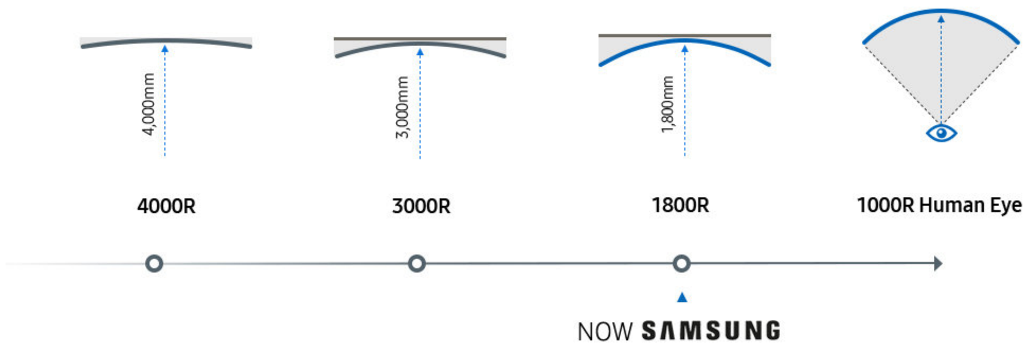


Figure 8.7: Curved screen, degree of curvature[6]

[6]From: https://www.samsung.com/sg/whycurved/

8.4.1 Parameters

When we buy a monitor (i.e. a device that holds a screen), we are interested in its specifications. What should we be interested in (without specifying the type yet)?

The usual values of *resolution* as the number of pixels in width and height are as follows:

640×480 (VGA),	1280×800 (WXGA),	1366×768 (HD Ready),
800×600 (SVGA),	1280×1024 (SXGA),	1920×1080 (Full HD, 1080p),
1024×768 (XGA),	1600×1200 (UXGA),	4096×2160 (4K),...

Word terms are often used:

- HD: 1280×720, also 720p
- Full HD (FHD): 1920×1080, also 1080p
- Quad HD (QHD): 2560×1440, $4\times$ HD displays
- Ultra HD (UHD): 3840×2160
- 4K: 4096×2160 (more precisely, there are two variants: DCI 4K is 4096×2160, 4K UHD is 3840×2160)
- 8K, 16K, etc.

Remark

It depends on the specific interface and its version how much data is transferred per time unit. The higher the resolution and the higher the refresh rate, the more bandwidth we need. Can you think of which interfaces and which version we would need to transmit a 4K image at 60 Hz?

Response time is the time it takes for a pixel to change color from black to white and back. It is the sum of two time intervals: rise+fall: the "rise" parameter is the time it takes for the pixel to light up to white, "fall" is the time it takes for it to turn off to black.

But more telling is the "grey to grey" parameter – the change from dark grey to light grey and back. For example, we can see "Response time 5 ms (gray-to-gray)", which means that the color change from dark gray to light and back takes 5 ms. For fast gaming, choose a monitor with a response time of less than 5 ms – the lower the number, the better.

Remark

In any case, the values given by the manufacturer must be taken with a grain of salt because they are measured under ideal laboratory conditions, which are rarely achieved in a normal environment with varying lighting conditions and when displaying normal screen content. In addition, there may be a problem with the image transmission – crosstalk in the cable, a problem on the graphics card side, etc.

Refresh rate is the rate at which the displayed content is refreshed periodically, i.e. how often the monitor updates the screen content. It is also referred to as FPS (Frames per Second). It can be 60 Hz (i.e. 60 FPS), gaming computers have a refresh rate of around 120 Hz or even higher.

Input lag (in general) is the time delay between the source of the input (here the graphics card) and the destination (here the screen). Input lag is not the same as response time. While the response time is purely a property of the monitor, the Input lag also depends on the interface (monitor and graphics card) and the quality of the cable, i.e. the entire data path.

The values between 40 and 60 ms are average, suitable for regular purposes. For playing games, lower values are more suitable.

Other parameters that may be of interest:

- power consumption in working/stand-by mode,
- pixel pitch,
- viewing angles (viewing the screen at an angle will cause deformation of the colour displaying),
- dimensions,
- technology used – CRT, LCD (TN, IPS, VA, etc.), OLED, FED, SED, etc.

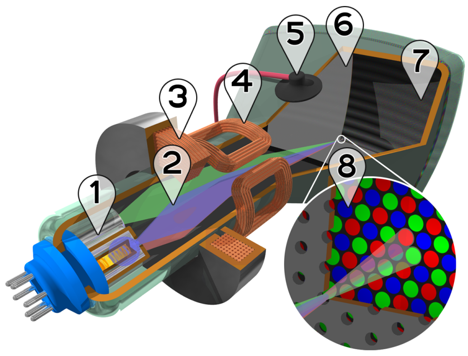
The connectors on the monitors (VGA, DVI, HDMI, Display Port, USB) are in Chapter 3, we won't discuss them here.

8.4.2 CRT Monitors

CRT (Cathode Ray Tube) monitors used to be the only type of monitor for personal computers. Their basis (also appearing in the name) is the *cathode ray tube*.

Usually it is a glass tube with a vacuum or very sparse gas inside. The tube contains electrodes (anode and cathode) between which an electric current passes when connected to an electric voltage. The cathode forms an "electron gun" which emits electrons, which are accelerated towards the anode in the tube and directed towards the screen by deflection coils. The shield in the fluorescent layer contains phosphors which light up when the electrons fall. The intensity of the electron stream indicates the intensity of the illumination of the pixel; for a black pixel the intensity is zero (no electrons).

Colour monitors (RGB) have three electron guns, one for each basic colour, in front of the screen (inside) there is a filter mask for each basic colour.



1. electron guns
2. beams of electrons
3. focusing coils
4. deflection coils for individual colours
5. anode
6. mask separating colours
7. phosphor layer with R, G, B zones
8. detail of the inside of the screen

Figure 8.8: Principle of CRT monitor[7]

Advantages and disadvantages of CRT monitors: The big disadvantage is the size of CRT monitors – they take up a lot of space on a desk. Another disadvantage over other types is the slightly higher power consumption.

