CSE261 Project 2: Building a Simple MIPS Simulator

Due 11:59 PM, Dec 15th KST

1 Overview

This project is to build a simulator of a subset of the MIPS instruction set. The simulator loads a MIPS binary into a simulated memory and executes the instructions. Instruction execution will change the states of registers and memory.

2 Simulation Details

For a given input MIPS binary (the output binary file from the assembler built-in Project 1), the simulator must be able to mimic the behaviors of the MIPS ISA execution.

2.1 States

The simulator must maintain the system states, which consist of the necessary register set (R0-R31, PC) and the memory. The register and memory must be created when the simulation begins.

2.2 Loading an input binary

For a given input binary, the loader must identify the text and data section sizes. The text section must be loaded to the simulated memory from the address 0x400000. The data section must be loaded to the simulated memory from the address 0x10000000. In this project, the simple loader does not create the stack region.

2.3 Initial States

- PC: The initial value of PC is 0x400000.
- Register: All values of R0 to R31 are set to zero.
- Memory: You may assume all initial values are zero, except for the loaded text and data sections.

2.4 Instruction execution

With the current PC, 4 Byte from the memory is read. The simulator must parse the binary instruction and identify what the instruction is and what are the operands. Based on the MIPS ISA, the simulator must accurately mimic the execution, which will update either the PC, registers, or memory.

2.5 Completion

The simulator must stop after executing a given number of instructions.

2.6 Supported instruction set

ADDIU	ADDU	AND	ANDI	BEQ	BNE	J
JAL	JR	LUI	LW	LA	NOR	OR
ORI	SLTIU	SLTU	SLL	SRL	SW	SUBU

3 Download Project2 Repository

You can download the skeleton code from the 2021-Fall-Computer-Architecture/unistudentID/Project2 repository to the server or local machines. Then you are ready to start the project.

1. Go to the each gitlab page. The page will contain project2 directory.
2. Change directory to the location you want to clone your project and clone!.
3. You will get the clone repo in your machine.

Be sure to read the README.md file for some useful information. It includes the explanation of each file and which files you are allowed to modify for this project.

If you do not want to use the skeleton code, it is allowed to write code from scratch. However, you are supposed to follow the input and output file format because the grading script works on the provided *sample_input* and *sample_output* files described in the following section.

4 Simulator Options and Output

4.1 Options

```
python main.py [—m addr1:addr2] [−d] [−n num_instr] inputBinary
```

- -m : Dump the memory content from addr1 to addr2
- -d : Print the register file content for each instruction execution. Print memory content too if -m option is enabled.
- -n : Number of instructions simulated

The default output is the PC and register file content after the completion of the given number of instructions. If -m option is specified, the memory content from *addr1* to *addr2* must be printed too.

If –d option is set, the register (and memory dump, if –m is enabled) must be printed for every instruction execution.

4.2 Formatting Output

PC and register content must be printed in addition to the optional memory content. You should print the output with standard output.

1. If you type the command line as below, the output file should show only PC and register values like Figure 1.

   ```
   $ python main.py −n 0 input.o
   ```

2. If you type the command line as below, the output file should show memory contents of specific memory region, PC and register values like Figure 2.

   ```
   $ python main.py —m 0x400000:0x400010 −n 0 input.o
   ```

3. The functions for printing the memory and register values are provided in the *util.py* file.

5 Grading Policy

Grades will be given based on the 7 test cases project provided in the *sample_input* directory. Your simulator should print the exact same output as the files in the *sample_output* directory.

We will be automating the grading procedure by seeing if there are any differences between the files in the *sample_output* directory and the result of your simulator executions. Please make sure that your outputs are identical to the files in the *sample_output* directory.

You are encouraged to use the diff command to compare your outputs to the provided outputs.

```
$ python main.py -m 0x10000000:0x10000010 -n 50 example01.o > my_output
$ diff -Naur my_output sample_output/example01
```

If there are any differences (including whitespaces) the diff program will print the different lines. If there are no differences, nothing will be printed.

There are 7 test codes to be graded and you will be granted 10 score for each correct binary code and **being ”correct” means that every digit and location is the same** to the given output of the example. If a digit is not the same, you will receive **0 score** for the example.

6 Submission

The grading will be on the code at **Master branch** at the deadline. Any other branches are not considered as submission of the assignments. Please make sure that you can see the same result on the submission on the master branch as on your local machine.

7 Updates/Announcements

If there are any updates to the project, including additional tools/inputs/outputs, or changes, we will post a notice on the BB, and will send you an email using the BB system. Please check the notice or your email for any updates.

One concern is that we are using Python as an assignment language, there is plenty of packages you can leverage. Therefore, we restrict you not to import external packages. If you have question about using external package, you must email to TAs to discuss about it. Also, if you send email of using external package, you need to demonstrate ”Why you have to use this package?” to TAs. If not sending email and use external packages, we regard this behavior as a cheating.

8 Misc

We will accept your late submissions; the deduction is 10% of each day. The minimum score is 50% of total score. Also, you have 4 **sleep days** as professor announced. Please do not give up the project.

Be aware of plagiarism! Although it is encouraged to discuss with others and refer to extra materials, copying other students or opening your source code is strictly banned. The TAs will compare your source code with other’s code. If you are caught, you will receive a penalty for plagiarism.

Last semester, we found a couple of plagiarism cases through an automated tool. Please do not try to cheat TAs. If you have any requests or questions regarding administrative issues (such as late submission due to an unfortunate accident, git push or repository is not working) please send an email to the TAs (tykim8191@unist.ac.kr / heelim@unist.ac.kr).