LECTURER: TAI LE QUY

INTRODUCTION TO COMPUTER SCIENCE

UNITS 4 & 5

PROPOSITIONAL LOGIC, BOOLEAN ALGEBRA AND CIRCUIT DESIGN & HARDWARE AND COMPUTER ARCHITECTURES

STUDY GOALS

- Understand the basic language and concepts of propositional logic.
- Learn how to create truth tables.
- Find out how to use the conjunctive and disjunctive normal form.
- Learn about the basic concepts of digital circuits and logic gates.

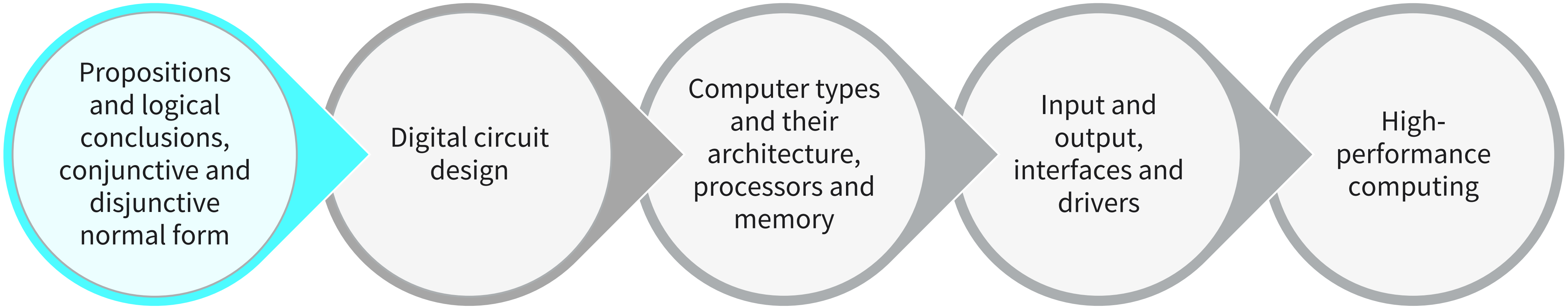
STUDY GOALS

- Learn the basic elements of computer architecture.
- Understand how processors and memory work.
- Explore how a computer processes input and output.
- Learn how operating systems and hardware communicate.
- Get an overview of high-performance computing.

EXPLAIN SIMPLY

1. What is a register?
2. What is the difference between a data bus and an address bus?
3. What is a computer cluster?

PROPOSITIONAL LOGIC, BOOLEAN ALGEBRA AND CIRCUIT DESIGN



- **logical constants**: true, false
- **propositional symbols**: P, Q, S, … (**atomic sentences**)
- wrapping **parentheses**: (…)
- literal: atomic sentence or negated atomic sentence
- sentences are combined by **connectives**:

Connectives	Type	Example
$\wedge$ ...and	[conjunction]	action or instance of two or more events or things occurring at the same point in time or space ***"a conjunction of favorable political and economic circumstances"***
$\vee$ ...or	[disjunction]	A disjunction is a compound statement formed by joining two statements with the connector OR. ***"a disjunction by either favorable political or economic circumstances"***
$\Rightarrow$...implies	[implication / conditional]	
$\Leftrightarrow$...is equivalent	[biconditional]	
$\neg$ ...not	[negation]	

EXAMPLES OF PL SENTENCES

$(P \wedge Q) \rightarrow R$	– "If it is hot and humid, then it is raining."
$Q \rightarrow P$	– "If it is humid, then it is hot."
Q	– "It is humid."
A better way:	– Ho = "It is hot." – Hu = "It is humid." – R = "It is raining."

- a simple language useful for showing key ideas and definitions
- User defines a set of propositional symbols, like **P** and **Q.**

- User defines the **semantics** of each propositional symbol:
  - Ho means "It is hot"
  - Hu means "It is humid"
  - R means "It is raining"

A sentence (well-formed formula) is defined as follows:

- A symbol **S** is a sentence.
- If **S** is a sentence, then $\neg$ **S** is a sentence.
- If **S** is a sentence, then **(S)** is a sentence.
- If **S** and **T** are sentences, then **S** $\vee$ **T**, **S** $\wedge$ **T**, **S** $\rightarrow$ **T**, and **S** $\leftrightarrow$ **T** are sentences.
- A sentence results from a finite number of applications of the above rules.

The definition of conjunctive normal form is a logic formula that is **a single conjunction** of any **number of disjunctions**.

$$p \lor \neg q \land r \lor s$$

There can be **any number** of disjunctions as long as they are **joined** by a conjunction.

To evaluate the results of the logic formula above, we can create a truth table.

The other normal form, disjunctive, is one where any number of conjunctions are connected by a disjunction. In other words, something like this:

$$p \land q \lor \neg r \land \neg s$$

This could be read as “p and q or not r and not s.”

This is two conjunctions connected by a disjunction, and the parentheses tell us which to evaluate first.

TRUTH TABLES

p	q	$(p \wedge q)$ and
F	F	**F**
F	T	**F**
T	F	**F**
T	T	**T**

p	q	$(p \rightarrow q)$ If … then
F	F	**T**
F	T	**T**
T	F	**F**
T	T	**T**

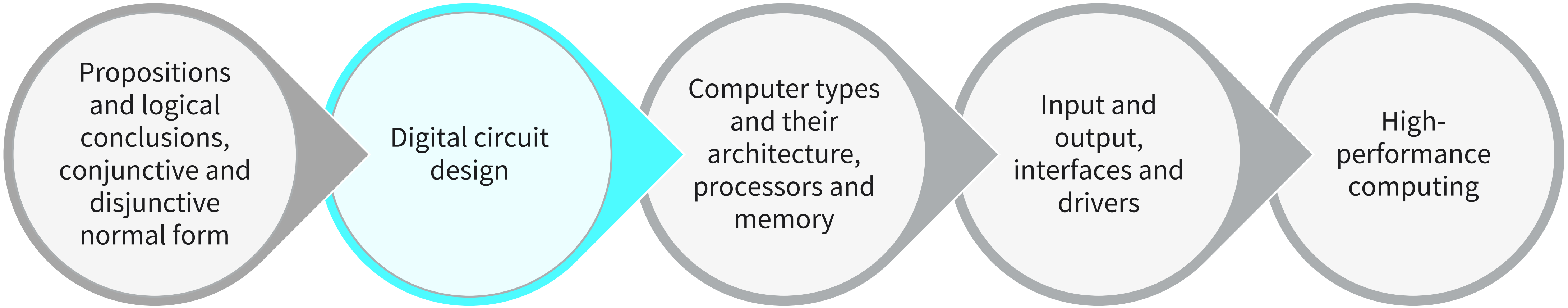
p	q	$(p \vee q)$ or
F	F	**F**
F	T	**T**
T	F	**T**
T	T	**T**

p	$\neg p$ not
F	**T**
T	**F**

Try it out:

https://web.stanford.edu/class/cs103/tools/truth-table-tool/

PROPOSITIONAL LOGIC, BOOLEAN ALGEBRA AND CIRCUIT DESIGN



Logical AND

a AND *b* is 1, if *a*=1 and *b*=1

Logical OR

a OR *b* is 1, if *a*=1 and/or *b*=1

Exklusive OR (XOR)

