1. INTRODUCTION

1. **Computers** can be divided into three functional groups:
    a. **Personal computers** (PC)
        i. Made for individual use
        ii. Laptops, tablets
    b. **Servers**
        i. Better the performance, higher the price
        ii. Ranging from powerful desktop computers to Supercomputers
    c. Embedded computers
        i. The most numerous group of computers
        ii. Microcontrollers (consisting of CPU, Memory, and IO) in automobiles, mobile phones, gaming consoles, household appliances, …

2. **Computer architecture** is what the programmers see on an assembly-level language, independently of the physical and logical realization behind it.

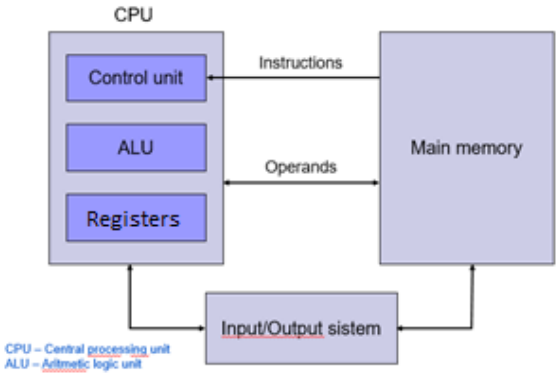
   Computer organization is the physical and logical realization behind computer architecture. It is closer to the hardware level.

   One architecture can be realized with different types of organizations and vice versa.

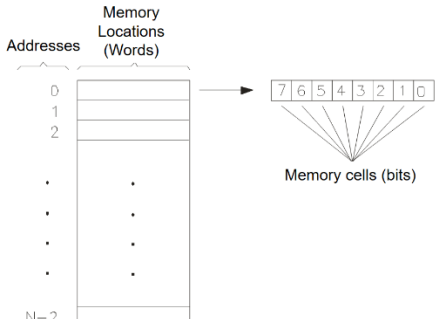
3. **A machine language** consists of instructions that can be directly executed by the computer. Those instructions are called **machine instructions**, and are built-in the computers as their native "language". Generally, machine instructions can be different in different computers made by different manufacturers.

4. **How does the computer work?**
   a. **Digital computers** are machines for solving problems by the execution of instructions that are set (programmed) by people.
   b. The sequence of those instructions is called a **program**.
   c. Before execution, every high-level program has to be translated into a limited set of machine instructions that the electronic circuit (computer) understands.
   d. Different processors can have different machine code instructions.
   e. Every higher-level programming language or concept (Java, Python, C, C++…) has to be somehow translated to those basic machine code instructions.

5. Structure of a typical computer (Von Neumann's computer model)



   a. **CPU** – Central processing unit
      i. Control unit (Gives the commands)
      ii. ALU – Arithmetic logic unit (does the calculations)
      iii. Registers (memory in the CPU)
   b. Main memory



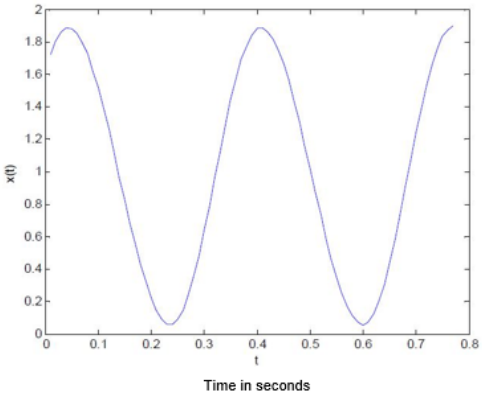
i. It is composed of memory locations called words, which contain memory cells, called bits, and every word has its own address.

c. **Input/Output** (I/O) system
d. **Buses** (connect the components, so instructions, operands, and other data can flow between system components)

6.

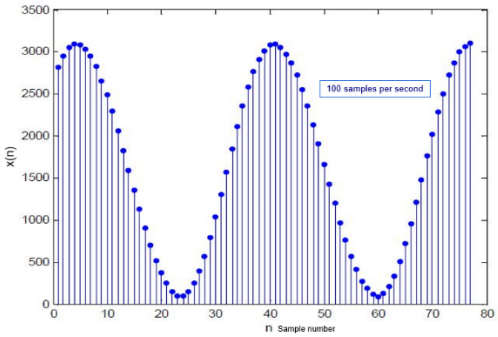
7. **Signals can be:**
    a. **Analog** (Continuous)

i. Signal's value is defined for every value on the x axis
ii. It is a continuous function (x and y axes)
iii. Usually a result of an observation of certain physical quantity
iv. Example: Analogue watch: based on the watch hands, you can tell the time with greater (continuous) accuracy.



    b. **Digital** (Discrete)

i. It is based on sampled signal values - numbers
ii. It is a discrete function (on both axes)
iii. The correct value is approximated due to finite word (number of bits)
iv. Example: Digital watch: The accuracy depends on the numbers the watch shows, usually minutes, sometimes seconds.



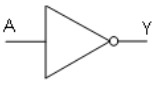
Digital computing

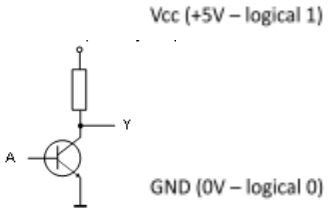
c. With digits 0 and 1, we can form the **binary numeral system**
d. A binary digit is called a **bit**, and it is represented either by 0 or 1
e. Digital computers are based (built) on using the binary numeral system

8.

