香 港 中 文 大 學
The Chinese University of Hong Kong

CENG3420

Lab 2-2: RISC-V RV32I Simulator

Chen BAI& Hongduo LIU
Department of Computer Science & Engineering
Chinese University of Hong Kong
cbai@cse.cuhk.edu.hk&
hdliu21@cse.cuhk.edu.hk

Spring 2023

Outline

Introduction

Introduction
Overview

Use C programming language to finish lab assignments in following weeks.

- Lab 2.1 – implement an RISCV-LC Assembler
- Lab 2.2 – implement an RISCV-LC ISA Simulator
- Lab 3.x – implement an RISCV-LC Simulator

NOTICE

Lab2 & Lab3 are challenging!
Once you have passed Lab2 & Lab3, you will be more familiar with RV32I & a basic implementation!

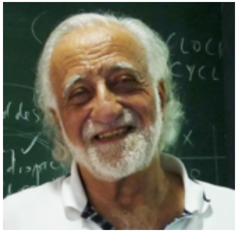
ISA Simulator

- Mimic the behavior of an instruction execution.
- ISA Simulator input: a binary file generated in Lab 2.1.

Introduction

Our Lab2 & Lab3 are Inspired by LC-3b

- LC-3b: **Little Computer 3, b** version.
- Relatively simple instruction set.
- Most used in teaching for CS & CE.
- Developed by Yale Patt@UT & Sanjay J. Patel@UIUC.



RV32I Instructions

Introduction to RV32I

- The instruction encoding width is 32-bit.
- Each instruction is aligned with a **four-byte** boundary in memory.
- RV32I only manipulates integer numbers and no multiplication or division.
- Our labs are based on the official RISC-V ISA manual: `https://github.com/riscv/riscv-isa-manual/releases/download/Ratified-IMAFDQC/riscv-spec-20191213.pdf`.

RV32I Instructions

31	30 25	24 21	20	19 15	14 12	11 8	7	6 0	
funct7		rs2		rs1	funct3	rd		opcode	R-type
imm[11:0]				rs1	funct3	rd		opcode	I-type
imm[11:5]		rs2		rs1	funct3	imm[4:0]		opcode	S-type
imm[12]	imm[10:5]	rs2		rs1	funct3	imm[4:1]	imm[11]	opcode	B-type
imm[31:12]						rd		opcode	U-type
imm[20]	imm[10:1]		imm[11]	imm[19:12]		rd		opcode	J-type

RV32I instructions base formats.

Integer Computational Instructions
Integer Register-Immediate Instructions

31 20	19 15	14 12	11 7	6 0
imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7
I-immediate[11:0]	src	ADDI/SLTI[U]	dest	OP-IMM
I-immediate[11:0]	src	ANDI/ORI/XORI	dest	OP-IMM

addi, andi, ori, xori

31 25	24 20	19 15	14 12	11 7	6 0
imm[11:5]	imm[4:0]	rs1	funct3	rd	opcode
7	5	5	3	5	7
0000000	shamt[4:0]	src	SLLI	dest	OP-IMM
0000000	shamt[4:0]	src	SRLI	dest	OP-IMM
0100000	shamt[4:0]	src	SRAI	dest	OP-IMM

slli, srli, srai

31 12	11 7	6 0
imm[31:12]	rd	opcode
20	5	7
U-immediate[31:12]	dest	LUI
U-immediate[31:12]	dest	AUIPC

lui

Integer Computational Instructions

Integer Register-Register Instructions

31 25	24 20	19 15	14 12	11 7	6 0
funct7	rs2	rs1	funct3	rd	opcode
7	5	5	3	5	7
0000000	src2	src1	ADD/SLT/SLTU	dest	OP
0000000	src2	src1	AND/OR/XOR	dest	OP
0000000	src2	src1	SLL/SRL	dest	OP
0100000	src2	src1	SUB/SRA	dest	OP

add, and, or, xor, sll, srl, sub, sra

Control Transfer Instructions
Jumps & Conditional Branches

31	30 — 21	20	19 — 12	11 — 7	6 — 0
imm[20]	imm[10:1]	imm[11]	imm[19:12]	rd	opcode
1	10	1	8	5	7
	offset[20:1]			dest	JAL

jal

31 — 20	19 — 15	14 — 12	11 — 7	6 — 0
imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7
offset[11:0]	base	0	dest	JALR