On the other hand, the advantages are better contrast, color reproduction, viewing angles, and response time, especially when compared to the currently most common types of LCD panels. The lifetime of CRT monitors is also reported to be slightly higher than that of LCDs.

[7]From: http://www.absoluteastronomy.com/topics/Cathode_ray_tube

8.4.3 LCD Panels

LCD (Liquid Crystal Display) is named after liquid crystals. A liquid crystal is a material that changes its molecular structure when an electric current is passed through it. LCDs are usually based on passing or blocking light (from an external source or light from a back or side backlight).

✎ **Backlight.** There are two basic technologies used for *backlighting*:

1. CCFL (Cold Cathode Fluorescent Lamp) – cold cathode fluorescent lamps, either several horizontally on top of each other, or the lamp is placed on the side of the panel and its light is passed across the panel by a light guide and a reflective layer (often multiple reflective layers).

   It is almost impossible to display black on a panel using CCFL, there is always some light reflected on the pixel. This technology is on the decline.
2. LED – the light is generated by LEDs. There are several types, differing mainly in the placement of the LEDs. Today it is common to have *Edge LED*, where the LEDs are placed on the side of the panel and their light is distributed throughout the panel by light pipes and reflective elements. There is also *RGB LED*, where the LEDs are in a matrix across the entire panel, each pixel having four LEDs (red, blue and two green), and *Direct LED*, which is similar, but the LEDs are just white.

Edge LED can't display pure black, but it has another advantage – panels backlit in this way can be very thin, which is especially advantageous for notebook panels. Edge LED has another advantage: lower cost.

Originally, the LEDs in Edge LEDs were on two opposite sides of the screen, but today we tend to see them placed on only one side (shorter) and the light is distributed throughout the panel by a light pipe, thus lowering the price at the expense of backlight homogeneity.

We can also encounter the term WLED (White LED), which is practically the same thing (white LEDs are used as backlight, the colour is created differently, for example by a colour filter at the subpixel).

✎ For all technologies, it is important how well the light is distributed across the panel (this parameter is called *homogeneity of backlighting*).

☞ **Remark**

The uniformity of the backlight is tested using an optical probe (any device that reads brightness intensity with coordinate recording can be used). The results are often found in comparative tests in the form of a 3D graph or 2D image where the intensity of the backlight is expressed in different colours.

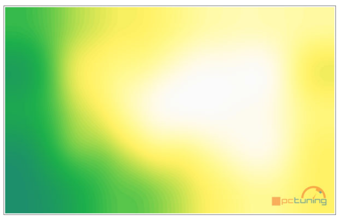


Figure 8.9: Illustration of backlight homogeneity[8]

In the picture on the right we can see the result for the screen of the IPS panel Dell U2311H. The white color indicates the places with maximum backlight (100 %).

In this particular case, the backlighting is not quite homogeneous, but even in the "worst" pixels it is not bad – in this case, that is because the green color for this panel indicates 80 %. For a given location, the variance from the border values is displayed, the border values are not important. ☜

[8]From: http://pctuning.tyden.cz/

Active and passive LCD. There are two basic types of LCD displays – active and passive. *TFT (Thin-Film Transistors)* are "active" LCDs, the most common today. Underneath each pixel there are transistors, some technologies have two transistors for each primary colour, i.e. 6 per pixel. Active backlighting is required, usually with LEDs.

Passive LCDs work a little differently. Each row has one transistor, each column has one transistor. This type of LCD has slower response, fewer colors, interference between wires, but it is cheaper and uses less power.

Next, we will focus on mostly TFT (active) LCDs.

TFTs are divided into subgroups according to the technology used – the functionality of the liquid crystal layer. The basic types are TN, IPS, VA, the second and third types are again found in several variations. The differences are mainly in the principle of liquid crystal treatment, followed by color display quality, viewing angles, response time, etc.

Basic principle. A liquid crystal is a material that changes its molecular structure when an electric current is passed through it. LCDs work on the principle of *light transmission and blocking* (from an external source or light from a back or side backlight), blocking or transmitting light is what liquid crystals do by changing their molecular structure into a light permeable or light impermeable form (or something in between).

We will explain the TNT operation on TN panels, the difference will be shown in explanation for other types.

Liquid crystal molecules align themselves with microscopic grooves on electrodes into a spiral or other regular structure (thus forming a crystal) when an electric current passes through them.

The colour of a pixel (or a subpixel for the base colour) is affected by the following sequence:

light source ⇒ vertical polarizing filter ⇒ transparent electrode ⇒ liquid crystals ⇒ transparent electrode ⇒ horizontal polarizing filter ⇒

Twisting of the crystals is controlled by transistors below the pixels.

Remark

A bit of physics: Light is simply a stream of photons oscillating in a certain plane (note, the world is 3D, this plane can be tilted in various angles) in a certain direction. We can think of a vertical polarizing filter as a barrier with vertical slits. Only those photons that oscillate vertically (i.e. the plane of their motion is vertical) can pass through (they fit to some of slits), the others cannot. A vertical polarizing filter is shown in Figure 8.10.

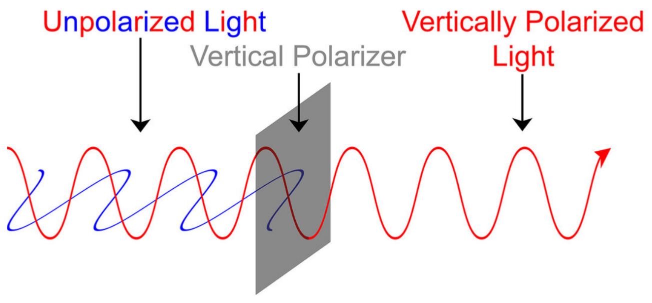


Figure 8.10: Vertical polarization procedure[9]

A horizontal polarizing filter has horizontal slits, so only photons that oscillate in a horizontal plane will pass through it. So, if a photon passes through the vertical filter and reaches the horizontal filter without further change in polarization (plane of oscillation), it will not pass. If the photon behind the vertical filter is reflected by the liquid crystal so that its plane of oscillation changes to horizontal, it will pass.