a XOR *b* is 1, if either *a*=1 or *b*=1

Logical NOT

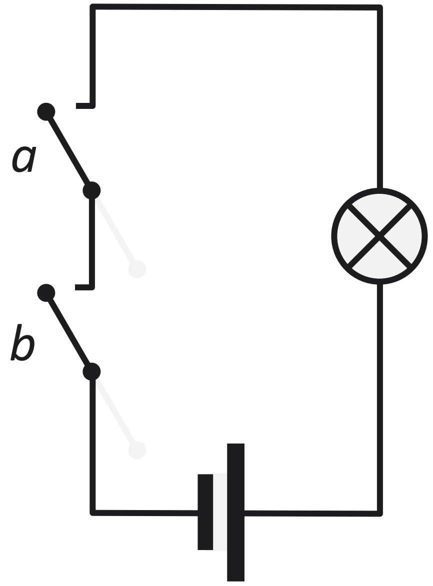
NOT *a* (NOT *b*) inverts the argument

a	b	AND	OR	XOR	NAND	NOR	NXOR*
0	0	0	0	0	1	1	1
0	1	0	1	1	1	0	0
1	0	0	1	1	1	0	0
1	1	1	1	0	0	0	1

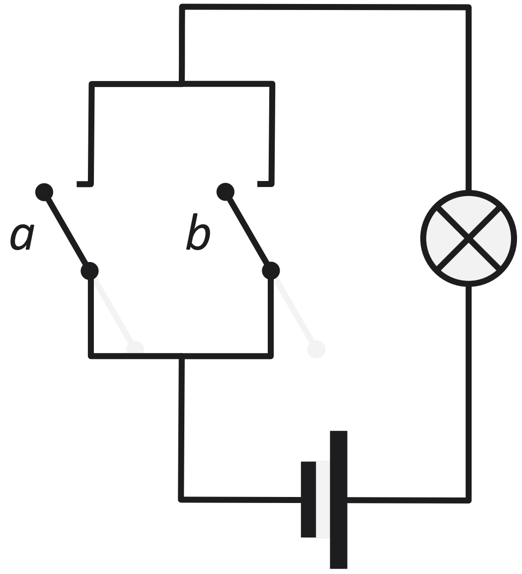
a/b	NOT
0	1
1	0

*also: XNOR

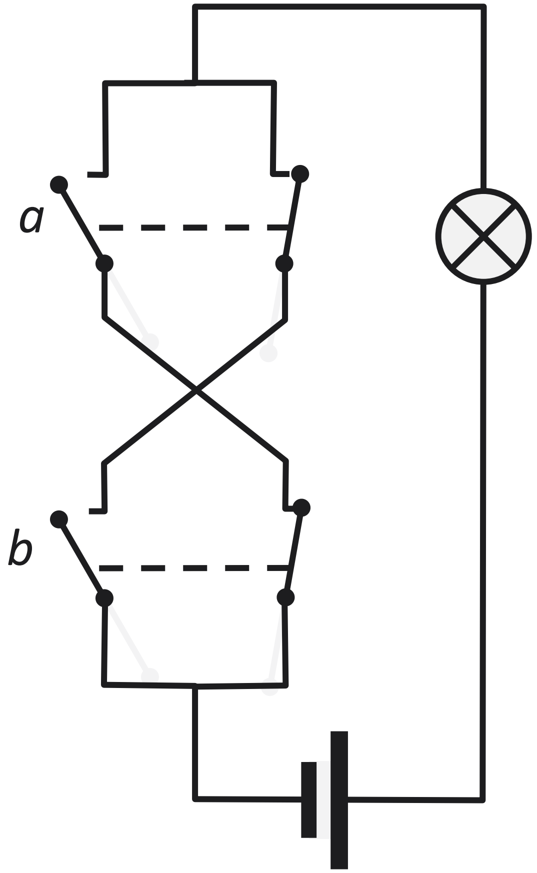
AND Gate



OR Gate



XOR Gate



a	*b*	AND	OR	XOR
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	1	0

We have looked at Boolean functions in abstract terms.

- In this section, we see that Boolean functions are implemented in digital computer circuits called gates.

A **gate** is an electronic device that **produces a result** based **on** two or more **input values**.

- In reality, gates consist of one to six transistors, but digital designers think of them as a single unit.
- Integrated circuits contain collections of gates suited for a particular purpose.

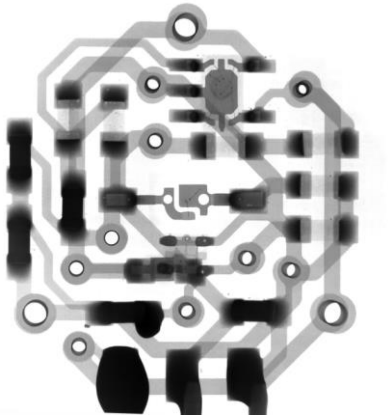
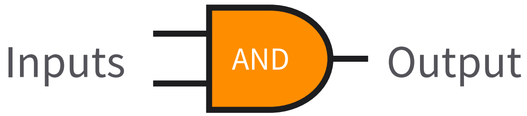


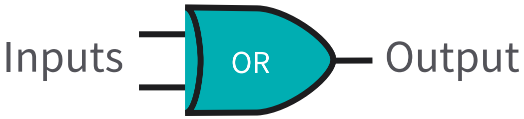
Photo: Mathew Schwartz on Unsplash

Source of the video: https://youtu.be/ZoqMiFKspAA

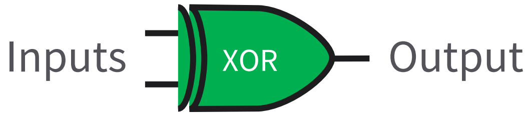
A pictorial representation of AND, OR, XOR, and NOT gates as well as their input and output values



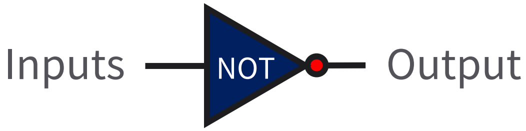
Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1



Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1



Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0



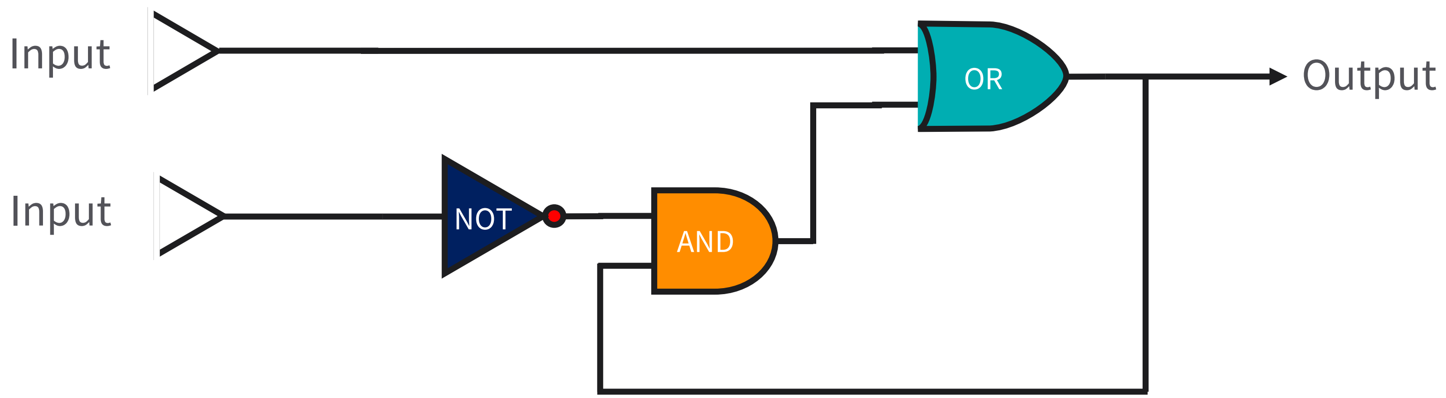
Inputs	Output
0 0	1
0 1	0

Source of the text: Brookshear & Bylow, 2011, p. 22.

