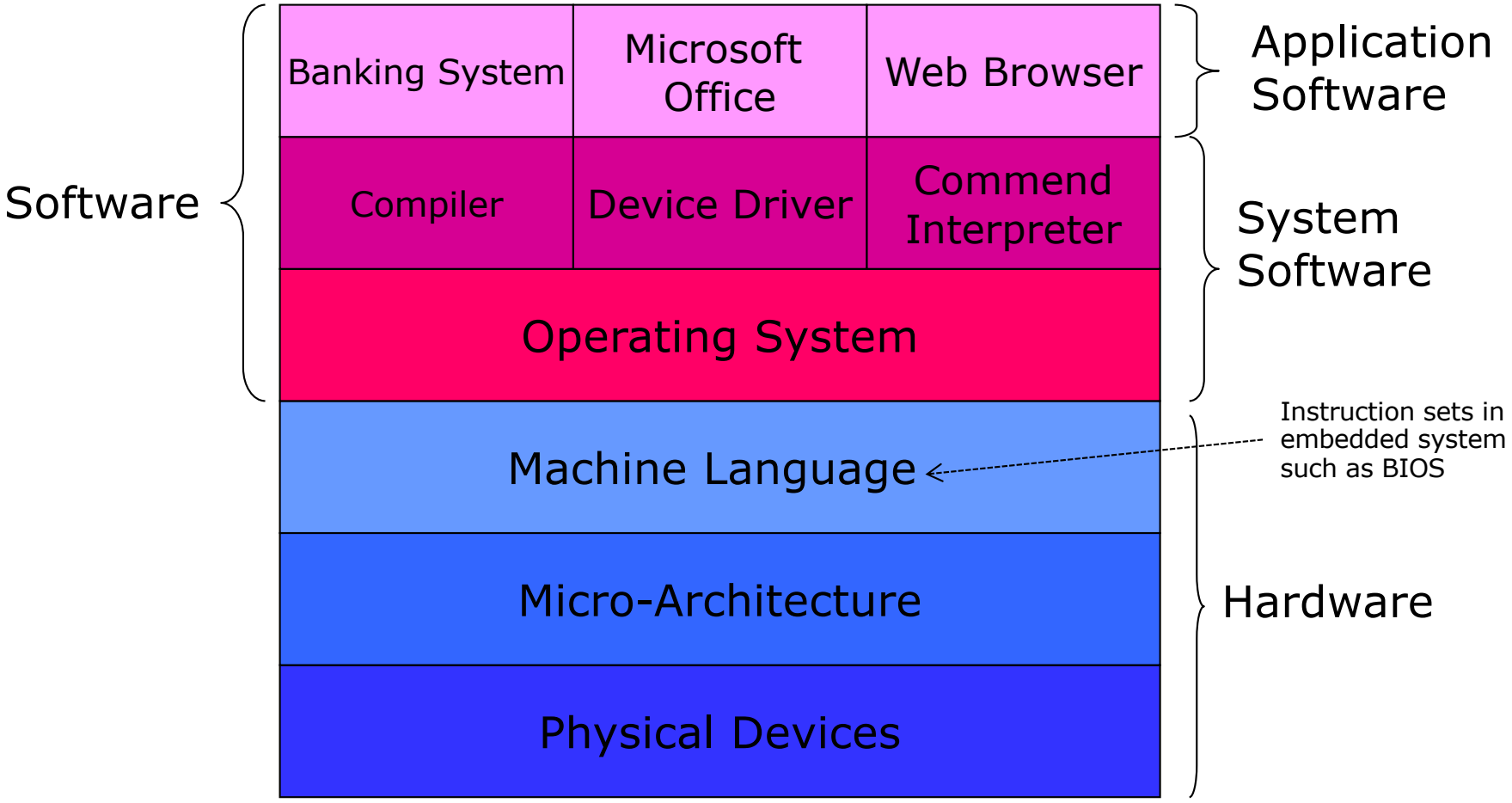
Preview

- Computer Systems
- Computer Structure
- Von Neumann Architecture

- Digital System
- What is Digital System?
- Why Digital System use Binary Number?
- Information Representation with binary number

Computer Systems

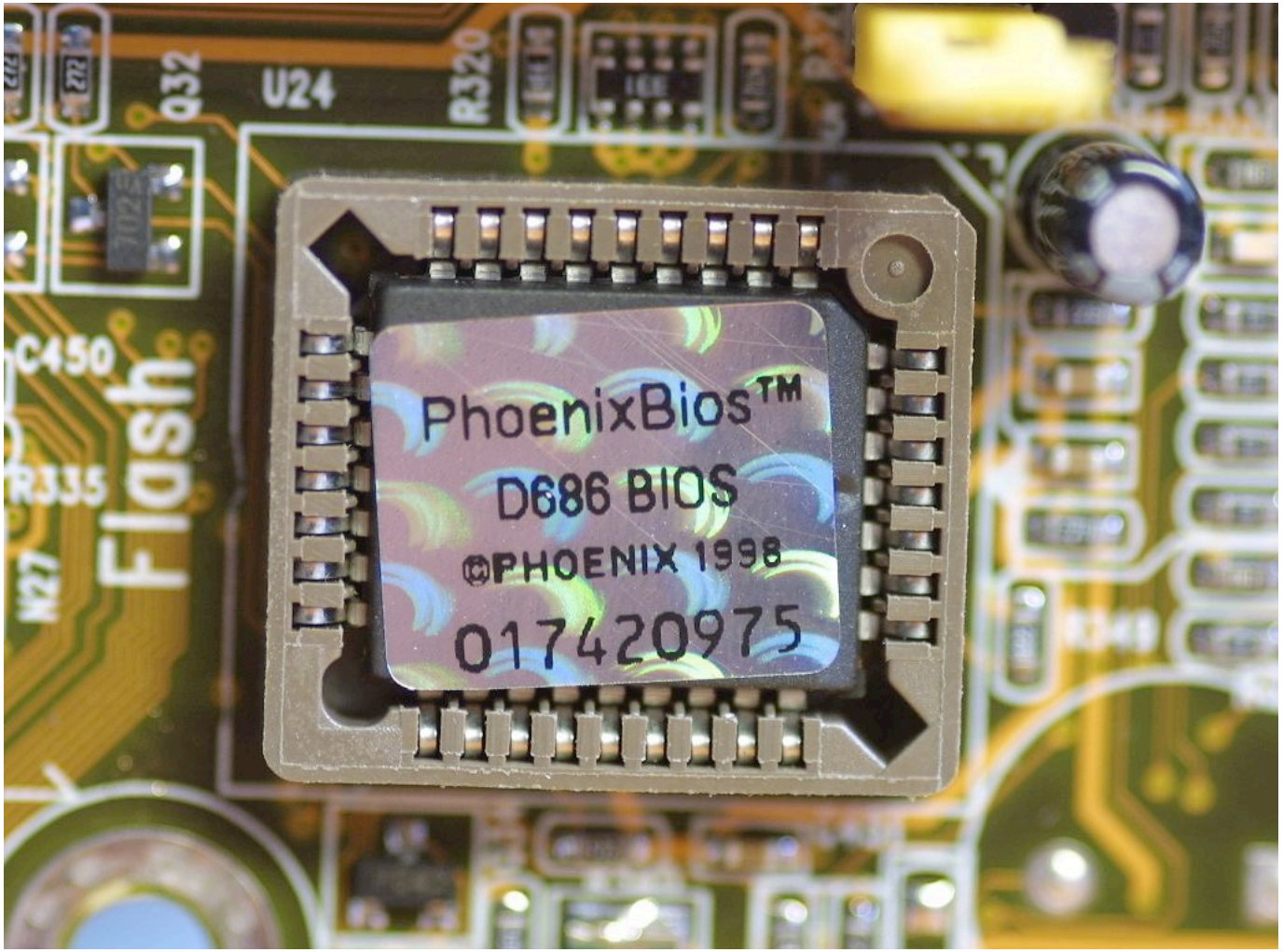


Computer Systems (Hardware)

