

Introduction to Computer Architecture

Project 1

RISC-V Binary Code

Hyungmin Cho

Department of Computer Science and Engineering
Sungkyunkwan University

Projects: Programming Assignments

- Project 1: Interpret RISC-V binary code
- Project 2: Simulate a single-cycle CPU
- Project 3: ?

Project 1 Goal

- Your program reads a binary file containing RISC-V machine code and **prints the assembly representation of the code**

Assembly language program (RISC-V)

```
slli x6, x11, 3
add  x6, x10, x6
ld   x5, 0(x6)
ld   x7, 8(x6)
sd   x7, 0(x6)
sd   x5, 8(x6)
jalr x0, 0(x1)
```

Machine (object, binary) code (RISC-V)

```
00000000001101011001001100010011
000000000011001010000001100110011
00000000000000110011001010000011
00000000100000110011001110000011
00000000011100110011000000100011
00000000010100110011010000100011
0000000000000001000000001100111
```



Disassemble

Program Interface

■ Executable file name

- ❖ The name of the program should be “**riscv-sim**”
- ❖ If using Python, name the main file “**riscv-sim.py**”

■ Input

- ❖ The input is a **binary file** containing RISC-V machine codes
- ❖ The input file name is provided as the first command-line argument.
- ❖ The file name length will not exceed 255 characters.

■ Output

- ❖ Print the disassembled instructions.
- ❖ Each line should display one instruction in the following format:

```
inst <instruction number>: <32-bit binary code in hex format> <disassembled instruction>
```

Disassembled Instruction Format

- Use lowercase instruction names
 - ❖ add, sub, sw, jal, ...
- Print registers using their register number
 - ❖ x0, x1, x20, ...
 - ❖ Do not to use the register name (e.g., sp, ra, ...)
- Follow these formatting rules:
 - ❖ Use a single space between the instruction name, registers, and immediate values.
 - ❖ Place a comma between registers (no comma at the end).
 - ❖ Do not use spaces for load / store / jalr offsets.
 - ❖ Examples of incorrect formatting:
 - ADD x1 x2 x3 ← uppercase instruction name, no commas between registers.
 - or x1 , x2, x3, ← incorrect comma placement, double space, comma at the end.
 - lw x1, 20 (x3) ← space between the offset value and parenthesis

Immediate

- Represent immediate and address values as **signed decimal (base 10) integers**.
 - ❖ `lw x16, 20(x29)`
 - ❖ `addi x29, x29, -16`
- Print all immediate values in signed decimal format after sign-extending to 32 bits.
 - ❖ Note: This may cause confusion for the branch offsets and `lui` immediates..

Assembly code

```
HERE:  
    addi x1, x2, 0x100  
    beq x1, x3, HERE  
    lui 0x1
```

Binary
Code

What you need to print

```
addi x1, x2, 256  
beq x1, x3, -4  
lui 4096
```

Instructions to support

- lui, auipc, jal, jalr, beq, bne, blt, bge, bltu, bgeu, lb, lh, lw, lbu, lhu, sb, sh, sw, addi, slti, sltiu, xori, ori, andi, slli, srli, srai, add, sub, sll, slt, sltu, xor, srl, sra, or, and
- If an instruction cannot be interpreted, print “unknown instruction”

Execution Results

```
$ ./riscv-sim /home/swe3005/2023s/proj1/proj1_1.bin
inst 0: 00208033 add x0, x1, x2
inst 1: 41450fb3 sub x31, x10, x20
inst 2: 008319b3 sll x19, x6, x8
inst 3: 00a4d433 srl x8, x9, x10
inst 4: 40a4d433 sra x8, x9, x10
inst 5: 0010e0b3 or x1, x1, x1
$
```

- Print results to stdout using standard print functions (e.g., print, printf).
- DO NOT save the output to a text file.

