



Topic 2 - Computer organization

2.1 Computer organization

Computer architecture

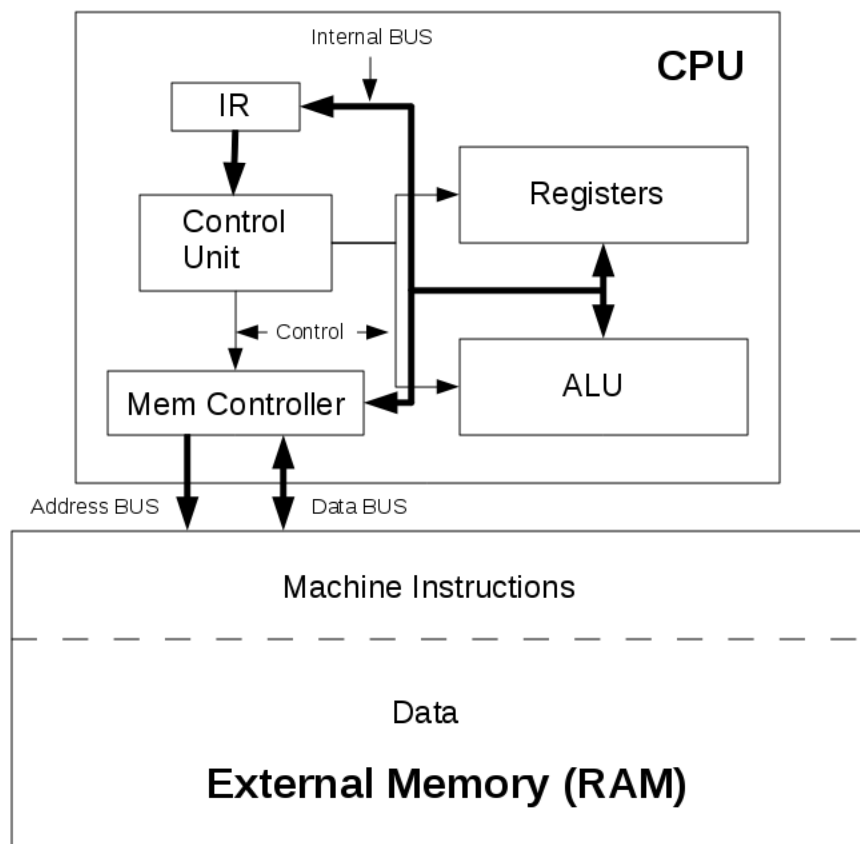
2.1.1 Outline the architecture of the CPU and the functions of the ALU and the CU and the registers within the CPU

CPU: the Central Processing Unit, is the brain of the computer. This highly compact and powerful chip controls all the peripherals and processes inside the computer and does the calculations on the data the system handles.

ALU: the Arithmetic Logic Unit, is a circuit that is part of the CPU and does all the arithmetic calculations.

CU: the Control Unit, is the control circuit of the CPU. It handles the loading of new commands into the CPU and the decoding of these commands. Also, it directs the data flow and the operation of the ALU.

Registers: they are small, very fast circuits that store intermediate values from calculations or instructions inside the CPU. They are often referred to as cache memory inside the CPU. There are two types of these memory we need to know: The **Memory Address Register (MAR)** and the **Memory Data Register (MDR)**. The MAR register holds the address of the next memory cell that data will need to be read from or written to, while the MDR holds data that will be written to the RAM or that was read from RAM. The MAR gives the address the data of the MDR will be read from or written to.



We are expected to be able to draw a diagram with

the CPU, ALU, CU, MAR and MDR!

2.1.2 Describe primary memory

Random Access Memory (RAM): memory unit that can be read to and written from at any location. Therefore it is used as primary memory in computers to hold data and instructions. It is volatile, so it loses its contents when power is turned off.

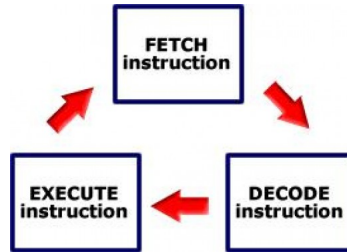
Read-Only Memory (ROM): memory unit that can only be read from. It is non-volatile, so it keeps its contents when power is turned off. Used to store the BIOS (today UEFI) of the computer, the small control program that directs turning on the computer.

2.1.3 Explain the use of cache memory

A type of small, high-speed memory inside the CPU used to hold frequently used data, so that the CPU needs to access the much slower RAM less frequently.

