Organization of Digital Computers Lab
EECS 112L

LAB 2

EECS Department
Henry Samueli School of Engineering
University of California, Irvine

Winter 2019

Due Date : Feb 8th at 4:00 pm

1 Introduction

The goal of this lab is to write a testbench to test your simplified processor. Last week, you tested RISC-V processor with given instructions. In this lab, you should write your own instruction set for all instructions. You are not supposed to add any new instructions to your processor in this session.

You have 2 weeks to finish this lab. Submit a zip file containing all your files to Canvas.

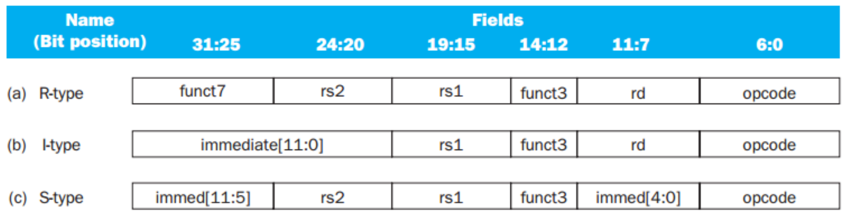
2 Review

Table 1 shows three of the instruction classes used in RISC-V:

- **R-type:** *Arithmetic instructions*, (opcode = 51_{ten}), both ALU operands are from the Register-File, the result is saved in Register-File as well. *rs1* and *rs2* are addresses of the operands 1 and 2 on the Register-File. *rd* is the address on the Register-File which the result should be saved to. The **R-type instructions** that we implement are **add**, **sub**, **and**, and **or**.
- **I-type:** is used by *Arithmetic* operations with one constant operand, including *addi*, and by *load instructions*. opcode = 3_{ten}, 19_{ten} and 103_{ten}. In the *load instructions* (opcode = 3_{ten}), the register *rs1* is the base register that is added to the 12-bit immediate field to form the memory address (just 9 LSB bit will be used). Field *rd* is the destination register (on the register file) for the loaded value.
- **S-type:** is used by *store instructions* (opcode = 35_{ten}). The register *rs1* is the base register that is added to the 12-bit immediate field to form the memory address which the data should be read from. in S-type instructions, the immediate field is split into a 7-bit piece and a 5-bit piece. Field

rs2 is the the address of the register on the Register-file, which the read value should be stored into.

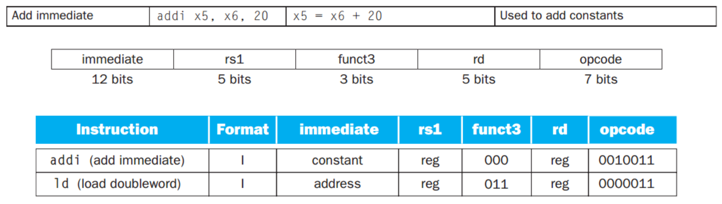
Table 1: **Three instruction classes of RISC-V.**



Name (Bit position)	Fields 31:25	24:20	19:15	14:12	11:7	6:0
(a) R-type	funct7	rs2	rs1	funct3	rd	opcode
(b) I-type	immediate[11:0]		rs1	funct3	rd	opcode
(c) S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode

Also, we have DDI instructions (ADDI, ORI, ANDI, ...). Table 2 shows the instruction format for this operations.

Table 2: **DDI Instruction.**



Add immediate	addi x5, x6, 20	x5 = x6 + 20	Used to add constants

immediate	rs1	funct3	rd	opcode
12 bits	5 bits	3 bits	5 bits	7 bits

Instruction	Format	immediate	rs1	funct3	rd	opcode
addi (add immediate)	I	constant	reg	000	reg	0010011
ld (load doubleword)	I	address	reg	011	reg	0000011

3 Instructions Binary File

1. make sure to pull new repository from GitHub.Then, cd to "Lab2" directory.

2. In design folder, open "instructionmemory.sv" file. As you can see, there is not any instruction in this file anymore.

3. In verif folder, there is another directory named "program". In program directory, there is file named "inst.bin". In this file, we should have the binary codes for instructions. We already copy-pasted the instruction set from last lab to this file.

