Chapter 3

General organisation of a digital computer

The von Neumann's model. Instruction Cycle.

§1. Definition of a digital computer. Computer architecture, computer organization, computer implementation.

- A digital computer in a complex equipment containing millions of elementary electronic components. (1.1)
- A digital computer represents a hierarchical system, composed of several interrelated subsystems up to the lowest level of elementary subsystems (components). (1.2)
- *Definition № 1*: a digital computer is a **system** intended to automate computations on **discrete information** in accordance with given **algorithms**. (1.3)
- *Definition № 2*: a digital computer is an union between a set of physical equipment, representing the **hardware** component, a set of microprograms, representing the **firmware** component and a set of programs representing the **software** component, allowing processing by means of arithmetical and logical operations of the **discrete information** at very **high speed** in accordance with given **algorithms**. (1.4)
- *Definition № 3*: a digital computer is a **finite automaton,** that is a finite state system, capable of processing at a very **high speed discrete information**, in accordance with given **algorithms**. (1.5)

- From the above definitions several common features are implied:
    1. The discrete nature of information.
    2. The computation process is totally automated.
    3. The processing is carried out according to given algorithms.
    4. The speed of processing is very high.
    5. The nature of processing is arithmetical and logical.

  (1.6)
- The term digital computer has a very broad significance, from single-chip microcomputers up to supercomputers. (1.7)
- Differences between different digital computers reside in size, cost, performance, range of applications, organizations. (1.8)
- Computer technology changes very fast covering all aspects of computer manufacturing, from underlying integrated circuits (IC) technology up to parallel organization concepts of building modern digital computers. (1.9)
- The development of Large Scale Integration (**LSI**) and Very Large Scale Integration (**VLSI**) technologies in the eighth decade of the preceding century ensured a dramatic reduction in the dimensions, a spectacular increase of the speed of operation and productivity, as well as incredible decrease in the costs and power consumptions of digital computers. (1.10)
- At present, due to the worldwide spreading of professional/personal computers (PC) more and more areas of human activities are supported by digital computers, a tremendous development of application programs had an impetus in further improving of digital computers characteristics. (1.11)
- In spite of the dynamics in the computer field, certain fundamental concepts apply consistently throughout, which are approached in this textbook. (1.12)

- In a classic paper, published in 1964, Amdahl, Blaaw and Brooks proposed dividing computer systems study and description into three levels:
  (1) Architecture
  (2) Organization
  (3) Implementation (1.13)
- **Computer Architecture** refers to attributes of a system visible to a programmer, that have a direct impact on the logical execution of a program. The Computer Architecture deals with the functional behavior of a computer system as seen by a computer programmer, referring to the nature of data types, range of realized operations, memory organization, the set of registers that are accessible to the programmer, the instruction set and format, addressing techniques, I/O mechanisms. (1.14)
- A computer architect develops the functional and performance specifications for various blocks of a computer system and defines the interfaces between these blocks, in consultation with hardware and software designers. (1.15)
- **Computer Organization** refers to the operational units and their interconnections that realize the architectural specifications. Organizational attributes include those hardware details transparent to the programmer, such as control signals, detailed structure of functional blocks, interfaces between the computer and peripherals, memory technology, extension techniques, clock frequency, etc. (1.16)
- **Computer Implementation** is defined as the actual hardware, actual physical structure, including logical design techniques, board layouts, physical interconnections, power supply, testing methods, interference between signals, mechanical design, etc. (1.17)
- Distinction between architecture, organization and implementation is important for any computer specialist. (1.18)

- Many computer manufacturers offer a family of computer models with the same architecture, but with different organizations, and, consequently, the different models in the family have different price and performance characteristics. Even if architecture remains unchanged, the available technology will determine a certain implementation. All implementations must execute the same programs and derive the same results. (1.19)
- In case of microcomputers the relationship between these facets of a computer system is very tight: changes in technology influence the organization, which in turn allows a more powerful architecture. (1.20)

- There is an important concept, that of **levels in computer architecture**, reflecting many different views at which a digital computer can be considered, from the highest level, where the user is running programs, to the lowest level, consisting of transistors and wires. (1.21)

§2. Short history on stored program computers concept

- The model on which present-day computers are based on originates in a scientific paper elaborated by the famous mathematician John von Neumann that was published in United States, in June 1945, which traced the essential lines to build a digital computer. (2.1)
- This model derived from a more general concept proposed earlier and known as “the principle of **stored program control”**; the corresponding structure was known as **stored program machine** (stored program computer). (2.2)
- The **stored program concept** considers that the machine language program is stored in the computer along with pertaining data and that the computer is able to manipulate the program as if it were data. (2.3)

- The stored program concept originated with John Eckert and John Presper Mauchley group of scientists including John von Neumann, as consultant, and technicians, which worked at the University of Pennsylvania, since 1943, on the first general-purpose electronic computer, **ENIAC** (Electronic Numerical Integrator And Computer). The project of designing and manufacturing **ENIAC** was supported financially by US Army, for developing range and trajectory tables for the new weapons created and used during the Second World War. (2.4)
- The **ENIAC** project was completed in 1946 and it was in operation until 1955. Among remarkable problems solved on **ENIAC** computer there can be mentioned performing a series of complex calculations help determine the feasibility of the H bomb. (2.5)
- Programming and data entry in **ENIAC** were performed by *setting switches and changing cables*; therefore, the task of entering and altering programs for the **ENIAC** computer was extremely tedious. (2.6)
- **ENIAC** computer had no central memory and had to be *programmed manually* by setting switches or plugging and unplugging cables. (2.7)
- In 1944, John Mauchley and Presper Eckert wrote a memo proposing the use of a magnetic drum as a main memory for the computer. (2.8)
- In 1945, John Von Neumann together with Mauchley and Eckert wrote another memo discussing the concept of the *stored program machines*, which was published on 30 June 1945. This was the draft for the next computer project, called **EDVAC** (Electronic Discrete Variable Automatic Computer), **the first conceived stored program digital computer**. The work on EDVAC started since 1945 with the same team, at the University of Pennsylvania (Moore School of Engineering) (2.9)
- The project EDVAC was delayed as the most important designers left Moore School of Engineering (Eckert, Mauchley, John Von Neumann, Burks, Goldstine etc.) (2.10)