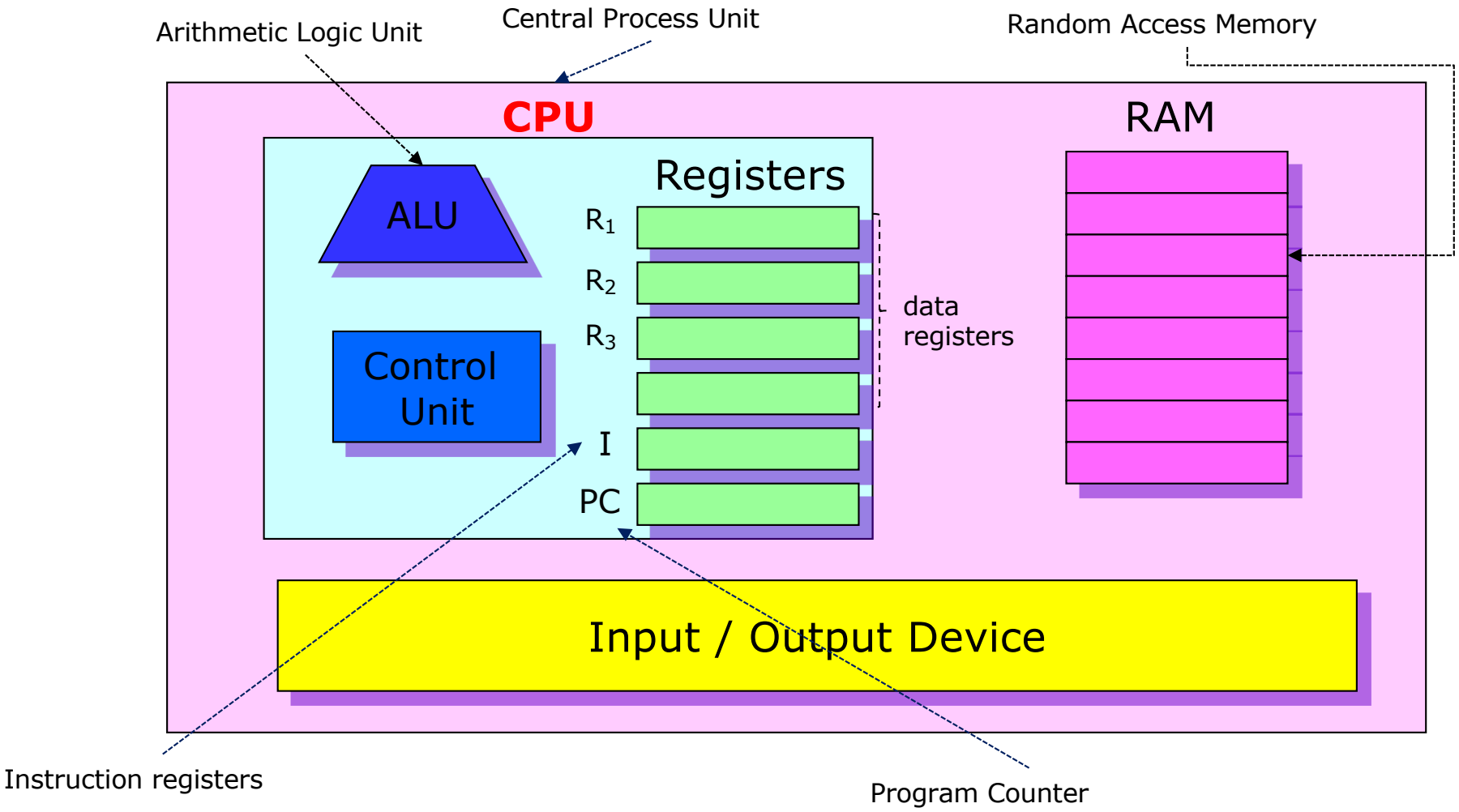
Hardware

- **Physical devices** – IC chips (CPU, Memory ..), wires, power supplies, Input Devices (keyboard, mice), Output Devices (printer, monitor)
- **Micro-architecture** – physical devices are grouped together to form a functional units
  - Ex) registers internal to CPU, a data path containing an ALU.
- **Machine language** – the purpose of the data path is to execute some set of instructions. There are typically 50 to 300 instructions in the system.

Computer Systems (Hardware)



Computer Systems (Hardware)



Computer Systems (Hardware)

Computer Systems (Hardware)

CPU (Central Processing Unit)

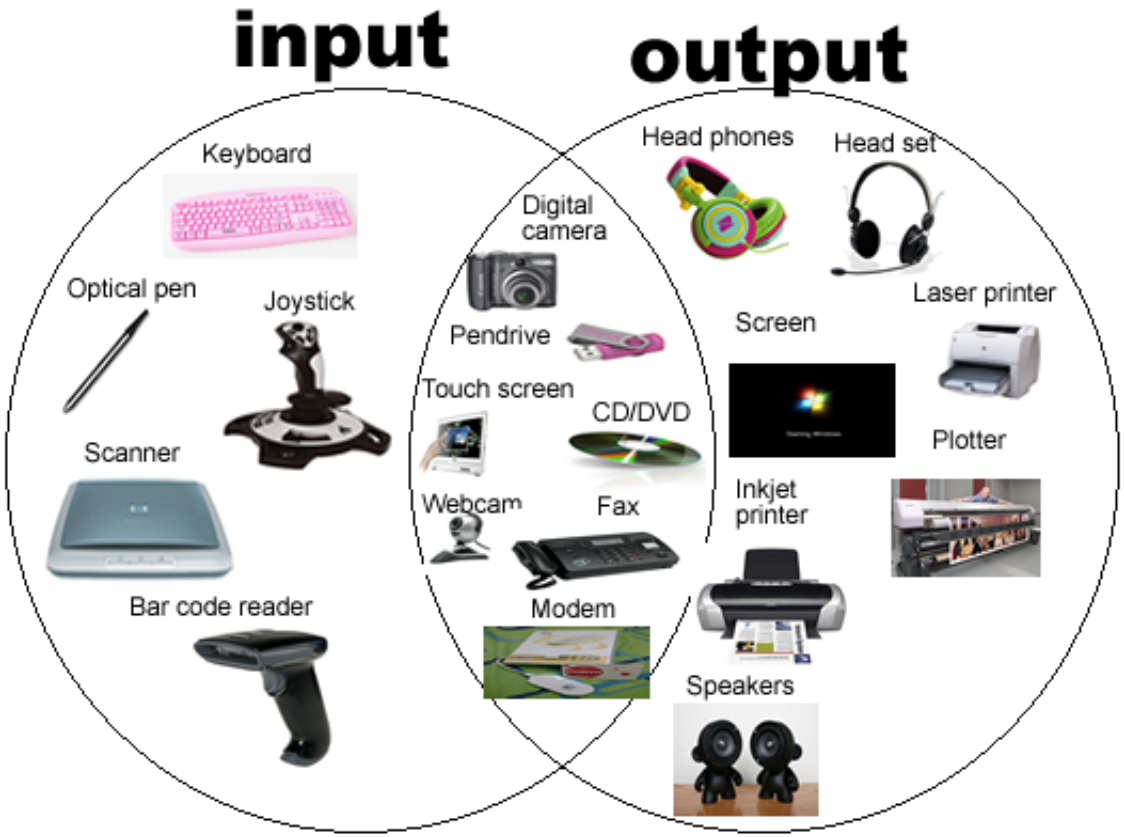
- **ALU** (Arithmetic Logic Unit) – performs arithmetic and logic operations.
  - Arithmetic operations – add, subtract, multiply, division.
  - Logic operations – NOT, AND, OR and XOR operation
- **Registers** –fast temporary storage
  - **Data Registers** (R_1, R_2, R_3 …) – hold variable data which are frequently used for calculation.
  - **Instruction Register** – The CPU is responsible for fetching instructions, one by one, from the memory, storing them in the instruction register, interpreting them and executing them.
  - **Program Counter** (PC) – keeps track of the instruction currently being executed.
- **Control Unit** – controls the operation of each part of body (Memory, ALU) in CPU and Input/output subsystems.

Computer Systems (Hardware)

- RAM (Random Access Memory)
  - RAM is a temporary space where data and instructions are saved for executing a program.
  - RAM's component is formed with capacitor (DRAM) or logic gates (SRAM)
  - A program must be loaded into RAM to execute.
  - Operating system control to execute each application programs.

Computer Systems (Hardware)

- Input / Output Devices



Computer Systems (Software)

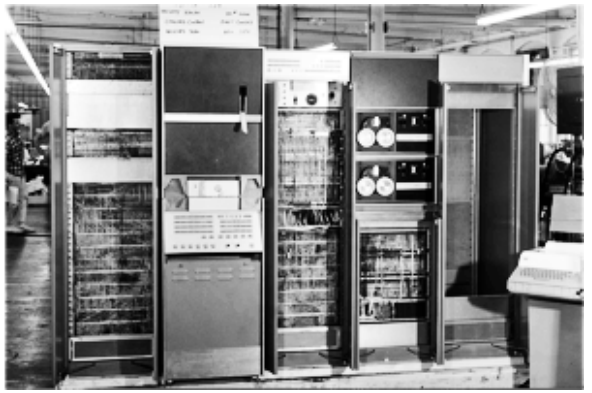
- System software is a software designed to operate and control the computer hardware and to provide a platform for running application software.
  - Operating System – Window 7, Window 8, Linux, Unix
  - Compiler – create a executable code
  - Device Drivers – device drivers for each input/output devices must be installed. Device drivers are interface between hardware and operating system.

Computer Systems (Software)

- Application software, also known as an application or an app, is computer software designed to help the user to perform specific tasks.
  - Online Banking System
  - Graphic System – Photoshop,
  - Office Applications – Microsoft Office
  - Game Software
- Application software are written by programming languages.

Designing Computers

- All computers more or less based on the same basic design, the Von Neumann Architecture!