9. Important ideas in Computer Architecture (and also more general in the Computer Science area):
    a. Moore's law
        i. The number of transistors on an integrated circuit doubles every two years
    b. Abstraction as simplification
        i. In the design of hardware, software, programming languages…
        ii. We divide problems/modules into smaller units and stack them in the hierarchy
    c. Speed
        i. It is most profitable to speed up the most commonly used procedures first
    d. Parallelism
        i. Simultaneous execution of tasks, the consequence of technology evolution
    e. Pipelines
        i. Series of stages where the output of one stage is the input for the next
        ii. All stages perform parallel execution of serveral instructions, each in its own stage
    f. Performance with speculations
        i. Better work in advance according to some foresight rather than wait in idle
    g. Memory hierarchy
        i. A result of the compromise between memory speed and cost
        ii. Consists of several levels with different compromises between speed and cost
    h. Reliability with redundancy
        i. Back-up systems, sometimes it is cheaper to have a redundant system

10. The physical structure of the computer:
    a. Mathematical (logical) view: logic gate – ideal
    b. Electrical realization: electronic circuit (transistor as basic element)
        i. Cons of realization:
            1. Analog voltages
            2. Inherent Time delays
            3. Vulnerability to noises



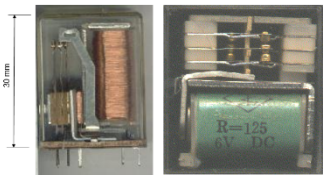


11. Switches
    a. In computers, information is represented in the binary system (0,1), using electrical signals.
    b. There are two states, which are represented with two voltages:
        i. State 0 is usually low voltage - switch is opened
        ii. State 1 is usually high voltage – switch is closed
    c. One switch operates 1 bit of information and therefore can be one of two states, 0 or 1.
    d. One basic memory cell is built upon switches and can store 1 bit of information

12. Switches in history:
    a. **Relay** – electromechanical switch – 1939+
    b. **Vacuum tubes** – electrical switch – 1945-1955

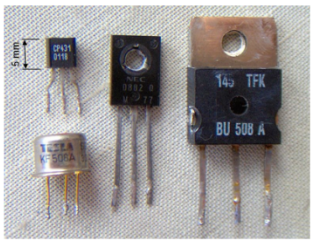
c. **Transistors** – electrical switch – 1955+
d. **Integrated circuit** (chip)– More transistors on one silicon board – 1958+

a)

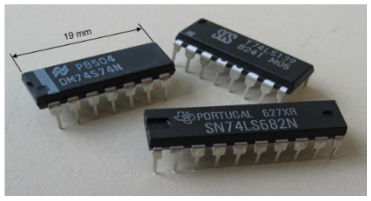


b)

c)

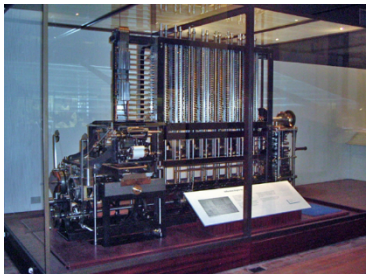


d)

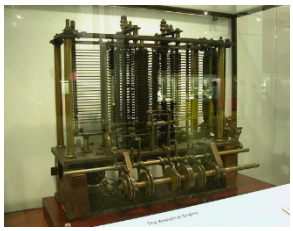


2. THE EVOLUTION OF THE COMPUTING MACHINES

1. Period of **mechanics** (1600)
    a. The first calculators were made, which were mechanical and manually operated.
    b. **Charles Babbage**
        - Differential machine

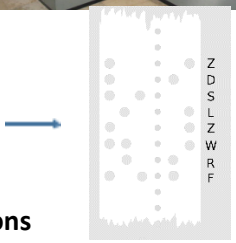


        - **Analytical machine** – first real precursor of today's computers
            - Operations were run by a program
            - Designed to solve arbitrary problems
            - Not fully completed



2. **Electro-mechanical computers** (1939)
    a. Electric motors were used to drive the gears
    b. Punch cards processed electrically instead of mechanically
    c. Konrad Zuse
        - Z1 – first working machine of Babbage's kind, completely mechanical.
        - Z2 – arithmetical unit built with telephone relays, unfinished.
        - **Z3**
            - First working **electromechanical general purpose computer controlled by programs**
            - Used binary based arithmetic
            - Storage of instructions on perforated (punched) film

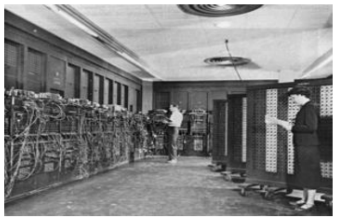
    d. Harvard
        - **Mark I**
            - Equivalent to Babbage's analytical machine
            - Decimal arithmetic
            - Storage of instructions on perforated (punched) tape



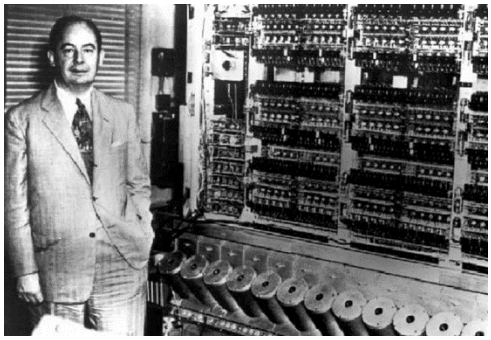
        - Mark II, III, IV

*Harvard Mark I and Zuse Z3 are **similar machines**, both having the **storage of instructions on perforated tape**. The difference is that Mark I uses **decimal arithmetic**, while Z3 uses **binary arithmetic**.

