Intro to Computer Science

Basis for the design and use of computers

SoftUni Team

Technical Trainers

Software University

SoftUni Foundation

Software University

http://softuni.bg

Table of Contents

SoftUni Foundation

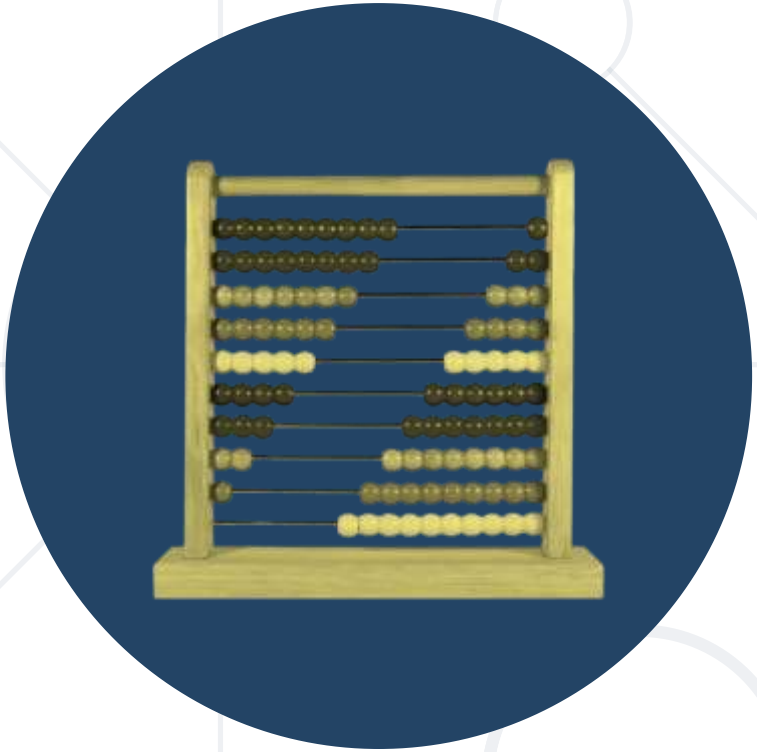
1. Early Computing
2. Boolean logic and Logical gates
3. ALU (Arithmetic and Logic Units)
4. Registers, RAM
5. CPU - Instruction cycle

Have a Question?

SoftUni Foundation

sli.do

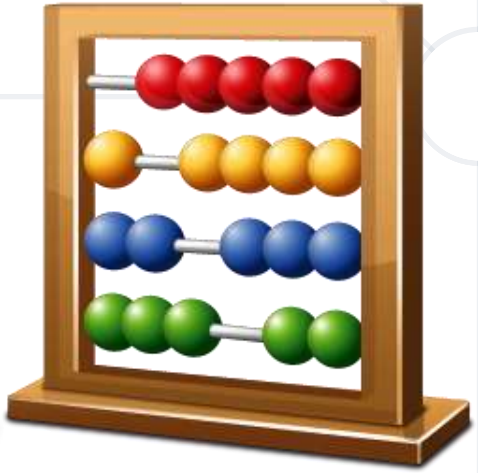
#TECH-FUND



Early Computing

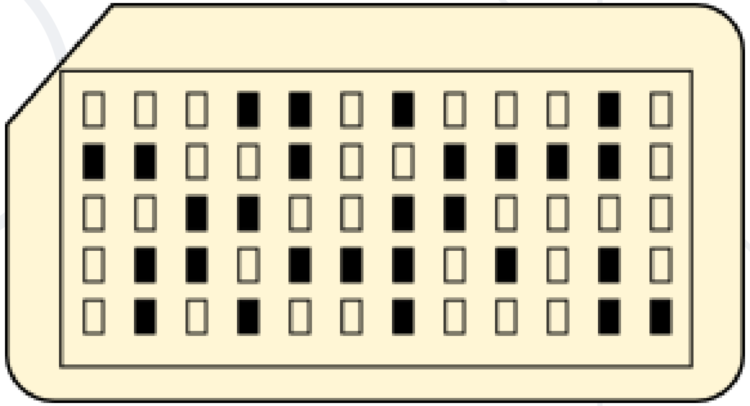
The Abacus

- **Calculating tool** that was in use in Europe, China and Russia
- Originally they were beans or stones moved in grooves in sand or on tablets
  of wood, stone, or metal.



The Punched Card

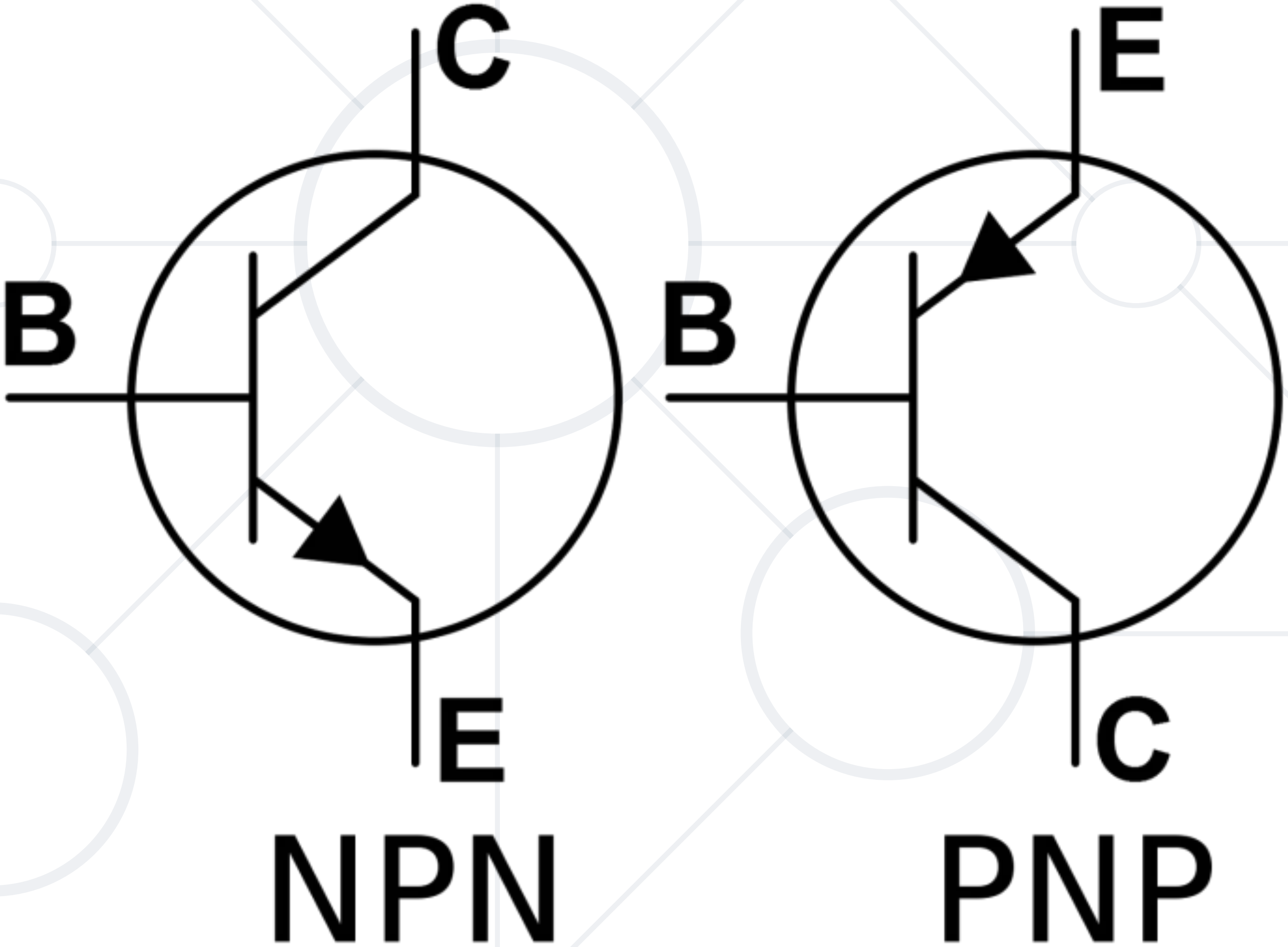
- Piece of stiff paper that can be used to contain **digital data** represented by the presence or absence of **holes** in predefined positions.
- Widely used through much of the 20th century in the **data processing** industry