Circuits built from gates that act as a fundamental unit of computer memory

- One input line is used to set its stored value to 1.
- One input line is used to set its stored value to 0.
- While both input lines are 0, the most recently stored value is preserved.

A simple flip-flop circuit

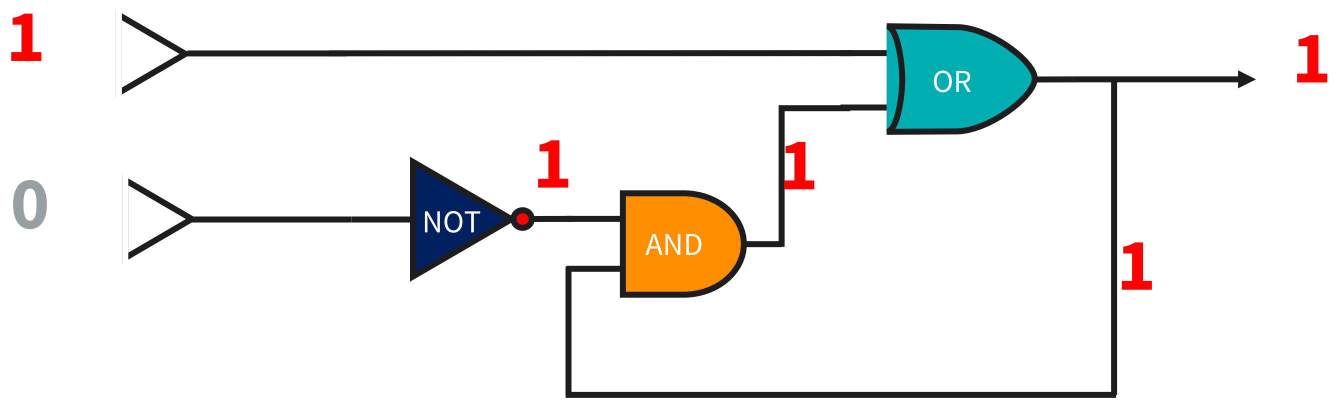


Source of the text: Brookshear & Bylow, 2011, p. 23.

SETTING THE OUTPUT OF A FLIP-FLOP TO 1

a. First, a 1 is placed in the upper input .

b. This causes the output of the OR gate to be 1 and the output of the AND gate to be 1.



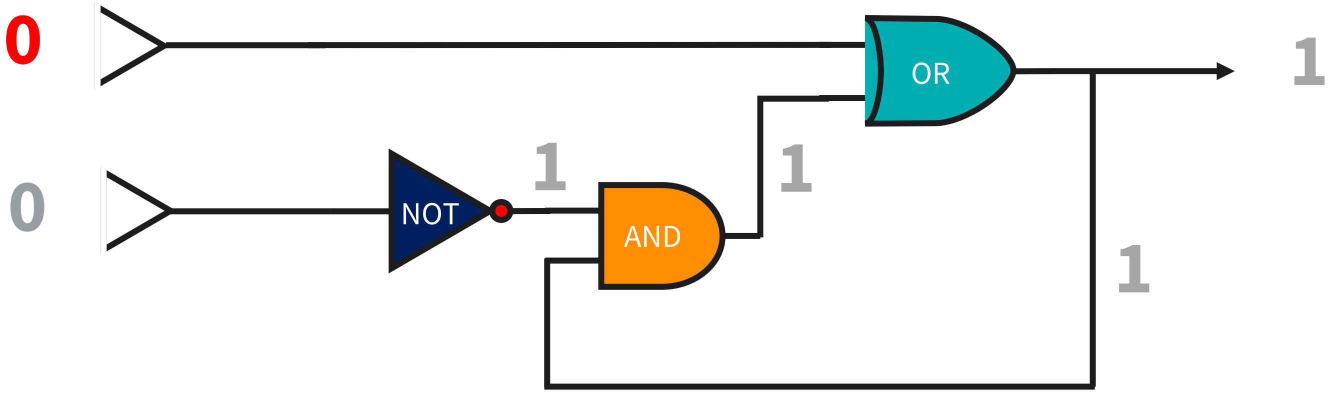
Source of the text: Brookshear & Bylow, 2011, p. 23.

SETTING THE OUTPUT OF A FLIP-FLOP TO 1

a. First, a 1 is placed in the upper input.

b. This causes the output of the OR gate to be 1 and the output of the AND gate to be 1.

c. Finally, the 1 from the AND gate keeps the OR gate from changing after the upper input returns to 0.



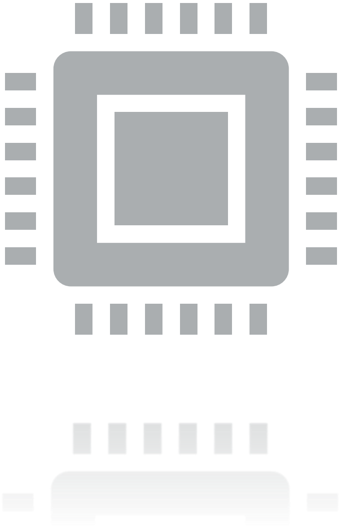
Source of the text: Brookshear & Bylow, 2011, p. 23.

We have seen digital circuits from two points of view: digital analysis and digital synthesis.

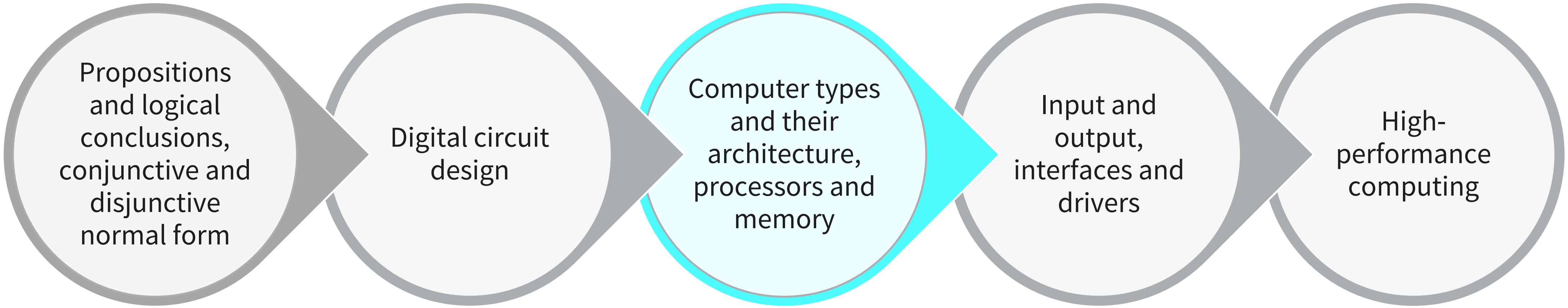
- Digital analysis explores the relationship between a circuit's inputs and its outputs.
- Digital synthesis creates logic diagrams using the values specified in a truth table.