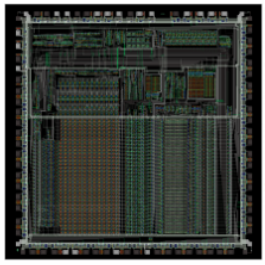
3. First **electronic computers** (1945)
    a. Electronic tubes replaced relays, increasing the speed of the computers
    b. Used for decryption of messages during WWII in the UK
    c. **ENIAC** – University of Pennsylvania
        - **E**lectronic **N**umerical **I**ntegrator **A**nd **C**alculator
        - Programming using switches and connecting cables



4. **Electronic stored program computers** (after 1945)
    a. John von Neumann
        - EDSAC
            - First operational stored program computer
            - Rule, which is still followed nowadays: ***"If the instruction does not require otherwise (JUMP, GOTO instructions), instructions are read and executed in ascending address order."***
        - EDVAC
            - The program was stored in the memory
        - IAS
            - RAM – random access memory
            - **Program counter** – register that contains the address of the next instruction



5. Rapid development of computers after 1950
    a. Development was more technological than architectural
    b. Tubes were replaced by **transistors**, which are smaller, faster and more reliable
    c. First **microprocessors** appeared (1971)
    d. First Personal computer IBM PC (1980)
    e. First ARM processor (1985)



3.BASIC PRINCIPLES OF COMPUTING

1. The **Von Neumann** computer model:
   - A machine that has its programs stored in the main memory.
   - The **program** leads the machine operation; the **instructions** tell the machine what to do.
   - The Von Neumann model consists of:

a. CPU

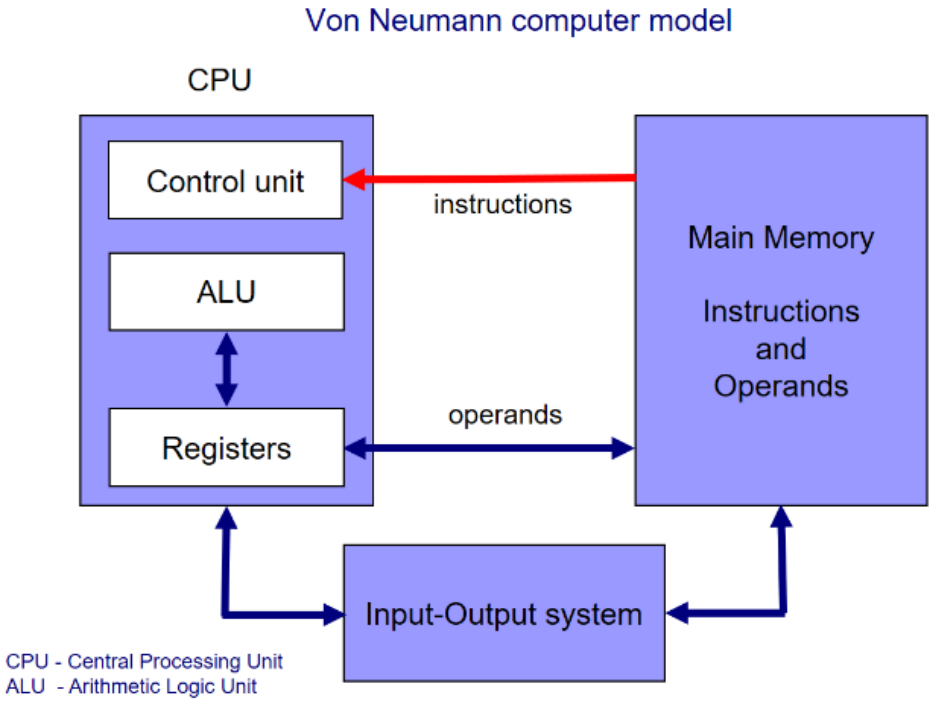
- Central Processing Unit
- Reads instructions from the main memory and executes them
- CPU consists of:
  - **Control unit** – Fetches the instructions and operands; activates the operations and devices specified by the instructions.
  - **ALU** – Arithmetic Logic Unit – Performs algebraic and logic operations (addition, and, or…)
  - **Registers** – Memory cells which serve to store values. Can be classified as:
    i. Program Inaccessible – necessary for CPU
    ii. Program Accessible (architectural) for storing operands, represent a small and fast memory in the CPU

b. Main memory

- Made up of memory words, where every memory word has its own location address
- Instructions and operands are stored in the main memory
- Denoted as “Main”, it should be distinguished from other memory devices, such as caches and virtual memory

c. Input-Output (I/O) system

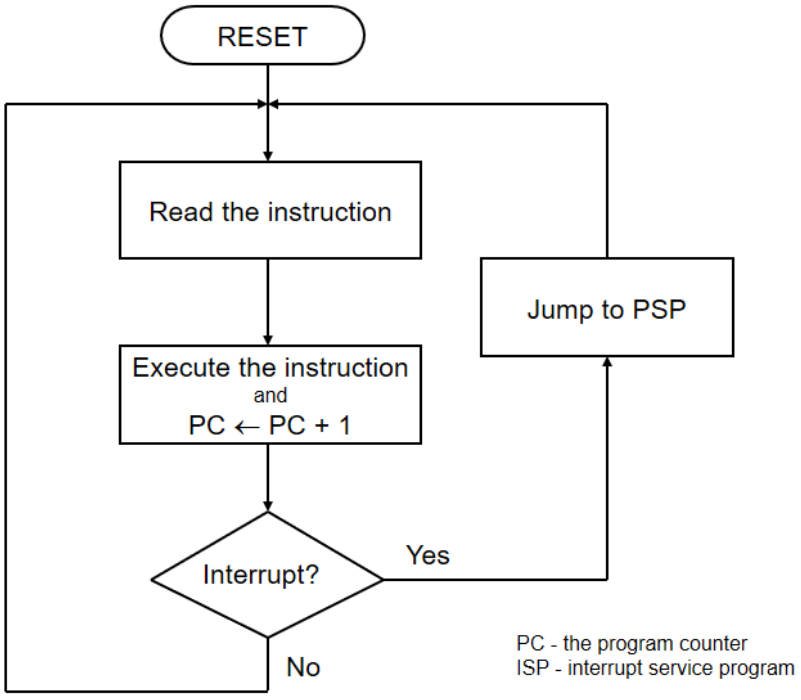
- Serves for the transfer of information between the machine and the outside world.
- An integral part of the I/O system are the I/O devices, which transform information to a form suitable for the user, or serve as a secondary memory