4. You should modify your testbench to be able to copy instructions from "inst.bin" to "instructionmemory.sv" automatically. So, from now on, you just need to add/delete any instruction in "inst.bin". Add the following command to your testbench file:

$readmemb ("$verif/program/inst.bin" , riscV.dp.instr_mem.Inst_mem);

- **$readmemb** will read a binary file, and initialize memory array. If you want to write instructions in hexadecimal format, use $readmemh.

- **"$verif/program/inst.bin"** is the address to the binary file (.bin).
- **riscV.dp.instr_mem.Inst_mem** is the path to instruction memory register. In riscv.sv, Datapath is instantiated. In Datapath, instruction memory is instantiated, and Inst_mem is the name of the 512*32bits register.
- Now, you should simulate riscv module. It should give you the same result as what you got in lab1.
- Finally, write your own instruction set in "inst.bin", and make sure that your riscv, works correctly for all possible intructions.

4 Material to be submitted

Report (20% of your total grade):

- Reports written in LaTeX are preferred. Submit pdf file of your report.
- Block diagram of the design with the detailed explanation of how your processor works. Your Block diagram should contain all Your design modules.
- Put a screenshot from the simulated waveform. your screenshot is from the result of testbench simulation so it should contain Clk, reset, and ALU-Result.
- After synthesis, put the riscv.elab.rpt and riscv.qor.rpt files in the doc folder of your project.
- Put the synthesis results in your report. Critical path length, critical path slack, and area and power report.

Important notes:

- Submit the project in zip format.
- The name of uploaded zip file should be your name.
- The compressed file should include design, sim, syn, verif, and doc, directories.
- The report should contain your name and student ID (on the cover page).
- No points if you submit a wrong file by mistake. We **do not** accept any changes in code after the deadline.
- **NO Late submission is permitted.**

5 Appendix

Table 3: **RISC-V Instruction set**

RV32I Base Instruction Set

imm[31:12]					rd	0110111	LUI
imm[31:12]					rd	0010111	AUIPC
imm[20|10:1|11|19:12]					rd	1101111	JAL
imm[11:0]			rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2		rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2		rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2		rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2		rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2		rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2		rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]			rs1	000	rd	0000011	LB
imm[11:0]			rs1	001	rd	0000011	LH
imm[11:0]			rs1	010	rd	0000011	LW
imm[11:0]			rs1	100	rd	0000011	LBU
imm[11:0]			rs1	101	rd	0000011	LHU
imm[11:5]	rs2		rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2		rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2		rs1	010	imm[4:0]	0100011	SW
imm[11:0]			rs1	000	rd	0010011	ADDI
imm[11:0]			rs1	010	rd	0010011	SLTI
imm[11:0]			rs1	011	rd	0010011	SLTIU
imm[11:0]			rs1	100	rd	0010011	XORI
imm[11:0]			rs1	110	rd	0010011	ORI
imm[11:0]			rs1	111	rd	0010011	ANDI
0000000	shamt		rs1	001	rd	0010011	SLLI
0000000	shamt		rs1	101	rd	0010011	SRLI
0100000	shamt		rs1	101	rd	0010011	SRAI
0000000	rs2		rs1	000	rd	0110011	ADD
0100000	rs2		rs1	000	rd	0110011	SUB
0000000	rs2		rs1	001	rd	0110011	SLL
0000000	rs2		rs1	010	rd	0110011	SLT
0000000	rs2		rs1	011	rd	0110011	SLTU
0000000	rs2		rs1	100	rd	0110011	XOR
0000000	rs2		rs1	101	rd	0110011	SRL
0100000	rs2		rs1	101	rd	0110011	SRA
0000000	rs2		rs1	110	rd	0110011	OR
0000000	rs2		rs1	111	rd	0110011	AND
0000	pred	succ	00000	000	00000	0001111	FENCE
0000	0000	0000	00000	001	00000	0001111	FENCE.I
000000000000			00000	000	00000	1110011	ECALL
000000000001			00000	000	00000	1110011	EBREAK
csr			rs1	001	rd	1110011	CSRRW
csr			rs1	010	rd	1110011	CSRRS
csr			rs1	011	rd	1110011	CSRRC
csr			zimm	101	rd	1110011	CSRRWI
csr			zimm	110	rd	1110011	CSRRSI
csr			zimm	111	rd	1110011	CSRRCI