Introduction to Computer Architecture
Project 2

Single-cycle RISC-V CPU Simulator

Hyungmin Cho
Department of Computer Science and Engineering
Sungkyunkwan University

Project 2 Overview

- In Project 2, you'll implement an ***instruction simulator*** that supports a subset of RISC-V instructions
  - What is an instruction simulator? Your program reads instructions from the binary file and execute instruction one-by-one.
  - We only consider the **register values (proj2)** and data **memory contents (from proj3)**
  - At the end of the execution, your program prints out the current value of the registers, and that should match with the expected output on a real RISC-V processor.
- The basic rules (submission rule, etc…) are the same as Project 1, but please ask TAs if anything is unclear.

RISC-V Instructions to Support in Proj2

- `lui, addi, slti, sltiu, xori, ori, andi, slli, srli, srai, add, sub, sll, slt, sltu, xor, srl, sra, or, and`

- FYI, the following instructions will be additionally required in Proj3
  - `auipc, jal, jalr, beq, bne, blt, bge, bltu, bgeu, lb, lh, lw, lbu, lhu, sb, sh, sw`

Simulator Program Behavior

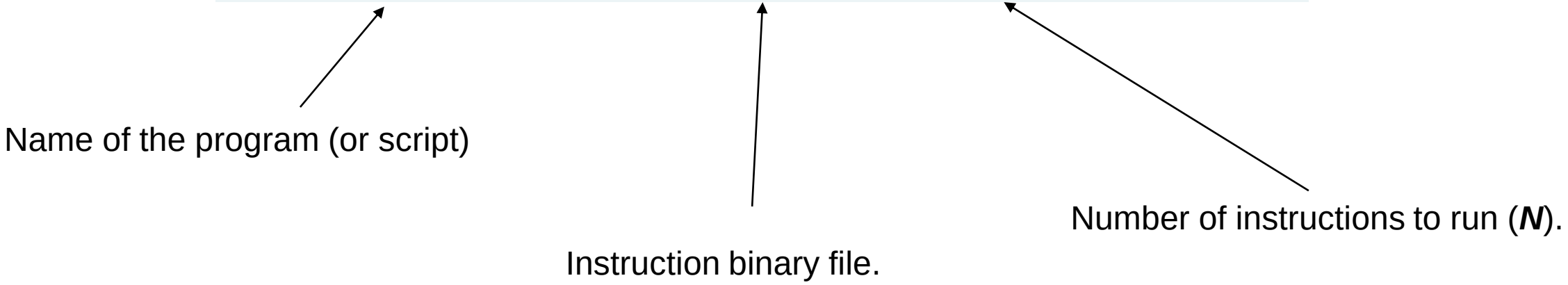
1. Your program takes 2 command-line arguments.

2. First argument: Input file name
   - This file includes binary instructions, same as proj1
   - For proj2, you can assume that there is no invalid instruction in the input

3. Second argument: Number of instructions to execute (***N***)
   - Your program simulates each instruction one-by-one
   - If ***N*** is smaller than the number of instructions in the input file, execute ***N*** instructions and then stop.
   - If ***N*** is bigger than the number of instructions in the input file, execute all instructions in the input file, and then print "`No more instructions`"

Simulator Program Behavior

```
# ./riscv-sim test1.bin 100
```



Registers

- All registers are initialized to zero (0x00000000) at the beginning.

- x0 is fixed to zero.

Register Output

- After the execution, print the final status of the registers (x0-x31)

```
# ./riscv-sim proj1_1.bin 10
x0: 0x00000000
x1: 0x00000000
x2: 0x00000123
x3: 0x87654321
x4: 0xffffffff
x5: 0x00000321
x6: 0xfffffffb
x7: 0xfffb0000
x8: 0x00fffb00
x9: 0x00000000
x10: 0x00000001
x11: 0x00000000
x12: 0x00000000
x13: 0x00000000
x14: 0x00000000
x15: 0x00000000
x16: 0x00000000
x17: 0x00000000
x18: 0x00000000
x19: 0x00000000
x20: 0x00000000
x21: 0x00000000
x22: 0x00000000
x23: 0x00000000
x24: 0x00000000
x25: 0x00000000
x26: 0x00000000
x27: 0x00000000
x28: 0x00000000
x29: 0x00000000
x30: 0x00000000
x31: 0x00000000
```