Test Input Files

- Obtain test input files from the department servers (swui.skku.edu, swye.skku.edu, swji.skku.edu)
 - ❖ `~swe3005/2023s/proj1/proj1_1.bin`
 - ❖ `~swe3005/2023s/proj1/proj1_2.bin`
 - ❖ ...
 - ❖ `~swe3005/2023s/proj1/proj1_7.bin`
- To check the contents of the binary file, use the 'xxd' program

```
$ xxd /home/swe3005/2023s/proj1/proj1_1.bin
00000000: 3380 2000 b30f 4541 b319 8300 33d4 a400
00000010: 33d4 a440 b3e0 1000
```

Test Result

- You may compare your program's output with the reference implementation.

```
$ /home/swe3005/2023s/proj1/riscv-sim /home/swe3005/2023s/proj1/ proj1_1.bin
inst 0: 00208033 add x0, x1, x2
inst 1: 41450fb3 sub x31, x10, x20
inst 2: 008319b3 sll x19, x6, x8
inst 3: 00a4d433 srl x8, x9, x10
inst 4: 40a4d433 sra x8, x9, x10
inst 5: 0010e0b3 or x1, x1, x1
```

Test Result

- Your output should EXACTLY MATCH with the reference output.
 - ❖ Any difference (e.g., extra character) is considered incorrect.
- Check your output correctness using the `diff` command.

```
$ ./riscv-sim /home/swe3005/2023s/proj1/proj1_1.bin > my_output.txt  
$ /home/swe3005/2023s/proj1/riscv-sim /home/swe3005/2023s/proj1/proj1_1.bin > ref_output.txt  
$ diff my_output.txt ref_output.txt  
$
```

← This will save your output to my_output.txt

← Nothing will be printed if two files match.
Otherwise, it will show the differences.

Project Rule – IMPORTANT!

- You may use C, C++, or Python.
 - ❖ If you want to use a different programming language, inform the TAs in advance.
- Your submission must be compilable and executable on the department Linux server.
 - ❖ Test your program on the server if you developed it on your own PC.
- Provide a Makefile to compile your code
 - ❖ The compiled executable should be named **riscv-sim**
 - ❖ If the build fails, your project score is **zero**.
 - ❖ No Makefile is needed for Python

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)
```

Script Example

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

`riscv-sim` ← Don't forget to give the execute 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

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
- First time users :
 - ❖ ID: your student ID (e.g., 2020123456)
 - ❖ Use the default password (unless you already changed your password...)
 - "pw"+Student_ID (last 8 digits)
 - e.g., The initial password for 2020123456 is pw20123456
 - ❖ MUST change your password after the first login (Use `yppasswd` command)

Submission

- Clear the build directory before submission.
 - ❖ Do not leave any executable or object files in your submission
 - ❖ `make clean`
- Use the submit program
 - ❖ `~swe3005/bin/submit project_id path_to_submit`
 - ❖ If you want to submit the 'project_1' directory...
 - `~swe3005/bin/submit proj1 project_1`

Submitted Files for proj1:

File Name	File Size	Time

proj1-2021123456-Sep.05.17.22.388048074	268490	Thu Sep 5 17:22:49 2021

- Verify the submission
 - ❖ `~swe3005/bin/check-submission proj1`

Multiple Submissions

- You may submit multiple times before the deadline.
- You will be scored using the **last** submission.

Project 1 Due Date

- 2023 Apr. 14th, Friday, 23:59:59
- **No late submission**

Project 0

- A dummy project to test the submission process.
- Not mandatory. It will not affect your grade.
- However, it is highly recommended to try the submission process
- Familiarize yourself with the submission system. Incorrect submissions from proj1 onwards will not be scored.

Project 0 – What to submit

- Submit your source code (+Makefile) that prints the following two lines of text to stdout (i.e., console).

```
inst 0: 00220020 add x0, x1, x2
inst 1: 8d420020 lw x2, 32(x10)
```

- No input file.
- Use project id “proj0”
 - ❖ `~swe3005/bin/submit proj0 your_project_0_directory`

Project 0 Due Date

- 2023 Apr. 7th, Friday, 23:59:59

Homework Discussions

- If you have questions about the programming assignment...
- Recommended way of discussion:
 - ❖ Use the i-Campus discussion section
 - ❖ Visit the office hour
- Avoid sending direct messages to the TAs unless the question requires privacy.