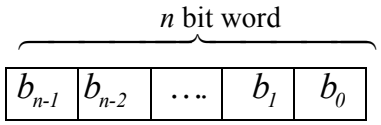
- Consequently, *the first stored program computer* called **EDSAC,** was completed in England, at the Cambridge University, under the supervision of Professor **Maurice Wilkes**, in 1949. Professor Wilkes had the opportunity to visit USA during ENIAC development, and to discuss with the main designers of ENIAC. He understood the importance of the concept of stored program computer and, after return in England, began his own project. (2.11)
- John Von Neumann moved to Princeton, at the Institute of Advanced Study together with Arthur Burks and Herman Goldstine. They started another project for a new stored program computer, **IAS** computer, which was completed in 1951. (2.12)
- Working on the new project, John Von Neumann together with Burks and Goldstine, published an important paper entitled "Preliminary discussion of the logical design of an electronic computer instrument". This was a detailed clearly argued discussion of many aspects of machine design based on *von Neumann model*. The project **IAS** was very important because it incorporated *many innovative concepts*, had a *broad range of applications* and was widely known. Several copies of **IAS** computer were built after 1952. (2.13)
- The **EDVAC** project was also completed in 1952 at the Pennsylvania University, but it had already a little impact on scientific world and market. (2.14)
- John Von Neumann model remained essentially unchanged since that time (1945) despite the fact that technological advances have opened the path to enormous progress, both in the direction of the higher levels (new programming languages, new operating systems), and to the lower levels (hardware, machine code, microprogramming). (2.15)

- The essence of stored program computer consists in representation of algorithms corresponding to the problem to be solved by a flowchart comprising two types of operators: (2.16)
  - **(a) Processing operators**
  - **(b) Sequencing operators**
- This refers to the fact that any problem must be "described" to the computer as a *sequence of operations* realized by the computer; the machine expects a program, that is a set of instructions, which tells it what to do from one moment to the next moment.The processing operators specify the method of changing/ transforming the input data, whilst the sequencing operators are decoding the succession of operation execution. (2.17)
- There can be derived *several schemes* to implement this principle, but the most widely accepted is the model proposed by John Von Neumann in 1945. (2.18)
- From a fairness stand point, it must be signaled a previous work, completed in 1942, by Crawford, at M.I.T, where it was suggested the use of magnetic drum as a main memory in a digital computer; hence, a precursor of the stored program computer existed since 1942. (2.19)
- In recent years several alternatives of models were proposed, all known as non-Neumann architectures, such as *Data-Flow architectures*, *Harvard architectures*, *Neural Computers* etc. (2.20)

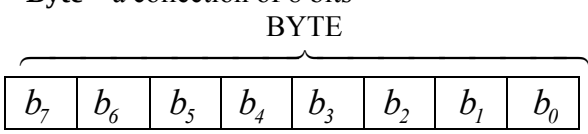
§3. The Von Neumann's principles

- The following 5 principles are the basic elements for definition of the Von Neumann's model of digital computers. (3.1)
  - *Principle 1*: the information within a digital computer is **binary coded**, which means that it is represented with binary digits, named **bits** (0 and 1). The term **bit** is an abbreviation for binary digit. (3.2)

- A set of *n* bits is called a **word**: (3.3)

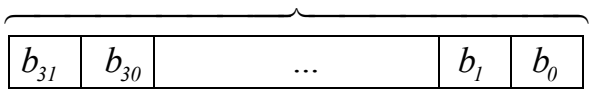


- Depending on the length *n* there are defined the following entities:
  - **Byte** – a collection of 8 bits (3.4)



  - **half-word** - a collection of 16 bits or 2 Bytes (3.5)

  - **word** - a collection of 32 bits or 4 Bytes (3.6)



  - **double-word** a collection of 64 bits or 8 Bytes (3.7)

- A word may represent either a *command* (*instruction*) or a *datum* (with plural data), so that there are defined two kinds of words: (3.8)
  - (a) Datum words
  - (b) Command words (Instructions)

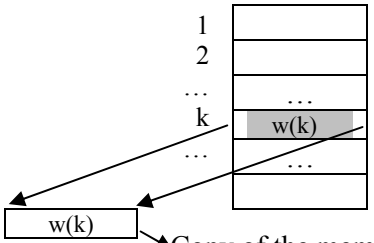
  Sometimes these are called simply *Data* and *Instructions*.

➢ *Principle 2*: the **distinction between the two types of words** is not carried out by the coding method (coding rule), but on the way of words utilization – the, so called, *utilization context*. (3.9)

- As it is known, data words represent numbers (binary, decimal, etc.) or alphanumeric characters. (3.10)
- Considering a binary combination on 8 bits, 10011001, it can be interpreted as a numeric datum or an instruction; hence, inside the computer the code is the same, both for the data and for commands, but it is the task of the user, by the way of manipulating the information, to ensure a proper distinction. (3.11)
- An evident advantage of such a coding, in particular of the command words, is outlined by the possibility to process these words in a similar way, if necessary, as the data words, ensuring thus a possibility of changing the structure of a program. (3.12)

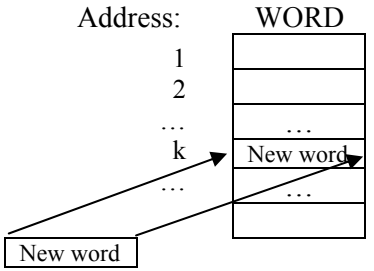
➢ *Principle 3*: the words are placed in **memory locations**, every location being assigned a specific number, called **address**. (3.13)