Reference Implementation

- We provide a reference implementation (without source code) in the following location.
  - `~swe3005/2022f/proj2/riscv-sim`

- If you have difficulties in implementing your simulator, try to compare the output with the reference implementation's output.
- If you think it is difficult to match the final results of the application at once, try to match the outputs one step at a time by changing the number of instructions (*N*)

```
~swe3005/2022f/proj2/riscv-sim ~swe3005/2022f/proj2/proj2_1.bin 1
~swe3005/2022f/proj2/riscv-sim ~swe3005/2022f/proj2/proj2_1.bin 2
~swe3005/2022f/proj2/riscv-sim ~swe3005/2022f/proj2/proj2_1.bin 3
...
~swe3005/2022f/proj2/riscv-sim ~swe3005/2022f/proj2/proj2_1.bin 10
~swe3005/2022f/proj2/riscv-sim ~swe3005/2022f/proj2/proj2_1.bin 11
```

Test Sample (1)

- ~swe3005/2022f/proj2/proj2_1.bin
- “proj2_1.bin” file represents the following assembly code with 10 instructions

```
addi x2, x0, 0x123
lui x3, 0x87654
ori x3, x3, 0x321
addi x4, x0, -1
andi x5, x3, 0x7FF
addi x6, x4, -4
slli x7, x6, 16
srli x8, x7, 8
slt x9, x8, x5
slti x10, x5, 0x7FF
```

```
./riscv-sim ~swe3005/2022f/proj2/proj2_1.bin 1
```

```
x0: 0x00000000
x1: 0x00000000
x2: 0x00000123
x3: 0x00000000
x4: 0x00000000
x5: 0x00000000
x6: 0x00000000
x7: 0x00000000
x8: 0x00000000
x9: 0x00000000
x10: 0x00000000
x11: 0x00000000
x12: 0x00000000
x13: 0x00000000
x14: 0x00000000
x15: 0x00000000
x16: 0x00000000
x17: 0x00000000
x18: 0x00000000
x19: 0x00000000
x20: 0x00000000
x21: 0x00000000
x22: 0x00000000
x23: 0x00000000
x24: 0x00000000
x25: 0x00000000
x26: 0x00000000
x27: 0x00000000
x28: 0x00000000
x29: 0x00000000
x30: 0x00000000
x31: 0x00000000
```

2nd argument is 1. Execute the first instruction only, and then print the register values.

Test Sample (1)

```
./riscv-sim ~swe3005/2022f/proj2/proj2_1.bin 10
```

```
x0: 0x00000000
x1: 0x00000000
x2: 0x00000123
x3: 0x87654321
x4: 0xffffffff
x5: 0x00000321
x6: 0xfffffffb
x7: 0xfffb0000
x8: 0x00fffb00
x9: 0x00000000
x10: 0x00000001
x11: 0x00000000
x12: 0x00000000
x13: 0x00000000
x14: 0x00000000
x15: 0x00000000
x16: 0x00000000
x17: 0x00000000
x18: 0x00000000
x19: 0x00000000
x20: 0x00000000
x21: 0x00000000
x22: 0x00000000
x23: 0x00000000
x24: 0x00000000
x25: 0x00000000
x26: 0x00000000
x27: 0x00000000
x28: 0x00000000
x29: 0x00000000
x30: 0x00000000
x31: 0x00000000
```