Von Neumann computer model
CPU
Control unit
ALU
Registers
instructions
operands
Main Memory
Instructions
and
Operands
Input-Output system
CPU - Central Processing Unit
ALU - Arithmetic Logic Unit

2. Operation of Von Neumann computer:
    - Its operation is controlled by machine instructions, read by the CPU from the Main memory, in consecutive order.
    - The machine instructions are stored in the memory one after another by increasing addresses
    - On start of the computer: the first instruction is usually read from a certain address in the memory, usually at the lowest or highest address

3. For each instruction, we distinguish:
    - Each instruction has two types of information:
        a. Information about the operation to be executed
        b. Information about the operands over which the operation is to be executed
    - Steps for execution of the instruction:
        a. FETCH
            - Read the instructions from the memory -> instruction fetch cycle
            - **Program Counter** – Special register that keeps the address of the next instruction that should be executed
                - Usually, the PC is increased by 1 (or more), except in cases where there is a Jump (conditions and cycles), or Interrupts (CPU starts another program – Interrupt Service Program (ISP))

        b. EXECUTE
            - Execution of the instruction – Execute cycle



4. Flynn's classification
   - Sequential instruction execution in the Von Neumann model is generally slow, therefore often it is sped up with extensions contained in Flynn's classification
   - Classification is made by two criterions:
     a. Number of instructions executed at the same time
     b. Number of operands that one instruction processes

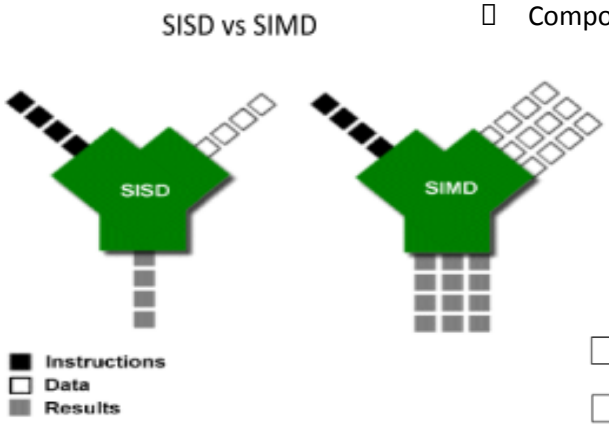
   - There are four groups:

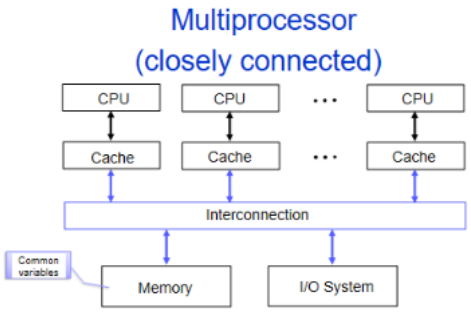
     a. **SISD** – Single Instruction Single Data
        - Classic Von Neumann computers without parallelism

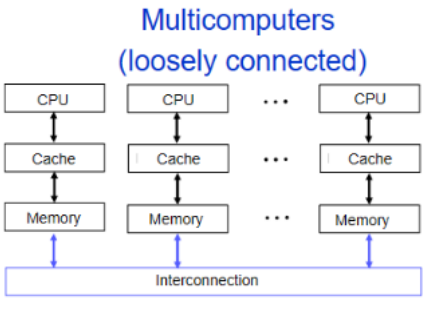
     b. **SIMD** – Single Instruction Multiple Data
        - Real vector computers
        - One instruction is executed over multiple data

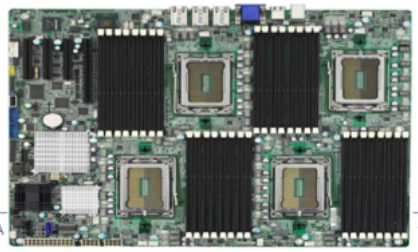
     c. **MISD** – Multiple Instructions Single Data
        - Unusual architecture, useful for reducing errors in computation
        - More instructions on a single operand

     d. **MIMD** – Multiple Instruction Multiple Data
        - Multiprocessor computers (Parallel computers)
        - Several instructions executed simultaneously, each on its operands
        - Composed of more CPUs that are interconnected