- Every data word or instruction word is located/stored in memory at a certain address, so that the address is a **pointer** to that word. (3.14)
- Any **Read** operation from the memory extracts a word from the memory. Since it is a non-destructive operation, it is obtained a *copy* of the word located in the memory at the specified address, the content of the location itself remains unaltered. (3.15)



Copy of the memory location with address k

- A **Write** operation implies insertion of a new word in the memory at the addressed location (k): (3.16)



- The concept of **address** plays an important role for the von Neumann's model. (3.17)

➢ *Principle 4*: the running algorithm is represented by means of a **sequence of command words** (instructions). (3.18)

- Every command word/instruction has to decide:
  (a) The *nature* of the performed operation (function)
  (b) The *operands involved* in the operation. (3.19)
- In general, any instruction is formed of two major *fields* (zones), called the **Operation Code** (OPCODE) and the **Address**. (3.20)
- The first part of an instruction contains the *opcode field*, whilst the second contains the *address field*. (3.21)
- Always, an instruction is divided in several fields or subfields, each corresponding to a specific role. (3.22)
- Usually the address field is subdivided into several smaller subfields pointing to several potential operands and address of the result. (3.23)

The general structure of an instruction: (3.24)

1…….L	1……k	1……...k	…	1..…… k
OPCODE	ADDR1	ADDR2	…	ADDRt

- The OPCODE field is a set of *L* bits representing a binary code which is assigned to a specific operation (function) performed by the CPU (processor) of the digital computer (addition, subtraction, multiplication, halt, rotation, complementation, division etc.) (3.25)

- Therefore, the OPCODE field is also called the **function of the instruction**. (3.26)
- By means of L bits there can be defined 2^L different patterns of 0 and 1, therefore there can be defined 2^L different operations (functions). (3.27)
- The address subfields ADDR1, ADDR2,…, ADDRt represent binary coded addresses on k bits specifying the address where operands are located, as well as where the result of the operation is to be placed. (3.28)
- By means of k bits there can be defined 2^k different addresses. This range represents the **address space** of the digital computer. (3.29)
- The address space is a key concept of the architecture of a digital computer. (3.30)
- The specific partition of an instruction in two or several fields and subfields corresponds to the general concept of the **instruction format**. (3.31)

➢ *Principle 5*: the data processing performed by the CPU (processor) is univocally determined by the **sequential instruction execution** constituting the **program** for the implemented algorithm. (3.32)

- Hence, any algorithm is carried out by a set of instructions which constitute the **computation program**. (3.33)
- The first executed instruction of a program corresponds to the initial address named **starting address**. (3.34)
- The next instruction is determined **automatically** by the CPU (processor) during the execution of the current instruction. (3.35)
- The next instruction is determined by its address in memory, which can be the **next** sequential address or another **different** address from the address space, like in case of *branching* instructions. (3.36)

- The simplest way to specify the address of the next instruction is to include this address in a subfield of the current instruction in execution. But, due to the length of the instruction, such a solution is not convenient. (3.37)
- It can be seen that by this principle it is ensured an automated way of program execution. (3.38)

- The program loaded in the memory that is executed by CPU defines a certain **input-output function** on the set of data. Thus, a program can realize the same input-output function on different sets of data, but changing the program a new input-output function is selected. (3.39)
- A given computer can accept a **finite set** of possible programs, thus defining a finite set of input-output functions.This range depends on the computer capabilities. (3.40)
- The set of accepted programs by a computer is called the **class of realizable functions** of a digital computer. (3.41)
- Therefore each computer has its own limits, allowing only a set of realizable applications. The user must correlate the approached application with the capability of the selected computer. (3.42)
- The main feature outlined from these principles is the **procedurality** in solving any problem submitted to a digital computer, materialized in an *algorithm* and a *program*; this refers to the fact that any problem must be presented to the computer as a **sequence of operations**, i.e. a **sequence of instructions**. (3.43)

§4. The von Neumann's model of a digital computer

- As it was previously mentioned, John von Neumann was the first to tie together all ideas developed in the team working on project EDVAC, publishing in June 1945 a report entitled " First draft for a report on EDVAC computer". In that document, John Von Neumann set forth the basic logical structure of the stored program computer. (4.1)

- The following five criteria define a digital computer in terms of conforming to the given von Neumann's principles:
  - It must have an *input medium*, by means of which an essentially unlimited number of operands (data items) or instructions may be entered;
  - It must have a *store*, from which operands (data) and instructions may be obtained and into which the results may be entered;
  - It must have a *processing section*, capable of carrying out arithmetic or logical operations on any operands taken from the store;
  - It must have an *output medium*, by means of which an unlimited number of results may be delivered to the user.
  - It must have a *control section* capable of interpreting instructions extracted from the memory and capable of choosing between alternative computer results.

(4.2)