Transistors

- Used to amplify or **switch** electronic signals and electrical power.
- Has at least **three terminals** for connection to an external circuit.
- The transistor is the fundamental building block of modern **electronic devices**

Structure of Transistors



Abstraction level:

1 2 3 4 5

Boolean logic and Logical gates

Boolean Algebra

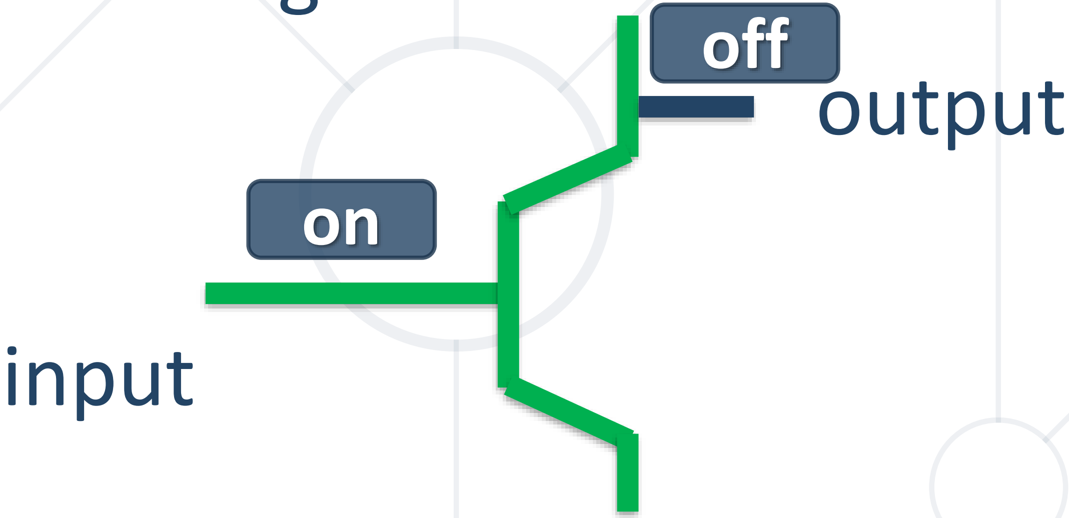
- Branch of algebra in which the values of the variables are the truth values **true** and **false**, usually denoted **1** and **0** respectively
- Formalism for describing logical relations
- There are 4 logical operations:
  - NOT
  - AND
  - OR
  - XOR

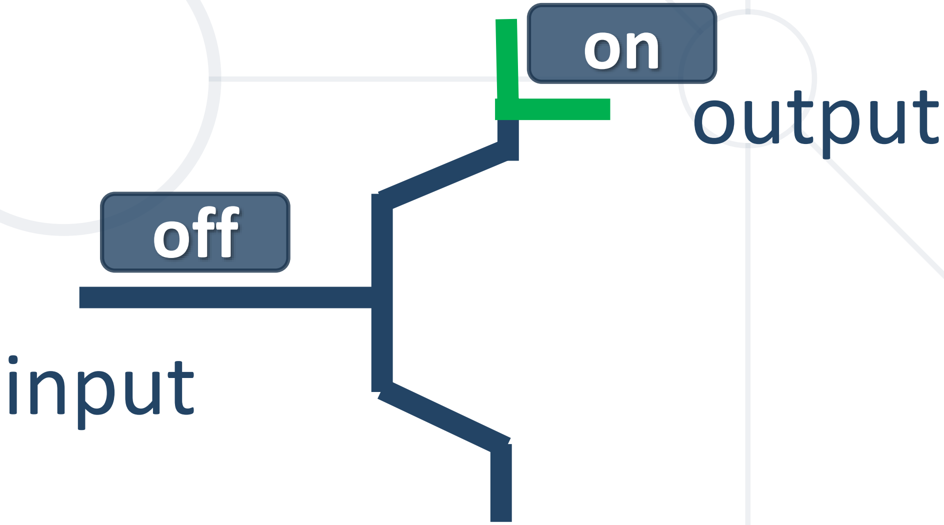
NOT Operation

- Takes a single value True or False and **flips** it

Input	Output
True	False
False	True

- NOT gate:



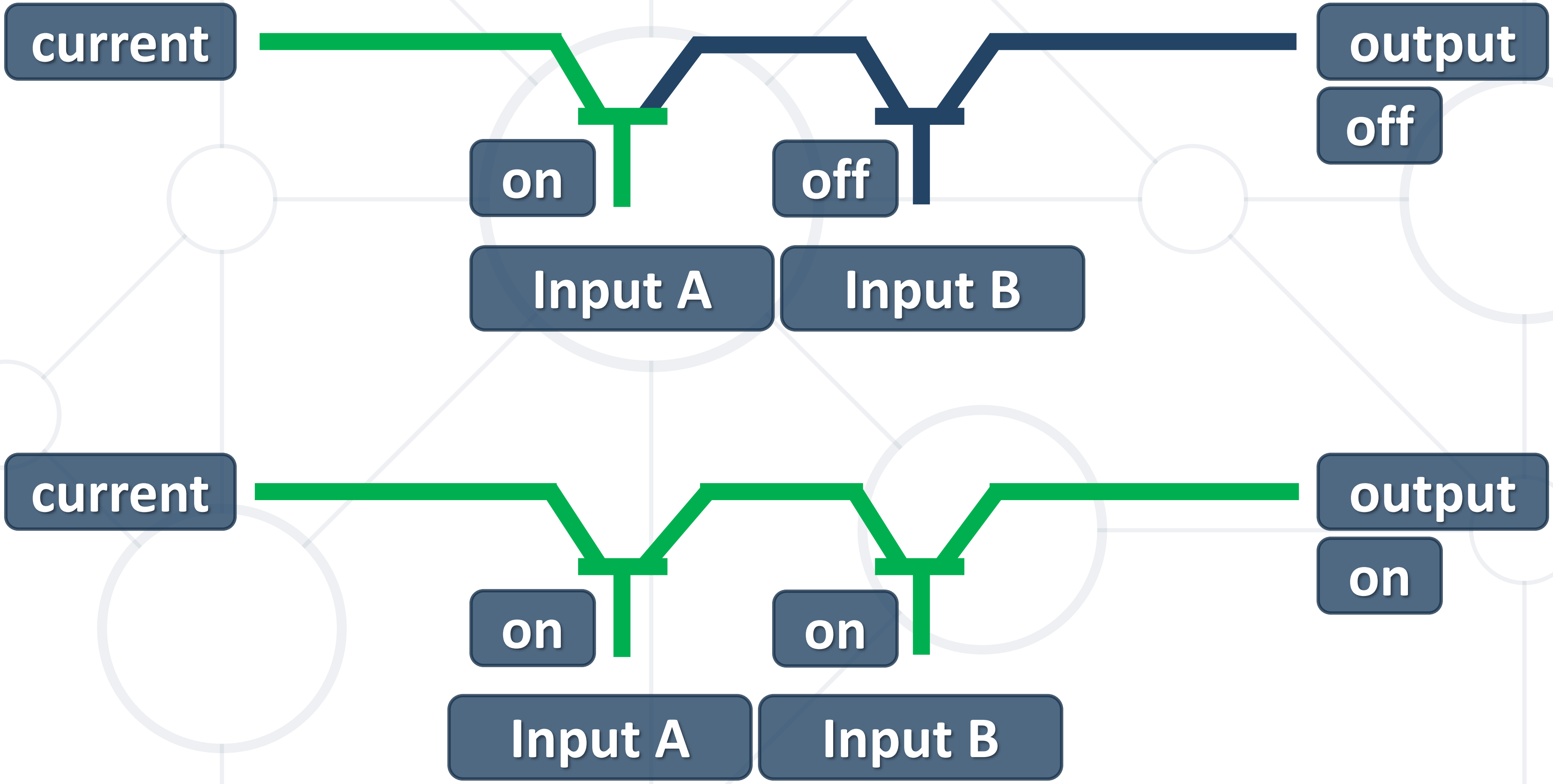


AND Operation

- Takes two input values and outputs a single value

Input A	Input B	Output
True	False	False
False	True	False
True	True	True
False	False	False

AND Gate

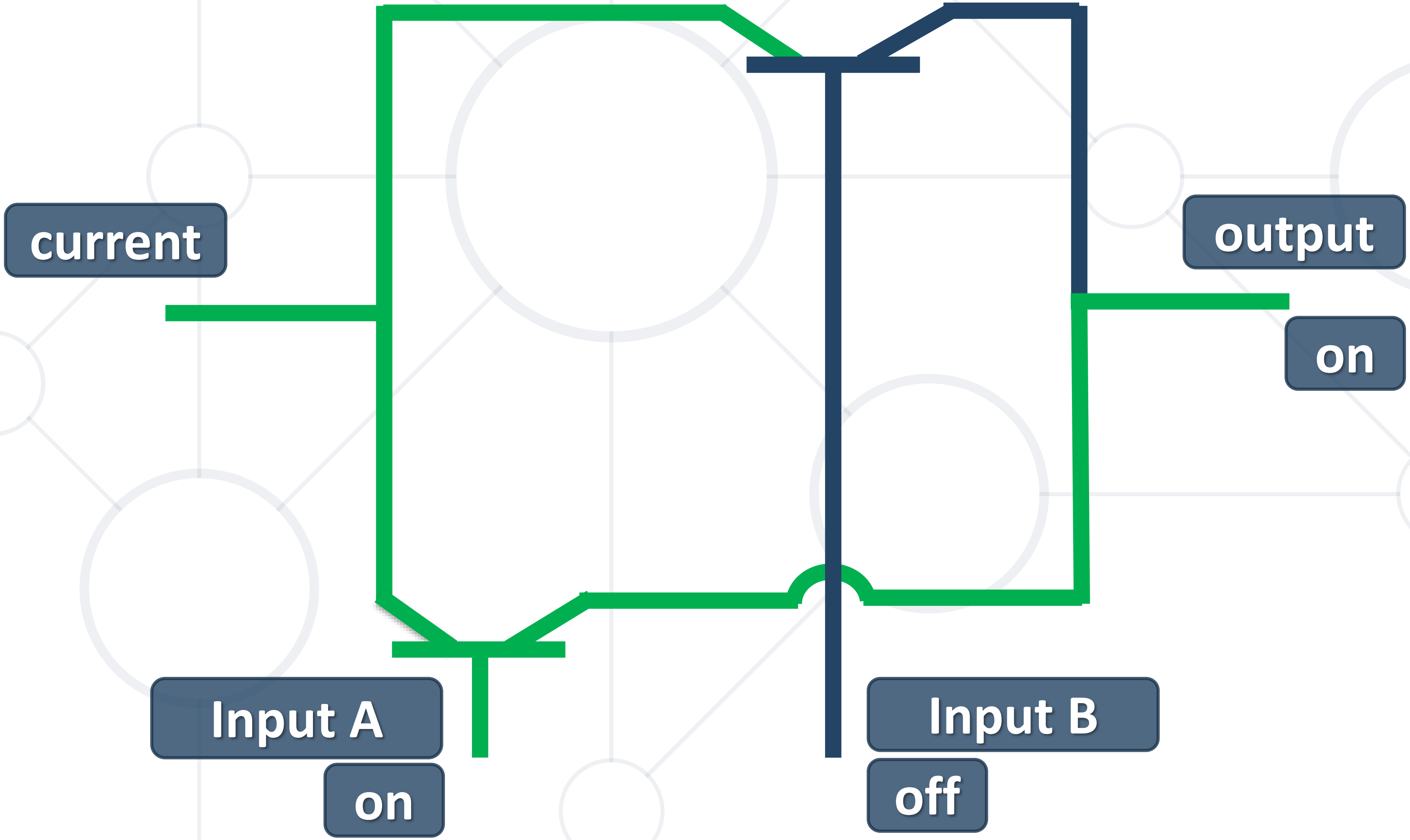

current
on
Input A
off
Input B
output
off
current
on
Input A
on
Input B
output
on