2.1.4 Explain the machine instruction cycle

The machine instructions cycle is also called the fetch-execute cycle. It consists of the steps: **fetching the next instruction**, **decoding the instruction**, **executing the instruction**. Often, a fourth step is added between the decoding and executing steps, called **getting data**, but because computers ultimately deal with data all the time, this step is often left out. This process is repeated all over again until the computer is turned off.



The role of data bus: data is transferred from or to the RAM using this bus.

The role of address bus: this bus tells the RAM from which location data will be needed to be read or written to.

A really good video explaining the fetch-execute cycle is on YouTube [<https://www.youtube.com/watch?v=IkdBs21HwF4>].

Secondary memory

2.1.5 Identify the need for persistent storage

Persistent storage is needed to store data and programs after the computer is turned off. Otherwise, all the programs and data would be need reinstalled every time the computer is restarted. Also, RAM is limited in a computer while large amounts of secondary storage are relatively cheap, so storing unused data and programs in secondary storage is useful to make space for running ones.


Operating systems and application systems

2.1.6 Describe the main functions of an operating system

Memory management: managing the usage of RAM by the single programs using the computer. To allow programs to be less concerned with computer architecture. Each program gets a space - a page - in RAM to run in. In this space the program believes it has all the computer for itself (it uses logical memory addresses that are then translated into physical memory addresses by the OS). Also, memory management manages the swapping of memory pages. When RAM gets full.

Control and monitoring of system functions: The OS controls the working of the peripherals and creates an abstraction of them for the programs to use. This way, the programs only need to be concerned with the tasks they are supposed to carry out and not with the different architectures of different computers and peripherals.

Distributing resources between competing programs:

 this section needs to be expanded

2.1.7 Outline the use of a range of application software

Word processor: A program to create and edit continuous texts

Spreadsheet: A program to manage calculations on complex amounts of data

Desktop Publishing: A program to edit media in a way to be published, like magazines, books or posters

Computer Aided Design (CAD): Software to digitally design and edit parts to be manufactured. Often includes a Computer Aided Manufacturing (CAM) module to be able to use the model to directly produce a prototype.

Web browser: Its clear, isn't it?

2.1.8 Identify common features of applications

- Possibility to terminate the application
- Possibility to save current working
- Toolbars
- Dialogue boxes
- Menus
- User interface (GUI) components

Binary representation

2.1.9 Define the terms: bit, byte, binary, decimal, hexadecimal

Bit: smallest unit in computers, either a 1 or a 0

Byte: a group of 8 bits

Binary: computers use the binary (base 2) number system

Decimal: This is the number system we use (base 10)

Hexadecimal: Base 16 number system. It has a base that is a power of 2, so we can use it to make reading binary codes shorter and thus easier to read. 4 binary characters grouped together give a hexadecimal character.

2.1.10 Outline the way in which data is represented in the computer

Computer work with the binary number system, so everything in them is represented in 1s and 0s. So, the data that we want to store in the computer needs to be **encoded** in 1s and 0s. For different media types, different encodings exist.

Text needs to be also transformed into 0s and 1s. To do this, a character set is used. A character set is a collection of characters and the binary codes that represent them. Today, the most used character sets are ASCII and UTF.

ASCII: stands for American Standard Code for Information Interchange. It uses 7 (later 8) bits to encode the characters, so in total $2^8=256$ different characters can be represented.

UTF: is a superset of ASCII. It was developed because people realized that the 256 characters of ASCII are not enough to satisfy our need for a suitable number of characters. The goal of the maker of UTF is nothing less than to represent every character humans have ever used.

Strings: representation of characters in a continuous bit chain

Integers: representation of numbers in computers. Whole numbers. Negatives can be represented by the 'two's complement' method.

Floats: representation of fractions within the computer. It is a special version of the scientific notation. It consists of a sign bit, the mantissa, the number itself, and the exponent which states how many characters left or right of the number the radix point 'floats'. The radix point is the version of the decimal point that can be applied in any number system. In the 64 bit float, 1 bit is used for the sign, 52 are used for the mantissa and 11 are used for the exponent.