Software is a collection of algorithms that could just as well be implemented in hardware.

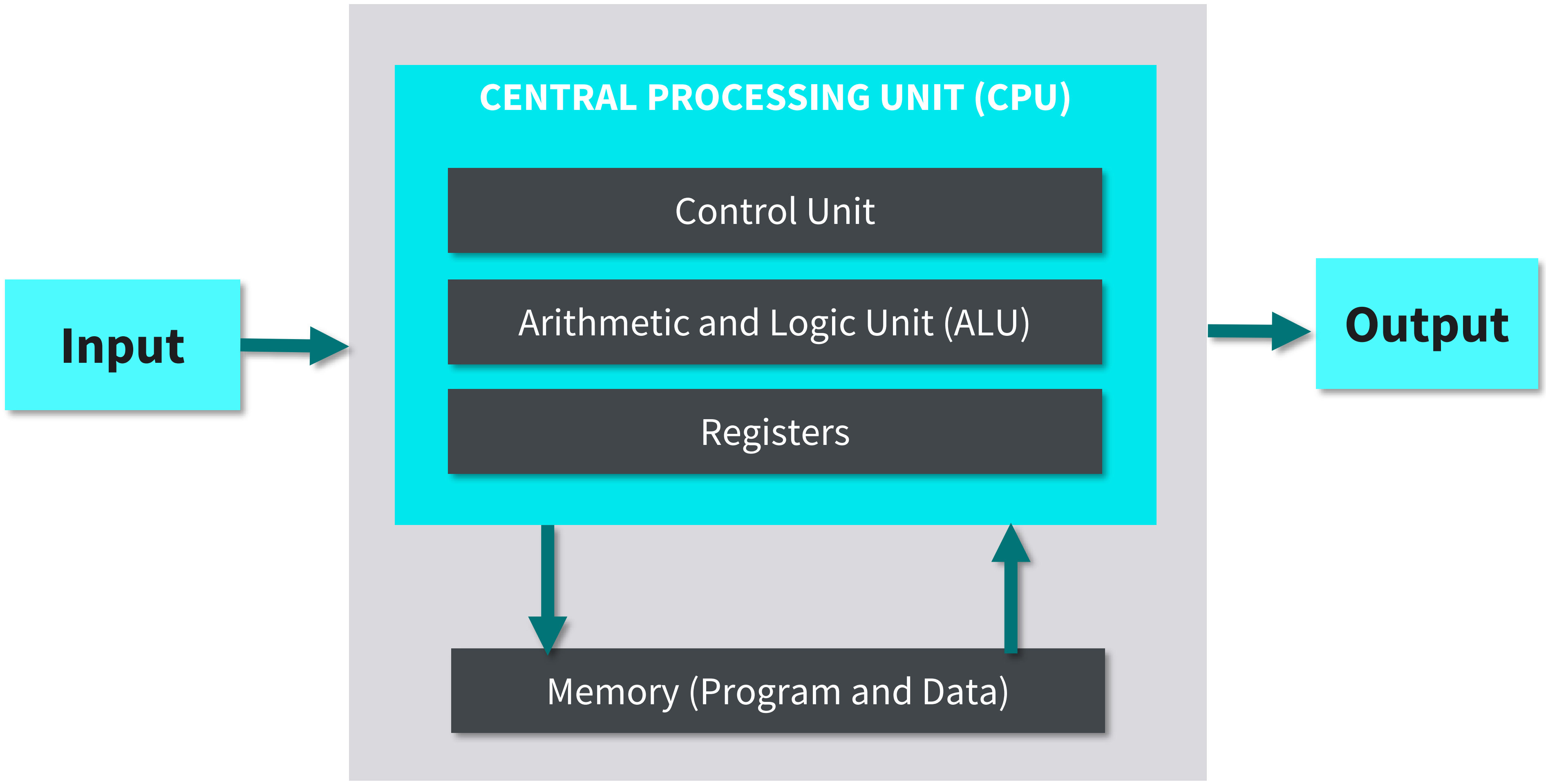
Principle of Equivalence of Hardware and Software.



PROPOSITIONAL LOGIC, BOOLEAN ALGEBRA AND CIRCUIT DESIGN



VON NEUMANN ARCHITECTURE (1/4)



Four registers are essential to instruction execution:

Program counter (PC)

- contains the address of an instruction to be fetched

Instruction register (IR)

- contains the instruction most recently fetched

Memory address register (MAR)

- contains the address of a location in memory

Memory buffer register (MBR)

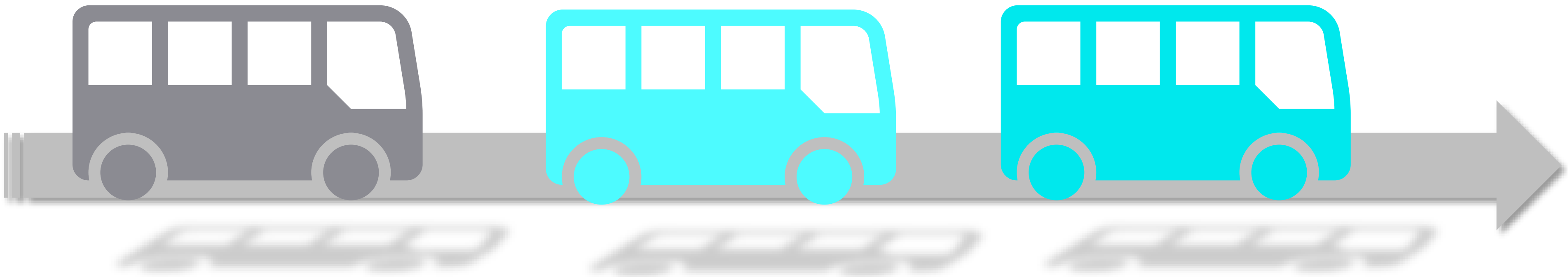
- contains a word of data to be written to memory or the word most recently read