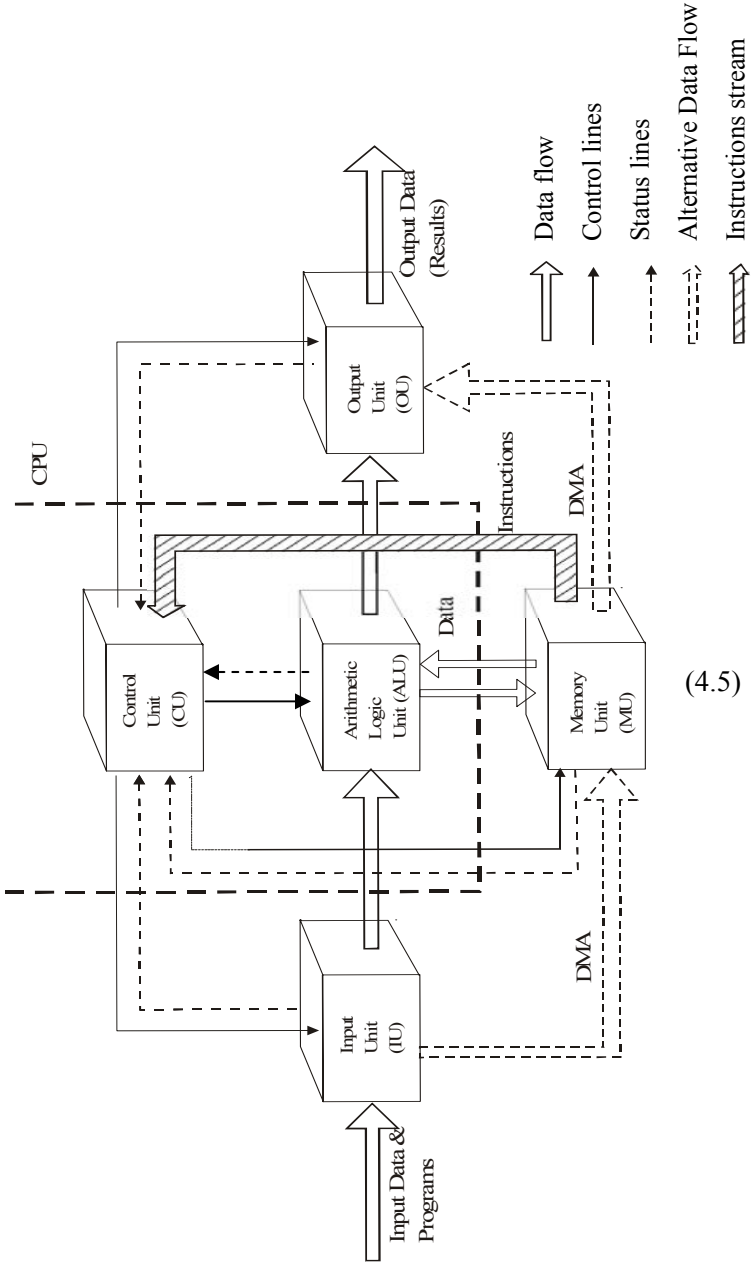
- The basic structure resulting from these criteria is known as the **von_Neumann's model** of digital computer. The model is also named the *five units model* of a digital computer.

(4.3)

- Virtually, all computers built since that time have been used this organization.

(4.4)

- The von Neumann's model consists of five major components as illustrated below in (4.5):



(4.5)

- It results that the general structure of any digital computer incorporates the following five units: (4.6)
  - Input Unit (IU)
  - Memory Unit (MU)
  - Arithmetic and Logic Unit (ALU)
  - Output Unit (OU)
  - Control Unit (CU)
- The Arithmetic and Logic Unit (ALU) together with the Control Unit (CU) are frequently referred to collectively as the **Central Processing Unit** (**CPU**). (4.7)
- Most computers commercially available can be decomposed into these five units. (4.8)
- A brief description of each unit:

1) **Input Units** (4.9)
- ensure the input of data and programs;
- data can be acquired from an industrial process as well;
- two modes of operation: 1) direct and 2) indirect;
- the indirect mode assumes the use of an intermediate medium(cards, tapes);
- the main feature imposed to any Input Unit is the high speed of operation which influences absorption of higher volumes of data and programs.
- *Examples*: 1) keyboard, mouse, scanner, digitizer
  2) card reader, microfilm reader, paper tape reader.

2) **Arithmetic and Logic Units (ALU)**
- fundamental units in any computer because they are capable of generating *new information* via carrying out arithmetic and logic operations on data. (4.10)

- ALU is the *most productive unit* of the digital computer; (4.11)
- The main constituent blocks are: adders, subtractors, complementers, registers, decoders, shifters, etc. (4.12)
- Inside ALUs there are implemented specific processing algorithms representing the **Computer Arithmetic**. (4.13)
- At present, ALUs realized through VLSI technologies incorporate very high performance; speeds of operation in the range of billions of operations per second. (4.14)

3) **Memory Units**
- Store programs and data;` (4.15)
- Although seemingly simple in concept, computer memory exhibits the widest range of type, technology, organization, performance and cost; (4.16)
- Functions: writing, storing, reading; (4.17)
- Any type of memory unit characterized by the following two main features: *capacity* and *access time.* (4.18)
- Capacity represents the amount of data items stored, measured in bits, bytes, words. The current measuring units are *kilo*(2^{10}), *mega*(2^{20}), *giga*(2^{30}); (4.19)
- Access time gives the speed of operation of the memory; (4.20)

- Access time is the interval between the instant when a command of reading is given and the instant of data availability; (4.21)

- An optimal structure of memory would have the highest possible capacity and the lowest access time; (4.22)

- Memory is organized as a **hierarchical structure** being layered in accordance to the capacity and speed; (4.23)

- The major levels in the hierarchy are: (4.24)
  - **Main Memory** (Primary Memory, Internal).
  - **Secondary Memory** (External).

- These levels present sharp differences in technology, speed, capacity, management, sublevels; (4.25)

