SCE212 Project 2: Building a Simple MIPS Simulator

Due 11:59PM, May 31th

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`.
- Registers: 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 (same as Project 1)

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 SCE212/Project2 repository to the server or local machines. Then you are ready to start the project.

1) Go to the following page: http://github.com/csl-ajou/sce212-project2. The page is the Project2 repository.
2) Change directory to the location you want to clone your project and clone!
   `$ git clone http://github.com/csl-ajou/sce212-project2.git`
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

```
sce212sim [-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`.

```
$ ./sce212sim -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`.

```
$] ./sce212sim -m 0x400000:0x400010 -n 0 input.o
```

3. The functions for printing the memory and register values are provided in the `util.c`, and `util.h` files.

```
Current register values :
-------------------------------------
PC: 0x00400000
Registers:
R0: 0x00000000
R1: 0x00000000
R2: 0x00000000
R3: 0x00000000
R4: 0x00000000
R5: 0x00000000
R6: 0x00000000
R7: 0x00000000
R8: 0x00000000
R9: 0x00000000
R10: 0x00000000
R11: 0x00000000
R12: 0x00000000
R13: 0x00000000
R14: 0x00000000
R15: 0x00000000
R16: 0x00000000
R17: 0x00000000
R18: 0x00000000
R19: 0x00000000
R20: 0x00000000
R21: 0x00000000
R22: 0x00000000
R23: 0x00000000
R24: 0x00000000
R25: 0x00000000
R26: 0x00000000
R27: 0x00000000
R28: 0x00000000
R29: 0x00000000
R30: 0x00000000
R31: 0x00000000
```

Figure 1. Dump Register Values

```
Current register values :
-------------------------------------
PC: 0x00400000
Registers:
R0: 0x00000000
R1: 0x00000000
R2: 0x00000000
R3: 0x00000000
R4: 0x00000000
R5: 0x00000000
R6: 0x00000000
R7: 0x00000000
R8: 0x00000000
R9: 0x00000000
R10: 0x00000000
R11: 0x00000000
R12: 0x00000000
R13: 0x00000000
R14: 0x00000000
R15: 0x00000000
R16: 0x00000000
R17: 0x00000000
R18: 0x00000000
R19: 0x00000000
R20: 0x00000000
R21: 0x00000000
R22: 0x00000000
R23: 0x00000000
R24: 0x00000000
R25: 0x00000000
R26: 0x00000000
R27: 0x00000000
R28: 0x00000000
R29: 0x00000000
R30: 0x00000000
R31: 0x00000000

Memory content [0x00400000..0x00400010] :
-------------------------------------
0x00400000: 0x00000000
0x00400004: 0x00000000
0x00400008: 0x00000000
0x0040000c: 0x00000000
0x00400010: 0x00000000
```

Figure 2. Additionally dump memory

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.

```
$ ./sce212sim -m 0x10000000:0x10000010 -n 50
sample_input/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. Furthermore, we have provided a simple checking mechanism in the `Makefile`. Executing the following command will automate the checking procedure.

```
$ make test
```

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.

5. Submission

Before submitting your code, it is highly recommended to complete the work on the Linux server we provided, the Linux Virtual Machine, or Windows Subsystem for Linux. In this project, you should compress `parse.c` and `run.c` file into the `pa2-submission.tar.gz` file, and you just need to upload the `tar.gz` file to https://sslab.ajou.ac.kr/pasubmit. Please execute the following command to make `pa2-submission.tar.gz` file.

```
$ make submission
```

After then, you should check whether your code works well or not. Of course, you can test your code on `PAsubmit` as many as you want. Please make sure that you can see the same result on the submission site as well.

6. Updates/Announcements

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

7. Misc

We will accept your late submissions up to 5 days. For each late day, your score will lose 10% so that up to 50% for 5 days. Please do not give up!

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, `PAsubmit` is not working) please send an email to the TAs (tome01@ajou.ac.kr / jjw8967@ajou.ac.kr).