jalr

31	30 — 25	24 — 20	19 — 15	14 — 12	11 — 8	7	6 — 0
imm[12]	imm[10:5]	rs2	rs1	funct3	imm[4:1]	imm[11]	opcode
1	6	5	5	3	4	1	7
offset[12|10:5]		src2	src1	BEQ/BNE	offset[11|4:1]		BRANCH
offset[12|10:5]		src2	src1	BLT[U]	offset[11|4:1]		BRANCH
offset[12|10:5]		src2	src1	BGE[U]	offset[11|4:1]		BRANCH

beq, bne, blt, bge

Load and Store Instructions

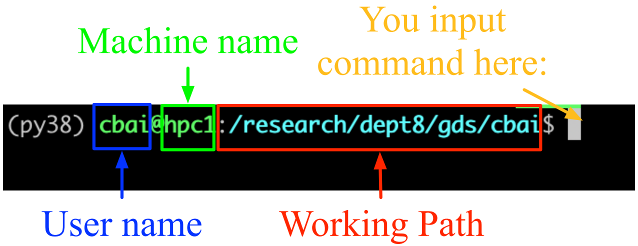
31 20	19 15	14 12	11 7	6 0
imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7
offset[11:0]	base	width	dest	LOAD

31 25	24 20	19 15	14 12	11 7	6 0
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
7	5	5	3	5	7
offset[11:5]	src	base	width	offset[4:0]	STORE

lb, lh, lw, sb, sh, sw

RISC-V LC ISA Simulator

Linux Terminal:



```
(py38) cbai@hpc1:/research/dept8/gds/cbai$
```

The user interface of Linux terminal

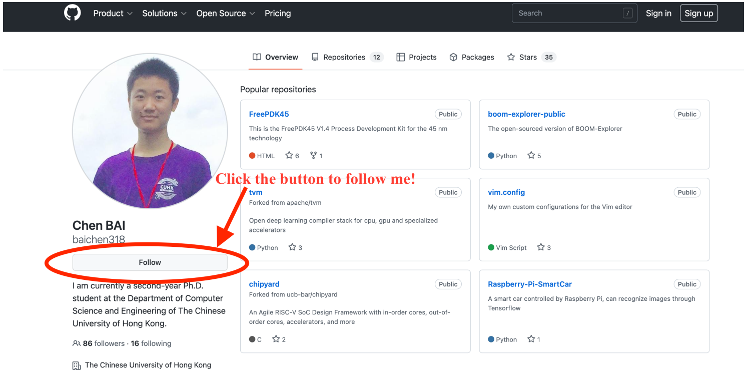
RISC-V LC ISA Simulator
Basics 1 – Linux Terminal Tutorial

A live demo to illustrate:

- `ls`: list files/directories.
- `cd`: change the working directory.
- `rm`: remove a file.
- `mv`: move a file/rename a file.
- `cat`: show file contents.
- `file`: check the file type.
- `man`: help menu for commands.
- For more information: `https://ubuntu.com/tutorials/command-line-for-beginners`.

RISC-V LC ISA Simulator
Basics 2 – Git

- We use `git` to manager our codes, please access `https://github.com/`, and register your account.
- Click `https://github.com/baichen318`.
- Click the **Follow** button.
  Follow me through GitHub, so that you can see any latest updates of the lab!



Chen BAI
baichen318

I am currently a second-year Ph.D. student at the Department of Computer Science and Engineering of The Chinese University of Hong Kong.

86 followers · 16 following

The Chinese University of Hong Kong

Popular repositories

FreePDK45 Public
This is the FreePDK45 V1.4 Process Development Kit for the 45 nm technology
HTML 6 1

boom-explorer-public Public
The open-sourced version of BOOM-Explorer
Python 5

tvm Public
Forked from apache/tvm
Open deep learning compiler stack for cpu, gpu and specialized accelerators
Python 3

vim.config Public
My own custom configurations for the Vim editor
Vim Script 3

chipyard Public
Forked from ucb-bar/chipyard
An Agile RISC-V SoC Design Framework with in-order cores, out-of-order cores, accelerators, and more
C 2

Raspberry-Pi-SmartCar Public
A smart car controlled by Raspberry Pi, can recognize images through Tensorflow
Python 1

RISC-V LC ISA Simulator
Basics 2 – Git

A live demo to illustrate:

- `git clone`: clone the code repository.
- `git checkout`: checkout a git branch.
- For more information:
  - `https://www.w3schools.com/git/`
  - `https://git-scm.com/docs/gittutorial`

RISC-V LC ISA Simulator
Basics 3 – Makefile

A live demo to illustrate:

- How to compile C codes into machine codes with GCC?
  - `https://medium.com/@laura.derohan/compiling-c-files-with-gcc-step-by-step-8e78318052` `https://www.wikihow.com/Compile-a-C-Program-Using-the-GNU-Compiler-(GCC)`
- How to automate the compilation process with Makefile?
  - `https://makefiletutorial.com/`
  - `https://www.tutorialspoint.com/makefile/index.htm`

RISC-V LC ISA Simulator

Get ISA Simulator Codes

Get the RISC-V LC ISA Simulator

In the terminal of your computer, use these commands:

- git clone https://github.com/baichen318/ceng3420.git
- cd ceng3420
- git checkout lab2.2

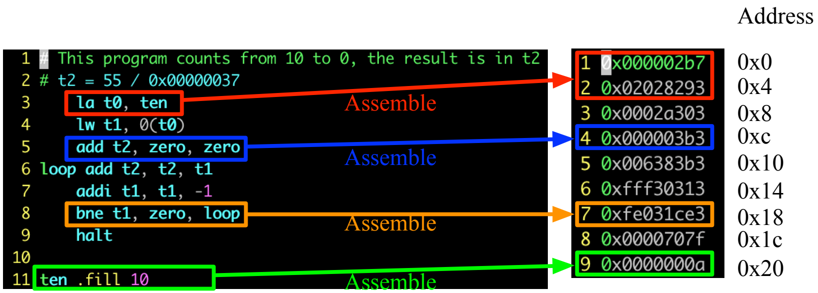
Compile (Linux/MacOS environment is suggested)

- make

Run the Simulator

- ./sim benchmarks/count10.bin # the simulator can execute successfully if you have implemented it.

RISC-V LC ISA Simulator
Assemble & Simulate



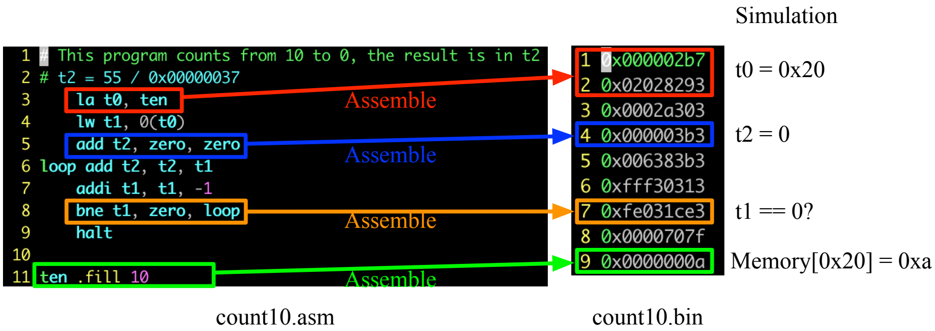
```
# This program counts from 10 to 0, the result is in t2
# t2 = 55 / 0x00000037
    la t0, ten
    lw t1, 0(t0)
    add t2, zero, zero
loop add t2, t2, t1
    addi t1, t1, -1
    bne t1, zero, loop
    halt

ten .fill 10
```

```
0x000002b7
0x02028293
0x0002a303
0x000003b3
0x006383b3
0xfff30313
0xfe031ce3
0x0000707f
0x0000000a
```

count10.asm count10.bin

RISC-V LC Assembler.