LCD panels are digital, so it is better to connect them via digital interface (DVI, HDMI, DisplayPort). If they are connected via an analogue interface (VGA), unnecessary conversions occur: Digital→Analog→Digital, which can affect the quality of the output (however, in office and often at home it doesn't matter).

LCD panel maintenance: Definitely do not use window cleaners or paper towels! Follow these steps:

- turn off the panel and let it *cool completely* (if we didn't do this, we would get "decorative" smudges),
- clean either with a special wipe designed for LCD panels, or use a good quality soft microfibre cloth (good quality, be careful with coarse thread, it could scratch the surface), which we either wet in mild water and wipe thoroughly, or use a spray designed for this purpose,
- let it dry, then use it.

If the surface of the panel is not glass (partially flexible plastic can be used), we should not push too hard, more pressure can damage the crystals.

8.4.4 Types of TFT Panels

TN (Twisted Nematic), B-TN is the oldest massively deployed TFT technology. It is characterised by poorer viewing angles, due to the spiral arrangement of liquid crystal molecules.

According to the above sequence affecting the pixel color, the following applies to TN:

- No current passes through: the crystals are in a "chaotic state", the light from the first filter is diffused through them in all directions and reflected everywhere, including the original direction in various polarizations – some of it passes through the second filter. So the pixel lights up.
- If the (maximum) voltage is: the liquid crystals are twisted in such a way that the photons pass by them with unchanging polarization, thus they do not pass through the second filter (because the filters are perpendicular to each other; the frequencies that passed through the first one do not pass through the second one). The pixel does not light up.

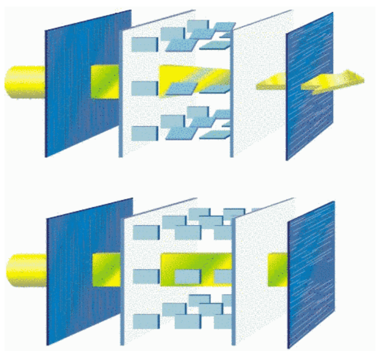


Figure 8.11: Principle of TN LCD, higher voltage means lower light intensity[10]

This implies that more light will pass through when the voltage is reduced. There is a *indirect proportion* between electrical voltage and brightness, so as the voltage increases, the brightness of the pixel decreases (no voltage = white). Faulty pixels light up white, which is quite annoying.

[9]From: https://www.diyphotography.net/what-is-cross-polarization-and-how-you-can-use-it-in-your-photography/
[10]From: http://pctuning.tyden.cz

TN typically has poorer color rendering – Adobe RGB gamut is not even close (this is related to the smaller color depth, usually only 6 bits per color/channel, the color is dithering for greater depth). On the other hand, these panels usually have better response than other types of TFT panels.

Remark

Summary: the advantage of TN panels is low price and good response time (handles fast image changes without ghosting), the disadvantage is everything else – worse color reproduction, viewing angles are small (especially vertical – colors darken quite significantly when viewed from below), defective pixel lights up. These panels are more common in very cheap laptops.

IPS (In-Plane Switching) developed by Hitachi is also called Super-TFT. Compared to TN, the molecules are not arranged in a spiral, but longitudinally when displaying white – resulting in better viewing angles (around 170 degrees), but on the other hand, contrast is not better (even vice versa) and response time is generally low with (original) IPS. Newer IPS panels already have quite decent response times, but it depends on the manufacturer and the specific model. The disadvantage may be a slightly higher power consumption than TN.

Colour rendering is much better than TN. Colours are more lively and accurate (except for very dark tones), but unfortunately the contrast is not as good. The defective pixel is black, it doesn't interfere in the picture.

The manufacturing technology is more demanding, which is why (among other things) IPS panels tend to be more expensive. The typical use was originally in DTP, but IPS panels can now also be found in regular monitors and laptops.

PLS (Plain-to-Line Switching) is a variant of IPS from Samsung, which in its characteristics basically coincides with the above mentioned about IPS panels. Unlike other IPS, it has slightly better viewing angles, color rendering in bright tones, and Samsung has better prepared this technology for flexible or curved panels. PLS may also be interesting for touchscreen displays (the colour of the pixels does not change when touched, so it does not leave a trace when swiped with a finger or pen). Also, the production is reportedly more financially optimal.

There are other variants of IPS from different manufacturers (S-IPS, H-IPS, E-IPS, AHVA and others), but most manufacturers are not investing so much in development lately, it is not worth it.

Remark

Summary for IPS and descendants: excellent color rendering, the defective pixel is black (does not interfere with the image), viewing angles are acceptable, response time is slightly worse than TN (on paper), but newer variants quite improve on this parameter. The disadvantage is the higher price, for professional panels very significantly.

VA (Vertical Align) is the work of Fujitsu. Unlike TN and IPS, the molecules (crystals) are arranged in “tree”, which slightly “expand” as the current passes through. The original VA technology was characterized by poor viewing angles (better than TN, worse than IPS), which has been much improved with the new variants – MVA and PVA. It is typical for VA technologies to have the same vertical and horizontal viewing angle.

The advantage over TN is higher contrast, the defective pixel is dark, colors are slightly better than

TN (depending on the variant, some VA variants are even comparable to IPS). In general, in many parameters VA is just somewhere between TN and IPS (viewing angles, color rendering, response). However, they are better in contrast and worse in response (but for both parameters it also depends on the specific devices).

In particular, *MVA* (Multidomain Vertical Alignment) technology has expanded viewing angles to almost IPS-like values. There are different variants: for example, P-MVA improves colour rendering, similarly S-MVA, A-MVA and others. In some cases, it's basically the same thing, just from a different manufacturer.

PVA (Patterned Vertical Alignment) by Samsung and Sony have similar characteristics to MVA, but generally have higher contrast and black rendering. The *S-PVA* (Super-PVA) variant is currently considered the cutting edge of LCD technology, often having a colour depth of 10 bits per channel.

Remark

Summary: compared to TN, VA panels have an order of magnitude better color rendering (the best ones achieve the color rendering qualities of IPS panels), the defective pixel is black, contrast is generally better than both TN and IPS, response times are approximately equivalent to IPS, viewing angles are the same in different directions and almost reach the parameters of IPS.

