RISC-V Simulator

-Devansh Verma (CS20B009)
-Sidhant Hanwate(CS20B041)
*[Team: **Valyrian Steel**]*

IDEAS

1) Introduction

This RISC V Simulator is written in **C++**.
It simulates the code written in RISC-V, with some of the modifications and assumptions (given below under *Assumptions and Limitations*). The simulator will take a

<filename>.asm file with the RISC V code written in it.

Sample codes and their outputs are present in *sample_codes* file in the github repo along with the screenshots of outputs – for both the cases- WITH FORWARDING and WITHOUT FORWARDING. The *main.cpp* file is the one that needs to be executed for simulating the assembly code.

GitFront link for the repository.

Functions

1) form_tokens(): takes any.*asm* file as input and converts it to a 2D vector of string tokens.

2) jump(): this function forms a map ‘jt’ that stores the indexes of the next instruction for all the locations mentioned in the code.

3) assign(): assigns all the registers their default values.

4) recog_instr(): recognizes the below mentioned instructions and executes them.

- li
- lw
- sw
- add
- sub
- mul
- div
- addi

- bne
- beq
- jal
- # (for comments)
- end
- bgt
- blt

5) display_reg(): displays registers along with their corresponding values.

6) syntax_error_checker(): checks and prints syntax errors, if any. *reg_name_check()* is invoked for that purpose.

7) stChecker(): checks if previous row in f_staller had a stall at same column

8) **wstChecker():** checks if previous row in wf_staller had a stall at same column

9) **with_forwarding():** pipelining with data forwarding is implemented here.

10) **without_forwarding():** pipelining without data forwarding is implemented here.

Assumptions and Limitations

1. The Simulator has no ecall functions.

2. We have provided an end keyword to end the program successfully.

3. The x2 register which is the Stack pointer register will be assigned value 0. But, this value can be changed later on.

4. As per reference from the Problem Statement pdf, no jalr function has been implemented.

5. All registers have been assigned with 0, as the default value.

6. The output of the program would be a list of all registers with the latest updated values, one can also query for value stored in a particular register.

   User is given the option to execute pipeling with/without data forwarding. Accordingly the pipeline stages are displayed.

   In case of any syntax error, eg. Invalid register name, appropriate error is displayed.

7. To create a location, the user has to mention the name of the location on the single line and leave the rest of the line blank(may add comments). The operations to be performed can be mentioned in the following lines.