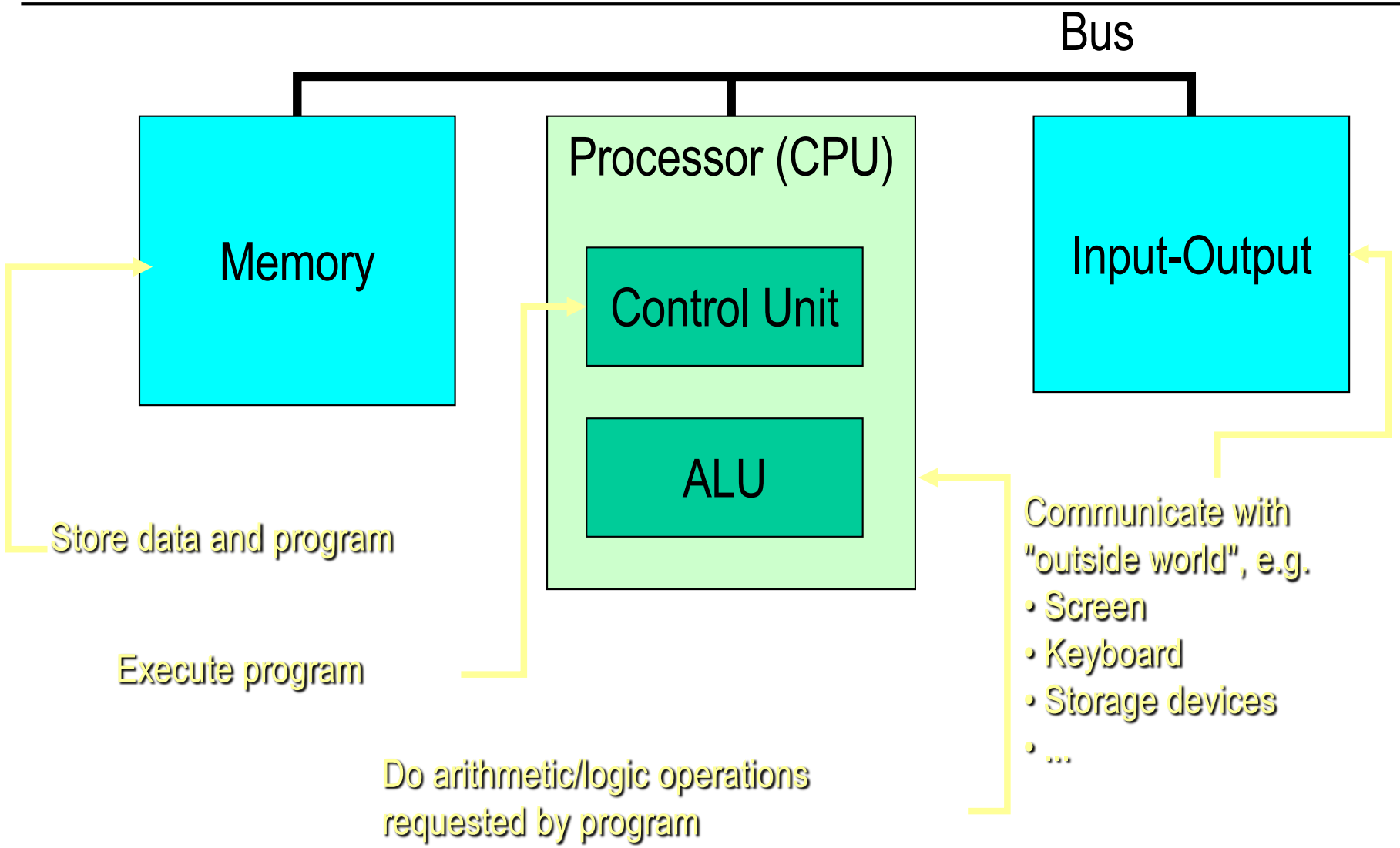


The Von Neumann Architecture

Model for designing and building computers, based on the following three characteristics:

1) The computer consists of four main sub-systems:
   - Memory
   - ALU (Arithmetic/Logic Unit)
   - Control Unit
   - Input/Output System (I/O)
2) Program is stored in memory during execution.
3) Program instructions are executed sequentially.

The Von Neumann Architecture



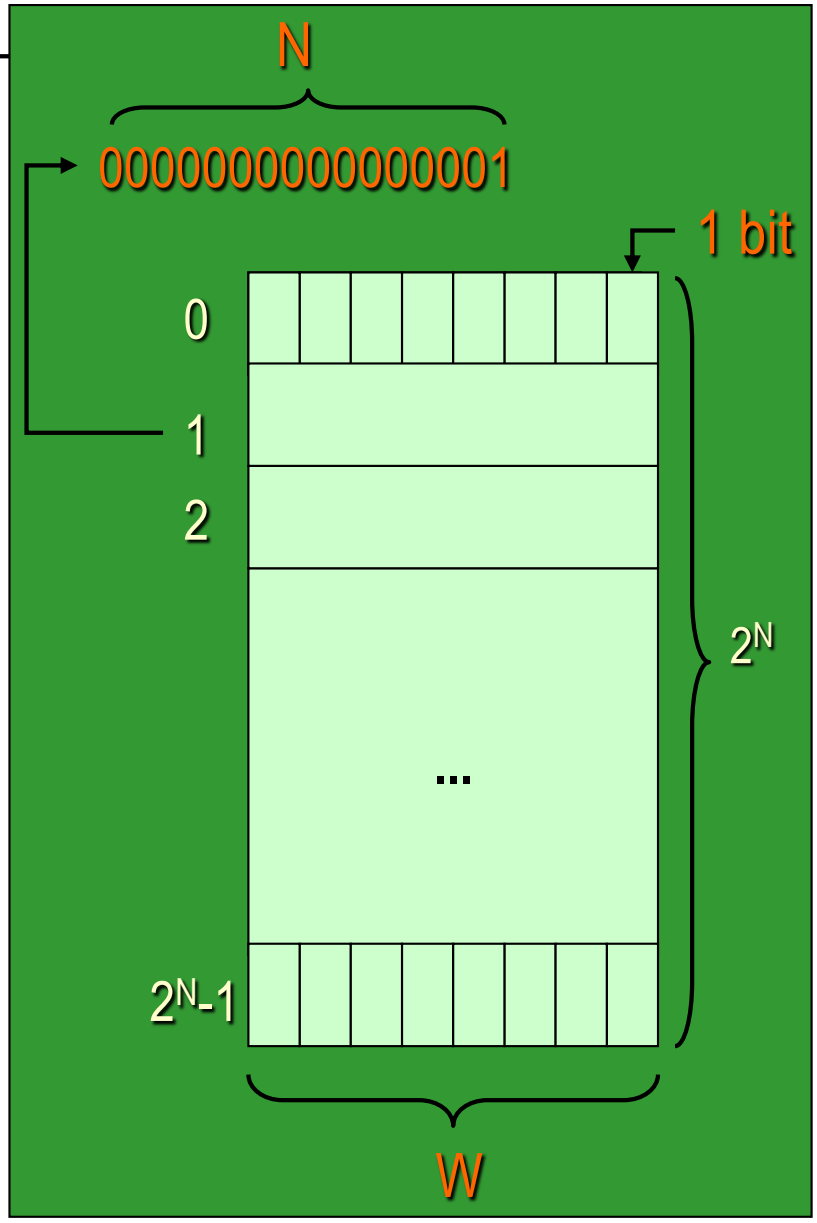
Memory Subsystem

- Memory, also called RAM (Random Access Memory),
  - Consists of many memory cells (storage units) of a fixed size. Each cell has an address associated with it: 0, 1, …
  - All accesses to memory are to a specified address. A cell is the minimum unit of access (fetch/store a complete cell).
  - The time it takes to fetch/store a cell is the same for all cells.
- When the computer is running, both
  - Program
  - Data (variables)

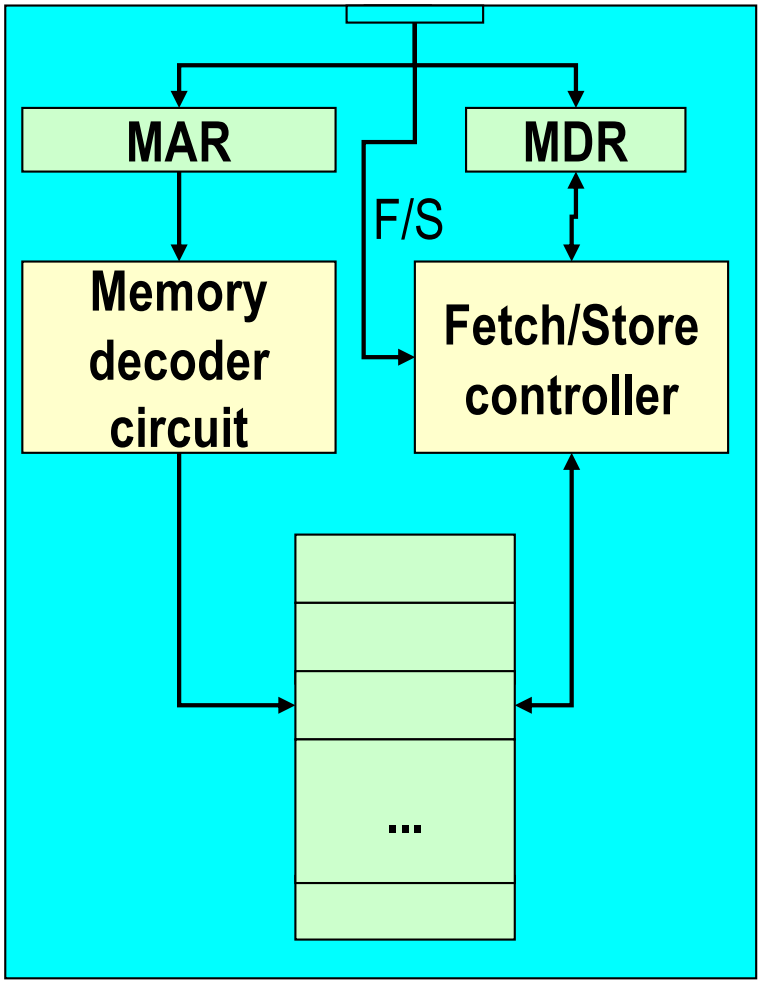
  are stored in the memory.

RAM

- Need to distinguish between
  - the address of a memory cell and the content of a memory cell
- Memory width (W):
  - How many bits is each memory cell, typically one byte (=8 bits)
- Address width (N):
  - How many bits used to represent each address, determines the maximum memory size = address space
  - If address width is N-bits, then address space is 2^N (0,1,...,2^N-1)