Simple logic gates

2.1.11 Define the Boolean operators: AND, OR, NOT, NAND, NOR and XOR

AND: true only if *both* of the inputs are true

OR: true if *at least one* of the inputs are true

NOT: inverts a value, makes a false from a true and vice versa

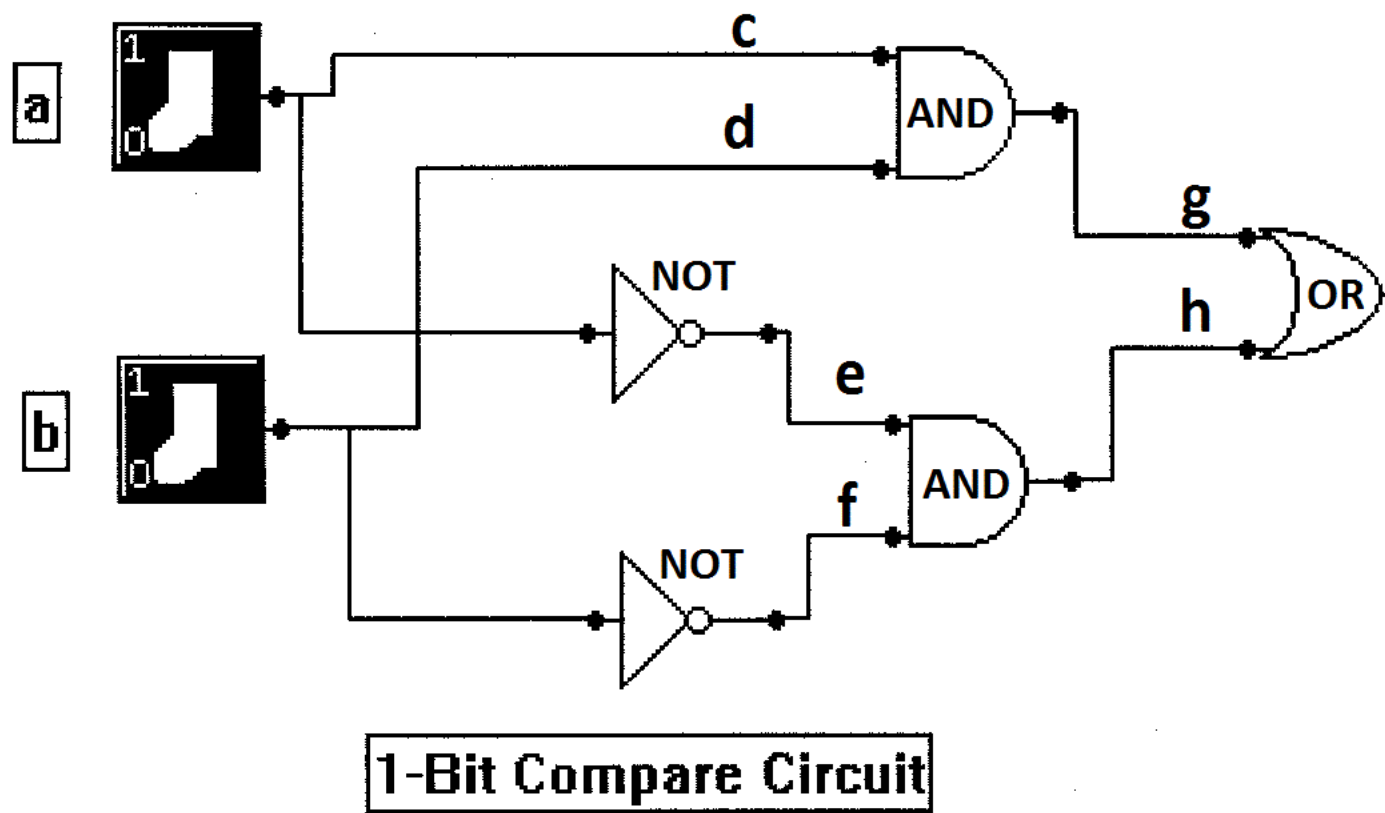
NAND: NOT + AND gates together. False if both the inputs are true, otherwise true

NOR: NOT + OR gates together. Only true if both of the inputs are false, otherwise true

XOR: exclusive OR. Only true if only *one* of the inputs is true, otherwise false

2.1.12 Construct truth tables using the above operators

A truth table is a table listing all the connections in a logic circuit and can thus be used to trace output to different inputs.



<dtable>

a	b	c	d	e	f	g	h	i
false	false	false	false	true	true	false	true	true
false	true	false	true	true	false	false	false	false
true	false	true	false	false	true	false	false	false
true	true	true	true	false	false	true	false	true

</dtable>

2.1.13 Construct a logic diagram using AND, OR, NOT, NAND, NOR and XOR gates

There is nothing much to say to this..... Just get the logic right.

The IB do not require us to draw the specific symbols for the gates, we should just draw a circle and write the name of the gate into it.

However, we are expected to draw straight lines and 90° angles.

Created by Matyas Mehn — [Matyas Mehn \[mailto:matyas.mehn@gmail.com\]](mailto:matyas.mehn@gmail.com) 2014/03/28 11:33