



香港中文大學
The Chinese University of Hong Kong

CENG3420

Lab 3-3: RISC-V Litter Computer (RISC-V LC)

Chen BAI

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

Spring 2022

Outline

① Introduction

② Details

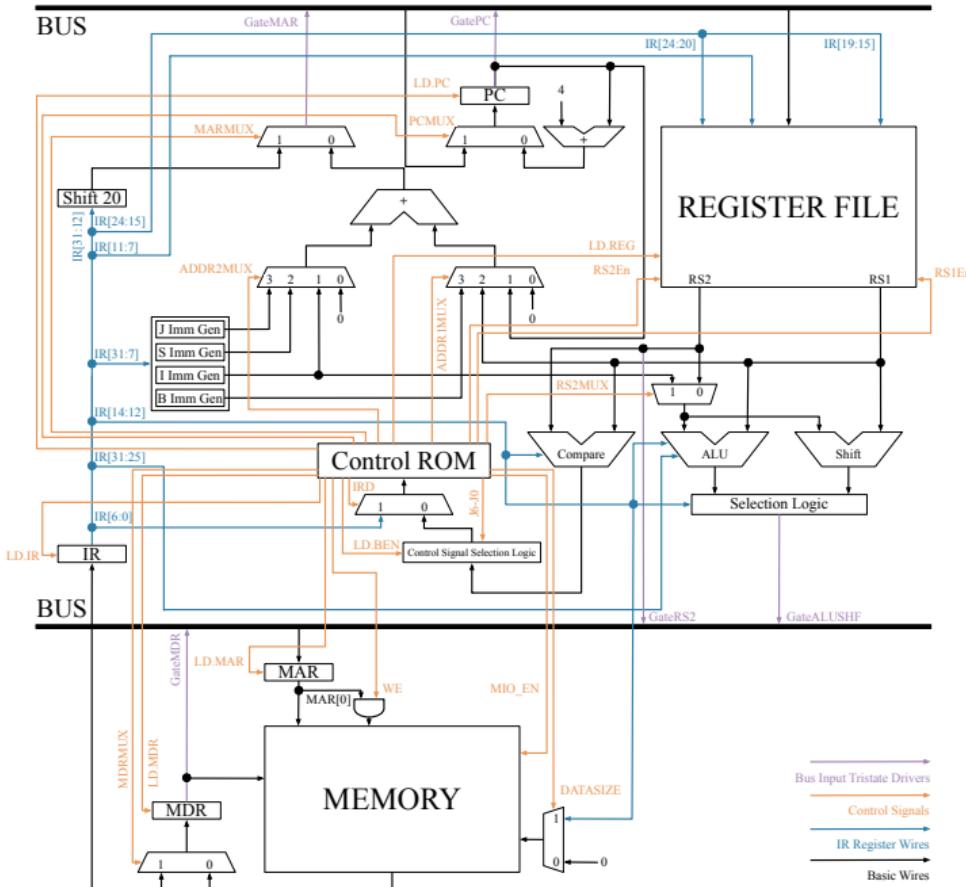
③ Implementations

④ Lab 3-3 Assignment

Introduction

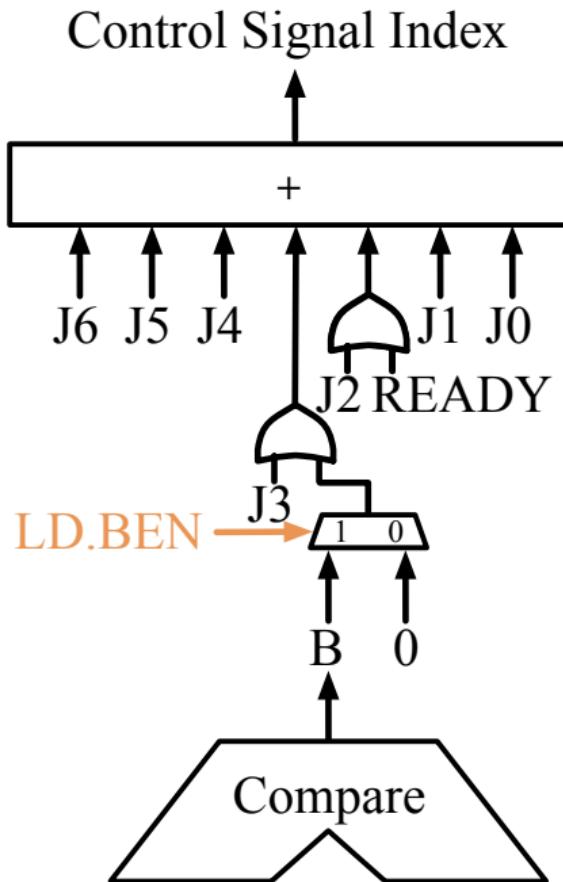
Introduction

RISC-V LC



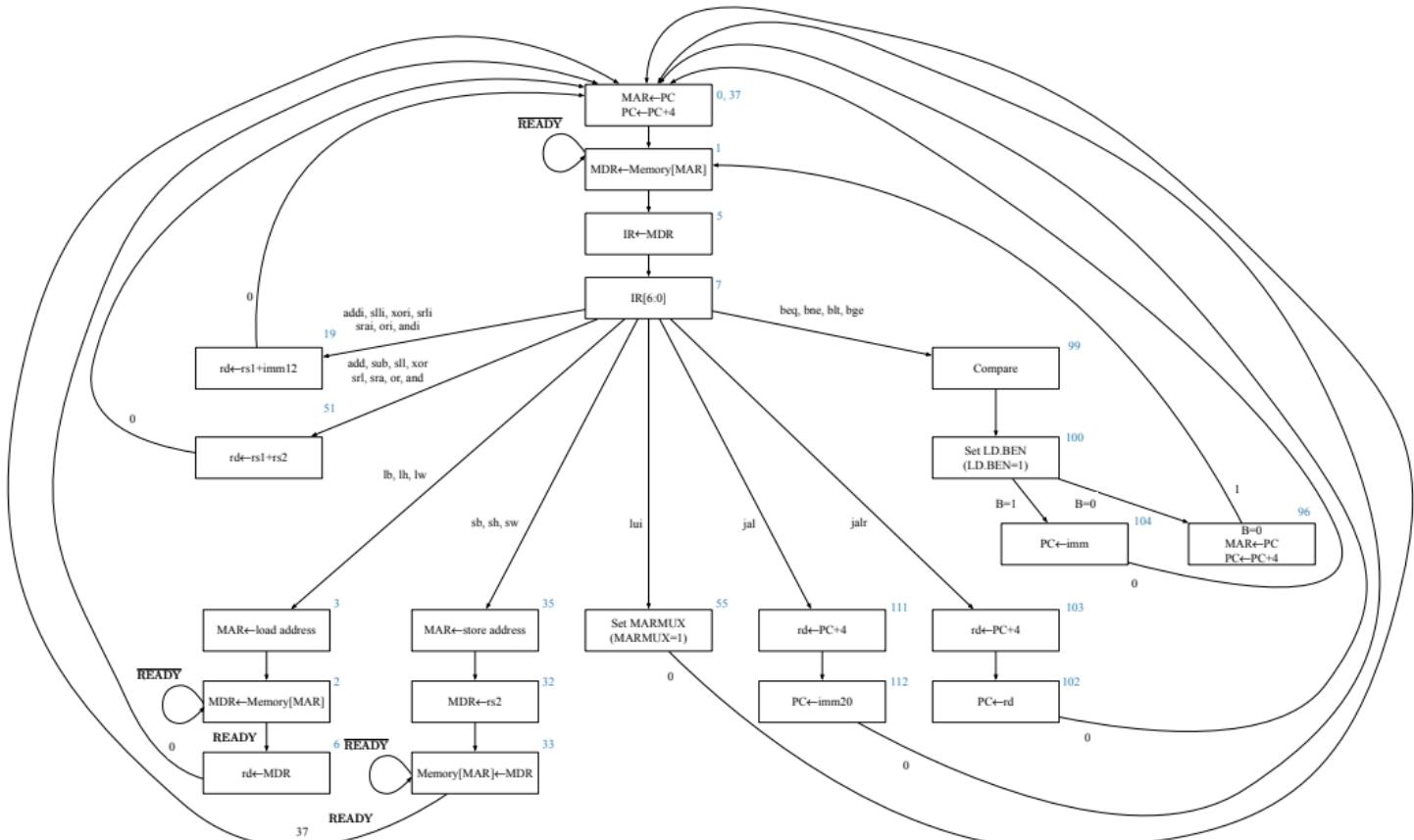
Introduction

Control Signal Selection Logic



Introduction

RISC-V LC Finite State Machine



Details

Details

Micro-ops – uop

	IRD	J6 → J0	LD_PC	LD_MAR	WE	DATASIZE	RESET
State 0	1	00000001	11000010000000000000000000000000					