Structure of the Memory Subsystem

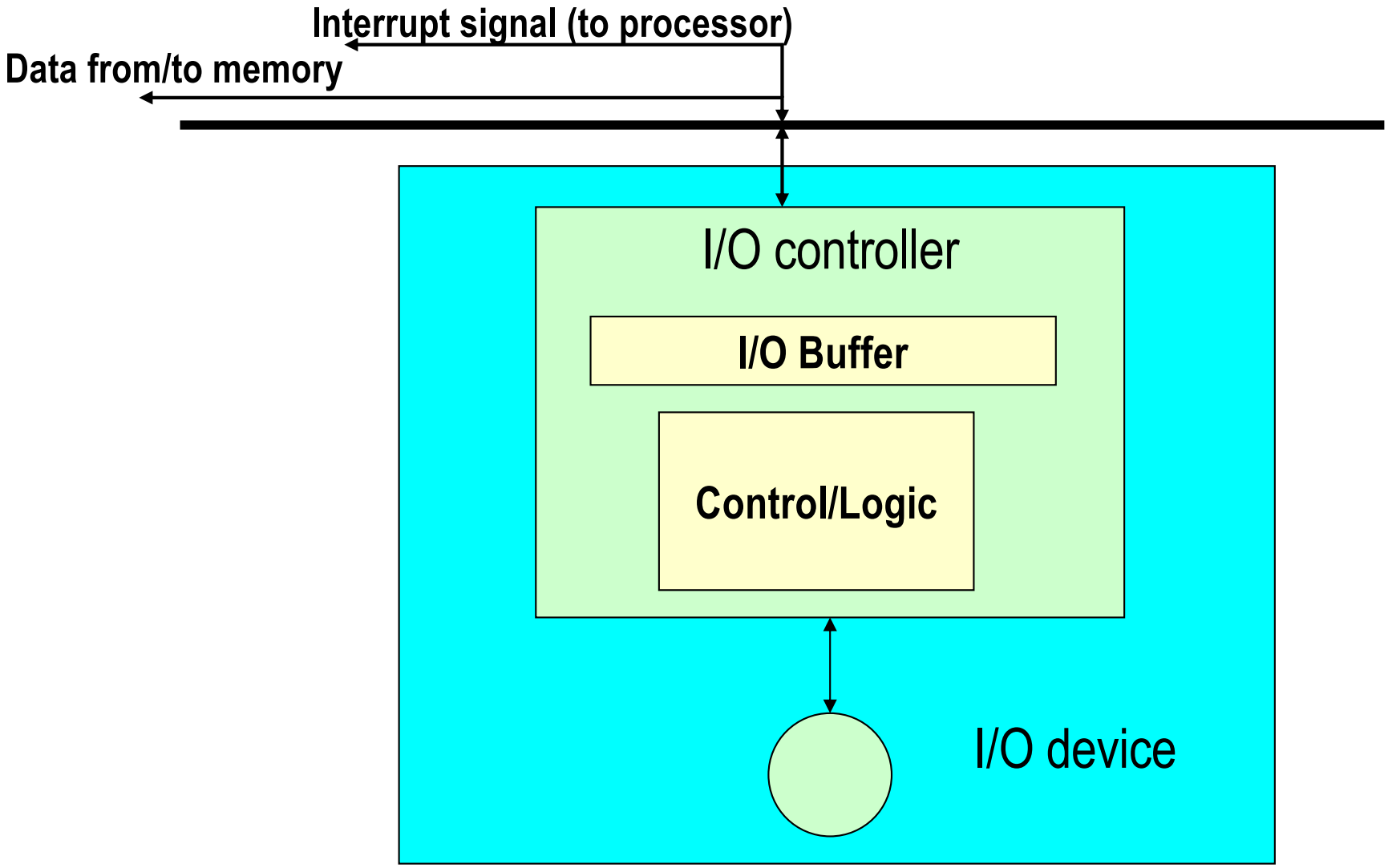


- Fetch(address)
  - Load address into MAR.
  - Decode the address in MAR.
  - Copy the content of memory cell with specified address into MDR.
- Store(address, value)
  - Load the address into MAR.
  - Load the value into MDR.
  - Decode the address in MAR
  - Copy the content of MDR into memory cell with the specified address.

Input/Output Subsystem

- Handles devices that allow the computer system to:
  - Communicate and interact with the outside world
    - Screen, keyboard, printer, ...
  - Store information (mass-storage)
    - Hard-drives, floppies, CD, tapes, …
- Mass-Storage Device Access Methods:
  - Direct Access Storage Devices (DASDs)
    - Hard-drives, floppy-disks, CD-ROMs, ...
  - Sequential Access Storage Devices (SASDs)
    - Tapes (for example, used as backup devices)

Structure of the I/O Subsystem

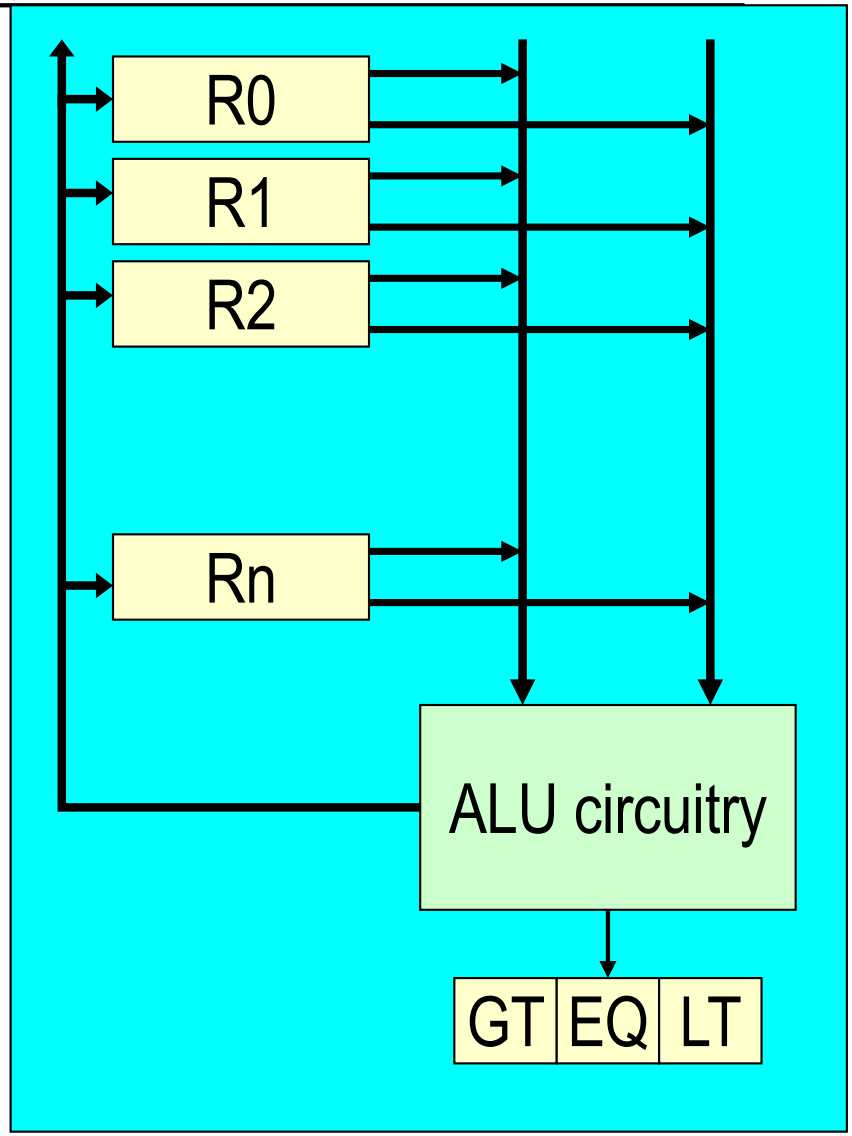


The ALU Subsystem

- The ALU (Arithmetic/Logic Unit) performs
  - mathematical operations (+, -, x, /, …)
  - logic operations (=, <, >, and, or, not, ...)
- In today's computers integrated into the CPU
- Consists of:
  - Circuits to do the arithmetic/logic operations.
  - Registers (fast storage units) to store intermediate computational results.
  - Bus that connects the two.

Structure of the ALU

- Registers:
  - Very fast local memory cells, that store operands of operations and intermediate results.
  - CCR (condition code register), a special purpose register that stores the result of <, = , > operations
- ALU circuitry:
  - Contains an array of circuits to do mathematical/logic operations.
- Bus:
  - Data path interconnecting the registers to the ALU circuitry.