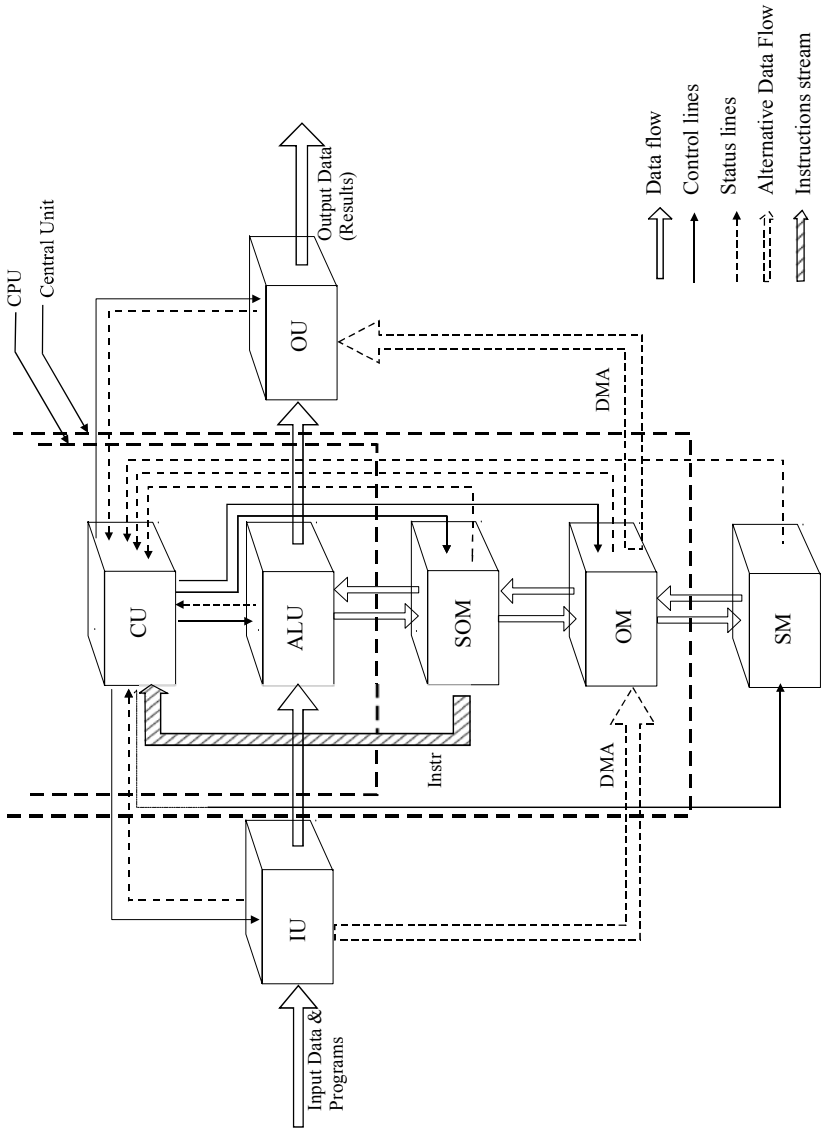
- Key characteristics of memory units are: (4.26)
  1) Performance: access time, cycle time, transfer rates, etc;
  2) Physical type: semiconductor, magnetic, optical, magneto-optical, etc;
  3) Unit of transfer: word, byte, block;
  4) Capacity: word size, number of words;
  5) Location: internal(main), external(secondary);
  6) Access mode: sequential, direct, random, associative;
  7) Organization;
  8) Physical characteristics: volatile/nonvolatile, erasable/non erasable, etc.

- **Main Memory** (**MM**)
  - At present, Main Memory is realized in VLSI technologies ensuring large volumes (high capacity) and high rates of accessing the data (in the range of nanoseconds). (4.27)
  - The Main Memory (**MM**) is split into two sublevels (4.28)
    1) superoperative memory ─ **SOM**
    2) operative memory ─ **OM**
  - **SOM**
    - It is the closest to the CPU; (4.29)
    - It has a small capacity and a very small access time (units or tens of nanoseconds); (4.30)
    - It is intended for storing current instructions in execution, intermediate results and the data involved in the current execution of the program; (4.31)
    - The main function of the SOM is the matching of the speeds of operation of CPU and the Operative Memory; (4.32)
    - It is an expensive resource of the digital computer; (4.33)
    - In a particular form of organization it is known as **CACHE** Memory (4.34)
    - Control of operation through specific hardware mechanisms transparent to the user; (4.35)
    - At present, CPUs incorporate a part of this memory. (SOM exhibits in turn a hierarchical structure). (4.36)

- **OM**
  - OM contains the program (or programs, in case of multiprogramming operation) in execution and the corresponding data; for any digital computer execution process implies that the current instruction and the attached data are read from the OM; (4.37)
  - The capacity of OM is larger than that of SOM, whilst the access time is worse; (4.38)
  - Nowadays the OM *capacity* is measured in Megabytes; (4.39)
  - Operation of OM is based on *Random Access* technique which means that the time to get access to *any location* is identical, regardless its address; (4.40)
  - At present, OM is realized using *semiconductor technologies* (including RAM, DRAM, SRAM memories); (4.41)
  - An important concept referring to OM operation is the *addressable information unit* (**AIU**); in many computer systems the AIU is just the **computer word**; but in other systems it is allowed addressing at the *byte level*. (4.42)
  - By *unit of transfer* it is understood the number of bits read out or written into memory at a time; the unit of transfer need not to be equal to a *computer word* but it coincides with AIU. (4.43)

- **Secondary memory** (**SM**)
  - **SM** is also called *External Memory*; (4.44)
  - External memory consists of *peripheral storage devices;* (4.45)
  - **SM** has much greater capacities than MM; (4.46)
  - **SM** has greater access times than MM;
  - **SM** offers smaller costs per bit than MM;
  - **SM** aimed at storing *large sets of data*, measured in **MB**, **GB**, **TB** (Terabytes); (4.47)
  - The access time ranging from microseconds to milliseconds; (4.48)
  - If a program is stored in **SM** in order to be executed it must firstly be loaded (transferred) into the MM; (4.49)
  - The transfers between MM and SM, organized in words or blocks, are carried out under *system software* control, known as *operating system*. (4.50)
  - *Transfer rate* is the rate at which the data can be transferred into or from a memory unit (4.51)
  - Many types of **SM** exist depending on the technology, principle of operation, storing media (usually magnetic), speed of operation etc; (4.52)
  - Main classification according to their speed of operation; (4.53)
  - *Examples*: hard disk units, floppy disk units, streamer tapes, CD-ROMs units, magneto-optical disks, DVD units etc; (4.54)