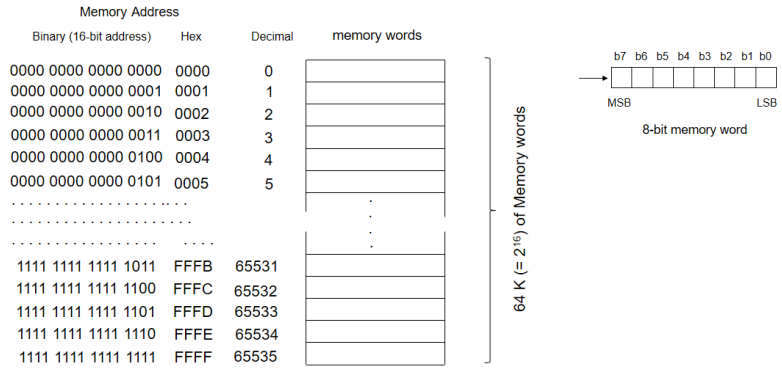




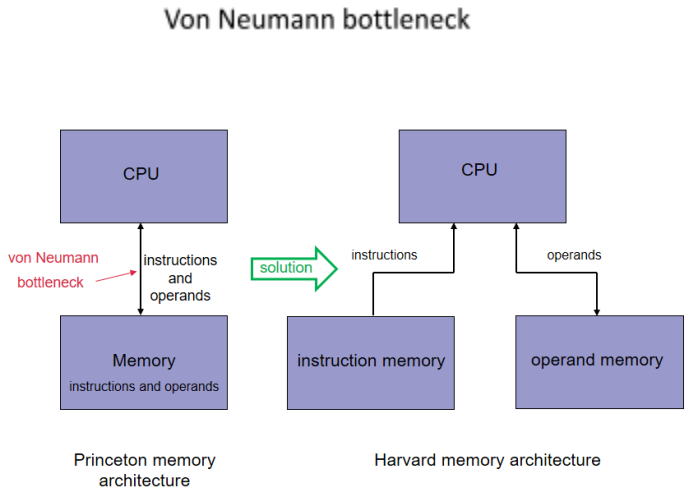




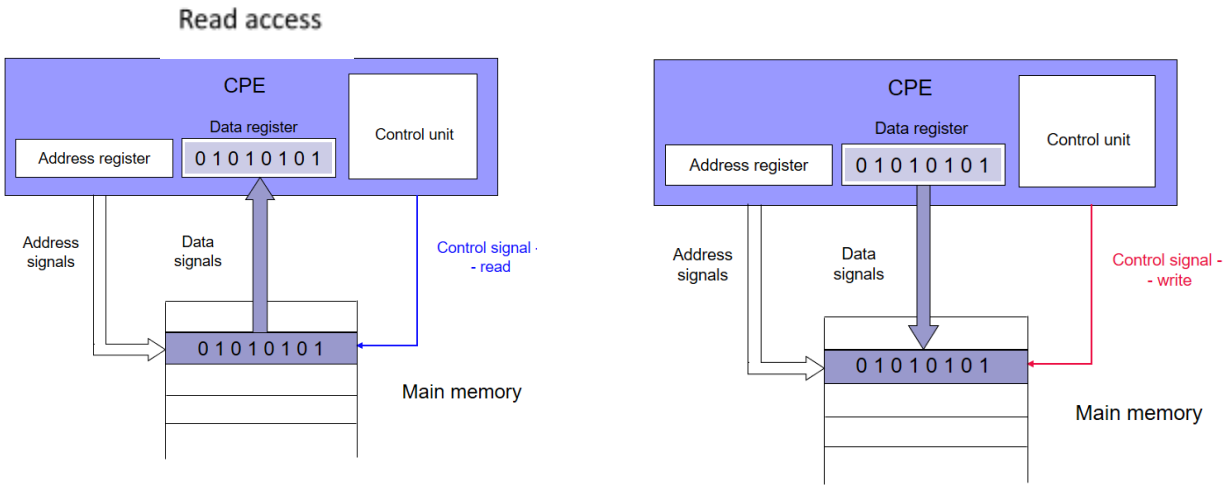
5. Main memory in Von Neumann based computers
    - It is a passive device used for storage of instructions and operands
    - The basic cell in the memory is called a **memory cell**, and it stores one bit of information (1 or 0)
    - More memory cells make up a **memory word**, which is the smallest number of bits that have their own **address**
    - The content of the memory words can change. In an n-bit memory word, we can store 2^n different contents (combinations).
    - Memory is a one-dimensional sequence of memory words
    - The memory address is a unique label for each memory word. It is unchangeable. The number of bits that compromise the address are denoted as **Address Length**.
    - The title length of the address is called an **address space**. It is the set of all the addresses, and it determines the maximum memory size.
    - Parts of the address space may be empty, as the main memory is usually smaller than the address space.
    - The prefixes for the memory units are always powers of two: $1K=2^{10}$, $1M=2^{20}$, $1G=2^{30}$…



6. Von Neumann **bottleneck**
    - It is the connection between the CPU and the main memory, where there is a lot of traffic produced. It is where both the instructions and operands are transferred.
    - One way to extend the bottleneck is to split the memory in two parts
    - Harvard memory architecture has two memories which operate simultaneously, one for instructions, and one for operands. It is used in cache memories in the lowest levels.
    - In the main memory, the Princeton architecture is used.



7. Access to memory
   - CPU accesses the memory by first sending its address to the memory, and the signal determines the direction of the transfer
   - There are two types of access:
     a. **Read access**: Main memory –> CPU (reading from the main memory, used approx. 80 % of the time)
     b. **Write access**: CPU –> Main memory (writing in the main memory, used approx.. 20 % of the time)



8. **Amdahl's law**
   - If the computer speeds up the operations except the portion of the operations that are not accelerated by a factor of N, then the increase in the speed of the entire computer, S(N) is:

$$S(N) = \frac{1}{f + \frac{1-f}{N}} = \frac{N}{1 + (N-1) * f}$$

f - the portion of operations that are not *accelerated* !

S (N) = Increase in the speed of the entire system
N = a scaling factor of the speed of (1 - f) portion of operations
f = portion of operations, which are not accelerated
1 - f = fraction of operations that are N times accelerated

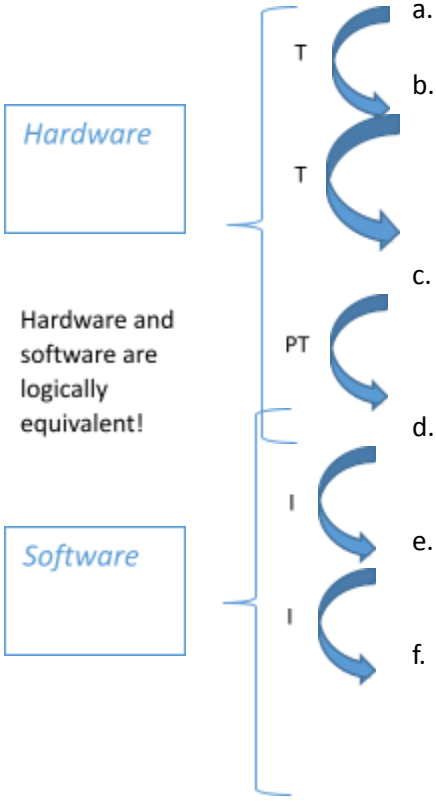
   - According to Amdahl, the computer is well designed (balanced), if it meets the following cases:

     a. **First Amdahl's rule**: The size of the main memory, in bytes, must be equal or higher to the number of instructions the CPU can execute in a second.