COMPUTER BUS

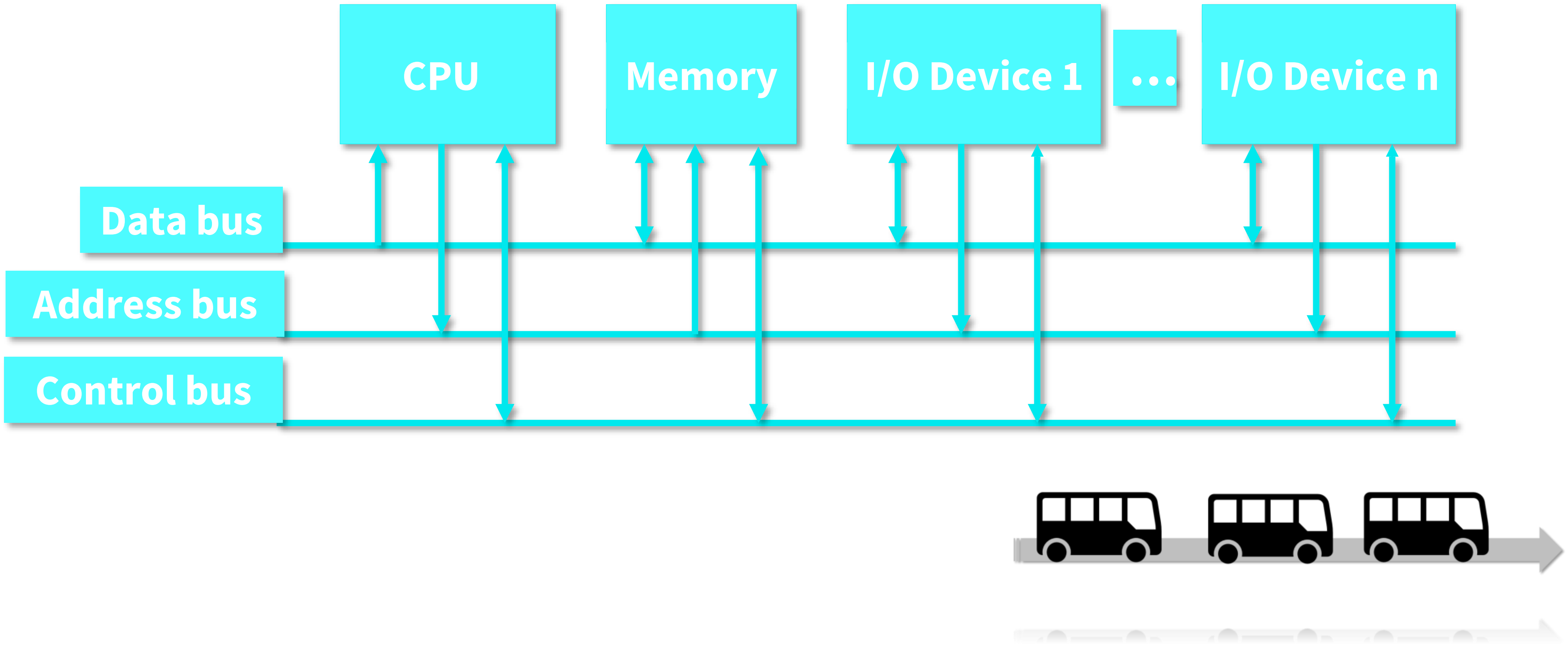
A **bus** is a set of parallel conductors **that transfer data** between different components of a computer.

A bus has three main parts:

- **Data bus**
  - carries data
- **Address bus**
  - carries the address of data
- **Control bus**
  - carries control signals



BUS-ORIENTED COMPUTER ARCHITECTURE

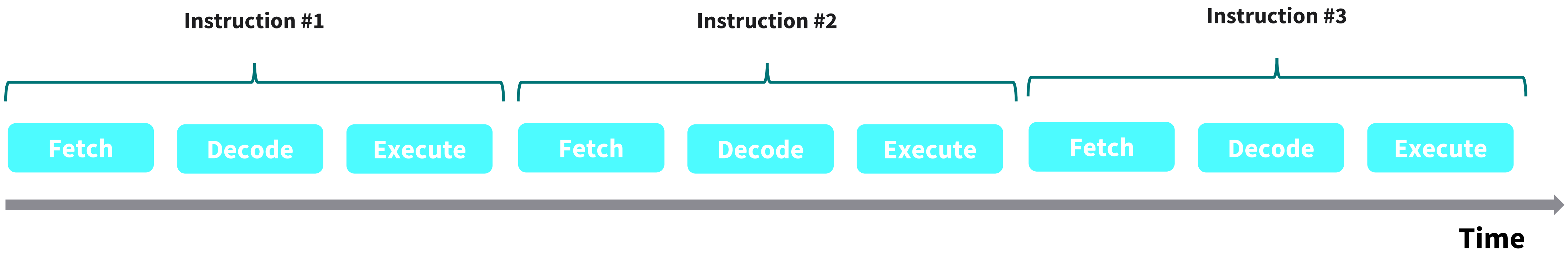


INSTRUCTION CYCLE

The control unit in all Central Processing Units (CPUs) follows the same basic instruction processing sequence:

1. **Fetch the instruction.**
2. **Decode the instruction.**
3. **Execute the instruction.**

On traditional CPUs, these phases are typically executed sequentially as shown:



Random Access Memory (RAM): Memory in which individual cells can be easily accessed in any order

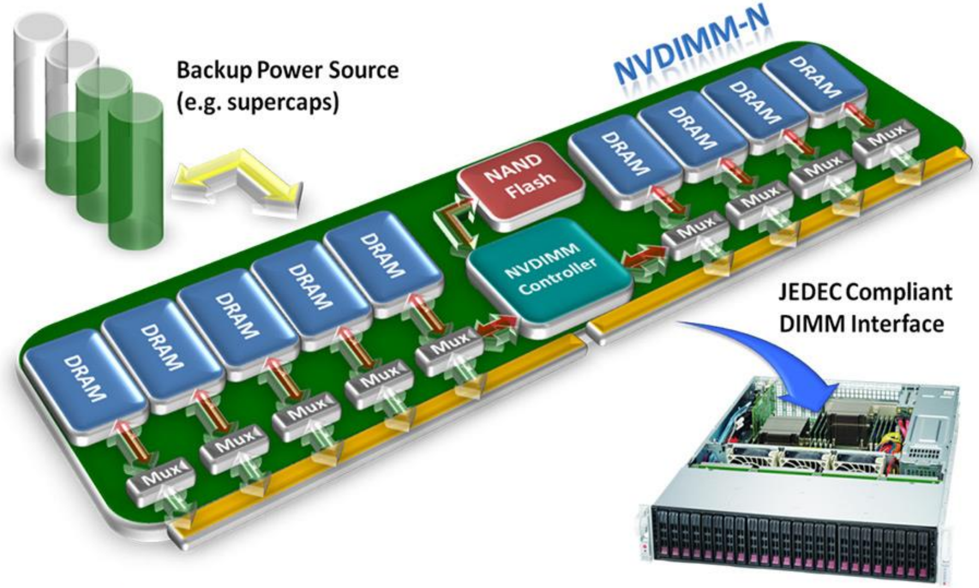
Dynamic Memory (DRAM): RAM composed of volatile memory