VA panels generally have a better price/quality ratio with very good color rendering and contrast, and if we don't need that good response, they are an excellent choice.

QLED (Quantum LED) by Samsung is in fact an improvement of TN technology by quantum dots, a technology designed mainly for TVs and is intended to compete with OLED displays (see below).

Samsung has managed to improve some of the TN properties in QLEDs: for example, the colour space covering is excellent, on average better than OLEDs, and they also achieve high brightness (i.e. in bright environments the image on the display is quite visible). On the other hand, some of the disadvantages of TN remain, and in these parameters QLED is worse than OLED, IPS etc., for example, black display is a problem (the screen just lights up a bit all the time) and contrast is not the best either. Static image "burn in" is also a problem.

Additional information

http://www.tftcentral.co.uk/articles/panel_technologies.htm

8.4.5 OLED Technology

OLED (Organic Light-Emitting Diode, organic LED) use organic electroluminescent diode technology. Organic because this technology is based on organic material, carbon. The screen is made up of diodes that emit light when an electric current is applied, hence no backlight or filter is required. The pixel is made up of blue, red and green subpixel with their own LEDs.

Between the anode and the cathode is an emissive organic layer (luminophore), which, when electrons pass through it, lets photons go through. All this is either enclosed between two glasses, or a glass and another material, or deposited on a foil to create a very thin and flexible display.

This design has advantages such as wide viewing angles, the ability to achieve high brightness, fast response time, good contrast, and can display true black (this is quite a problem when using backlighting, which is not a problem here), in general, color reproduction is excellent, OLED panel can be thinner than conventional LCD.

Since individual pixels have their own diodes, the power consumption depends on how these diodes are currently lit. OLEDs are quite efficient when displaying an "average image", but if a very bright image is emitted for a long time, the consumption is quite high.

The disadvantage is a lower lifetime especially of blue subpixels, which can be solved for example by making the blue subpixel larger (i.e. its wearing out will not be felt so soon). The disadvantage is also more expensive production than the above.

It was mentioned above that we will be looking more at active displays, but in the case of OLED, we are actually at the boundary, the technology is spread across both – there are two basic types of OLED: passive and active.

PMOLED (Passive Matrix OLED) is a passive variant, so we have a grid of wires (rows and columns). The pixel is always at the intersection of the row and column, the anode and cathode are connected to the conductors at the pixel location, and there is organic material between the conductors. The conductors determine whether the anode should emit electrons and therefore the organic material should generate photons. The greater the current, the more photons and therefore the pixel lights up more.

The disadvantage is that in order to activate just a single pixel it is necessary to pass a relatively large current to the entire row and column (even so much that in another type of display it would mean very high brightness) and the display frequency cannot be very high (the activation time of the wires depends on the number of rows or columns).

AMOLED (Active Matrix OLED) is an active variant, where each pixel has its own transistor (at least one, more likely more cooperating) and its luminance is controlled by this transistor independently of other pixels on the row/column.

Unlike passive variants, AMOLED has a shorter response time and consumes less power, it also generally has better contrast (black should be truly black, not shining). But on the other hand, AMOLED displays are harder to read in sunlight and more expensive to manufacture.

There are other variations depending on how the energy is converted to light, the carrier material used, etc. For example, PHOLED (phosphorescent) with much higher efficiency at lower power consumption, flexible FOLED for various shaped surfaces (including helmet visors), double-sided TOLED (transparent), and others.

Chapter 9

Power and Cooling

Quick preview: This chapter covers power and cooling – how the power supply works, what are its usual parameters, how notebook accumulators ("batteries") work, what is important in cooling computer components.

Keywords: Power supply, ATX, TFX, SFX, power supply certification, efficiency, PFC, battery, Ni-MH, Li-Ion, Li-Pol, active and passive cooling, active-passive heat sink, heatpipe, thermal paste, ACPI.

Objectives: After studying this chapter, you will learn how the components in a computer or other computing device are powered, how the battery in mobile devices works and how to operate it, and how to cool the components in a computer.

9.1 Power Supply

Every computer has a *power supply* that does not produce energy, but *transforms* from alternating current (AC, from the power socket) to direct current (DC, the computer needs this).

An important function of power supply is also to deal with occasional *energy fluctuations* (mainly overvoltage, which could damage components in the computer, but also undervoltage). If the power supply can handle an actual power failure, it is referred to as a UPS (uninterruptible power supply). Of course, even UPSs can't keep the power on forever, but a certain amount of time is guaranteed to provide power during an outage.

The sources are available in several different designs. In addition to the normal desktop power supply (ATX) there are lower variants (TFX), SFX power supplies are designed for even smaller computers, in laptops and some All-In-One computers we can find an external power supply (that's the "box" on the connected cable, the transformer, trafo).

9.1.1 How the Power Supply Works

From the power grid, the power supply draws AC voltage at 230 V, which it transforms into DC voltage, which continuously charges the *capacitor* (this component of the power supply stores energy for consumption by other components). However, the voltage across the capacitor is still very high (several hundred volts), but the integrated circuits that are usually powered in the computer require very low voltages (units of volts). Therefore, after the capacitor, there are components that change the high DC voltage to a low voltage according to the required values.

From the power supply, the energy goes to the *branches* to which the individual components are connected. Each branch is specific in the voltage it distributes. Usually there are 3.3V branches, 5V branches (+5V and -5V) and 12V branches (+12V and -12V). For each of the branches, the possible load it can handle is also an important indication. For example, the 3.3V branch will usually handle a load somewhere between 20 and 30 amperes.

In addition to the branches, the cabling or connectors are also important. The length of the cables is an important parameter for cabling (how far "reach"), the minimum required length from the specification is 26–28 cm, which is really not enough for a desktop (ideally 45–50 cm, more for a bigtower). The connectors are usually ATX or PCIe type (so called cubes), or SATA power for SATA drives or molexes for older PATA drives. Cables consist of multiple conductors, with one conductor going to each pin in the connector. The conductors are colour coded according to the branches.

The main power cable for the motherboard usually has a 24-pin connector. We also need plenty of 4pin, 6pin or 8pin (or 4+4) connectors (CPU, PCIe cards, etc.) and more. We also need to power hard drives, optical drives, and anything else in the computer case. Many dedicated graphics cards require additional power. The chipset is usually powered from the 3.3V branch, the 5V branch is mainly for PCI cards and older processors (newer ones use the 12V branch).

