Introduction to Computer Architecture
Project 1

RISC-V Binary Code Read

Hyungmin Cho
Department of Software
Sungkyunkwan University

Projects: Programming Assignments

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

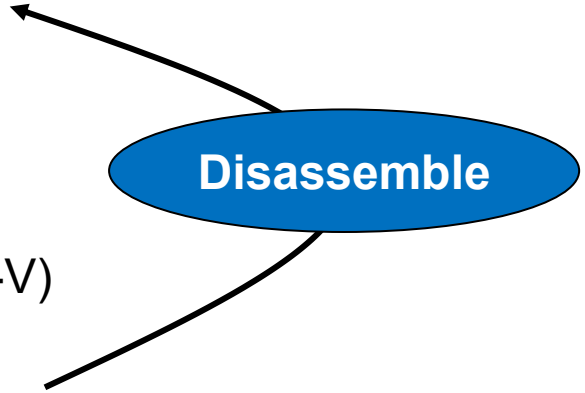
- Every step depends on the previous one.

Project 1 Goal

- Your program reads a binary file filled with RISC-V machine code, and **print 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
00000000011001010000001100110011
00000000000000011001100101000011
00000000100000110011001110000011
00000000011100110011000000100011
00000000010100110011010000100011
00000000000000001000000001100111
```

Program Interface

- Executable file name
  - The name of the program should be "**`riscv-sim`**"
  - If you're using Python, name the main file as "**`riscv-sim.py`**"

- Input
  - The input is a **binary file** that has RISC-V machine codes
  - The input file name is given by the first command-line argument
    - You can assume that the length of the input file name is no longer than 255

- Output
  - Prints the disassembled instruction
  - Each line prints in the following format

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

Disassembled Instruction Format

- Instruction name in lowercase
  - `add, sub, sw, jal, ...`
- Registers are printed by their register number
  - `x0, x1, x20`, ...
  - Do not to use the register name (`sp, ra`, ...)
- Formatting rule (strict!)
  - Single space between the instruction name, registers, immediate
  - Comma between the registers (no comma at the end)
  - No space for load/store/jalr offsets
    - Incorrect examples:
      - ADD x1 x2 x3 ← upper case instruction name, no commas between the registers.
      - or x1 , x2, x3, ← incorrect placement of comma, double space, comma at the end.
      - lw x1, 20 (x3) ← space between the offset value and parenthesis

Immediate

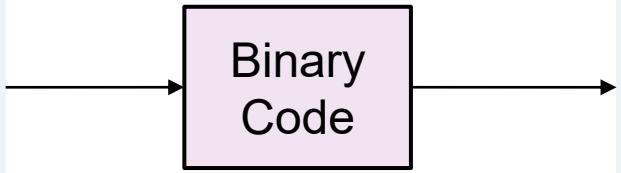
- Immediate and address values are represented in **signed decimal**
  - `lw x16, 20(x29)`
  - `addi x29, x29, -16`

- `All immediate values should be printed in signed decimal format after being sign-extended to 32bits`

- This may cause confusion for the branch offsets and `lui` immediates..

Assembly code

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



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 there is an instruction that can't be interpreted, print the following: "`unknown instruction`"

Execution Results

```
$ ./riscv-sim /home/swe3005/2022f/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
$
```

- Your program should print the results to `stdout`
  - i.e., just use normal print functions that prints to the console (e.g., print, printf, …)
- DO NOT save the output to a text file.

Test Input Files

- You can obtain test input files from the following location of the department servers (`swui.skku.edu`, `swye.skku.edu`, `swji.skku.edu`)
  - `~swe3005/2022f/proj1/proj1_1.bin`
  - `~swe3005/2022f/proj1/proj1_2.bin`
  - …
  - `~swe3005/2022f/proj1/proj1_6.bin`
- If you want to check the contents of the binary file, you may use the **xxd** program

```
$ xxd /home/swe3005/2022f/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/swe3005/2022f/proj1/riscv-sim /home/swe3005/2022f/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 as a wrong answer

- You can make sure your output is correct using the `diff` command

```
$ ./riscv-sim /home/swe3005/2022f/proj1/proj1_1.bin > my_output.txt
$ /home/swe3005/2022f/proj1/riscv-sim /home/swe3005/2022f/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 intend to use a different language, please inform the TAs in advance.

- Your submission must be compilable and executable on the department Linux server
  - Caution: some students complained their code is okay on their own PC but fails on the server. In most cases, such differences were caused by a bug in their code. The bug did not affect the output on their own PC, but the bug behaved differently on the server. Remember that your submission is scored on the department server. Make sure to test your program on the server if you created your program locally on your own PC.

- You need to provide a `Makefile` to compile your code
  - The name of the executable should be **`riscv-sim`**
  - If your build fails, your project score is **zero**.
  - Do not need if you're using 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)
```

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
  - Do not leave any executable or object file in the 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

- 2022 Oct. 14th, 23:59:59

- **No late submission**

Project 0

- A dummy project to test the submission process.
- Not mandatory. Will not affect your grade.

- Submit your source code (+Makefile) that prints the following

```
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

- 2022 Oct. 7th, 23:59:59

Homework Discussions

- If you need to ask something about the programming assignment…

- Recommended:
  - Use the i-Campus discussion section
  - Visit the office hour

- Please avoid sending direct messages **if the question is about the programming assignment.**
  - Use direct message or email only if you need some privacy…
  - If the question is not about the programming assignment, you can freely send direct messages.