State 1	2	00000000	xxxx00000000000000000000xxxx000					
State 2	3	0xxxxxx0001	00000000000000000000000000000000					
...								
5	5	00000000000000000000000000000000						
6	6	00000111000100001000000000000000						
7	7	00000000000000001000100000000000						
8	8	10000000000000000000000000000000						
9	9	00000000000000000000000000000000						
State 11	10	00000000000000000000000000000000						

Micro-ops specifications

Details

The Data Structure Organization in Memory

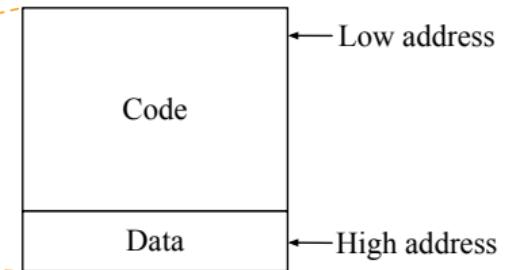
```
1 # This program counts from 10 to 0, the result is in t2
2 # t2 = 55 / 0x00000037
3 la t0, ten
4 lw t1, 0(t0)
5 add t2, zero, zero
6 loop add t2, t2, t1
7 addi t1, t1, -1
8 bne t1, zero, loop
9 halt
10
11 ten .fill 10
```

```
[INFO]: Welcome to the RISC-V LC Simulator
[INFO]: load the micro: uop
[INFO]: read 36 words (144 bytes) from program into memory.

RISCV LC SIM > md 0x0 0x30

memory content [0x00000000 .. 0x0000003F]:
0x00000000 (0) : 0x000002b7
0x00000004 (4) : 0xd2282291
0x00000008 (8) : 0x00002d10
0x0000000C (12) : 0x000003b3
0x00000010 (16) : 0x00063830
0x00000014 (20) : 0xfffff30313
0x00000018 (24) : 0x00031863
0x0000001C (28) : 0x0000707F
0x00000020 (32) : 0x00000000
0x00000024 (36) : 0x00000000
0x00000028 (40) : 0x00000000
0x0000002C (44) : 0x00000000
0x00000030 (48) : 0x00000000

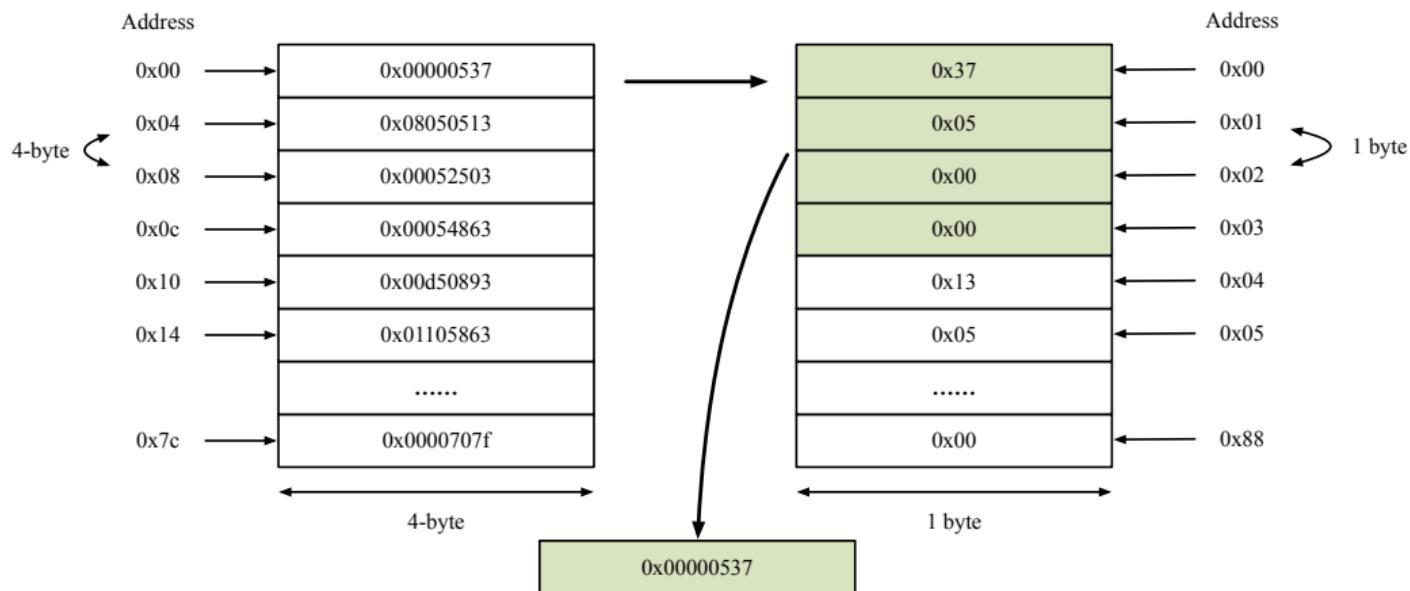
RISCV LC SIM >
```