9.1.2 Standards

✎ Nowadays, we mainly see *ATX* specification power supplies (previously, AT specification power supplies were used) – as we can see, the standardization of power supplies is closely linked to the form factories of motherboards (and also functionally related to them).

The ATX standard for power supplies exists in multiple versions. The most common versions we see are ATX v2.x (instead of "x" the subversion number) and ATX 12V x.y (again instead of "x" and "y" specific numbers). Various versions differ in minimum required values for possible branch loads, minimum required efficiency, connectors provided, requirements for protection against overvoltage short circuit, overheating overload, etc. ATX 12V 2.x uses "cube" connectors: 24pin main for motherboard power, 4pin for additional motherboard power, additional PCIe connectors for graphics cards and of course SATA, Molex and others.

We can also meet power supplies according to other specifications, for example there are TFX and SFX power supplies for very low cases (e.g. Mini-ITX).

In 2020, a new standard ATX12VO ("O" as "only") appeared, whose main connector is 10pin with smaller "cubes". Only the 12V branch leads from this power supply to the motherboard, everything else, including transformations and connectors, is handled by the motherboard.

✎ There are also *power supply certifications 80 PLUS*, provided by the ClearResult Company. We encounter several certification types differing mainly in the required power supply efficiency at a specific load, and depending the voltage level. The main characteristics of certified power supplies are outlined in Table 9.1 (only for 230V non-redundant power supplies), the latest is also the 80 Plus Titanium certification designed for so-called redundant power supplies[1] with an efficiency even higher than 80 Plus Platinum.

[1]Redundant power supply is an additional power supply used to supply power from a second power supply independent of the first, or to supply power from a central backup battery, it is not the same as a UPS.

% of rated load	**80 Plus**	**80 Plus Bronze**	**80 Plus Silver**	**80 Plus Gold**	**80 Plus Platinum**	**80 Plus Titanium**
	Efficiency at given load					
Load 10 %						90 %
Load 20 %	82 %	85 %	87 %	90 %	92 %	95 %
Load 50 %	85 %	88 %	90 %	92 %	94 %	96 %
Load 100 %	82 %	85 %	87 %	89 %	90 %	91 %

Table 9.1: Comparison of the efficiency of certified power supplies at various loads, 230V EU Internal Non-Redundant[2]

✎ If the power supply has a high *efficiency*, it not only means that it is sufficient for the job and does not consume too much energy, but also that it gets less hot and therefore the fan on it does not have to rotate as fast (which also has an effect on noise). Thus, efficiency can be thought of as an inverse and normalized (to a percentage) number to the amount of energy loss on the individual components of the power supply (the greater the energy loss that turns into heat, the lower the efficiency of the power supply).

☎ How can one find out if and which certification the selected power supply has? Apart from the fact that manufacturers are quite "bragging" about this information, we can also find out on the website http://80plus.org/, we will be redirected to the web https://www.clearesult.com/80plus/.

PFC (Power Factor Correction) is the part of the power supply whose job it is to smooth the current waveform into a sinusoid. The PFC is not so much important for the computer components, but on the contrary it is important for the surrounding network because it works with AC current. Because there are spikes in the current draw (typically when the computer turns on, the draw suddenly jumps up a lot), the resulting sinusoidal voltage distortion in the grid can cause interference in the grid.

PFCs are therefore important mainly where several devices are switched on (or increase their consumption) at the same time. If, for example, multiple computers without PFCs are switched on at the same time (give or take), the circuit breakers may be blown.

Currently, all power supplies sold are equipped with a PFC module because, although the standards (EMC standard on electromagnetic compatibility) do not require PFC, the requirements to meet certain parameters and thresholds are difficult to meet without PFC.

In the power supplies we can find either passive or active PFC (this is always indicated on the power supply). Passive PFCs are more likely to be found in cheaper power supplies, whereas active PFCs are typical of almost all quality power supplies. Both types do practically the same thing, but while a passive PFC is made up of only passive components without self-draw (usually an inductor at the input of the power supply), an active PFC also contains active components with self-draw (a transistor combined with an inductor, etc.). Active PFC is more efficient than passive PFC.

9.1.3 Properties

✎ **Power supply output power.** As we know, an important parameter is the efficiency at a given load (efficiency is often indicated by the above mentioned certifications). Another important

[2]Created by the source https://www.clearesult.com/80plus/program-details

parameter, which we decide on according to the way and intensity of using the computer, is the power. As we have seen above, the efficiency of a power supply depends on the current load, and the load in turn depends on the output power of the power supply to some extent.

The output power of a power supply is given in Watts and it is one of the most important parameters to be concerned with. Even less than 300W power supply is enough for easy use (a lot depends on the components used and their power consumption). However, if we have a more powerful processor with a higher TDP value, then we choose a power supply with a power of around 400–500 W, and if we have a high-end (e.g. gaming) system with a powerful dedicated graphics card or even multiple graphics, then we need to use an even more powerful power supply (even over 1000 W). It also depends on the number of hard drives connected (for example, RAID logically consumes more power than a single drive), optical drives and more.

Remark

We can determine that the computer is close to the performance limits of our resource by various indicators, such as freezing of the computer in some situations (e.g. when playing graphically and computationally demanding games), occasional spontaneous restarting of disks, etc.

It might seem that it is better to get a power supply with a very high performance, so that we have the maximum flexibility for further expansion or improvement of the system. However, we should know that unnecessarily high power usually means lower efficiency of the power supply (the power supply is only lightly loaded, and with a load that is less than 50 % for most of the time, the efficiency is usually low), which means wasting energy. 20% energy loss is a smaller real value in the case of a 200W power supply than in the case of an 800W power supply.

Noise and cooling. Everyone has noticed that there is often an active heat sink on the power supply. Some power supplies designed for silent HTPC machines have passive cooling. Since the power supply (especially at higher power consumption or with lower efficiency) puts out a lot of heat, cooling is very important, just because it also affects the overall climate (temperature) inside the case.