OR Operation

- Takes two input values and outputs a single value

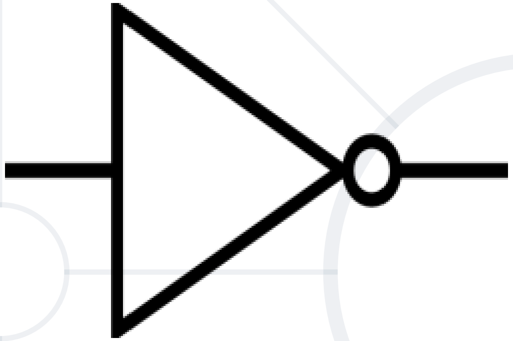
Input A	Input B	Output
True	False	True
False	True	True
True	True	True
False	False	False

OR Gate

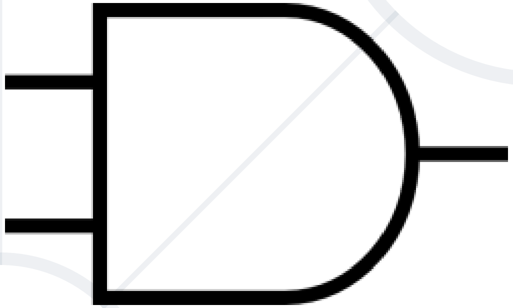


Logical operations notation

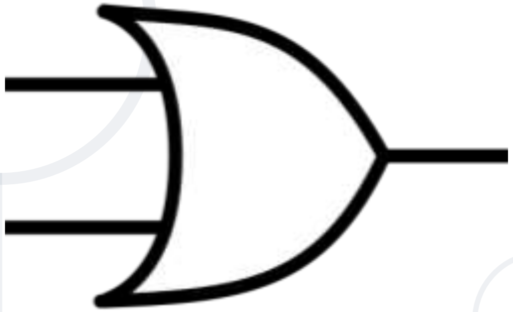
- NOT:



- AND:



- OR:



From now on, we are going to use them like this

Abstraction level:

1 2 3 4 5

XOR Operation

- Takes two input values and outputs a single value

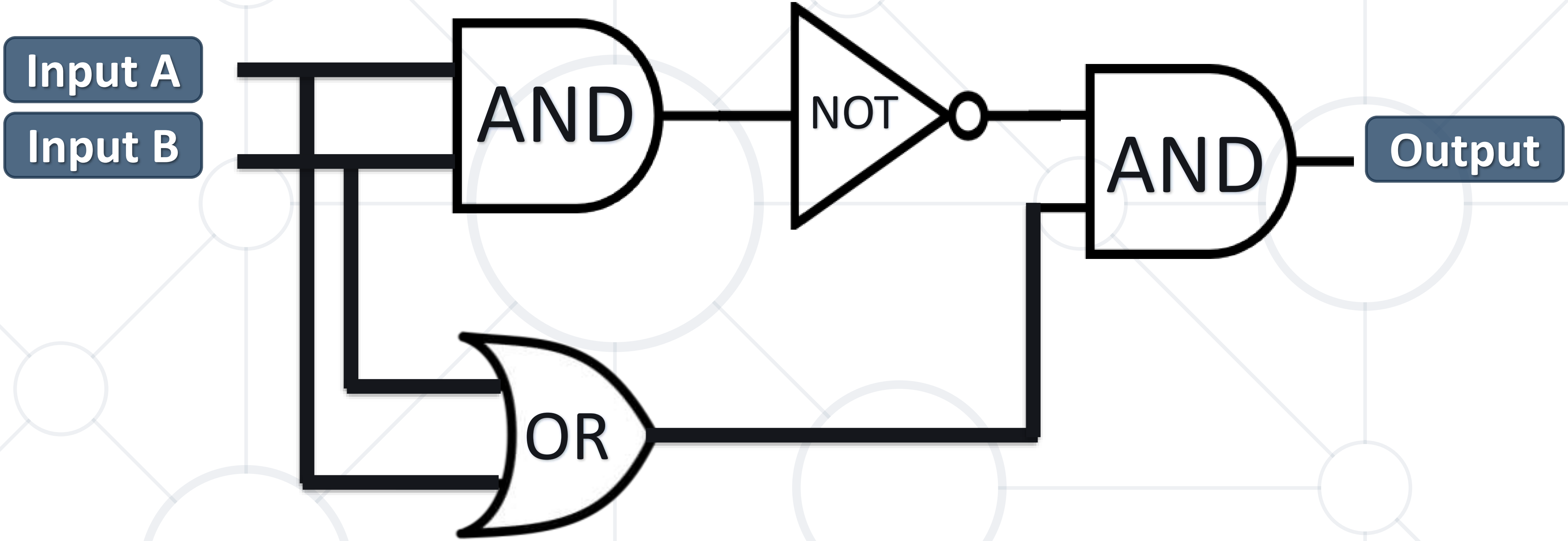
Input A	Input B	Output
True	False	True
False	True	True
True	True	False
False	False	False

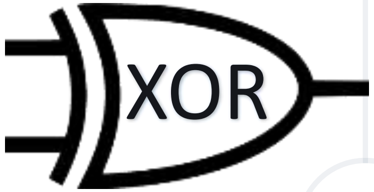
- XOR is **OR** but **not AND**

Problem: XOR Gate

- Try to recreate how the XOR gate will look like
- XOR is OR but not AND, so:
  - Include OR
  - Include AND
  - Include NOT after the AND
  - Connect all of them with AND

Solution: XOR Gate



- XOR notation: 

Abstraction level:

1 2 **3** 4 5

ALU

Arithmetic and Logic Units

What is ALU

- Electronic circuit that performs arithmetic and bitwise operations on integer binary numbers.
- To sum binary numbers, we use the following:

This is actually XOR

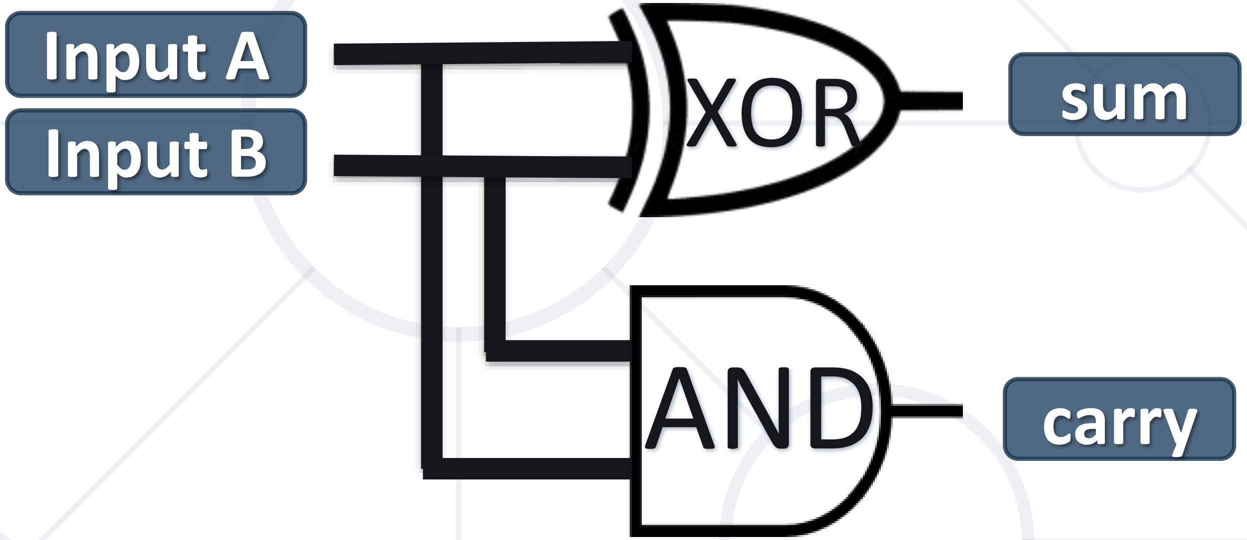
```
0 + 0 = 0
1 + 0 = 1
0 + 1 = 1
1 + 1 = 0 (1)
```

We carry over the 1

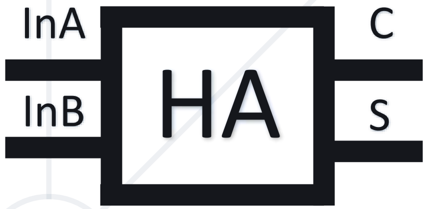
- To carry over we use **AND** gate

Half Adder

- So to represent adding binary numbers we use **XOR** + **AND**



- Half Adder notation:



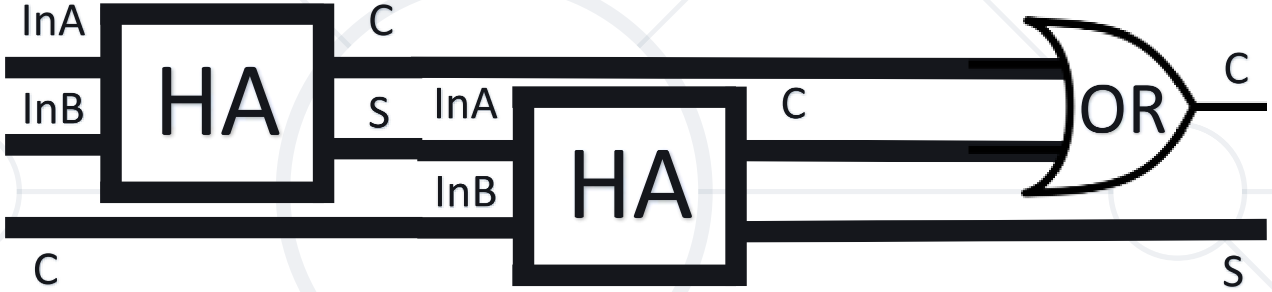
Full Adder

- To sum binary numbers, we have to send that 1 that we carried over to the next column
- To do that, we just need to take that carried over 1 and send it as input to another half adder
- We need two half adders to create full adder

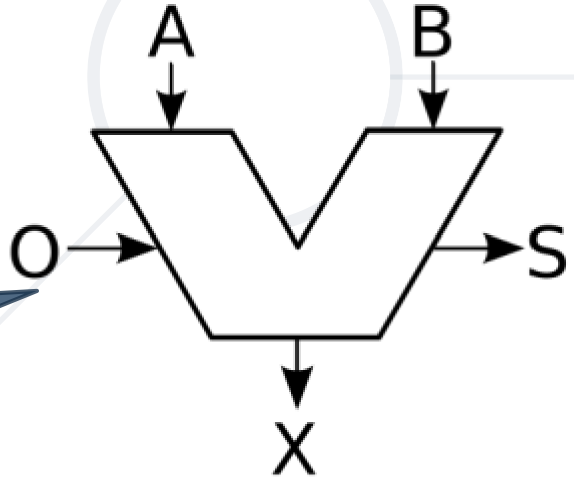
Problem: Full Adder

- Knowing what a half adder is, try creating full adder (used for carrying over ones)
- Full adder is chaining two half adders:
  - Take a half adder
  - Make its sum an input to another half adder
  - The other input of the second adder will be a carry (if one)
  - Then add an OR gate to calculate the carry of the two half adders

Solution: Full Adder



- So a full adder receives 3 inputs and produces two outputs (sum and carry)
- Full Adder notation:



OPCODE – 4 bit code to determine the operation (1000 – Add, 1100 - Subtract)

STATUS – 3 outputs (zero, negative, overflow).