- The expanded basic structure of a digital computer when the hierarchy of the Memory Unit is taken into account is presented in (4.56). (4.55)



(4.56)

4) **Output Units (OU)**

- **OU**s ensure communication between a digital computer and the outside world by extracting information for the user – called *Output Operation;* (4.57)
- Sometimes the output information is intended for industrial/ laboratory processes; (4.58)
- Two main modes of operation: 1) direct
  2) indirect (4.59)
- *Examples*:
  1) video monitors, printers etc.
  2) card punchers, tape punchers etc. (4.60)
- Input and Output Units are collectively called **I/O Peripheral Equipment**; (4.61)
- At present, a major I/O function is *remote communication*, enabling specialized I/O communication devices (MODEMS, communication cards, bridges, routers etc.). (4.62)

5) **Control Unit (CU)**

- **CU** has the role of managing, monitoring, supervising the operation of all units from a digital computer, including its own operation; (4.63)
- **CU** ensures the *complete automation* of the computing process; (4.64)
- **CU** determines the current operation based on the current instruction in execution and the *status information;* (4.65)
- **CU** is a complex *sequential logic network;* (4.66)
- **CU** contains mainly registers, counters, frequency dividers, controllers, decoders etc; (4.67)
- **CU** are classified in two classes:
  - hardwired **CU**s
  - microprogrammed **CU**s (4.68)

- **CU** implements the fundamental mechanism of instruction execution: it reads the current instruction from the memory (**OM** or **SOM**), decodes it (interprets it) in order to decide the function to be executed and to activate the corresponding control signals and ultimately executes it, yielding the result. (4.69)

- In the general structure of a digital computer there are represented some *additional data paths* from Input Units to Memory Unit as well as from Memory Unit to Output Units. These paths correspond to a special mechanism of speeding up the transfers from Input Units and to the Output Units. (4.70)

- By involving CPU in realizing the Input/Output operations, the CPU performance is seriously limited due to the time required by the Input/Output units to execute the instructions related to I/O transfers. There is a big disparity in the CPU and peripheral devices speeds of operation. Whenever large quantities of data are to be transferred the standard mechanism of involving the CPU is inefficient and alternative solutions are to be considered, by avoiding participation of the CPUs. (4.71)

- When large volumes of data are to be moved into or from the Main Memory it is more efficient to use the **Direct Memory Access** (**DMA**) techniques. (4.72)

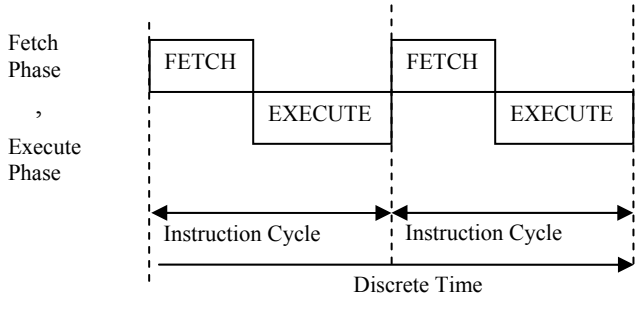
- **DMA** involves an additional unit, called **DMA controller**, that is capable of replacing the CPU, when such kind of transfers are initiated. Thus, the Input data can be transferred *directly* into the Main Memory *bypassing* the **CPU**; similarly, large volume of data are transferred *directly* from the Main Memory to Output Units, *bypassing* the **CPU**. (4.73)

§5. Instruction Cycle

§5.1. Principle of the Digital Computer Operation

- The program is stored in the memory of the computer system. The program is formed of a series of instructions.
- These instructions are called *machine language instructions* and the entire program is called *machine language program*. (1.1)
- The program is executed instruction by instruction. To accomplish this operation the following phases are to be followed: (1.2)
    1. The machine language instruction must be read from the memory and transferred in the CPU; this operation is called FETCH operation, and since it is fetched an instruction its exact name is FETCH INSTRUCTION (FETCH I) (1.3)
    2. The instruction must be executed in the functional units of the computer (mostly in ALU), which means to be processed by generating a result; this phase is called EXECUTE phase. (1.4)
- The above defined phases, FETCH and EXECUTE are collectively called **FETCH-EXECUTE CYCLES**, as they are repeatedly executed while running a program. (1.5)
- Thus, the basic computer operation can be summed up as being one of FETCH-EXECUTE-FETCH-EXECUTE…., performed repeatedly until the computer is halted. (1.6)

- A sequence FETCH-EXECUTE for an instruction is called **Instruction Cycle**. (1.7)
- Obviously, the CPU must be informed about where are stored the operands and where to store the result. (1.8)
- The CPU must also be provided information about where to find the next instruction. (1.9)
- According to Von Neumann's principles, the processing of a program is basically sequential, the instructions are executed one after another:



(1.10)

- By this simple sequence of operation at infinitum, a CPU can execute programs of any complexity. (1.11)
- This essential simplicity made von Neumann model of a computer very convenient and widespread; the interpretative mechanism as described above remained the same, independently of the level of technology. (1.12)