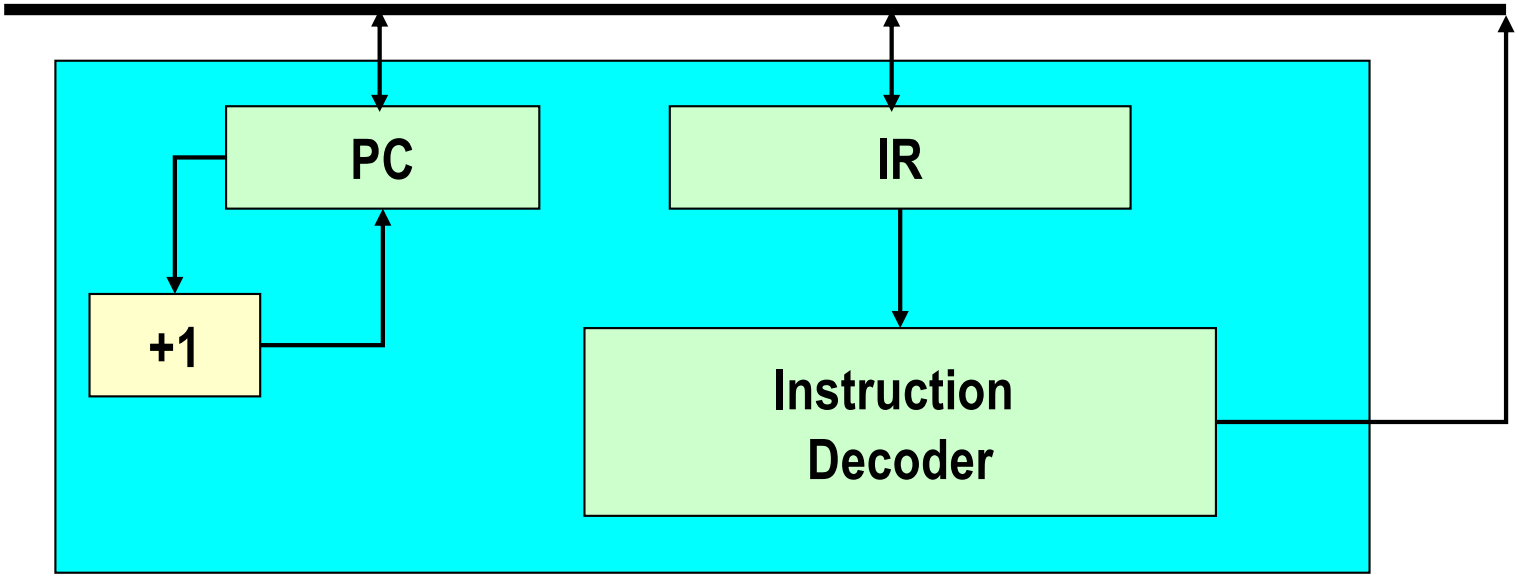


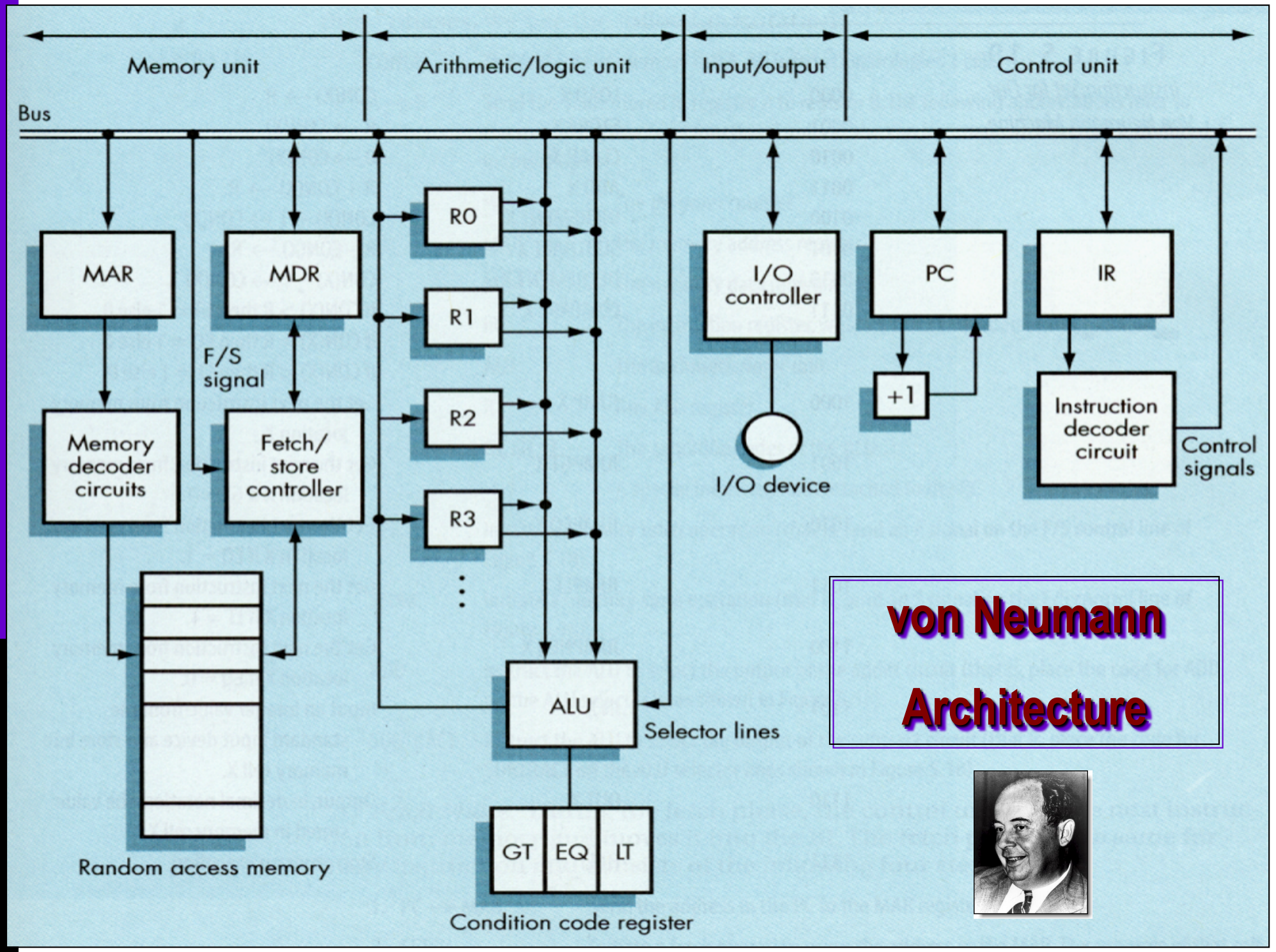
The Control Unit

- Program is stored in memory
  - as machine language instructions, in binary
- The task of the control unit is to execute programs by repeatedly:
  - Fetch from memory the next instruction to be executed.
  - Decode it, that is, determine what is to be done.
  - Execute it by issuing the appropriate signals to the ALU, memory, and I/O subsystems.
  - Continues until the HALT instruction

Structure of the Control Unit

- PC (Program Counter):
  - stores the address of next instruction to fetch
- IR (Instruction Register):
  - stores the instruction fetched from memory
- Instruction Decoder:
  - Decodes instruction and activates necessary circuitry




Memory unit
Arithmetic/logic unit
Input/output
Control unit
Bus
MAR
MDR
R0
R1
R2
R3
I/O controller
PC
IR
F/S signal
+1
Memory decoder circuits
Fetch/ store controller
Instruction decoder circuit
Control signals
I/O device
ALU
Selector lines
Random access memory
GT
EQ
LT
Condition code register
von Neumann Architecture

How does this all work together?

- Program Execution:
  - PC is set to the address where the first program instruction is stored in memory.
  - Repeat until HALT instruction or fatal error

    Fetch instruction

    Decode instruction

    Execute instruction

    End of loop

Program Execution (cont.)

- Fetch phase
  - PC --> MAR (put address in PC into MAR)
  - Fetch signal (signal memory to fetch value into MDR)
  - MDR --> IR (move value to Instruction Register)
  - PC + 1 --> PC (Increase address in program counter)
- Decode Phase
  - IR -> Instruction decoder (decode instruction in IR)
  - Instruction decoder will then generate the signals to activate the circuitry to carry out the instruction
- Execute Phase
  - Differs from one instruction to the next.

Instruction Set for Our Von Neumann Machine

Opcode	Operation	Meaning
0000	LOAD X	CON(X) --> R
0001	STORE X	R --> CON(X)
0010	CLEAR X	0 --> CON(X)
0011	ADD X	R + CON(X) --> R
0100	INCREMENT X	CON(X) + 1 --> CON(X)
0101	SUBTRACT X	R - CON(X) --> R
0101	DECREMENT X	CON(X) - 1 --> CON(X)
0111	COMPARE X	If CON(X) > R then GT = 1 else 0 If CON(X) = R then EQ = 1 else 0 If CON(X) < R then LT = 1 else 0
1000	JUMP X	Get next instruction from memory location X
1001	JUMPGT X	Get next instruction from memory loc. X if GT=1
...	JUMPxx X	xx = LT / EQ / NEQ
1101	IN X	Input an integer value and store in X
1110	OUT X	Output, in decimal notation, content of mem. loc. X
1111	HALT	Stop program execution

The Future: Non-Von Neumann Architectures

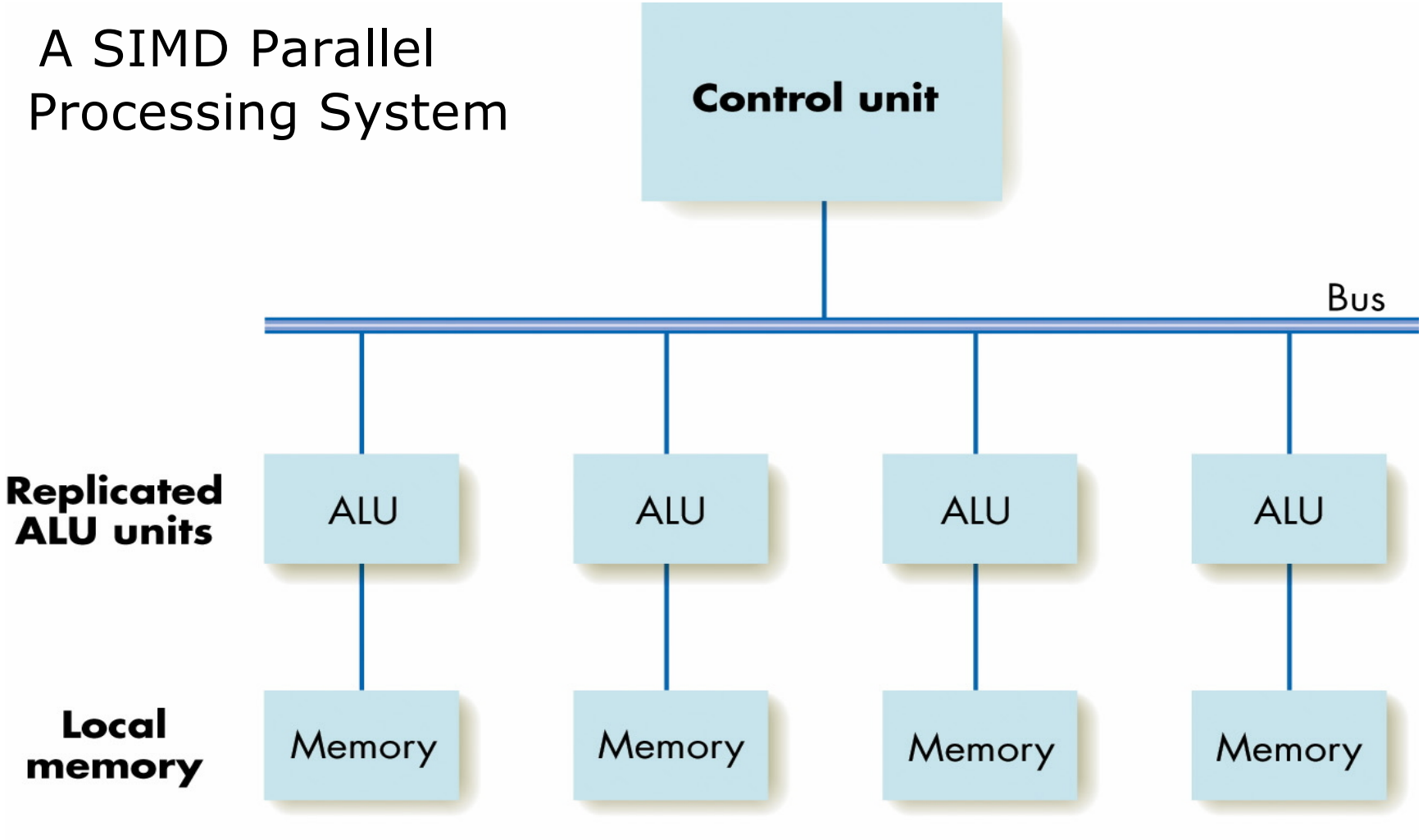
- Physical limitations on speed of Von Neumann computers
- Non-Von Neumann architectures explored to bypass these limitations
- Parallel computing architectures can provide improvements: multiple operations occur at the same time

