CSE26101 Project 2: Building a Simple MIPS Simulator

Due 11:59 PM, Nov 07th

1. Overview

In this project, you will 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 Supported Instruction Set

Your MIPS simulator should support the following instructions (same as in PS1):

ADD	ADDI	ADDIU	ADDU	AND	ANDI	BEQ	BNE	J
JR	LUI	LW	SLTIU	XOR	OR	ORI	SLT	SLTI
SLTU	SLL	SRL	SW	SUB	SUBU	JAL		

* Pseudo instructions

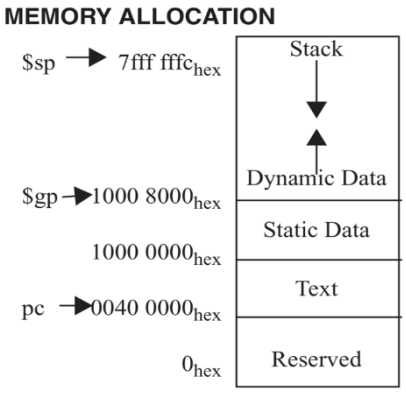
2.2 States

Your Simulator must maintain the system states, including the necessary register set (R0-R31, PC) and memory. The register and memory should be initialized at the beginning of the simulation.

2.3 Initial States

The memory allocation follows the same format as in PS1. The initial values are as follows:

PC: from 0x400000
Data Section: from 0x10000000
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 Loading an Input Binary

For a given input binary, the loader must identify the sizes of the text and data sections and load them to the corresponding memory addresses. **In this project, you do not need to create the stack region.**

2.5 Instruction Execution

To execute a given binary, the simulator should repeat the following process:

1. Read a 4-byte instruction from the current PC.
2. Parse and decode the binary instruction (PA1).
3. Mimic the execution of the decoded instruction accurately, according to the MIPS ISA, which will update either the PC, registers or memory.

2.6 Completion

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

3. Testing and Submission via PA Submit System (PASS)

We will use ***PA Submit System***, developed by Systems Software Lab of Ajou University, to manage the programming assignments. Please refer to the BlackBoard announcement for the PASS user manual.

3.1 Downloading the Skeleton Code

The skeleton code is available on BlackBoard under "Assignment" → "PA2: Building a Simple MIPS Simulator". Please download and read the skeleton code carefully before you begin working on PA2, you can start with the README file. Some macros and helper functions are given, please do not modify them and put your implementation under the "Fill the blanks" regions.

If you do not want to use the skeleton code, you are allowed to write code from scratch. However, you are supposed to follow the input and output file format exactly, as the grading script works on the provided sample input and sample output files provided to you.

3.2 Submission

You should zip the two files: *run.py* and *parse.py* as *ps2.zip*, and submit the zipped file on PASS, under "[Submission] PA2: Building a Simple MIPS Simulator". Please do not modify the file name or include unnecessary files in your submission. You can submit multiple times; **You can choose which submission will be used for grading.**

You should remove or comment out unnecessary codes before you submit to PASS, especially if it prints or generates log files. The automated grading program may fail to evaluate your assignment accurately.

3.3 Testing Locally

You can test your implementation locally with the following command:

```
$ python main.py -d inputBinary
```

-d: Print the register file content for each instruction execution. I

Your simulator should print the output with standard output. The output file should show memory contents of specific memory region, PC and register values.

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

You can simply run test.sh to test for default inputs.

3.4 Testing on PASS

You can test your implementation by submitting it on PASS, under "[Testing] PA2: Building a Simple MIPS Simulator". Your code will be automatically tested with the testcases we uploaded. You are allowed up to 100 submissions for testing.

4. Grading Criteria

Your assignment will be graded with 12 testcases. Each testcase is worth 1 point, no partial points will be given.

For late submissions, your slip days will be automatically subtracted; post-slip-day submissions will receive a 20% penalty from your score for each day late.

5. Updates/Announcements

We will post a notice on BlackBoard and Pizza if there are any updates to the assignment, please check BlackBoard and Piazza regularly.

Note that you are allowed to import and use external packages for this assignment. If you think it is necessary to use an external package, you must first get permission from the TAs by posting in Piazza. Your post must explain clearly why the package you intend to use is necessary. **Using an external package without permission will be regarded as plagiarism.**

7. Misc

Be careful about plagiarism! Last semester, we found a couple of plagiarism cases through an automated tool. If you are caught in "deep collaboration" with other students, you will split the score equally with your collaborators.
Please do not upload your code when posting in Piazza as it is considered cheating. If you think your question contains hints to other students, please make sure you post it as a "private post".

If you have any requests or questions (technical difficulties, late submission due to inevitable circumstances, etc.), please ask the TAs on Piazza.

We generally encourage the use of Piazza for discussions. However, for urgent issues, you can send an email to the TAs (minseok1335@unist.ac.kr(Head TA) / dyryu@unist.ac.kr / xinyuema@unist.ac.kr / garvel@unist.ac.kr).