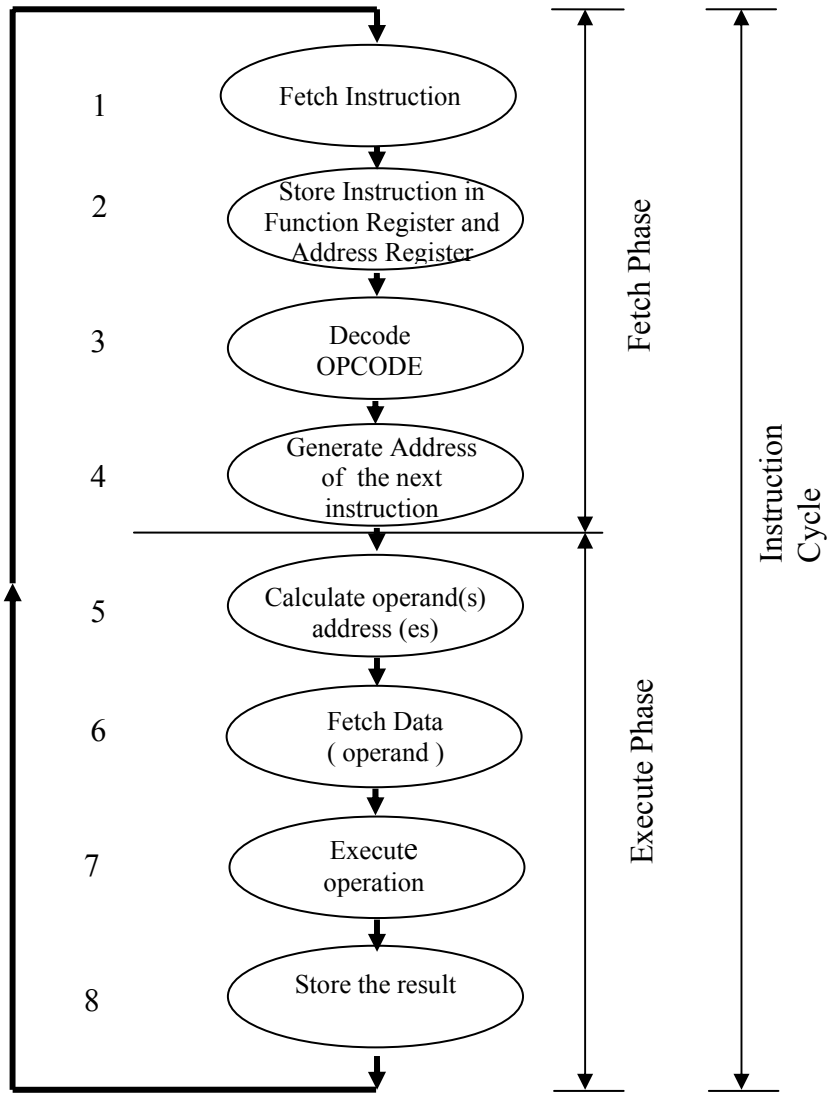
§5.2. Implementation of the Instruction Cycle

- The complete execution of an instruction corresponds to an **Instruction Cycle**, which is formed of two phases called FETCH and EXECUTE. (2.1)
- Each phase can be divided in several subphases or steps, according to the required internal operations linked to the instruction processing. (2.2)

- During a FETCH phase the following steps are encountered:
  1. The CPU reads from the memory at the address specified by the **Program Counter** the instruction to be executed and stores it in a register from the Control Unit , called **Instruction Register** (IR). (2.3)
  2. In the Control Unit the Instruction is divided into its two major fields, OPCODE and ADDRESS, and stored accordingly into the **Function Register** and **Address Register**. (2.4)
  3. The Control Unit decodes the OPCODE, i.e. the content of the Function Register; this step is called decoding the OPCODE, or interpreting the OPCODE. To do this, the Control Unit contains a functional unit fulfilling the role of a decoder. Based on this interpretation, the Control Unit can issue all commands to the other functional units to execute the current instruction. These commands are issued by the Control Sequencer block from the Control Unit. (2.5)
  4. The Control Unit is generating the next instruction address, by incrementing the Program Counter. The Program Counter (PC) is a pointer to the next instruction to be executed by the CPU, for which reason it is also called **Instruction Pointer** (IP). (2.6)
- After generating the next instruction address, the FETCH phase is over and a new phase begins, called EXECUTE phase, formed of the following steps (assuming an arithmetical processing instruction):
  5. Using the information from the Address Register, the Control Unit must determine the **effective addresses** of the operands. (Addressing techniques principles are implied) (2.7)

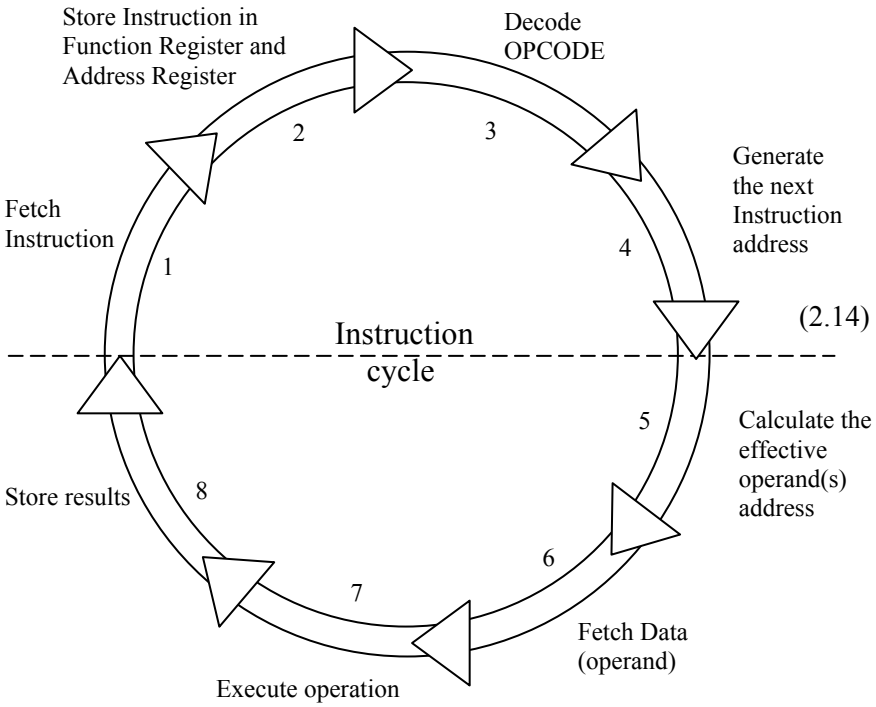
6. The CPU reads the memory again to extract the necessary operand (operands); this subphase or step is known to be FETCH DATA. The data (operand) is transferred in the ALU. Sometimes, operands are placed inside the CPU, in the register file of the ALU, constituting the *local memory* of the CPU; in such cases a new Read Memory operation is not required, the data (operands) are much faster supplied directly from the local memory of the CPU. (2.8)
7. The CPU effectively realizes the operation specified by the OPCODE, by processing the operands according to the given operation in the instruction and generating the result in the ALU. (2.9)
8. The result can be stored in the memory; to realize this a Write Cycle in the memory is started, by which the result of processing is stored in the memory at an address determined at step 5. But, most frequently this step is missing from the instruction cycle, as the result is saved inside the CPU's registers, for instance in a dedicated register, the Accumulator, having in view further processing. (2.10)