     b. **Second Amdahl's rule**: The performance of the I/O system, in bits per seconds, must be equal or higher to the number of instructions the CPU can execute in a second

9. The levels of the computer:
   - At each level, we see the computer through a different programming language, represented as the machine language of a certain virtual machine.
   - The simplest machine instructions are executed at the lowest level, by electronics.
   - Micro-programmed computer (6 levels) vs RISC Computer (5 levels)

   - The levels:



a. **Level 0**: Digital electronics
   - made of logic gates, executes the simplest machine instructions

b. **Level 1**: Micro-program language
   - RISC computers don't have it.
   - Every instruction of "usual" machine language is executed as a sequence of micro-instructions
   - Computers with this level area called micro-programmed.

c. **Level 2**: Usual Machine Language (ISA – Instruction Set Architecture)
   - With the conventional machine language, the programmer has full control over all parts of the computers
   - At the early computers, this was the highest level of computers

d. **Level 3**: Operating system
   - Program that facilitates computer work, and it serves as an interface between the user and the computer hardware

e. **Level 4**: Assembly language
   - Programs in the assembly language must be translated to the language from level 3 or 2 before execution

f. **Level 5**: Higher programming languages
   - More common programming languages, such as Java, Python, C, C++…
   - Programs written in these language must be translated to the languages from levels 4 or 3

   - **Mechanisms of transition** from one language to another:
     a. **Translation** (Compilation) – Marked as T on the scheme above
     b. **Interpretation** – Marked as I on the scheme above
        - 89The difference between translation and interpretation is that in interpretation, the translated (compiled) program does not exist
     c. **Partial translation** – Marked as PT on the scheme above
        - Solution between translation and interpretation
        - Faster than interpretation, slower than translation

4. INSTRUCTIONS

1. The basic types of information on the computer are operands and **instructions**.

2. Different computers have different architectures, and therefore different machine instructions, therefore, the **computer architecture is determined by specifying a set of machine instructions**.

3. Instructions are executed by the CPU, and that can be done in two ways:
    a) With hard-wired logic (faster, but less flexible for later changes or additions of instructions- it takes whole new CPU to do this)
    b) Microprogramming (slower, but easier to modify, as change is only needed in the microprogram)

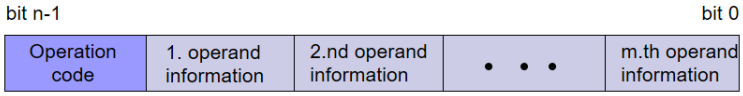
4. Each instruction contains information about:
    a) The operation which should be executed
    b) The operands on which the operation will be executed

This information is determined with bits in the fields of the instruction itself:

    a) Operation code: field that contains information about the operation
    b) Fields containing information on the operands: can contain the operand, or its address

For some instructions, the information about the operands is stored in the operation code.

5. The instruction format defines the division into fields for the information concerning the instruction. It depends on the number of operands, the number of registers in the CPU, the memory word size…

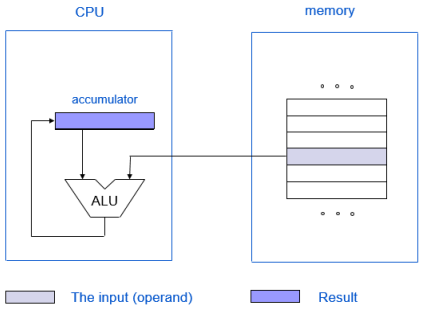


Instruction format of length *n* - bits
with *m* - explicitly defined operands

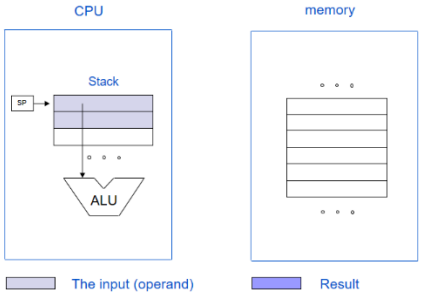
6. There can be more ways of executing the same operation, which leads to the number of instructions being greater than the number of operations.

7. Basic properties of the instructi ons:
    a) Ways of storing operands in the CPU
    b) Number of explicit operands in instruction
    c) Location of the operands and addressing modes
    d) Operations
    e) Type and length of the operands

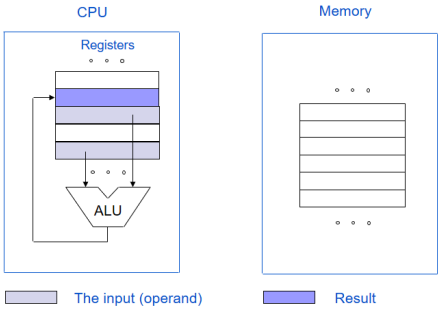
8. Ways of storing operands in the CPU:
    a) Accumulator – one accessible register in the CPU
        - Holds only one operand
        - In most cases, the result is also stored in the accumulator
        - Instructions: Load, Store (transfer Memory-ACC)
        - Shorter instructions
        - Many transfers between the CPU and the main memory



    b) Stack – can also be in the CPU
        - Only the top location is accessible
        - LIFO – Last in first out: New information always stored on top, loads only from the top of the stack
        - Instructions: Push, Pull/Pop



    c) Set of registers-in the CPU
        - All are accessible by their unique address
        - Faster
        - Shorter instructions
        - Reduced transfer between CPU and main memory
        - Two types of registers:
            - General purpose registers – All registers are equivalent
            - Base or index registers – Calculating addresses



9. Number of explicit operands in the instruction:
    a) Small
        - Shorter instructions = less memory space
        - Less powerful instructions
    b) Large
        - Longer instructions
        - More powerful instructions
10. Most elementary operations have up to three operands: 2 input operands and a result. Computers that use instructions with at most m operands are called m-operand (m-address) computers. They can be:
    a) 3+1 operand computers