MEASURING MEMORY CAPACITY

Kilobyte = 1024 bytes
Example: 3 KB = 3 times 1024 bytes

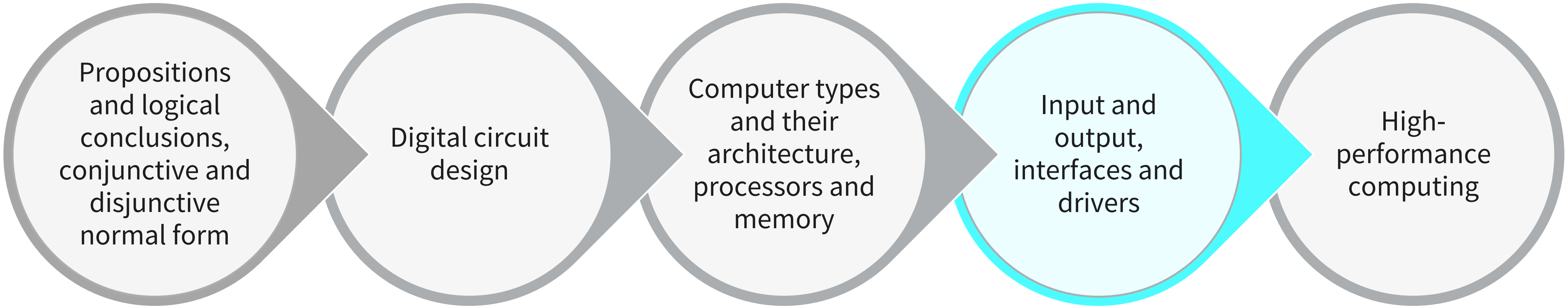
Megabyte = 1,048,576 bytes
Example: 3 MB = 3 times 1,048,576 bytes

Gigabyte = 1,073,741,824 bytes
Example: 3 GB = 3 times 1,073,741,824 bytes

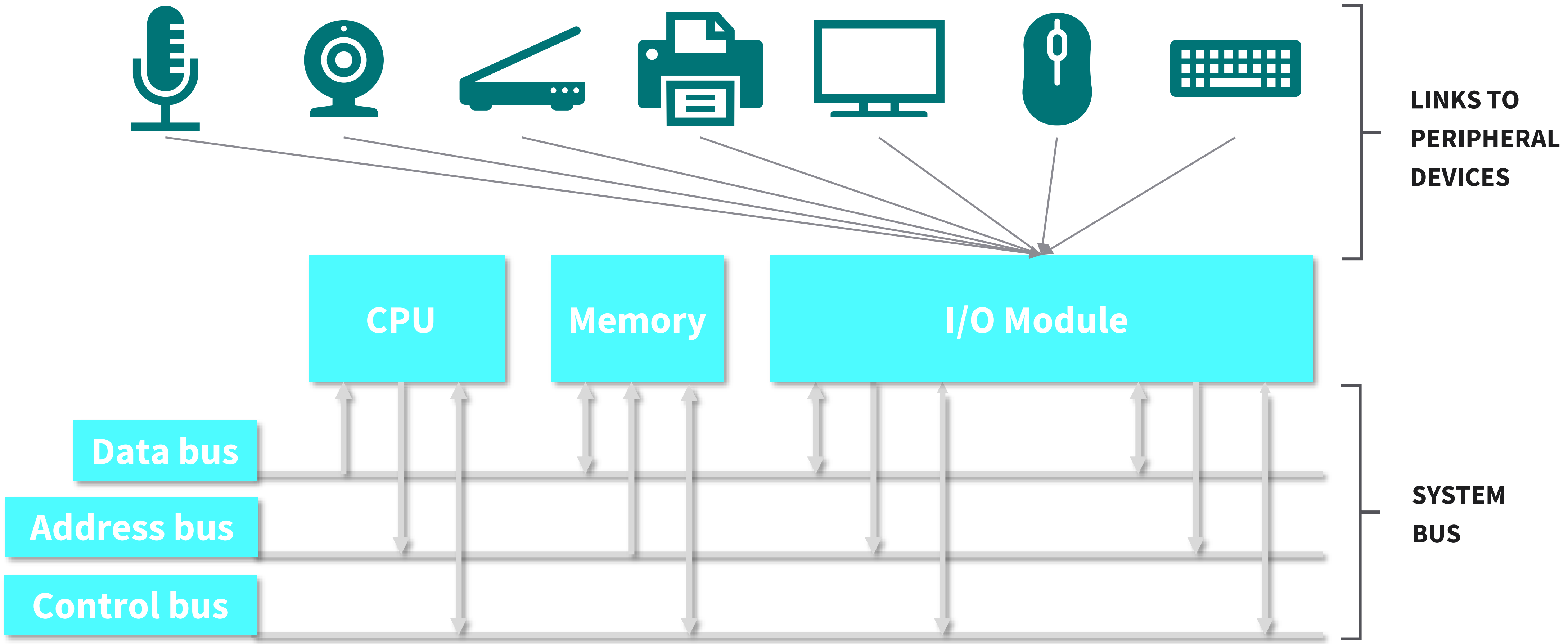


Source of the image: SNIA Global Education, 2015.

HARDWARE AND COMPUTER ARCHITECTURES



GENERIC MODEL OF AN I/O MODULE



EXTERNAL DEVICES

- Provide a means of exchanging data between the external environment and the computer
- Attach to the computer by a link to an I/O module
  - The link is used to exchange control, status, and data between the I/O module and the external device
- Peripheral device
  - An external device connected to an I/O module

Three categories:

Human readable	Machine readable	Communication
– suitable for communicating with the computer user – video display terminals (VDTs), printers	– suitable for communicating with equipment – magnetic disk and tape systems, sensors and actuators	– suitable for communicating with remote devices such as a terminal, a machine-readable device, or another computer

The major functions for an I/O module fall into the following categories:

Control and timing

– coordinates the flow of traffic between internal resources and external devices

Processor communication

– involves command decoding, data, status reporting, address recognition

Device communication

– involves commands, status information, and data

Data buffering

– performs the needed buffering operation to balance device and memory speeds

Error detection

– detects and reports transmission errors

Three techniques are possible for I/O operations:

Programmed I/O

- Data is exchanged between processor and the I/O module.
- Processor executes a program that gives it direct control of the I/O operation.
- When the processor issues a command, it must wait until the I/O operation is complete.
- If the processor is faster than the I/O module, it wastes processor time.

Interrupt-driven I/O

- Processor issues an I/O command, continues to execute other instructions, and is interrupted by the I/O module when the latter has completed its work.

Direct memory access (DMA)

- The I/O module and main memory exchange data directly without processor involvement.

DEVICE DRIVER AND OPERATING SYSTEM

What do you with a device?