2nd argument is 10. Execute 10 instructions from the input file, and then print the register values.

```
./riscv-sim ~swe3005/2022f/proj2/proj2_1.bin 11
```

```
No more instructions
x0: 0x00000000
x1: 0x00000000
x2: 0x00000123
x3: 0x87654321
x4: 0xffffffff
x5: 0x00000321
x6: 0xfffffffb
x7: 0xfffb0000
x8: 0x00fffb00
x9: 0x00000000
x10: 0x00000001
x11: 0x00000000
x12: 0x00000000
x13: 0x00000000
x14: 0x00000000
x15: 0x00000000
x16: 0x00000000
x17: 0x00000000
x18: 0x00000000
x19: 0x00000000
x20: 0x00000000
x21: 0x00000000
x22: 0x00000000
x23: 0x00000000
x24: 0x00000000
x25: 0x00000000
x26: 0x00000000
x27: 0x00000000
x28: 0x00000000
x29: 0x00000000
x30: 0x00000000
x31: 0x00000000
```

2nd argument is 11. Since there are only 10 instructions in the file, execute the 10 instructions from the input file, and then print “no more instructions”. After that, print the register values.

Test Samples

- There are total 5 test cases.

```
~swe3005/2022f/proj2/proj2_1.bin
~swe3005/2022f/proj2/proj2_2.bin
...
~swe3005/2022f/proj2/proj2_5.bin
```

- Try to match the output of your simulator program to the output of the reference program.

Project Environment

- We will use the department's ~~In-~~Ui-Ye-Ji cluster
  - `swui.skku.edu`
  - `swye.skku.edu`
  - `swji.skku.edu`
  - ssh port: 1398

- If you have a problem with the account, send an e-mail to the server admin
  - inuiyeji-skku@googlegroups.com
  - Do not send an email that is not related to the account itself!
  - If you're not sure, ask the TAs first.

Makefile / Script

- If you're using C/C++, you need to submit Makefile as proj1
  - The output executable file name should be the same as proj1 (i.e., **`riscv-sim`**)
  - Beware about the **tabs** (not spaces) in Makefile
  - You need to include Makefile in your submission

- If you're using Python, add a script file named "`riscv-sim`"
  - Don't forget to add executable permission (`chmod +x riscv-sim`)
  - You need to include this script file in your submission

Makefile Example

- C

Makefile

```
CC=gcc
CCFLAGS=

#add C source files here
SRCS=main.c

TARGET=riscv-sim

OBJS := $(patsubst %.c,%.o,$(SRCS))

all: $(TARGET)

%.o:%.c
	$(CC) $(CCFLAGS) $< -c -o $@

$(TARGET): $(OBJS)
	$(CC) $(CCFLAGS) $^ -o $@

.PHONY=clean

clean:
	rm -f $(OBJS) $(TARGET)
```

- C++

Makefile

```
CXX=g++
CXXFLAGS=

#add C++ source files here
SRCS=main.cc

TARGET=riscv-sim

OBJS := $(patsubst %.cc,%.o,$(SRCS))

all: $(TARGET)

%.o:%.cc
	$(CXX) $(CXXFLAGS) $< -c -o $@

$(TARGET): $(OBJS)
	$(CXX) $(CXXFLAGS) $^ -o $@

.PHONY=clean

clean:
	rm -f $(OBJS) $(TARGET)
```

These are tabs (one Tab key), not spaces

Script Example

- Python (if your python file is riscv-sim.py)

`riscv-sim` ← Don't forget to give the excute permission: `chmod +x riscv-sim`

```
python3 riscv-sim.py ${@}
```

- Also, be aware of the python version on the server
  - python: python 2.7.17
  - python3: python 3.6.9

${@} means all arguments

Submission

- Clear the build directory
  - Do not leave any executable or object file in the submission
  - `make clean`

- Use the `submit` program
  - `~swe3005/bin/submit project_id directory_to_submit`

```
Submitted Files for proj2:
File Name                                 File Size       Time
--------------------------------------------------------------------------------
proj2-2020123456-Sep.05.17.22.388048074        268490                  Thu Sep  5 17:22:49 2020
```

- Verify the submission
  - **`~swe3005/bin/check-submission proj2`**

Submission

- Submit example
  - Your current working directory is “`/home/2022123456/`”
  - You want to submit the directory “`/home/2022123456/project_2`”

```
~swe3005/bin/submit proj2 project_2
```

Project 2 Due Date

- 2022, Nov. 14th, 23:59:59
  - Deadline extended

- No late submission