- By taking into consideration the enumerated steps, the Instruction Cycle can be represented in the following graphical manner (2.12): (2.11)



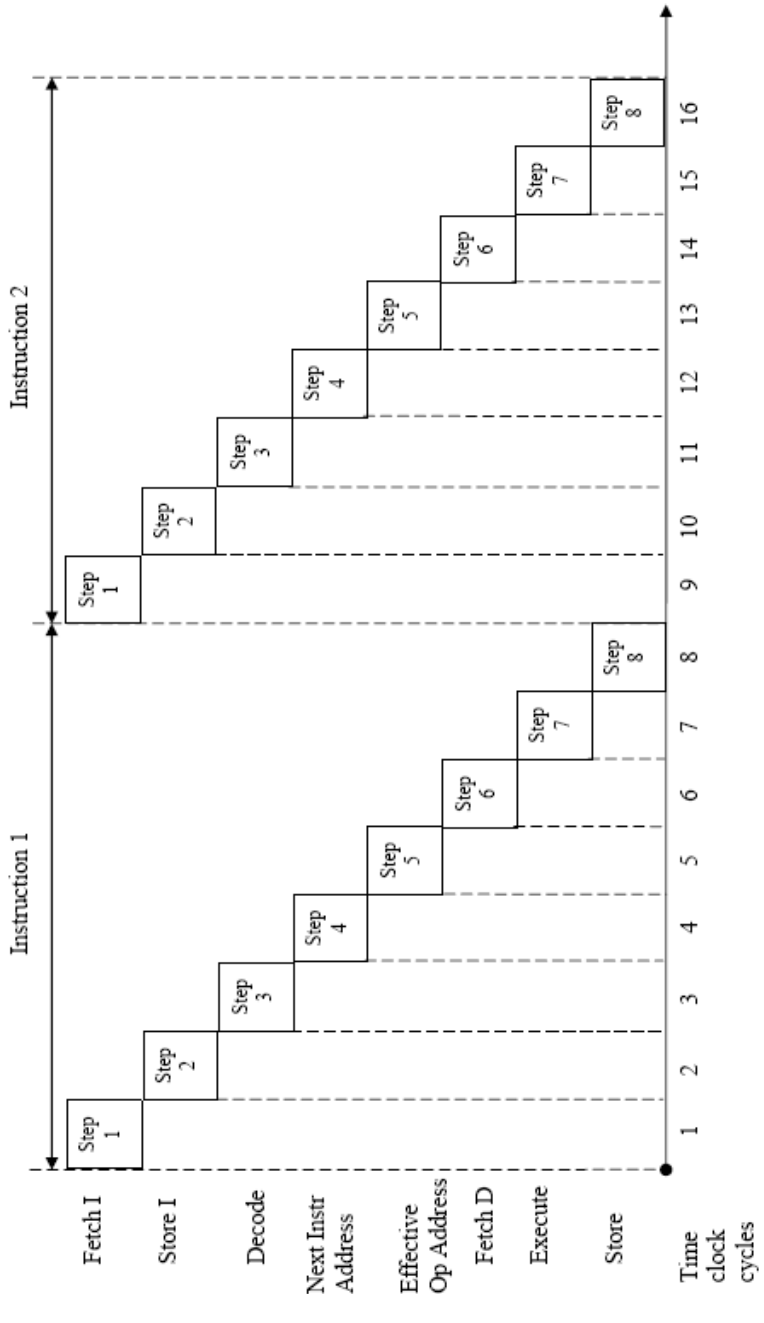
(2.12)

- This flowchart can be rearranged in a circular, more illustrative, form as depicted in (2.14): (2.13)



(2.14)

- The flow of running a program, i.e. a sequence of instructions, can be represented on the time axis as illustrated in (2.16) (2.15)



(2.16)

- Some of the steps from an Instruction Cycle may be absent in certain instructions; for instance, there exists a class of instructions without operands for which the steps (5),(6) are not required at all; in case of JUMP instructions there can be skipped steps (4), (5), (8);in case of comparison instructions the result is not stored, as only the flags (condition bits)from ALU are updated; for instructions defining multiple operands, the steps (5) and (6) are repeated several times until all operands are fetched in ALU. (2.17)
- A digital computer must go through thousands, millions or even billions of Instruction Cycles to fully process a single program. (2.18)
- Always the **FETCH phase is unique**, whereas **EXECUTE phase** presents manifold aspects, depending on the concrete instruction to be executed. Therefore, **EXECUTE phase** can be realized in *different arrangement of steps* specific to class of instructions (at machine level). (2.19)
- The time required for running an Instruction Cycle depends on the machine cycle, defined by the time clock cycle; in the computer world speeds are rated in Megahertz (MHZ) or Gigahertz (GHZ). Each MHZ represents one million clock pulses per second, while each GHZ represents one billion clock pulses per second. Currently there are manufactured microcomputers running at a speed over 1 GHZ. Therefore, at present the instruction cycles are measured in microseconds and nanoseconds (10^{-9} sec). (2.20)