- {read, write}
- {read only}
- {write only}

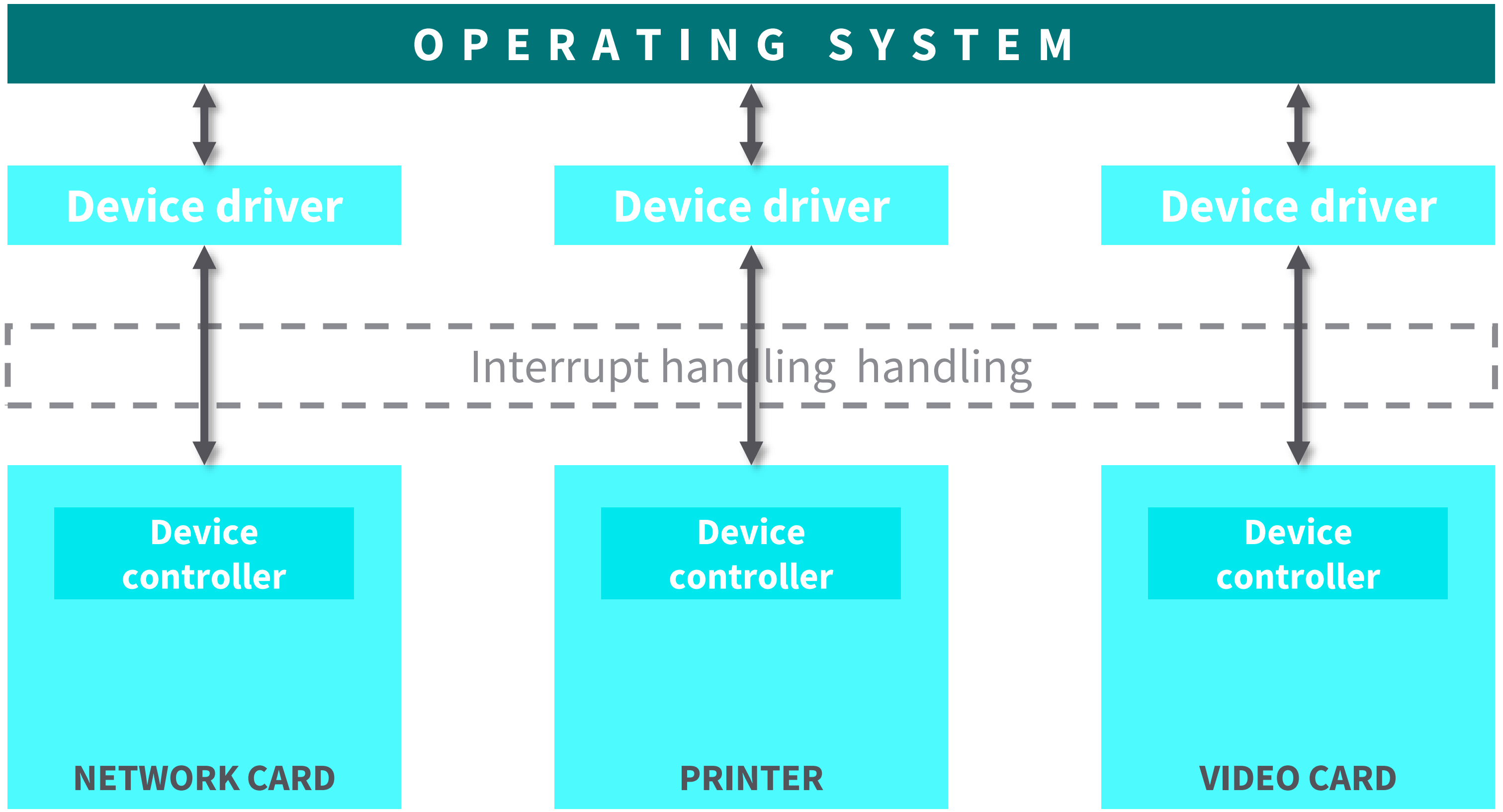
Let’s look at some examples:

- USB device
- CD-ROM
- LED Display
- …

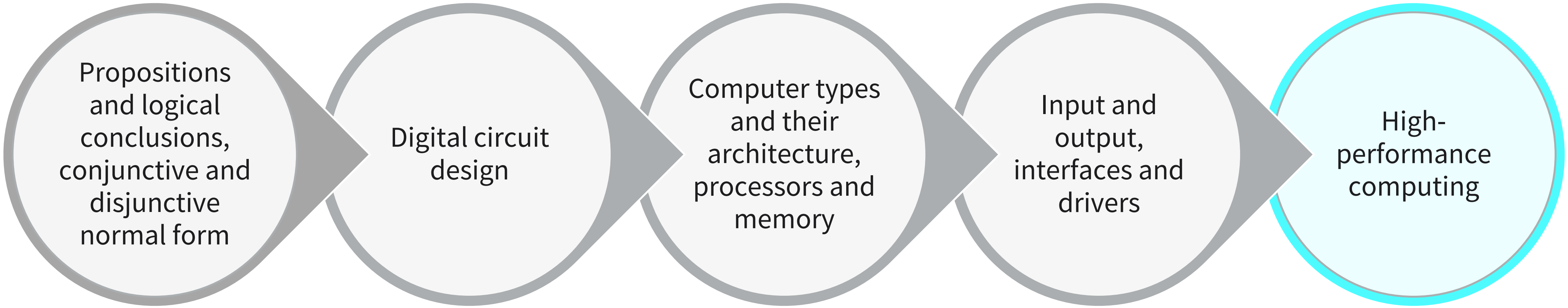
What do you do with a file?

- open, close, read, write, …

File is an excellent abstraction for devices.



HARDWARE AND COMPUTER ARCHITECTURES



Supercomputers are designed for high calculation performance and are used in simulation, product development, and research to solve **Grand Challenges.**

Performance is measured in floating point operations per second (FLOPS).

Grand Challenges are fundamental complex problems from engineering and natural sciences.

Examples are:

- quantum states for quantum computing
- properties of matter for quantum chromodynamics
- electronic structures of catalysts and semiconductors
- plasma dynamics in fusion reactors
- combustion dynamics
- fluid raw material dynamics
- weather dynamics

Computer performance

Name	Unit	Value
kiloFLOPS	kFLOPS	10^3
megaFLOPS	MFLOPS	10^6
gigaFLOPS	GFLOPS	10^9
teraFLOPS	TFLOPS	10^{12}
petaFLOPS	PFLOPS	10^{15}
exaFLOPS	EFLOPS	10^{18}
zettaFLOPS	ZFLOPS	10^{21}
yottaFLOPS	YFLOPS	10^{24}

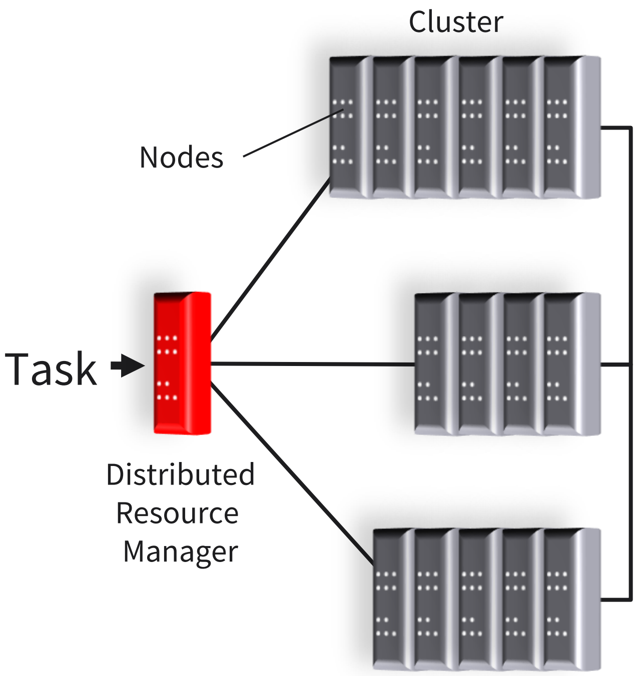
Computer cluster: High-performance computers typically consist of multiple homogeneous computers in parallel processing with centralized administration.