    $OP3 \leftarrow OP2 \oplus OP1$ ($\oplus$ means any operation on the two operands)
    $PC \leftarrow OP4$

    b) 3-operand computers

    $OP3 \leftarrow OP2 \oplus OP1$

    $PC \leftarrow PC + 1$

    c) 2-operand computers

    $OP2 \leftarrow OP2 \oplus OP1$
    $PC \leftarrow PC + 1$

    d) 1-operand computers

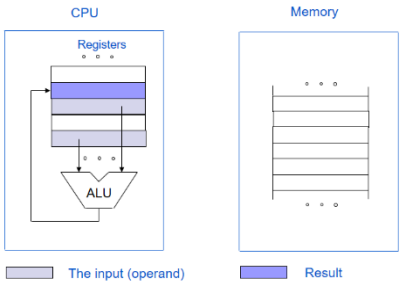
    $AC \leftarrow AC \oplus OP1$ (AC is abbreviation for accumulator)
    $PC \leftarrow PC + 1$

    e) No-operand computers (Stack computers)

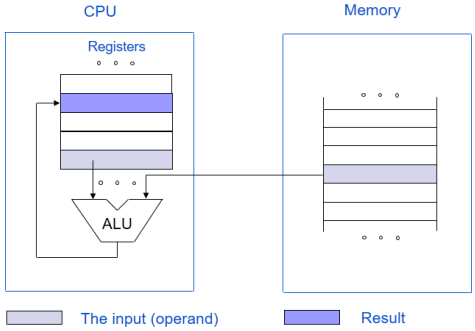
    $Stack_{TOP} \leftarrow Stack_{TOP} \oplus Stack_{TOP-1}$

    $PC \leftarrow PC + 1$
11. Operands can be stored in:
    a) Registers in the CPU
        - Register operands
        - Address given by instruction
    b) Consecutive memory word in the main memory
        - Memory operands
        - Complex addressing
    c) Registers in controllers for I/O devices
    d) Embedded in the instructions (immediate operands)
        - Available immediately

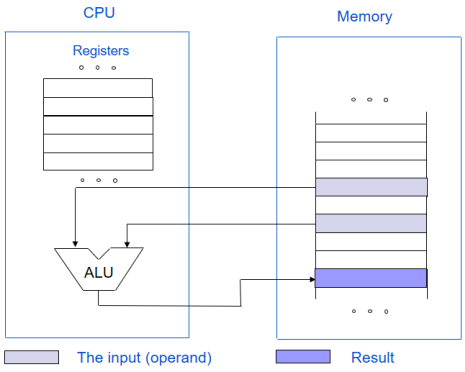
12. Considering the location of the operands, computers can be:
    a) Register-register computers
        - All the operands for the ALU instructions are stored in the CPU registers
        - Uses instructions LOAD and STORE for access to the main memory
        - Execution time for the ALU instructions is always the same



    b) Register-memory computers
        - One operand is either in the memory or the register, and the others are all in the registers
        - ALU instructions can use memory operands without transferring them to the registers with the LOAD instruction
        - Longer instructions, but shorter programs
        - Execution time depends on the operand location.



    c) Memory-memory computers
        - Every operand can be in the memory or the registers
        - Complex instructions
        - Longer execution time



13. Addressing modes - used to address the operands:

    a) Immediate addressing
        - The operand is embedded in the instruction and represented by its value. So, it is called literal or immediate operand
            - MOV r0, #128

    b) Direct addressing
        - The operand is represented by its address and is located in
            - Register
                - ADD R3, R2, R1
            - Memory location
                - LOAD r1, 20512

    c) Indirect addressing
        - Operand is represented by some other value or address - mediator
        - Mediator can be :
            - Register
                - LDR R3, [R2]
            - Another memory location
                - LOAD r1, @(20512)
        - Useful for processing arrays
        - Variants:
            - Base addressing
                - Address of the base register Rb, and of the offset D are given
                - Address of operand = Rb+D
            - Indexed addressing
                - Address of index register = Rx, offset D1 equal to the sum of the content of the base register and offset D
                - Address of operand = Rx+D1 = Rx+Rb+D
            - Auto-indexing addressing
                - Pre-decrement addressing
                - Post-increment addressing
            - PC-relative addressing
                - PC is used as the base register

14. Number of operations implemented by instructions on the computer:
    a) Instruction set should be complex enough
    b) The operations should be similar to the already existing common types of operations

15. Basic groups of operations (instructions):

a) Arithmetical and logical operations
- Can be scalar and vector
- Arithmetic – focused on integers
- Logical – Booleans
  - Setting, clearing individual bits in register
  - Shifts (in bits) – division by 2, multiplication by 2…

b) Data transfer
- Transfer data from one part of the computer to another
- Duplication of operands
- LOAD, STORE, MOVE, PUSH, POP…

c) Control operations
- Usual flow: Program counter PC = PC+1
- 3 types of unusual change of PC:
  - Conditional jumps (if-else, loops)
  - Unconditional jump
  - Call and return from procedures or subprograms

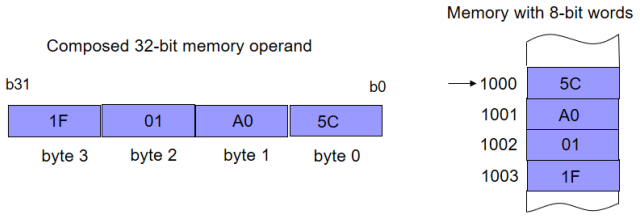
d) Floating point operations
- Floating Point Unit – FPU using arithmetic-logical operations on FP values
- Also: Square root, logarithm, exponential and trigonometric functions

e) System operations
- Privileged instructions
- Change the parameters of the computer's operation, and monitors its behavior
- Influence on: interrupts, traps, halting CPU, operation of cache, operation of virtual memory…

