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

CENG3420

Lab 2-2: RISC-V RV32I Simulator

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

Spring 2022

Outline

Introduction

Assembler & Simulator

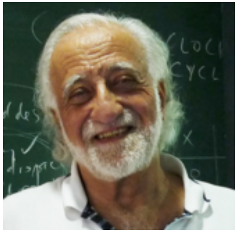
Assembler

- Turn symbols into machine binary instructions, *e.g.*, lc3b_asm, riscv64-unknown-elf-as, ...

Simulator

- Mimic the behavior of a processor, *e.g.*, lc3b_sim, spike, QEMU, rv8, ...

- 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

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
Unconditional Branches & 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

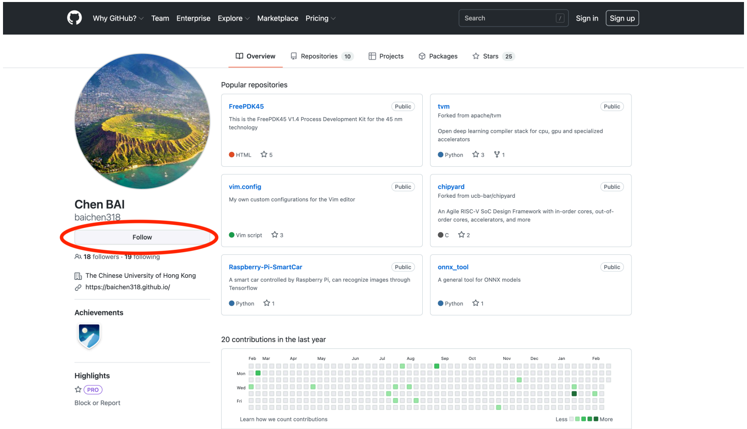
lb, lh, lw, sb, sh, sw

Lab 2-2 Assignment

Lab 2-2 Assignment
Pre-requisites

Get Latest Updates of the Lab

- Click `https://github.com/baichen318`.
- Follow my GitHub account.
  Follow me through GitHub, so that you can see any latest updates of the lab!



Chen BAI
baichen318
Follow
18 followers · 19 following
The Chinese University of Hong Kong
https://baichen318.github.io/
Achievements
Highlights
PRO
Block or Report

Popular repositories

FreePDK45 — Public
This is the FreePDK45 V1.4 Process Development Kit for the 45 nm technology
HTML ☆ 5

tvm — Public
Forked from apache/tvm
Open deep learning compiler stack for cpu, gpu and specialized accelerators
Python ☆ 3 1

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

onnx_tool — Public
A general tool for ONNX models
Python ☆ 1

20 contributions in the last year
Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb
Mon Wed Fri
Learn how we count contributions
Less More

Lab 2-2 Assignment

Pre-requisites

Get the RV32I Simulator

- $ 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.

Lab 2-1 Assignment
Assignment Content

Finish the RISCV LC simulator including 25 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: lb, lh, lw, sb, sh, sw

These unimplemented codes are commented with Lab2-2 assignment.

Lab 2-2 Assignment
Verification

Benchmarks

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

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

Verification

- isa.bin $\rightarrow$ a3 = -18/0xffffffee and MEMORY[0x84 + 16] = 0xffffffee
- count10.bin $\rightarrow$ t2 = 55/0x00000037
- swap.bin $\rightarrow$ NUM1 changes from 0xabcd to 0x1234 and NUM2 changes from 0x1234 to 0xabcd

Lab 2-2 Assignment
Verification

Submission Method:

Submit the source code and report into **Blackboard**, including

- Your implementations, *i.e.*, asm.c, and *sim.c* with the name of `name-sid-lab2-1.c` and `name-sid-lab2-2.c` (*e.g.*, `zhangsan-1234567890-lab2-1.c`, `zhangsan-1234567890-lab2-2.c`, *etc.*)
- A lab report (`name-sid-lab2.pdf`) illustrates your implementation for two parts of Lab 2 and all console results (screenshots).
- Deadline: 23:59, 31 Mar (Thu)

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