Laboratory consists of a set of exercises in **LogiSim**.

(If you have a MacBook, use the online version https://circuitverse.org/simulator)

Each exercise level has its own weight:

- **Easy** – 0.2 points
- **Medium** – 0.4 points
- **Hard** – 0.7 points

You must complete exercises from **all three levels**.

Choose a different exercise *from each subcategory* and level.

The goal of each is to accumulate **10 points**.

Report structure:

- ⇨ **Title page**
- ⇨ **Purpose of the work**
- ⇨ **Screenshots of completed exercises**
- ⇨ **Explanation for each exercise**
- ⇨ **Description of circuit functionality**
- ⇨ **Conclusion**

P.S. – The exercise number is represented as follows:

Exercise 8 Easy (+subcategory name) = (8E) and task for every exercise.

Easy:

⇨ ***Basic Logic Gates:***

1. NOT Gate – Build a NOT gate that inverts the input signal.
2. AND Gate – Build an AND gate with two inputs that outputs "1" only when both inputs are "1".
3. OR Gate – Build an OR gate with two inputs that outputs "1" when at least one input is "1".
4. NAND Gate – Build a NAND gate with two inputs that outputs "0" only when both inputs are "1".

5. NOR Gate – Build a NOR gate with two inputs that outputs "0" when at least one input is "1".
6. XOR Gate – Build an XOR gate with two inputs that outputs "1" when only one input is "1".
7. XNOR Gate – Build an XNOR gate with two inputs that outputs "1" when both inputs are the same.

⇨ ***LED Control Circuits:***

8. Single LED Control – Build a circuit to turn an LED on/off using a button.
9. Dual LED Control – Build a circuit to control two LEDs using a button.

⇨ ***Advanced Logic Gate Implementations:***

10. 4-input AND using NAND Gates – Implement a 4-input AND gate using only NAND gates.
11. 5-input OR using NOR Gates – Implement a 5-input OR gate using only NOR gates.
12. 4-input XOR using XNOR Gates – Implement a 4-input XOR gate using only XNOR gates.

Medium:

⇨ ***Flip-Flops & Registers:***

1. SR Flip-Flop – Build an SR flip-flop that stores an output state when set/reset signals are applied.
2. D Flip-Flop – Build a D flip-flop that stores the input state when the clock is activated.
3. J-K Flip-Flop – Build a J-K flip-flop.
4. T Flip-Flop – Build a T flip-flop.
5. Register – Build a register that stores multiple bits of data.
6. Binary Counter with Overflow – Build a binary counter that counts from 0 to (2^n)-1, where n is the number of bits.

⇨ ***Number System Conversions:***

7. Binary-to-Decimal Converter – Design a circuit that converts a binary number into decimal.
8. Decimal-to-Binary Converter – Design a circuit that converts a decimal number into binary.

⇨ ***Multiplexers & Demultiplexers:***

9. 2-Input Multiplexer – Build a multiplexer with 2 inputs and 1 select input.
10. 4-Input Multiplexer – Build a multiplexer with 4 inputs and 2 select inputs.
11. 2-Output Demultiplexer – Build a demultiplexer with 2 outputs and 1 select input.
12. 4-Output Demultiplexer – Build a demultiplexer with 4 outputs and 2 select inputs.

Hard:

⇨ ***Registers:***

1. Clocked Register – Build a register with a clock signal that stores input data sequentially.
2. Bit Reversal Register – Build a register that reverses the order of bits in an input signal.

⇨ ***Timing & Signal Processing Circuits:***

3. Delay Circuit – Implement a circuit that introduces a specified time delay in the output signal.
4. Oscillator Circuit – Implement a circuit that generates a periodic oscillation signal.

⇨ ***Comparators & Encoders/Decoders:***

5. 4-bit Comparator – Compare two 4-bit numbers and determine if one is greater than, less than, or equal to the other.
6. 4-to-16 Decoder – Implement a decoder circuit with 4 input bits and 16 output lines.
7. 4-input Priority Encoder – Implement a priority encoder where the highest-priority input determines the output.

⇨ ***Arithmetic Circuits:***

8. 8-bit Full Adder – Perform binary addition with carry propagation on two 8-bit numbers.

9. 8-bit Multiplier – Implement a circuit that multiplies two 8-bit binary numbers.
10. 8-bit Divider – Implement a circuit that divides an 8-bit binary number by another 8-bit binary number.
11. Booth's Multiplication Circuit – Implement an 8-bit multiplication circuit using Booth's algorithm.
12. Exponentiation Circuit – Compute the exponentiation of an 8-bit number raised to another 8-bit number.
13. Montgomery Exponentiation Circuit – Perform modular exponentiation using Montgomery's method.

⇨ ***Bitwise Operations & Logical Transformations:***

14. Base-2 Logarithm Circuit – Compute the binary logarithm ($\log_2$) of an 8-bit number.
15. Left Rotation Circuit – Perform a left rotation operation on an 8-bit number.
16. Bitwise Negation Circuit – Invert all bits of an 8-bit number (bitwise NOT).
17. Two's Complement Circuit – Compute the two's complement of an 8-bit number (for signed arithmetic).

⇨ ***Number System Conversions:***

18. Binary to Decimal Conversion Circuit – Convert an 8-bit binary number to decimal.
19. Decimal to Binary Conversion Circuit – Convert an 8-bit decimal number to binary.
20. Decimal to Hexadecimal Conversion Circuit – Convert an 8-bit decimal number to hexadecimal.
21. Hexadecimal to Decimal Conversion Circuit – Convert an 8-bit hexadecimal number to decimal.
22. Binary to Hexadecimal Conversion Circuit – Convert an 8-bit binary number to hexadecimal