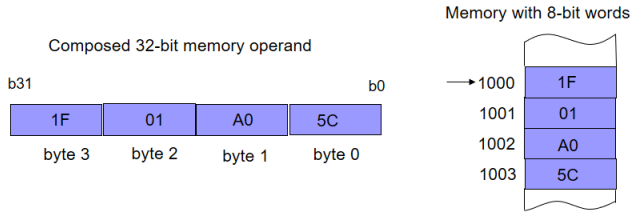
f) I/O operations
- Instructions for input/output transfers, or transfers between I/O devices and the CPU or the main memory
- Usually privileged

16. Types of operands:
    a) Bit
    b) Character – 8 or 16 bits
    c) Integer – 8, 16, 32 or 64 bits
    d) Real number – Floating point numbers, - 32, 64 or 128 bits
    e) Decimal number – String of 8 bit characters

17. Operands longer than 8 bits occupy more memory words, and are called composed memory operands. The words must be sequential, and are organized by:
    a) Little Endian Rule



    b) Big Endian Rule



18. Alignment:
    a) If operands are on natural (aligned) addresses, accesses can be faster.
    b) A is called a natural (aligned) address if :
        - A mod s = 0,
            - where A is an address, and s is the number of memory words in operand

19. The instruction format is influenced by:
    a) The size of the memory word
    b) The number of explicit operands in the instruction
    c) The type and number of registers in the CPU
    d) The size of the memory address

20. Orthogonal instructions:
    a) Information about the operation is independent of the information about the operands
    b) Information about one operand is independent of the information about the other operands

21. Computers according to the number of instructions:
    a) RISC – Reduced Instruction Set Computer
    b) CISC – Complex Instruction Set Computer

Historical evolution:

- Reasons to increase the number of instructions (up to 1980):
  - **Semantic gap** – the difference between how programmer in a higher-level programming language sees the computer in comparison to a programmer in machine language.
  - **Microprogramming** – allows simple addition of new instructions
  - **The ratio between the speed of the main memory and the CPU** – in the years 1960 to 1980, the speed of access to information in the CPU (to microinstructions) was more then 10-times higher than access to main memory.
- Reasons to decrease the number of instructions (after 1980):
  - **Difficulties of using complex instructions in compilers** – its better that the architecture enables simple elements for solving problems than solutions that are complex and often useless
  - **Change in the ratio between the speed of the main memory and the CPU** – microprogram solutions have become slower compared to fixed wired solutions, and complex instructions difficult to realize in a fixed wired logic; caches have bridged the speed gap between CPU and main memory.
  - **The introduction of parallelism in the CPU** – the realization of the pipelines is easier for simple instructions than for complex

Definition: A computer has a RISC architecture, if it meets the following six criterions:

1. Most of the instructions are executed in one clock cycle
2. Register-register design (load/store computer)
3. Instructions are executed with fixed wired logic and not microprogramed
4. Small number of instructions and addressing modes
5. All instructions have the same length
6. Good compilers (take into account the structure of the CPU)

5.OPERANDS

- Characters
  - BCD alphabet
  - EBCDIC alphabet
  - ASCII alphabet
    - Basic 7 bit
    - Extended 8 bit
  - UNICODE alphabet
    - UTF-8
    - UTF-16
- Numbers in fixed-point format (unsigned, signed – two's complement)
  - Unsigned numbers(0 <= x <=2^n -1)
    - Carry : Adding or subtracting unsigned numbers outside of range,there is carry from the highest bit
  - SIgned numbers
    - Sign and magnitude
      - The process of conversion from the decimal number into an n-bit binary number
        - Convert a number in binary to n-1 bits 2.
        - The highest bit is set according to the sign
    - Offset
      - Process conversion from the decimal number into an n bit binary number
        - Add the offset to the value
        - Convert like unsigned number
      - The conversion process from n-bit binary number in decimal number
        - Convert like unsigned number Subtract offset to get value
    - Ones' complement(for negative numbers)
      - The process of conversion from the decimal number into an n-bit binary number
        - Convert as unsigned number
        - If the number of negative, negate ("invert") all bits
    - Twos' complement(for negative numbers)
      - The process of conversion from the decimal value into an n-bit binary number
        - Convert as unsigned number
        - If the number is negative, invert the bits and add 1

- The process of conversion from n-bit binary presentation to the decimal value
  - If the presentation is negative, invert the bits and add 1
  - Converts as unsigned number (add a sign)

Overflow : If the result is outside the range that is presentable in two‘s complement

- Arithmetic with numbers in fixed-point
  - Arithmetic - four basic operations: addition, subtraction, multiplication and division
  - Arithmetic operations are executed in the arithmetic-logic unit (ALU), which is part of the CPU.
  - The basic element, with which we build n-bit adder, is 1-bit full adder.
    - 1-bit full adder has three inputs
      - two summands x_i and y_i
      - input carry c_i
    - and two outputs
      - sum s_i
      - output transfer c_(i+1)
- Real numbers and floating point
  - The general form -> m * r^e -> ex. : 0,03200000 * 10^8
    - m - mantissa (significand, fraction) = 0.03200000
    - r - base (radix) = 10
    - e - exponent = 8
- Arithmetic with numbers in floating point
  - Arithmetic in floating point has always been considered in computers separately from the fixed-point arithmetic
  - Basic differences with respect to the fixed-point arithmetic operations are:
    - The operations should use in addition to the mantissa also exponent - these operations requires arithmetic in fixed-point arithmetic
    - Rounding - the result of the operation should be the mathematically correct values, which are then rounded to the length of mantissa
    - When the result of floating point operations, in addition to the overflow, also underflow can occur
- Supplementing the IEEE Standard 754-2008
  - The most important additions:
    - Two new binary format with a base r = 2 ■
      - 128-bit format (quadruple precision) with 112-bit mantissa and 15-bit exponent. ■
      - 16-bit format (half-precision) with a 10-bit mantissa and a 5-bit exponent.
    - Two new decimal format with a base r = 10
      - 64-bit format: 16-digit mantissa (16 decimal digits)
      - 128-bit format: 34-digit mantissa