Cooling is discussed in the following subsection, so we will only mention here that the bigger the fan (the diameter of a large fan is about 12 or 14 cm), the better, because a fan with large blades can rotate slower and therefore quieter with the same cooling efficiency.

9.1.4 Laptop Accumulator

An accumulator (popularly known as the battery) of a laptop or other small electrical device is used to store energy that has already passed through the power supply (i.e. DC voltage, and rather lower values). There are usually several cells in a single accumulator (e.g. it may be a 6-cell accumulator).

Let's look at some accumulator-related terms.

Cell (galvanic cell) binds electrical energy, which can be easily released chemically in the form of an electric field. There are two basic cell types – primary (cannot be recharged, can only release energy) and secondary (can be repeatedly charged and discharged).

Battery consists of several interconnected galvanic cells. If the battery is made up of secondary cells (i.e. ones that can be charged), we call it an *accumulator*.

Accumulator Memory Effect (battery memory effect) is a phenomenon encountered in some types of accumulators (see below) when they are charged although they have not been fully discharged. In these types of accumulators, the capacity decreases due to repeated charging without full discharge. But beware – modern accumulators react differently!

Accumulator formatting is particularly important for those types that are prone to memory effect, and may even lead to a slight correction of the capacity reduction caused by memory effect. It is done by letting the accumulator discharge completely several times (three times) and then charging it to full capacity. In contrast, for accumulator types that are not susceptible to the memory effect, formatting can reduce capacity as the cells wear out through repeated charging and discharging.

✎ **Types of accumulators.** There are several types of accumulators for the most common mobile devices (laptops, netbooks, smartphones, etc.):

- *nickel-cadmium* (Ni-Cd) – rather historical, the problem was mainly cadmium toxicity and lower capacity,
- *nickel-metal hydride* (Ni-MH) – also not much used in mobile devices anymore, but we can buy them in the form of an AA battery and other "rechargeable batteries" for consumer electronics, the advantage is that it can be stored in a discharged state,
- *lithium-ion* (Li-Ion) – in laptops and other portable devices we see them most often today,
- *lithium-polymer* (Li-Pol) – in some smartphones and other small devices.

Ni-MH (and also old Ni-CD) accumulators suffer from the memory effect described above. Therefore, it was recommended to charge devices with this type of accumulator only after a complete discharge and to perform a so-called formatting from time to time. Nowadays, we encounter them in the form of AA or AAA batteries, whose chargers usually include a discharge function before recharging (in fact, we usually buy these batteries discharged).

Today's computers use Li-Ion and Li-Pol accumulators, which do not suffer from the memory effect, so they do not discharge before recharging, and even a complete discharge is harmful! If, for example, a laptop Li-Ion accumulator is repeatedly discharged fully hard, its lifetime will be reduced or even destroyed. There is some advice on the web in discussion forums about regulating the capacity of a accumulator by fully discharging it, but this is completely beside the point for Li-Ion and Li-Pol accumulators (it would apply to Ni-MH). By the way, this is the reason why modern operating systems in Li-Ion devices go into sleep mode or shut down completely when a low accumulator is detected (a few percent).

If we decide to format the accumulator "once in its lifetime", it doesn't help much, but it doesn't hurt that much either. If we format it regularly, even once a month, we are unnecessarily reducing its capacity.

✎ **Maintenance and use of accumulator.** Every accumulator gradually loses its capacity, this is quite normal. A particular type of accumulator has its own maximum number of charge cycles (i.e. lifetime), but even before this value is reached, the capacity gradually decreases. This decline can be slowed down by proper handling (the following applies to Li-Ion and Li-Pol accumulators):

- Never let the accumulator discharge completely. Operating systems are usually set up to shut down or put the computer to sleep in time before the accumulator runs out, do not change this setting (or if the operating system is not set up this way, make it so).
- If we often work connected to the power supply, we should set the operating system to the following settings. We do not let the accumulator overcharge unnecessarily and prefer to remove it (if possible). By overcharging unnecessarily, the accumulator is not damaged, but its lifetime is

reduced (we are approaching the maximum number of charging cycles, even if it is not necessary). This manifests itself in a gradual reduction of capacity.

- As with other components, accumulators work better in a certain temperature range than in another. Accumulators work best at room temperature (around 20 ℃), at higher temperatures their efficiency and lifetime decrease.

Better laptops are equipped with electronics (charge management circuits) that monitor accumulator status and react appropriately, for example keeping the accumulator at around 50 or 75% state of charge.

If a laptop with a lithium accumulator is powered alternately from the mains or from the accumulator, then the optimal handling is to maintain the state of charge in the range of 20-80 %, when the lifetime (capacity) hardly decreases.

If the accumulator is not used for a long time (the device is being plugged into the mains), it must be stored properly. Li-Ion accumulators should be stored in a cool place (but not in frost), in temperatures below room temperature. Excessive humidity is harmful, so if you decide to put the accumulator in the fridge (not the freezer!), wrap it tightly. Before storing, the accumulator should be charged to a capacity of about 75 % and approximately once every six months (depending on the total capacity of the accumulator) unpacked and recharged again to 75 %. Accumulators gradually discharge themselves, so it is important to maintain them in this way for Li-Ion.

9.2 Cooling

9.2.1 The Principle of Cooling

Almost all computer components need a temperature close to room temperature to run, but at the same time many components "heat" – produce heat. Typical "heaters", even when using cooling, are mainly the dedicated graphics card (over 60 ℃), the processor (up to 90 ℃), the hard disk (about 35–50 ℃), the memory modules and the power supply (about 30–40 ℃), and of course the motherboard itself. Therefore, cooling the computer is a very important issue, the temperature in the case should not exceed 40-45 ℃).

If the cooling is insufficient, the lifetime of components is reduced, but we can also encounter instabilities, stalling and "falling" (usually spontaneous restarts of components or even the whole computer).

✎ We distinguish between active and passive cooling. *Active cooling* requires its own power and has certain active components (such as a fan), whereas passive cooling does not need to be powered separately. For more thermally demanding components (such as a processor), it is now common to see a combination of these methods, i.e. active-passive cooling.