Source codes \leftrightarrow Machine codes \leftrightarrow Organization in memory

Details

Little Endian One-Byte Addressed Memory



RISCV-LC adopts little endian one-byte addressed memory.

add a4, a2, a0

- PC → BUS
 - In state 0, GatePC is asserted.
- BUS → MAR
 - In state 0, LD_MAR is asserted.
- PC +4 → PC
 - In state 0, PCMUX is deasserted, and LD_PC is asserted.
- Memory[MAR] → MDR
 - In state 1, the step will take MEM_CYCLES clocks.)
 - J0, LD_MDR, MIO_EN are asserted.
- MDR → BUS
 - In state 5, J2, J1, J0 are asserted, GateMDR is asserted.
- BUS → IR
 - In state 5, LD_IR is asserted.

Details II

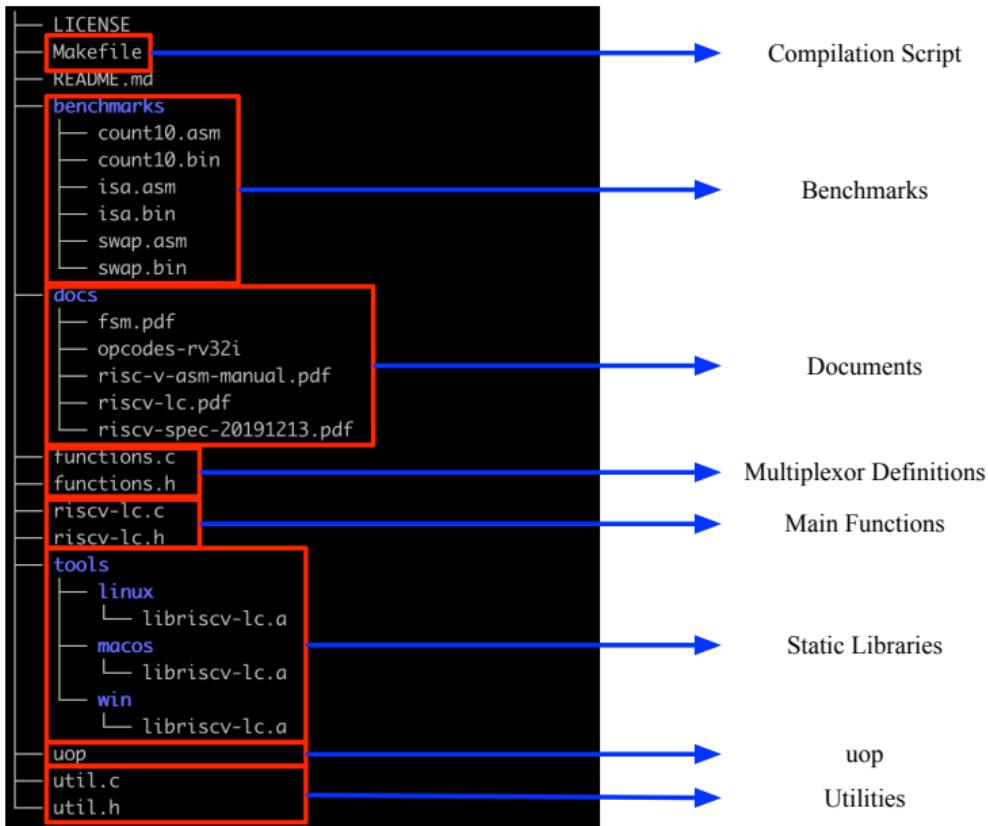
Data/Control Flow

- Generate control signals according to IR[6:0]
 - In state 7, IRD is asserted.
- R-type addition: $a_2 + a_0$
 - In state 51, J6 ~ J0 are deasserted, RS2En, RS1En are asserted.
- R-type addition: results write back to a4
 - In state 51, LD_REG, GateALUSHF are asserted.
- Go back to state 0