Abstraction level:

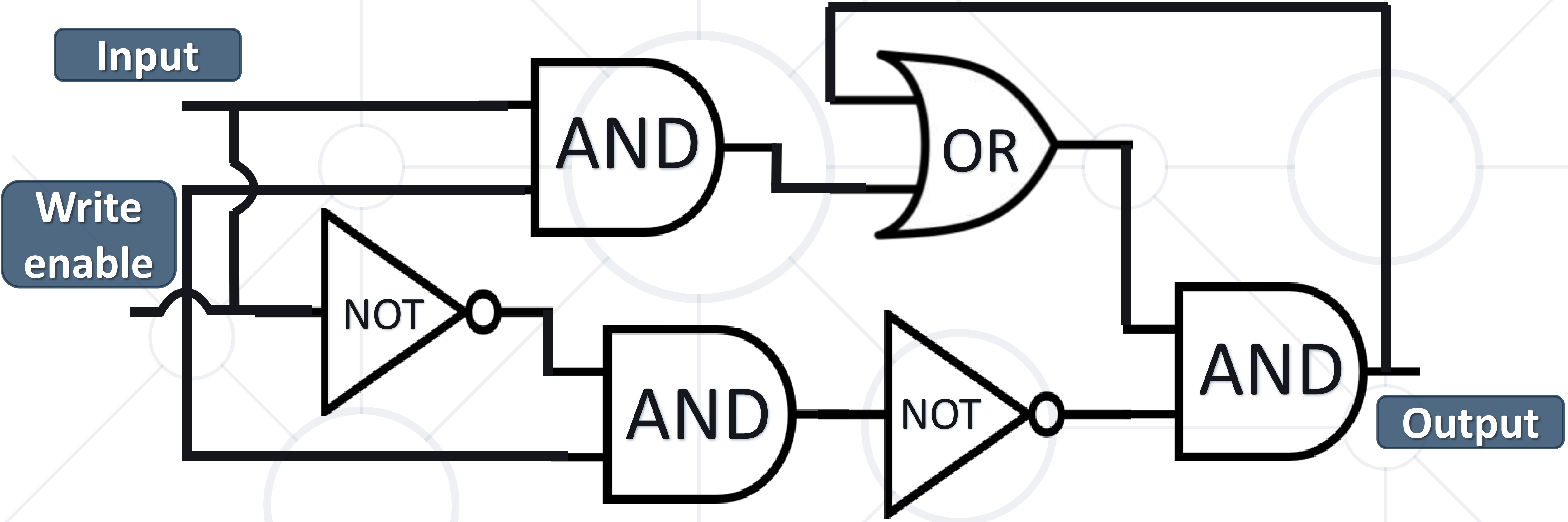
1 2 3 4 5

Registers, RAM

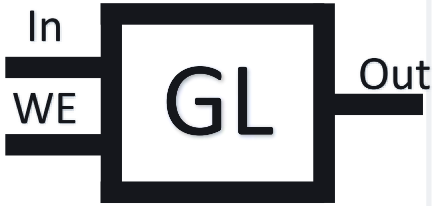
Gated Latch

- Circuit that has two stable states and can be used to **store** state **information**.
- It has a third input that must be active in order for the **SET** and **RESET** inputs to take effect.
- This third input is sometimes called **ENABLE** because it enables the operation of the SET and RESET inputs.