We can also distinguish different types of cooling according to the way the heat is transferred. The most common is cooling by air or water (or another fluid with suitable properties).

✎ Basic arrangements:

- *placement of the computer* is important. Air should flow around the computer as much as possible, the heater should not be next to it and the sun should not shine on the computer. We don't use the computer case as a storage space for books, papers and other insulating things.

The computer case can be padded so that it is not directly on the floor (or even carpet), and there should be a solid flat base (such as wood) underneath.

- *Dust* is an excellent thermal insulator. Unfortunately, any electronic device directly attracts dust, so even in a well-cleaned room, dust will get in. That's why it's a good idea to remove dust occasionally (either with a can of compressed air – be careful where you blow the dust, or with a lower-speed vacuum cleaner). Before cleaning, we should turn off the computer and unplug it.
- Airflow inside the computer is often obstructed by *cables*. Fortunately, wide IDE (PATA) cables are no longer used, but even thinner cables can obstruct airflow or cause turbulence. Therefore, we should keep cables organized and ideally tied with insulated wires.

Remark

If we use a vacuum cleaner to remove dust from inside the computer case, we should secure all the fans against rotation beforehand (e.g. catch them with duct tape or block them for a while with a pencil). After dusting, we must remember to loosen all the fans again and check that all the connectors are where they should be and that they are fully plugged in.

It is important to know how the air actually flows in the case. In newer cases, air is usually drawn in through side or front vents (they must be free!) and case leaks and blown out by a fan behind the power supply. The airflow should not be obstructed by the individual components around which the air flows and draws heat, and of course not by cables (or at least as little as possible). Dedicated graphics cards with their own heatsink usually blow the warm air out through a secondary slot in the back of the case.

9.2.2 Types of Heat Sinks

Passive Heat Sinks are designed for less thermally demanding components. They are usually based on the use of the thermal conductivity of certain materials as well as convective airflow (warm air rises) or use another method of forced airflow (for example, in combination with an active cooler). A typical and very important characteristic of passive heat sinks is that they operate quietly (they have no active parts, they have nothing to make noise with), a slightly worse characteristic is their lower efficiency than other cooling methods.

A large cooling surface is important for passive heat sinks. Therefore, they are usually quite densely finned components. Usually, the denser the finning, the better the cooling, but only up to a point. Too dense finning can interfere with proper airflow and cause turbulence.

Passive heatsinks usually complement active heat sinks (see below), but can also be used on their own (e.g. on a chipset).

Active and active-passive heat sinks can be found e.g. at the power supply or often at the processor, sometimes also at other components. As mentioned above, they are usually found in combination with passive cooling.

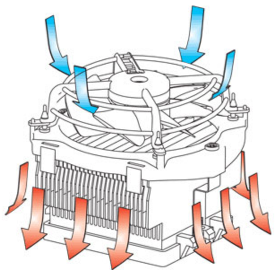


Figure 9.1: Active-passive heat sink[3]

[3]From: http://pctuning.tyden.cz/

Figure 9.1 shows the direction of airflow around the combined heat sink above the CPU (the fan draws in cold air and blows it onto the CPU, the heated air is blown away by the passive heat sink).

✎ **Heatpipe** is a hermetically sealed copper tube filled with a liquid that conducts heat well (water, ethanol, etc.). The inner surface of the heatpipe is distorted, it has a kind of spongy texture. This is passive cooling.

The boiling point of the liquid inside the heatpipe must be rather low (ideally around 50 ℃), which is achieved by a suitable choice of liquid (ethanol) or pressurisation (for water). At the heated end, the liquid evaporates – thus removing heat, then condenses at the cooler end and returns to the heated end through the inner "sponge" surface. As a result, no matter the specific position and orientation of the tubes, evaporation and rising work against the Earth's gravity.

Heatpipes are characterized by very high efficiency, even though they are passive cooling equipment in principle. We usually see them in combination with active cooling, an example is shown in the picture 9.2 (on the side there is a fan as an active heat sink, then there is a regular passive heat sink – finning, at the bottom there are heatpipes in the form of "U").

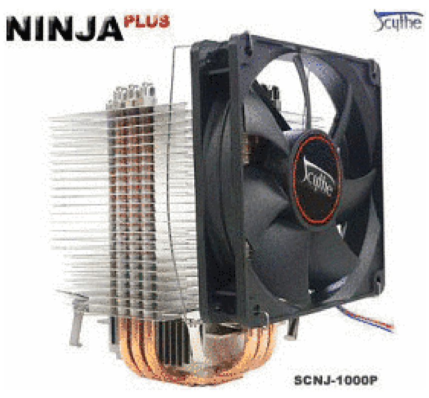


Figure 9.2: Combination of active heat sink and heatpipes[4]

✎ **Thermal paste** is applied to the area between the cooled component and the heat sink (most often in processors). Its purpose is to ideally bond the component and heat sink so that there is no air between them, thus improving heat dissipation. The paste should be applied only in a thin layer (usually a drop in the middle of the processor, or a few drops for larger processor chips) so that the interconnection is full. Too thick layer can degrade heat dissipation.

9.2.3 Temperature Monitoring

Modern computers commonly have temperature sensors on "risky" components that can be monitored by software. Monitoring can be done in various programs, even specialized for a particular component. In addition to monitoring, we can also often adjust the speed of the fans (their speed) – if a fan is too noisy and there is no danger of overheating the component, we can reduce the speed of the fan slightly.

✎ Examples of programs for Windows include:

- SpeedFan[5] – fan speed control, monitors temperature sensors and accesses S.M.A.R.T.,
- CPUID HWMonitor[6] – there's a whole suite of system monitoring tools by CPUID on the web,
- HDD Health, HDDLife – for the hard drive,
- Notebook Hardware Control[7] – mainly for laptops, including power management,
- SiSoft Sandra Lite.

Typical manufacturers of heat sinks are Arctic Cooling, Evercool, Nexus, Noctua, Thermalright, Xigmatek, Thermolab, Zalman and others.

[4]From: http://pctuning.tyden.cz/
[5]http://www.almico.com/speedfan.php
[6]http://www.cpuid.com/softwares.html
[7]http://www.pbus-167.com/

9.3 Power Management

Power Management presents options for managing the power consumption of computer components in certain situations. There are two ways in which power management can be performed: *APM* (older, completely BIOS-driven) and *ACPI* (newer, operating system driven).