Implementations

Implementations

Repo. Organization



Repo. Organization

Implementations I

Operations in One Clock Cycle

```
/*
 * execute a cycle
 */
void cycle() {
    /*
     * core steps
     */
    eval_micro_sequencer();
    cycle_memory();
    eval_bus_drivers();
    drive_bus();
    latch_datapath_values();

    CURRENT_LATCHES = NEXT_LATCHES;

    CYCLE_COUNT++;
}
```

Implementations I

Five Input Tristate Drivers

```
value_of_GatePC = 0;  
value_of_GateMAR = 0;  
value_of_GateMDR = 0;  
value_of_GateALUSHF = 0;  
value_of_GateRS2 = 0;
```

Implementations I

Three Intermediate Values for Data Path

```
int value_of_MARMUX = 0,  
value_of_alu,  
value_of_shift_function_unit = 0;
```

Implementations I

Implementation of value_of_MARMUX

```
value_of_MARMUX = addr2_mux(
    get_ADDR2MUX(CURRENT_LATCHES.MICROINSTRUCTION),
    0,
    sext_unit(mask_val(CURRENT_LATCHES.IR, 31, 20), 12),
    sext_unit(
        s_format_imm_gen_unit(
            mask_val(CURRENT_LATCHES.IR, 11, 7),
            mask_val(CURRENT_LATCHES.IR, 31, 25)
        ),
        12
    ),
    sext_unit(
        j_format_imm_gen_unit(
            mask_val(CURRENT_LATCHES.IR, 31, 31),
            mask_val(CURRENT_LATCHES.IR, 30, 21),
            mask_val(CURRENT_LATCHES.IR, 20, 20),
            mask_val(CURRENT_LATCHES.IR, 19, 12)
        ),
        20
    )
)
```

Implementations II

Implementation of value_of_MARMUX

```
)  
) + addr1_mux(  
    get_ADDR1MUX(CURRENT_LATCHES.MICROINSTRUCTION),  
    0,  
    CURRENT_LATCHES.PC,  
    rs1_en(  
        get_RS1En(CURRENT_LATCHES.MICROINSTRUCTION),  
        0,  
        CURRENT_LATCHES.REGS[mask_val(CURRENT_LATCHES.IR, 19,  
            15)]  
,  
    sext_unit(  
        b_format_imm_gen_unit(  
            mask_val(CURRENT_LATCHES.IR, 7, 7),  
            mask_val(CURRENT_LATCHES.IR, 11, 8),  
            mask_val(CURRENT_LATCHES.IR, 30, 25),  
            mask_val(CURRENT_LATCHES.IR, 31, 31)  
,  
        12
```

Implementations III

Implementation of value_of_MARMUX

```
)  
) ;
```

Implementations I

Implementation of value_of_alu

```
value_of_alu = alu(
    mask_val(CURRENT_LATCHES.IR, 14, 12),
    mask_val(CURRENT_LATCHES.IR, 31, 25),
    rs1_en(
        get_RS1En(CURRENT_LATCHES.MICROINSTRUCTION),
        0,
        CURRENT_LATCHES.REGS[mask_val(CURRENT_LATCHES.IR, 19,
            15)]),
    rs2_mux(
        get_RS2MUX(CURRENT_LATCHES.MICROINSTRUCTION),
        rs2_en(
            get_RS2En(CURRENT_LATCHES.MICROINSTRUCTION),
            0,
            CURRENT_LATCHES.REGS[mask_val(CURRENT_LATCHES.IR,
                24, 20)]),
        sext_unit(mask_val(CURRENT_LATCHES.IR, 31, 20), 12)
)
```

Implementations II

Implementation of value_of_alu

) ;

