MARMARA UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

# CSE 3015 TERM PROJECT REPORT

Esra UĞURBAŞ

Umut Çağlar ÇELİK

# 1. Project Description

This project involves the design and implementation of a processor that supports specific instruction set architecture (ISA). The processor is developed using Logisim and Verilog HDL. Additionally, an assembler is implemented to convert assembly language instructions into machine code. The project phases include designing the ISA, developing the processor in Logisim, implementing it in Verilog HDL, and testing the design thoroughly.

# 2. Instruction Set Architecture (ISA)

The processor supports the following instructions:

- **Arithmetic/Logical Operations:** ADD, OR, NAND, SUB, SLL
- **Immediate Operations:** ADDI, ORI, NANDI, SUBI, SLLI
- **Memory Operations:** LD, ST
- **Branching:** JUMP, BEQ, BLT, BGT, BLE, BGE

## Instruction Format

- **Opcode:** 5 bits
- **Register Addresses:** 4 bits each (SRC1, SRC2, DST)
- **Immediate Value:** 12 bits (for immediate instructions)
- **Addressing Mode:** PC-relative for branching and jump instructions

# 3. Design and Implementation

## a. Assembly Language Design

This Python script functions as an assembler, converting assembly language instructions into machine code and formatting them as hexadecimal for use in a simulated processor. It supports a defined set of opcodes and validates registers and immediate values during the conversion process. Each instruction is parsed, translated into binary based on its type (e.g., R-type, I-type), and finally converted to hexadecimal.

The script reads assembly code from an input file, removes comments, assembles the instructions into machine code, and writes the formatted hexadecimal output to an output file. This output is compatible with Logisim and similar simulators, allowing seamless testing of the processor design.

## b. Logisim Design

The processor was implemented in Logisim using the following components:

### i. Datapath Components

1. Program Counter (PC): A 20-bit register to store the current instruction address.
2. Instruction Memory: Stores program instructions.
3. Register File: Contains 16 registers for storing data.
4. ALU: Performs arithmetic and logic operations based on control signals.
5. Data Memory: A read-write memory for data storage.

### ii. Control Unit

- A finite state machine (FSM) was used to generate control signals for the datapath components.

## c. Verilog Implementation

The Verilog code translates the theoretical design of the processor into a functional, testable hardware model. It ensures that all components work together as a cohesive unit to execute the program correctly.