Gated Latch Structure

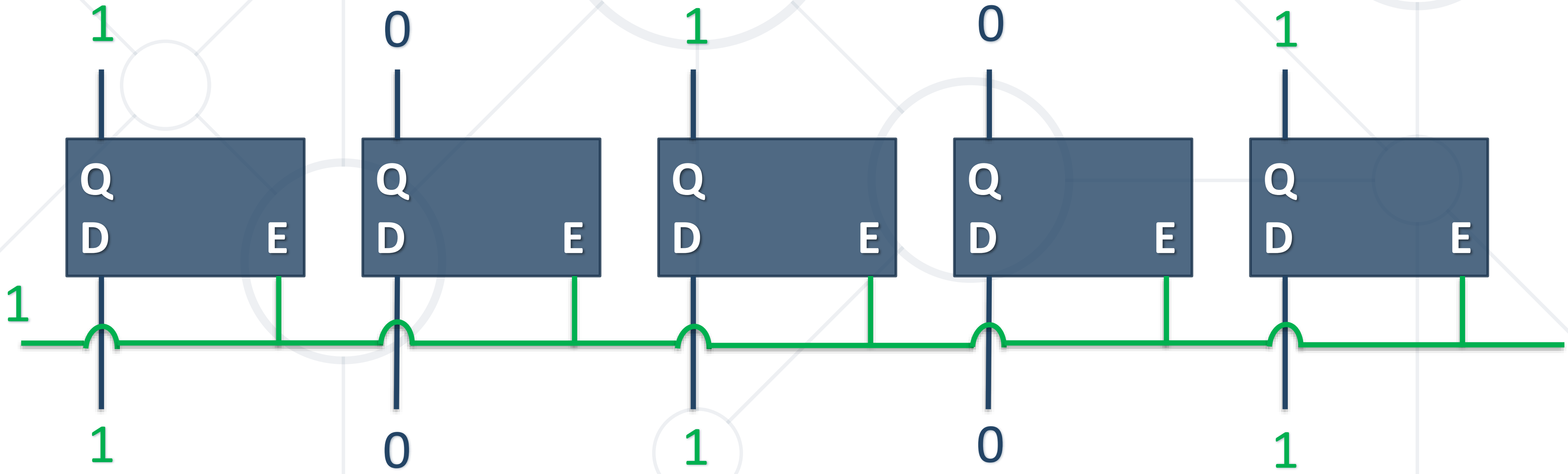


- When turning **Write enable** on, we save the output



8-bit Registers

- D – data in
- E – write enable
- Q – data out

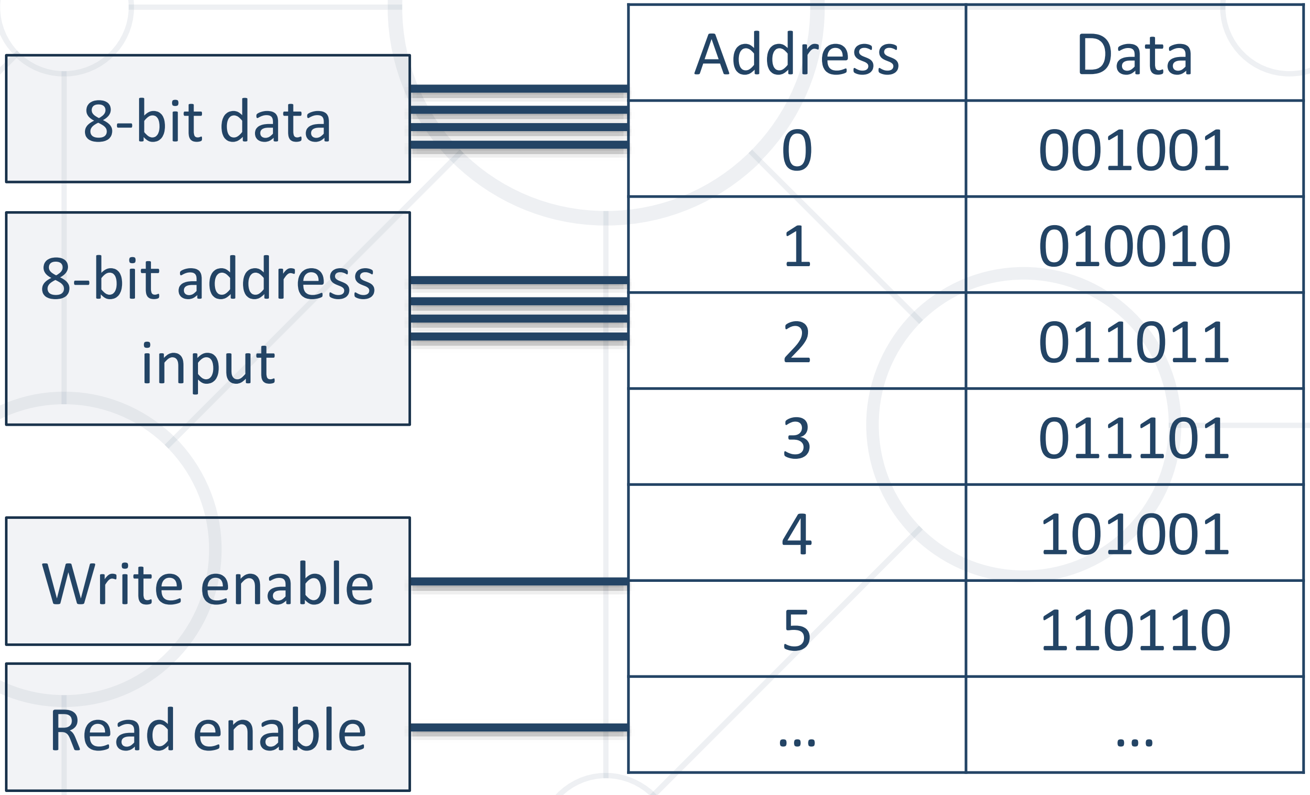


Multiplexer

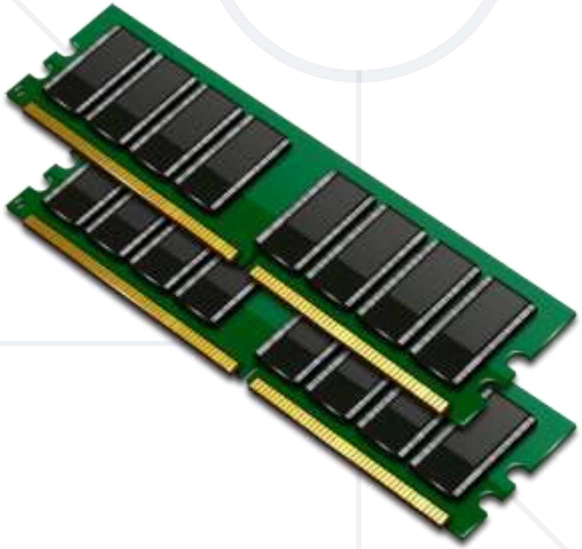
- Grid of many latches
- Selects one of several analog or digital input signals and forwards the selected input into a single line.
- Is used to increase the amount of data that can be sent over the network
- Multiplexers can also be used to implement Boolean functions of multiple variables.
- **RAM** contains many multiplexers

RAM

- Stores data and machine code currently being used.
- Allows data items to be read or written



Address	Data
0	001001
1	010010
2	011011
3	011101
4	101001
5	110110
…	…



Abstraction level:

1 2 3 4 5

CPU and Instruction cycle

What is a CPU?

- Carries out the **instructions** of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations
- Fundamental operation of most CPUs is to execute a sequence of stored instructions that is called a **program**.

Instructions Table

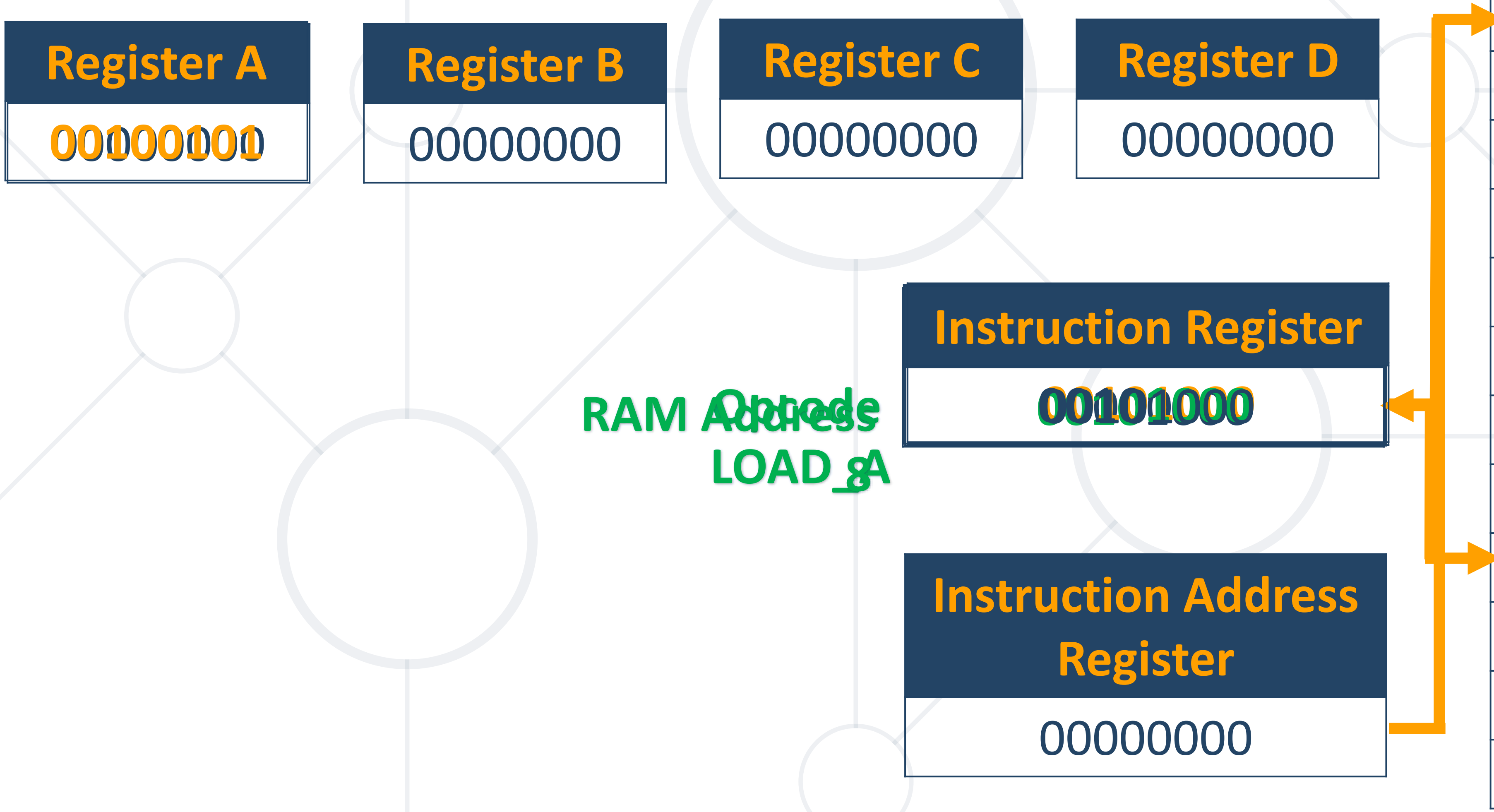
Instruction	Description	4-bit opcode	Address/Register
LOAD_A	Read RAM location into register A	0010	4-bit RAM address
LOAD_B	Read RAM location into register B	0001	4-bit RAM address
STORE_A	Write from register A into RAM location	0100	4-bit RAM address
ADD	Add two registers and store into second register	1000	2-bit register ID, 2-bit register ID