The Future: Non-Von Neumann Architectures (continued)

- SIMD architecture
  - Single instruction/Multiple data
  - Multiple processors running in parallel
  - All processors execute same operation at one time
  - Each processor operates on its own data
  - Suitable for “vector” operations

The Future: Non-Von Neumann Architectures (continued)

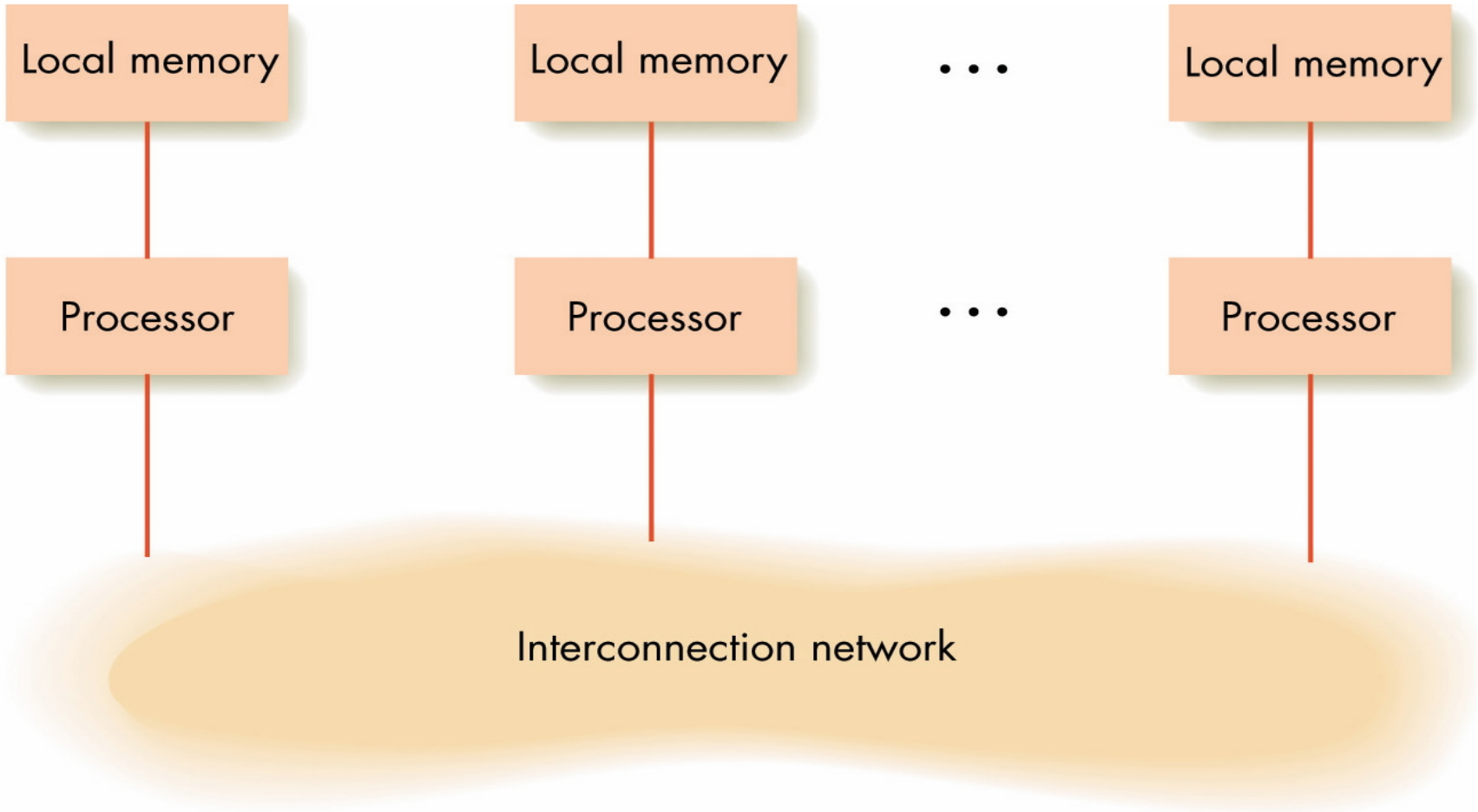
A SIMD Parallel Processing System



The Future: Non-Von Neumann Architectures (continued)

- MIMD architecture
  - Multiple instruction/Multiple data
  - Multiple processors running in parallel
  - Each processor performs its own operations on its own data
  - Processors communicate with each other

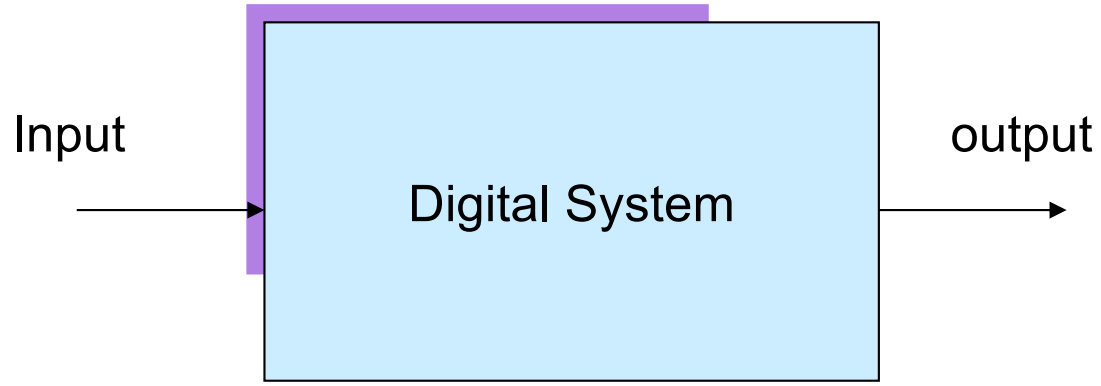
The Future: Non-Von Neumann Architectures (continued)



Model of MIMD Parallel Processing

Digital Computer and Information

Digital System?



- Digital System - device that can read, write, or store information that is represented in numerical form.

Digital Computer and Information

- Digital System use a physical quantities called signal to represent discrete information.- Voltage (usually use between 0 volt and 5 volt), Currents.

- How to represent discrete information?

Digital Computer and Information

- Analog representation: a quantity that can vary over a continuous range of values.
- Digital representation: a quantity that changes in discrete steps.
- Analog == Continuous
- Digital == discrete (step by step)

Digital Computer and Information

- Which of the following involves analog quantities and which involve digital quantities?

(a) Ten-position switch

(b) Current flowing out of an electrical outlet

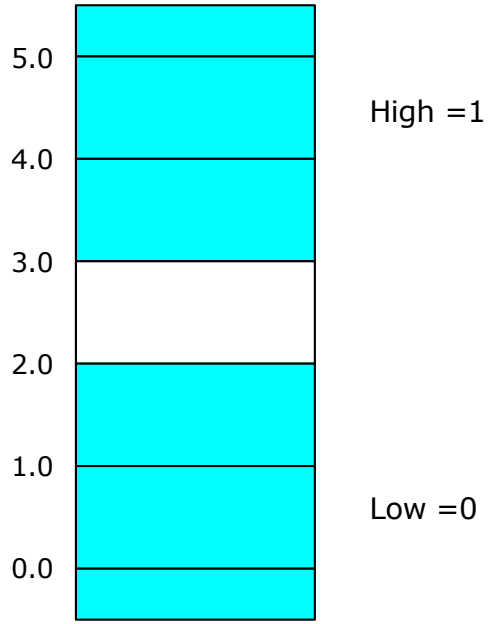
(c) Temperature of a room

(d) Sand grains on the beach

(e) Automobile speedometer

Digital Computer and Information

Ex) Two discrete information can be presented by the ranges of voltage values

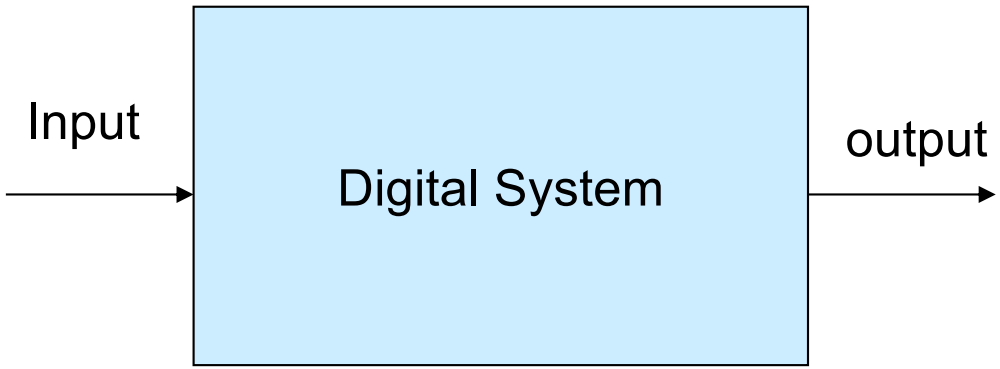


Input Voltage Range

output Voltage Range

Digital Computer and Information

- The Input ranges are usually longer than the output ranges, Why?
  - Since inputs are coming from outside through output line such as wire or wireless connection, input might be with noises !



Digital Computer and Information

- Why is Binary used?
- Is Possible with Decimal?