APM (Advanced Power Management) is configured in the BIOS in a section usually called "Power Management Setup" or similar. It defines several modes that the processor can switch between (Enabled, Standby, Suspend to RAM, Suspend to Disk, Off).

ACPI (Advanced Configuration and Power Interface) also uses the BIOS (to work with the motherboard configuration) and uses the hardware registers (in the chipset), but the configuration is done in the operating system. With the laptop we can (and usually do) get additional software from the manufacturer of the laptop (or motherboard or other component), which can also be used for more detailed power management settings.

Similar to APM, ACPI recognizes several modes of operation.

- Working (S0) – full power,
- Sleeping S1 (S1/POS) – similar to Standby and Suspend to RAM from APM, RAM and CPU cache remain under power, other components run in save mode,
- Sleeping S2 – similar to Sleeping S1, but the CPU cache is not powered, the contents of the RAM are not affected,
- Sleeping S3 (Save to RAM) – similar to Sleeping S2, but all components except RAM are completely unpowered, waking up is slightly longer than in previous sleeping modes,
- Save to Disk (S4, Soft-Off) – all components are without power, the contents of the memory are saved to disk, the wake-up time is several times longer than in S3, the operating system (here actually turned off, except RAM), which can handle this state, can restore itself from the swap space on disk after starting its bootloader,
- Mechanical-Off (S5) – system powered off.

The communication between controlled components, BIOS and operating system is done in a special language *AML* (ACPI Machine Language), which is universal for all types of components from different manufacturers. If a device (component) supports ACPI, it must understand this language.

Bibliography

[AMD] *AMD: Product Resource Center* [online]. WWW: http://products.amd.com/ [2022-10-10].

[ARMCompiler] ARM Compiler v5.06 for mVision armasm User Guide. *Assembler User Guide* [online]. 2016 [2019-02-01]. WWW: http://www.keil.com/support/man/docs/armasm/default.htm

[ARM] ARM Instruction Reference [online]. *ARM Information center* [2019-01-30]. WWW: http://infocenter.arm.com/help/topic/com.arm.doc.dui0068b/CIHEDHIF.html

[FOG] FOG, Agner. Software optimization resources [online]. *Agner's pages* WWW: https://www.agner.org/optimize/ [2022-10-10].

[Hamacher2012] HAMACHER, Carl, Zvonko VRANESIC, Safwat ZAKY a Naraig MANJIKIAN. *Computer Organization and Embedded Systems*. 6th ed. New York, NY: McGraw-Hill, c2012. ISBN 978–0–07–338065–0.
WWW: http://www.staroceans.org/kernel-and-driver/Computeranization%20And%20Embedded%20Systems,%20Hamacher,%20Vranesic,%20Zaky,%20Manjikian,%206Ed,%20Mgh,%202012.pdf

[HWSecrets] *Hardware Secrets: Uncomplicating the Complicated* [online]. WWW: http://www.hardwaresecrets.com/ [2022-10-10].

[intel] *Intel: Architectures and Products* [online]. WWW: http://ark.intel.com/ [2022-10-10].

[Intel2016] Intel 64 and IA-32 Architectures Software Developer's Manual: Volume 3 [online]. *Intel.com* [2019-03-19]. WWW: https://www.intel.co.uk/content/www/uk/en/architecture-and-technology/64-ia-32-architectures-software-developer-system-programming-manual-325384.html

[WikiChip] *Intel Microarchitectures* [online] WWW: https://en.wikichip.org/wiki/intel/microarchitectures [2019-04-02].

[IXBTLabs] *IXBTLabs: Computer Hardware in Detail* [online]. WWW: http://ixbtlabs.com/ [2022-10-10].

[Jaitak] JAITAK, Er. Nagesh. *Computer Organization & Architecture*. Self publishing. ISBN 978-1-31-230609-7.
Most pages on: https://books.google.cz/books?id=0ujYBgAAQBAJ&printsec=frontcover

[Lalonde] LALONDE, Jean-Francois. *ARM Instruction Set* [online]. [2019-02-01] Université Laval, Canada. WWW: http://vision.gel.ulaval.ca/j̃flalonde/cours/1001/h17/docs/arm-instructionset.pdf

[IntelASM] LOMONT, Chris. Introduction to x64 Assembly [online]. *Intel Developer Zone* [2019-03-15].
WWW: https://software.intel.com/en-us/articles/introduction-to-x64-assembly

[Mano1993] MANO, M. Morris. *Computer system architecture.* 3rd ed. Englewood Cliffs, N.J.: Prentice Hall, c1993. ISBN 978-013-1755-635. WWW:
https://faculty.psau.edu.sa/filedownload/doc-10-pdf-d171a71acbe44cd5cd2f78a40570a069-original.pdf

[Stallings2010] STALLINGS, William. *Computer Organization and Architecture: Designing for Performance.* 8th ed. Upper Saddle River, NJ: Prentice Hall, c2010. ISBN 978-0-13-607373-4.
WWW: https://inspirit.net.in/books/academic/Computer%20Organisation%20and%20Architecture %208e%20by%20William%20Stallings.pdf

[Tanenbaum2006] TANENBAUM, Andrew S. *Structured Computer Organization.* 5th ed. Upper Saddle River, N.J.: Pearson Prentice Hall, c2006. ISBN 01-314-8521-0.
WWW: https://eleccompengineering.files.wordpress.com/2014/07/structured_computer_organization _5th_edition_801_pages_2007_.pdf

[LinuxKernel] The Linux Kernel Archives [online]. WWW: https://kernel.org/ [2022-10-10].

[ThingComp] *Think Computers: Computer Hardware Reviews* [online]. WWW:
http://www.thinkcomputers.org/ [2022-10-10].

[TomsWH] *Tom's Hardware: The Authority of Tech* [online]. WWW: http://www.tomshardware.com/ [2022-10-10].

[LinuxKernel] *Understanding the Linux Kernel, Third Edition.* Notes [online]. WWW:
https://notes.shichao.io/lkd/ [2022-10-10].