Implementations I

Intermediate Variables for Bus Drivers

```
int _GateMAR = get_GateMAR(CURRENT_LATCHES.MICROINSTRUCTION);
int _GateALUSHF = get_GateALUSHF(CURRENT_LATCHES.
    MICROINSTRUCTION);
int _GatePC = get_GatePC(CURRENT_LATCHES.MICROINSTRUCTION);
int _GateRS2 = get_GateRS2(CURRENT_LATCHES.MICROINSTRUCTION);
int _GateMDR = get_GateMDR(CURRENT_LATCHES.MICROINSTRUCTION);
```

Implementations I

Drive Bus

```
switch (((_GateMDR << 4) + (_GateRS2 << 3) + (_GatePC << 2) + (_GateALUSHF << 1) + (_GateMAR)) {  
    case 0:  
        BUS = 0;  
        break;  
    case 1:  
        error("Lab3-3_assignment:_when_value_=1,_BUS_=?;\n");  
    case 2:  
        error("Lab3-3_assignment:_when_value_=1,_BUS_=?;\n");  
    case 4:  
        error("Lab3-3_assignment:_when_value_=1,_BUS_=?;\n");  
    case 8:  
        error("Lab3-3_assignment:_when_value_=1,_BUS_=?;\n");  
    case 16:
```

Implementations II

Drive Bus

```
error("Lab3-3_assignment:_when_value_=1,_BUS_=?;\n")
      ;
default:
    BUS = 0;
    warn ("unknown_gate_drivers_for_BUS\n");
}
```

Lab 3-3 Assignment

Lab 3-3 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!

The screenshot shows Chen Bai's GitHub profile page. At the top, there is a large circular profile picture of a green landscape. Below it, the user's name 'Chen Bai' and GitHub handle 'baichen318' are displayed. A red circle highlights the 'Follow' button, which is located below the user's name. To the right of the 'Follow' button, it says '18 followers - 18 following'. Below this section, there is a summary of the user's contributions: 'The Chinese University of Hong Kong' and a link to their GitHub page (<https://github.com/baichen318>). Further down, there are sections for 'Achievements' (with a blue trophy icon) and 'Highlights' (with a purple star icon). The 'Highlights' section includes a 'PRO' badge. At the bottom of the page, there is a chart titled '20 contributions in the last year' showing activity across different months. The chart uses a color-coded legend: light gray for 'Less', green for 'More', and dark gray for 'More' (the highest level).

Lab 3-3 Assignment

Pre-requisites

Get RISC-V LC

- \$ git clone https://github.com/baichen318/ceng3420.git
- \$ cd ceng3420
- \$ git checkout lab3.3

Compile (Linux/MacOS environment is suggested)

- \$ make

Run the RISC-V LC

- \$./riscv-lc <uop> <*.bin> # RISCV-LC can execute successfully if you have implemented it.

Lab 3-3 Assignment

Assignment Content

In **riscv-lc.c**,

- Finish `eval_bus_drivers`
- Finish `drive_bus`

These unimplemented codes are commented with [Lab3-3 assignment](#).

Lab 3-3 Assignment

Verification

Benchmarks

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

- [isa.bin](#)
- [count10.bin](#)
- [swap.bin](#)

Verification

- isa.bin → a3 = -18/0xfffffee and MEMORY[0x84 + 16] = 0xfffffee
- count10.bin → t2 = 55/0x00000037
- swap.bin → NUM1 changes from 0abcd to 0x1234 and NUM2 changes from 0x1234 to 0abcd

Lab 3-3 Assignment

Submission

Submission Method:

Submit a zip file into **Blackboard**. The zip file includes

- Your implementations, *i.e.*, three riscv-lc.c source codes for three parts of Lab 3. The sources should be renamed to name-sid-lab3-1.c, name-sid-lab3-2.c, and name-sid-lab3-3.c, respectively (*e.g.*, zhangsan-1234567890-lab3-1.c, zhangsan-1234567890-lab3-2.c, *etc.*).
- A lab report (name-sid-lab3.pdf) illustrates your implementation for three parts of Lab 3 and all console results (screenshots).

Deadline: 23:59, 30 Apr. (Sat)

Lab 3-3 Assignment

Tips

Tips

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

- `riscv-lc.pdf`
- `fsm.pdf`
- `opcodes-rv32i: RV32I opcodes`
- `riscv-spec-20191213.pdf: RV32I specifications`
- `riscv-asm-manual.pdf: RV32I assembly programming manual`