Ex) with Decimal

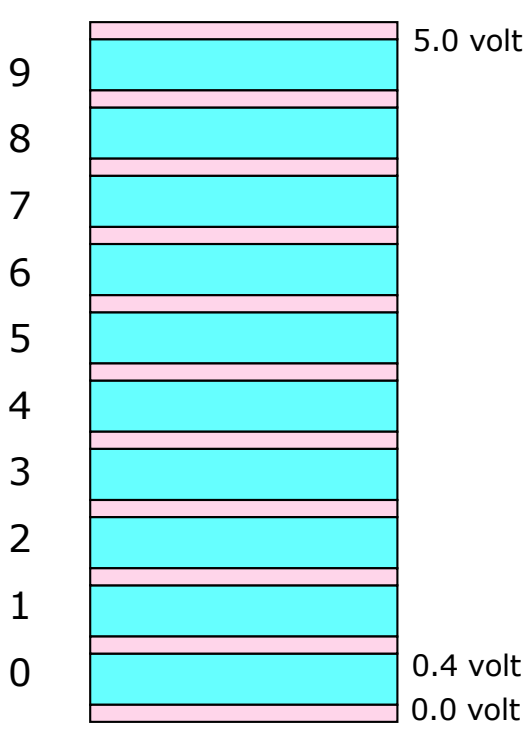
Approach 1)

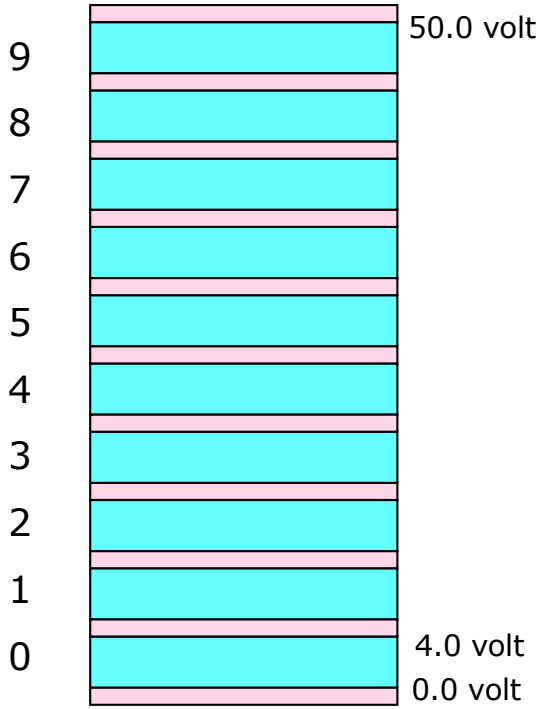
Divide voltage range 0 ~ 5.0 volt by ten to represent each ten digit number.

Approach 2)

Use high voltage 50 volt. Then divide voltage range 0 ~ 50.Volt by ten to represent ten discrete numbers.

Digital Computer and Information





Digital Computer and Information

- Approach 1 (use 5V with decimal)
  - Might possible to build a system (in future), but expect input/output error.
- Approach 2 (use 50V with decimal)
  - We can build a big size system since small digital component cannot stand for a high voltage
  - Small size CPU cannot stand for 50V.

Representing Binary Quantities

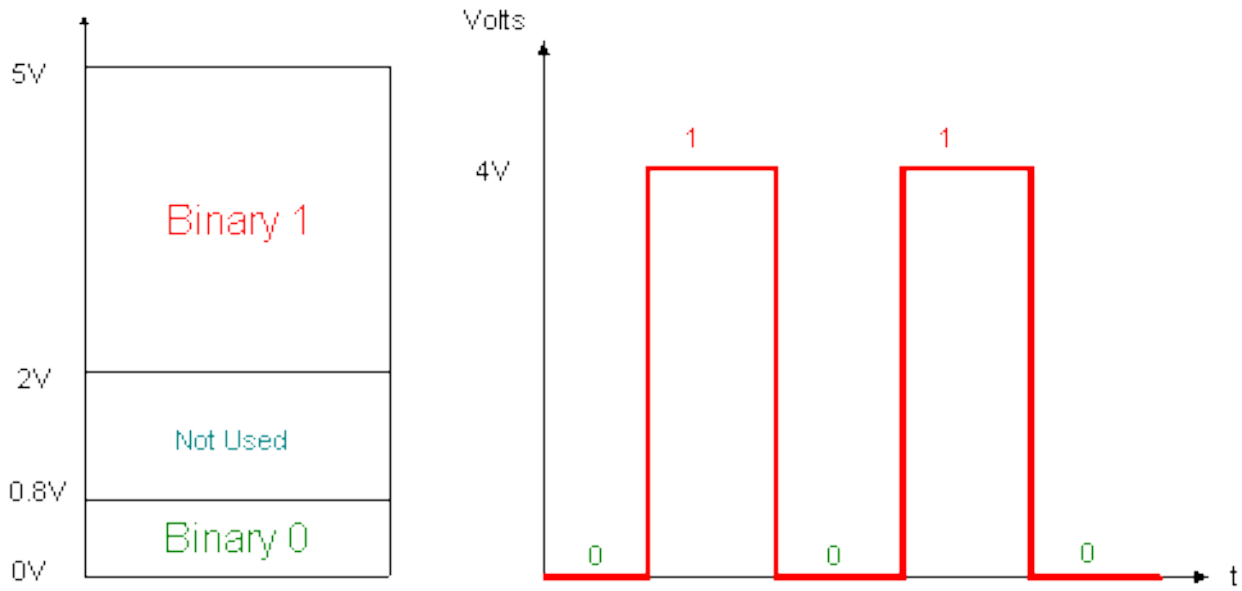
- In digital systems the information that is being processed is usually presented in binary form. Binary quantities can be represented by any device that has only two operating states or possible conditions.

- For example, a switch has only open or closed. We arbitrarily (as we define them) let an open switch represent binary 0 and a closed switch represent binary 1.

- Thus we can represent any binary number by using series of switches.

Typical Voltage Assignment

- **Binary 1**: Any voltage between 2V to 5V
- **Binary 0**: Any voltage between 0V to 0.8V
- Not used: Voltage between 0.8V to 2V, this may cause error in a digital circuit.



Information Representation with Binary Digits

- A binary digit is called a *bit*.
- Information is represented by a group of bit in the digital system
- With a bit how many discrete information can be represented?

  0 = FALSE = OFF

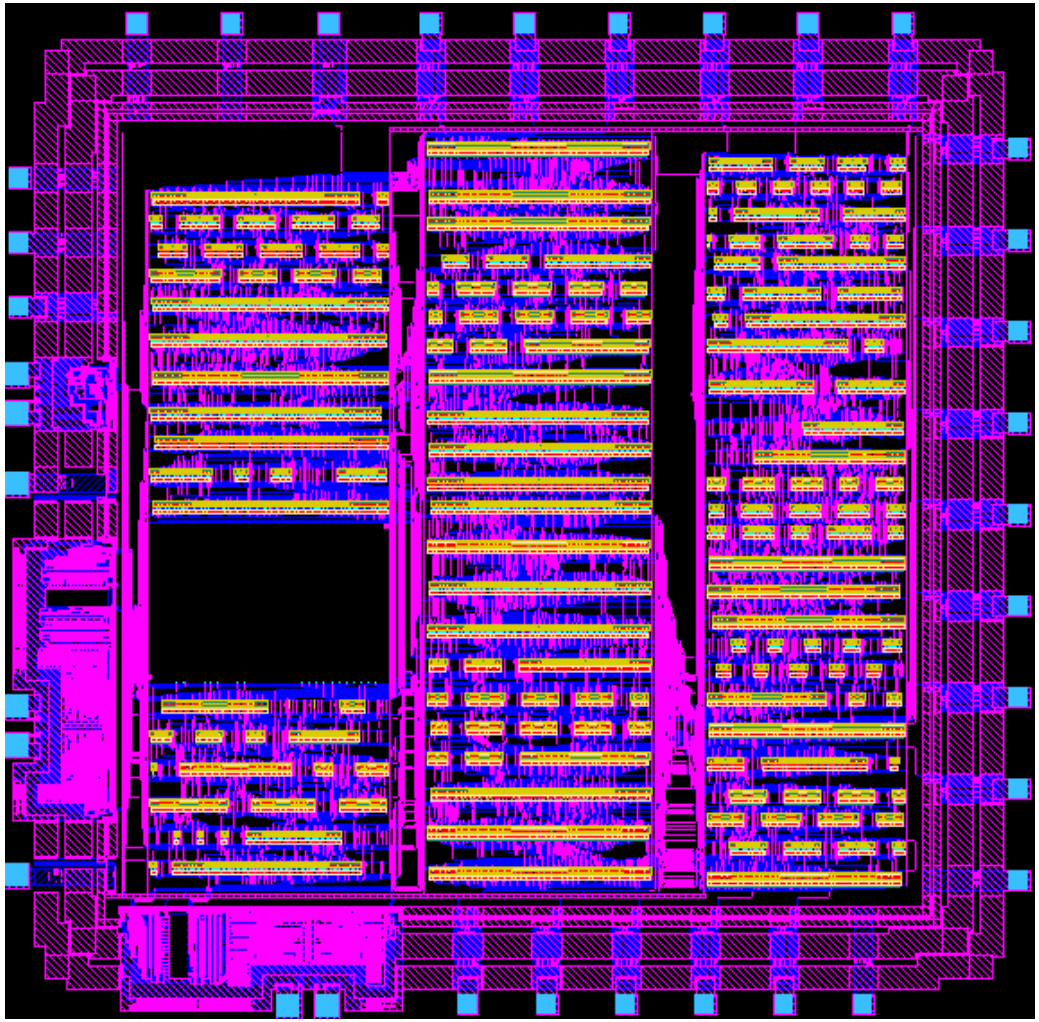
  1 = TRUE = ON
- With n bit how many discrete information can be represented?

