Course Project: MIPS Simulator
CS3339, Fall 2025
Due: Friday, Dec 5 11:59 PM

## Description

You will develop an instruction-by-instruction simulator for a pipelined MIPS processor. The simulator will execute instructions from the input assembly file one-by-one and display the contents of the register file and memory after the program has finished execution.

Your simulator must implement the MIPS instructions listed in the table below.

| Opcode | Description |
|---|---|
| ADD | signed integer addition |
| ADDI | add immediate |
| SUB | signed integer subtraction |
| MUL | integer multiplication |
| AND | bitwise and operation |
| OR | bitwise or operaton |
| SLL | shift left logical |
| SRL | shift right logical |
| LW | load word |
| SW | store word |
| BEQ | branch if equal to |
| J | jump |
| NOP | no op |

## Implementation

You can implement the simulator in any language of your choosing. However, make sure that your README has clear instructions for building and running the simulator. Below are some more implementation notes:

1. Although you will not be graded on the quality of the code, you should follow good design principles: choose appropriate data structures (or classes) to represent registers, memory, ALU and other elements of the MIPS processor
2. The simulator should be faithful to the 5-stage pipelined MIPS processor we have studied in class. For example, although it is possible to simulate the pipelined design without the use of state registers, you should not do so. Make sure all critical elements are represented in your simulator.
3. Internally, the assembly instructions do not need to be stored in binary format. However, your processor should still perform the task of decoding (i.e., extract opcode, register numbers, etc.)

For example, after reading the input file of an assembly program, you can provide the listing of what would be the binary representation of the program.

## Input

Input to your program is going to be a valid assembly program in MIPS. You may assume that your MIPS program is free of data and control hazards. The input program can be written by hand or generated by a compiler. In either case, it will contain only the instructions listed above.

## Output

The final output of your simulator will be the contents of the register file and memory. These two components represent the state of the machine after completing the execution of the program. **The simulator should also support a debug mode that prints the contents of the register file, the state registers, and the emitted control signals after executing each instruction.** You are free to choose the exact formatting of this output, but you should make an attempt to make it intuitive. Take a look at the MARS simulator or CPUlator implementation for ideas.

1. MARS Homepage
2. Github - MARS
3. CPU Simulator

## Submission

You will submit a link to the git repository that hosts the source. The repository should have a README with detailed instructions for building and running your simulator. Use git.txstate.edu not github. You can publish on github after the semester is over.

## Presentation & Grading

You should prepare a 20 minute presentation (15 minute presentation and demo, 5 minute Q&A) to be delivered on the final weeks of class. Slides should be used to convey high-level ideas about the project. The presentation should cover at a minimum:

1. **15 pts** - An overview of the components of the project (e.g. how it's built, how it's used, etc)
2. **30 pts** - An example run where an input assembly file that uses each instruction is fed through and processed

3. **20 pts** - A showcase of how we can observe the state of the simulator between instructions
4. **25 pts** - Peer evaluation - part of your grade will be determined on how your peers rate your contribution to the project.

## Teams

The project will have to comprise of teams of 3 - 4 individuals. **By the end of Week 7 (Friday, Oct 17th, 2025), have 1 person from the team email me the team members. Presentation dates are on a first-come-first-served basis and must be selected from the dates specified for presentations from the class schedule. The rest of the presentation dates/teams will be randomly-drawn for determining who is to present first versus last.**