Example

Fetch Phase

Decode Phase

Execute Phase



RAM	
Address	Data
0	00101000
1	00011111
2	10000100
3	01001101
4	00000000
5	00000000
6	00000000
7	00000000
8	00100101
9	00000011
10	00001110
...	...

What to discuss

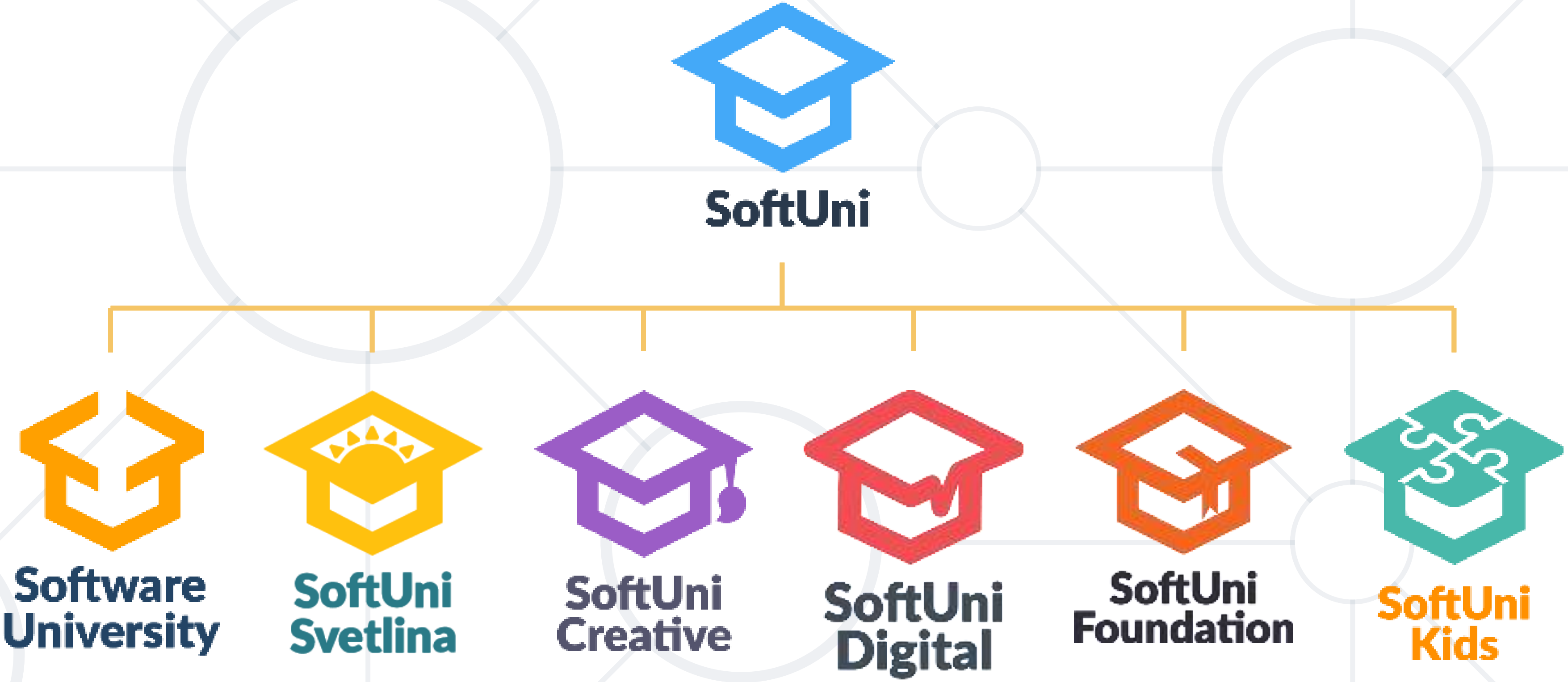
- Discuss how addition will work in the CPU
- Try adding ALU in the system
- How will ALU be used
- How will multiplication and division work
- How will if-statements and for-loops work

Live Exercises

Summary

- **Computer science** is form the basis for the design and use of computers.
- **Transistors** and **logic gates** are the basics in computing
- **Latches** build **Multiplexers**, which build **RAM**
- All the operations in the **CPU** are build on top of logical operations

Questions?


SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

https://softuni.bg/trainings/courses

SoftUni Diamond Partners

XSsoftware

SBTech
we know sports

telenor

SoftwareGroup
doing it right

NETPEAK

SmartIT

Postbank
Решения за твоето утре

SUPER HOSTING ®.BG

INDEAVR
Serving the high achievers

INFRAGISTICS®

LIEBHERR

æternity

codexio

SoftUni Organizational Partners

IS ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

WORLD OF MYTHS

Lukanet.com

codexio

Trainings @ Software University (SoftUni)

SoftUni Foundation

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

SoftUni Foundation

License