RISC-V LC ISA Simulator
Assemble & Simulate



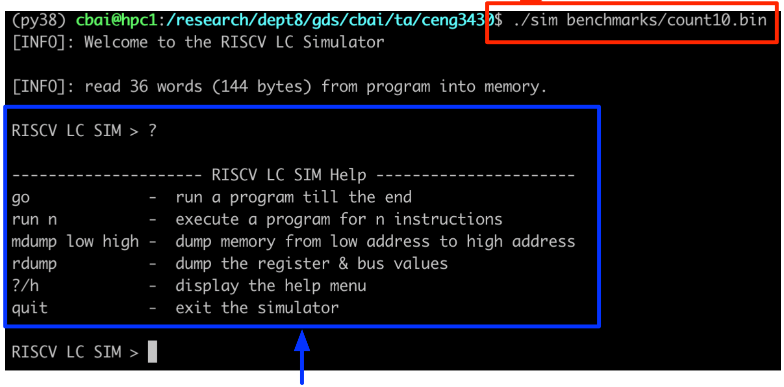
```
# This program counts from 10 to 0, the result is in t2
# t2 = 55 / 0x00000037
    la t0, ten
    lw t1, 0(t0)
    add t2, zero, zero
loop add t2, t2, t1
    addi t1, t1, -1
    bne t1, zero, loop
    halt

ten .fill 10
```

```
1 0x000002b7
2 0x02028293
3 0x0002a303
4 0x000003b3
5 0x006383b3
6 0xfff30313
7 0xfe031ce3
8 0x0000707f
9 0x0000000a
```

RISC-V LC ISA Simulator.

How to launch the simulator?

Launch the simulator using the command



```
(py38) cbai@hpc1:/research/dept8/gds/cbai/ta/ceng3430$ ./sim benchmarks/count10.bin
[INFO]: Welcome to the RISCV LC Simulator

[INFO]: read 36 words (144 bytes) from program into memory.

RISCV LC SIM > ?

-------------------- RISCV LC SIM Help ---------------------
go              -  run a program till the end
run n           -  execute a program for n instructions
mdump low high  -  dump memory from low address to high address
rdump           -  dump the register & bus values
?/h             -  display the help menu
quit            -  exit the simulator

RISCV LC SIM > 
```

Help menu

An overview of the simulator.

RISC-V LC ISA Simulator
Assemble & Simulate

RISC-V LC ISA simulator options:

- go: execute the program till the end.
- run n: execute the program by n instructions, *e.g.*, run 3
- mdump low high: dump the memory content, *e.g.*, mdump 0x0 0x30
- rdump: dump the registers
- h: help menu
- quit: exit the simulator

RISC-V LC ISA Simulator
Assemble & Simulate

Implementation examples (in `sim.c`):

- `handle_addi`
- `handle_add`
- `handle_beq`
- `handle_lb`

Lab 2-2 Assignment

Lab 2-2 Assignment
Assignment Content

Finish the RISCV LC simulator including 24 instructions in sim.c

- Integer Register-Immediate Instructions: slli, xori, srli, srai, ori, andi, lui
- Integer Register-Register Operations: sub, sll, xor, srl, sra, or, and
- Unconditional Jumps: jalr, jal
- Conditional Branches: bne, blt, bge
- Load and Store Instructions: lh, lw, sb, sh, sw

These unimplemented codes are commented with Lab2-2 assignment.

.. jalr $\underline{x1}$ 20 $\rightarrow$ $\boxed{20-1}$

20

Lab 2-2 Assignment
Verification

Benchmarks

Verify your codes with these benchmarks (inside the `benchmarks` directory)

- isa.bin
- count10.bin
- swap.bin
- add4.bin

Verification

- isa.bin → a3 = -18/0xffffffee and MEMORY[0x84 + 16] = 0xffffffee
- count10.bin → t2 = 55/0x00000037
- swap.bin → NUM1 (memory address: 0x00000034) changes from 0xabcd to 0x1234 and NUM2 (memory address: 0x00000038) changes from 0x1234 to 0xabcd
- add4.bin → BL (memory address: 0x00000038) changes from -5 (0xfffffffb) to -1 (0xffffffff)

Submission Method

Submit one zip file into **Blackboard**, including

- The source codes zip files, i.e., lab2-1.zip, lab2-2.zip.
  - Pay attention to the submission name format, i.e., `name-sid-lab2-x.zip`. For example, `zhangsan-1234567890-lab2-1.zip`, and `zhangsan-1234567890-lab2-2.zip`.
- One report. (name format: `name-sid-report.pdf`) The report summarizes your implementations and all screenshots of lab results (lab 2-1 & lab2-2).
- Deadline: 23:59, 29 Mar (Wed), 2023

Lab 2-2 Assignment
Tips

Tips

Inside `docs`, there are three valuable documents for your reference!

- opcodes-rv32i: RV32I opcodes
- riscv-spec-20191213.pdf: RV32I specifications
- risc-v-asm-manual.pdf: RV32I assembly programming manual