  2^n discrete information

Information Representation with Binary Digits

- n = 2
  - 00, 01, 10, 11
- n = 3
  - 000, 001, 010, 011, 100, 101, 110, 111
- n= 4
  - 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111,
  - 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111

Digital System

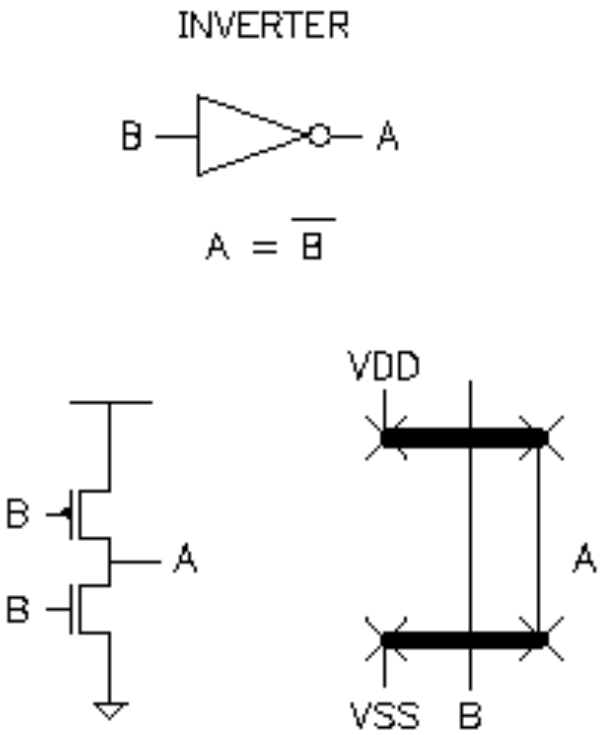


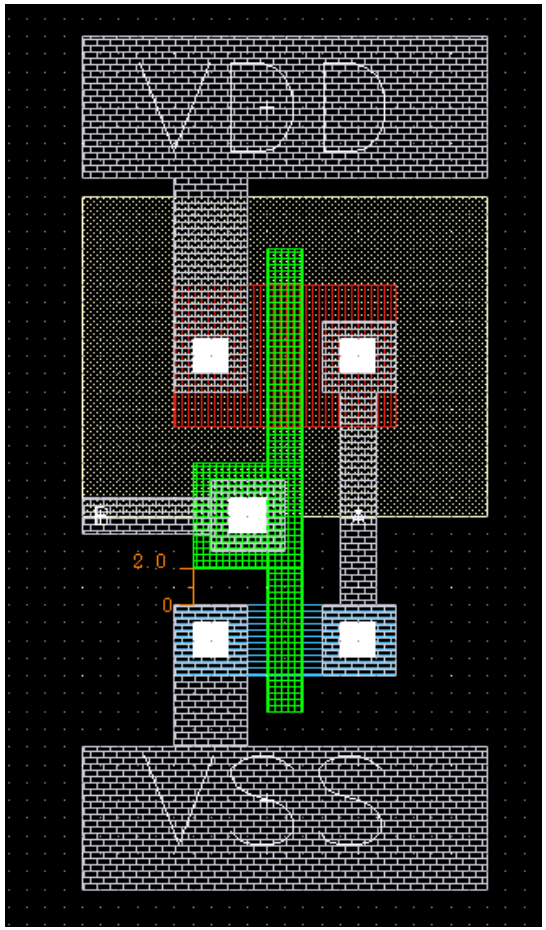
Digital System

- A Digital system is composed with several components (module).
- Each component itself is digital system.
- Each module are connected to build a digital component.
- Smallest modules are gates.
- To understand higher level digital system, we need to understand lower level digital module's behavior.

Examples: digital computers, calculators, digital audio/video equipment, telephone system…

Digital System

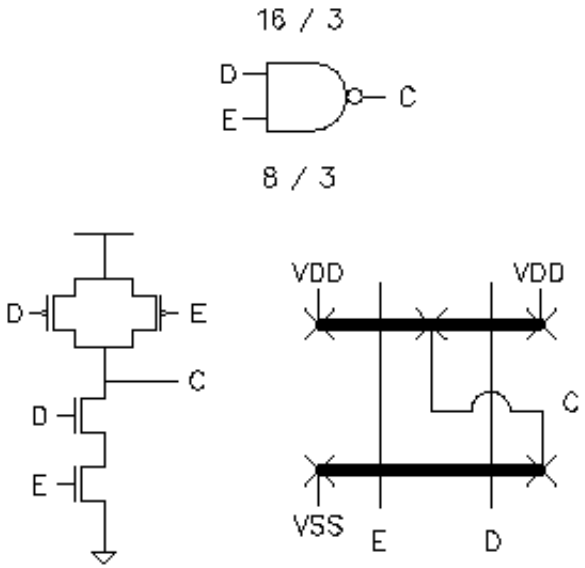


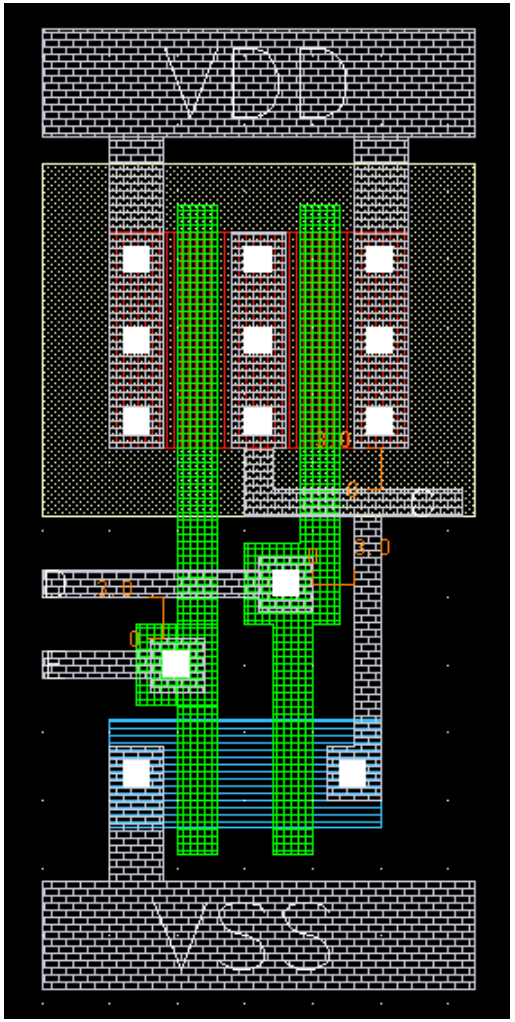


CMOS Technology

c*omplementary* ***m****etal* ***o****xide* ***s****emiconductor*

Digital System





CMOS Technology

c*omplementary* ***m****etal* ***o****xide* ***s****emiconductor*

Advantages of Digital Techniques

- Digital systems are generally easier to design.
- Information storage is easy.
- Accuracy and precision are greater.
- Operation can be programmed.
- Digital circuits are less affected by noise.
- More digital circuitry can be fabricated on IC chips.

Limitations of Digital Techniques

- *The real world in mainly analog*.
- To deal with analog inputs, three steps must be followed:
  - Convert the real-world analog inputs to digital form
    (analog-to-digital converter, ADC)
  - Process (operate on) the digital information
  - Convert the digital output back to real-world analog form (digital-to-analog converter,DAC)

Digital vs. Analog

- Added complexity and expense due to ADC, DAC
- Extra time required to perform conversions
- In most applications, digital techniques are favored because of the advantages discussed before.
- One notable exception: signal amplification is most easily achieved using analog circuitry.
- Hybrid systems: combination of digital and analog parts.
- *The future is digital*.

Digital System

Definitions

- **Gate** - the smallest module (unit) in digital system. There are AND, OR, NOT, XOR, NAND, and NOR basic gates. Each gate performs different operation based on input.
- **Circuits** - Several gates are connected to create a circuit to perform more complicated tasks.

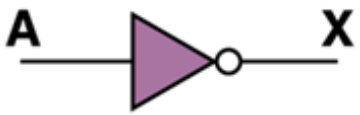
Logical Gates
(NOT Gate)

- A NOT gate accepts one input signal (0 or 1) and returns the opposite signal as a output

Boolean Expression

$$X = \overline{A}$$

Logic Diagram Symbol



Truth Table

A	X
0	1
1	0

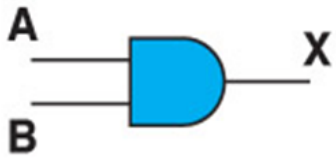
Logical Gates
(AND Gate)

- An two input AND gate accepts two input signals.
- If both are 1, the output is 1; otherwise, the output is 0

Boolean Expression

$X = A \cdot B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

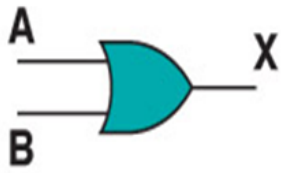
Logical Gates
(OR Gate)

- An two input OR gate accepts two input signals.
- If both are 0, the output is 0; otherwise, the output is 1

Boolean Expression

$$X = A + B$$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

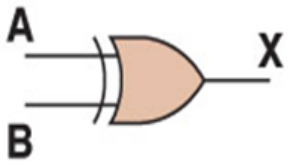
Logical Gates
(XOR Gate)

- An XOR gate (exclusive OR) accepts two input signals.
- If both are the same, the output is 0; otherwise, the output is 1

Boolean Expression

$X = A \oplus B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

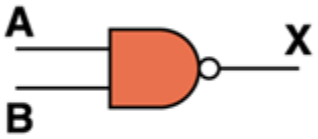
Logical Gates
(NAND Gate)

- The two input NAND gate accepts two input signals
- If both are 1, the output is 0; otherwise, the output is 1

Boolean Expression

$$X = \overline{(A \cdot B)}$$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

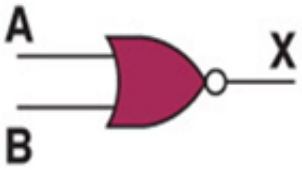
Logical Gates
(NOR Gate)

- The two input NOR gate accepts two input signals.
- If both are 0, the output is 1; otherwise, the output is 0.

Boolean Expression

$$X = \overline{(A + B)}$$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0