Grid-Computing uses multiple loosely coupled clusters to create a virtual high-performance computer. While a grid is usually not managed centralized, it requires a **Distributed Resource Manager** to distribute the tasks.

Typical fields of application are

- scientific calculations (e.g., CERN, SETI).
- simulation (e.g., drag coefficients).
- product innovation/development.

Service Grids are conceptual predecessors of **Cloud Computing**.



REVIEW STUDY GOALS

- Understand the basic language and concepts of propositional logic.
- Learn how to create truth tables.
- Find out how to use the conjunctive and disjunctive normal form.
- Learn about the basic concepts of digital circuits and logic gates.

REVIEW STUDY GOALS

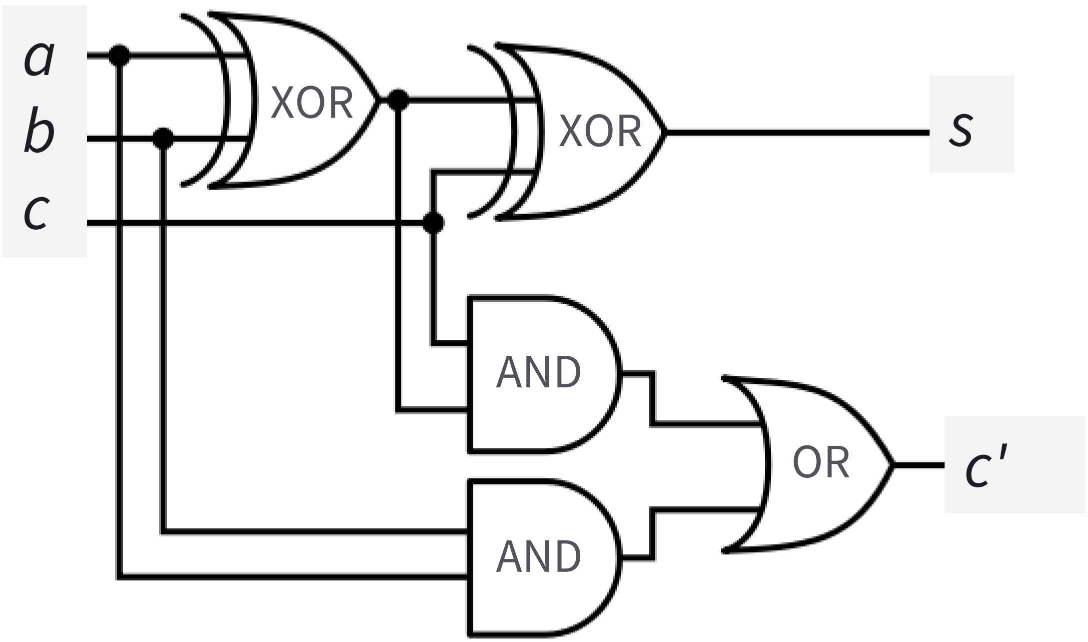
- Learn the basic elements of computer architecture.
- Understand how processors and memory work.
- Explore how a computer processes input and output.
- Learn how operating systems and hardware communicate.
- Get an overview of high-performance computing.

SESSION 3

TRANSFER TASK

TRANSFER TASK

The following is a combination of logical operators as electronic circuits.



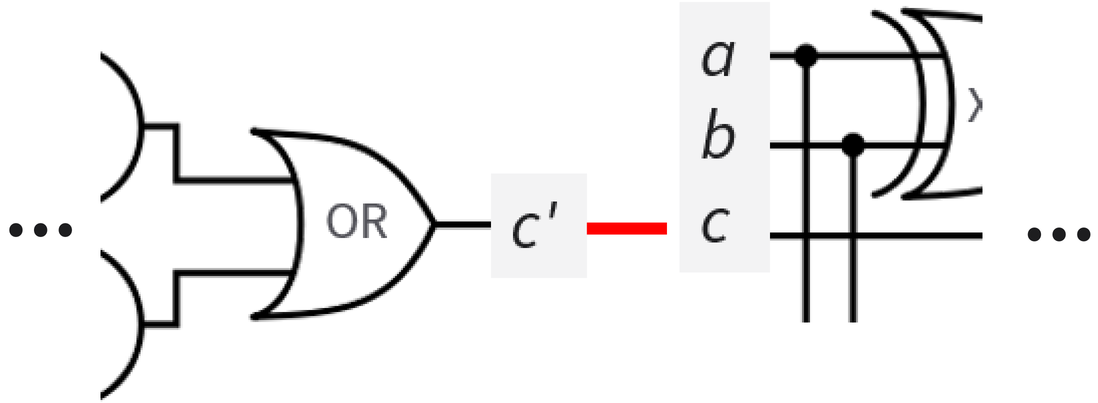
TRUTH TABLE

a	***b***	***c***	***s***	***c′***
0	0	0		
0	1	0		
1	0	0		
1	1	0		

a	***b***	***c***	***s***	***c′***
0	0	1		
0	1	1		
1	0	1		
1	1	1		

TRANSFER TASK

1. Complete the truth table next to the circuit.
2. What is the circuit doing?
3. Combine 4 of these circuits by connecting **c’** with **c**:



4. What is the combined circuit doing then?
5. How are multiplications e.g., 12x17 performed using such a simple circuit?

TRANSFER TASK
PRESENTATION OF THE RESULTS

Please present your results.
The results will be discussed in plenary.

1. If you wanted to make sure that p and q both were true, you would use which connector?
   a) disjunction
   b) conjunction
   c) negation
   d) implication

2. To list all possible outputs for logic inputs, you would create what kind of table?
   a) logic
   b) data
   c) truth
   d) circuit

3. An XOR gate means that the inputs:
   a) are the same
   b) are different
   c) are both “1”
   d) are both “0”

4. A supercomputer uses what kind of processing?
   a) serial
   b) intermittent
   c) parallel
   d) NAND

5. What does an operating system provide to interact with the user?

   a) interface
   b) CPU
   c) motherboard
   d) device driver

LIST OF SOURCES

Brookshear, G. & Bylow, D. (2011). *Computer science: An overview* (11th ed.). Pearson.

Dale, N. & Lewis, J. (2020). *Computer science illuminated* (7th ed.). Jones & Bartlett Learning.

Downey, A. B. & Mayfield, C. (2020). *Think Java: How to think like a computer scientist.* O'Reilly.

Filho, W. F. (2018). *Computer science distilled: Learn the art of solving computational problems.* Code Energy LLC.

Petzold, C. (2000). *Code: The hidden language of computer hardware and software.* Microsoft Press.

SNIA Global Education (2015). https://www.snia.org/sites/default/files/SDC15_presentations/persistant_mem/JeffChang-ArthurSainio_NVDIMM_Cookbook.pdf

Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM 9* (1), 36-45. https://doi.org/10.1145/365153.365168

Whitington, J. (2016). *A machine made this book: Ten sketches of computer science.* Coherent Press.