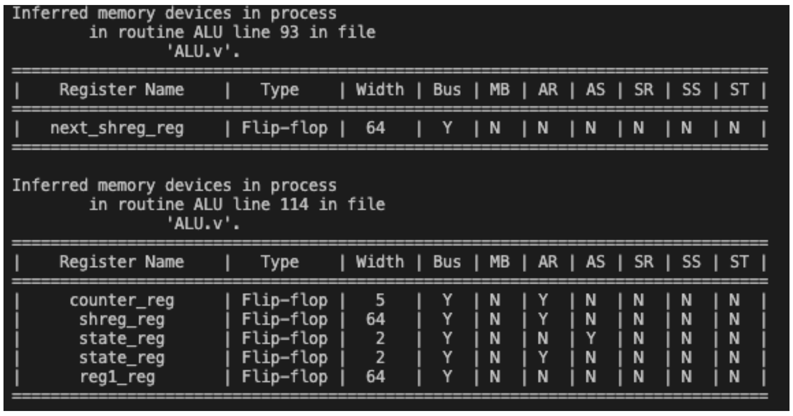
Computer Architecture Final Project Report

電機三 B09901079 劉碩 電機三 B09901140 于瑄

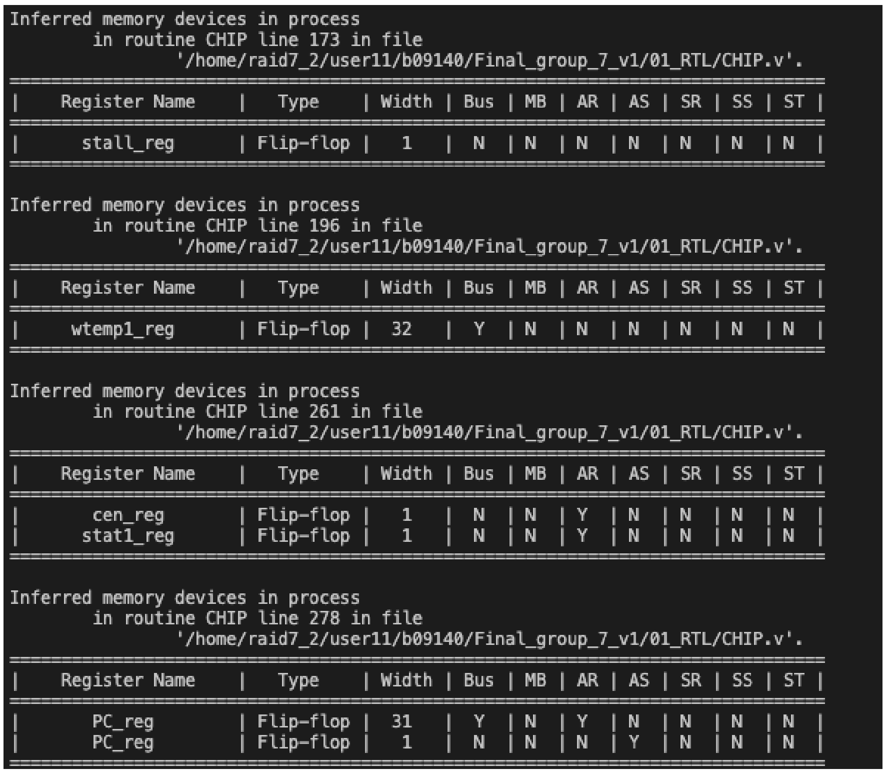
1、 Instruction Sets and their execution cycle

Instruction Set	Execution Cycle
I0	103
I1	853
I2	273
I3	703

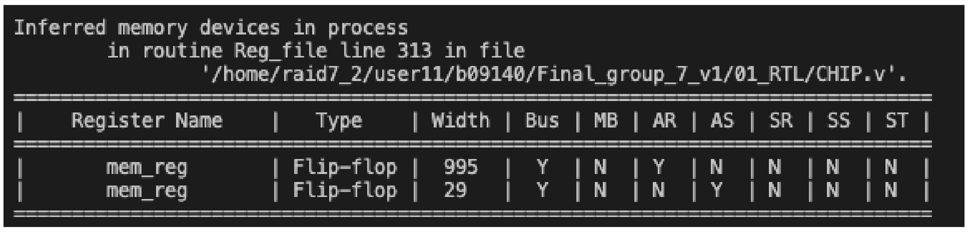
2、 Register table



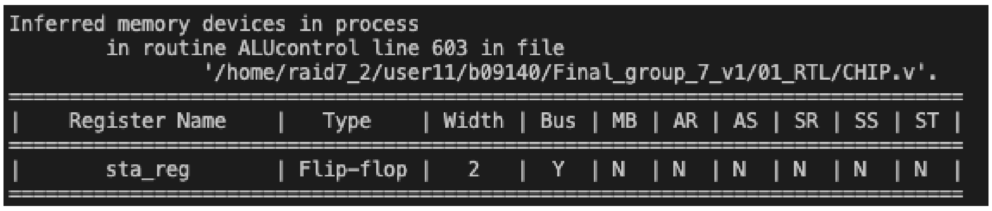
```
Inferred memory devices in process
        in routine ALU line 93 in file
                'ALU.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|   next_shreg_reg    | Flip-flop |  64   |  Y  | N  | N  | N  | N  | N  | N  |
===============================================================================

Inferred memory devices in process
        in routine ALU line 114 in file
                'ALU.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|     counter_reg     | Flip-flop |   5   |  Y  | N  | Y  | N  | N  | N  | N  |
|      shreg_reg      | Flip-flop |  64   |  Y  | N  | Y  | N  | N  | N  | N  |
|      state_reg      | Flip-flop |   2   |  Y  | N  | N  | Y  | N  | N  | N  |
|      state_reg      | Flip-flop |   2   |  Y  | N  | Y  | N  | N  | N  | N  |
|      reg1_reg       | Flip-flop |  64   |  Y  | N  | N  | N  | N  | N  | N  |
===============================================================================
```



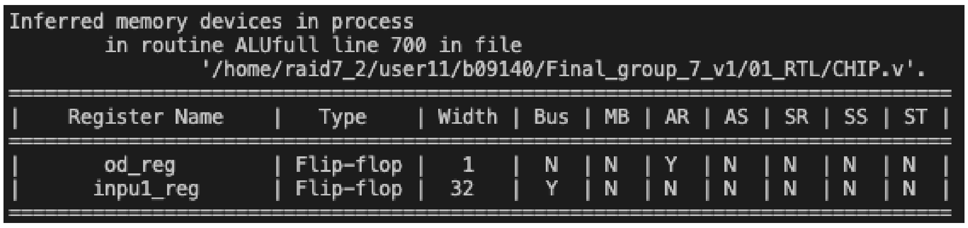
```
Inferred memory devices in process
        in routine CHIP line 173 in file
                '/home/raid7_2/user11/b09140/Final_group_7_v1/01_RTL/CHIP.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|      stall_reg      | Flip-flop |   1   |  N  | N  | N  | N  | N  | N  | N  |
===============================================================================

Inferred memory devices in process
        in routine CHIP line 196 in file
                '/home/raid7_2/user11/b09140/Final_group_7_v1/01_RTL/CHIP.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|     wtemp1_reg      | Flip-flop |  32   |  Y  | N  | N  | N  | N  | N  | N  |
===============================================================================

Inferred memory devices in process
        in routine CHIP line 261 in file
                '/home/raid7_2/user11/b09140/Final_group_7_v1/01_RTL/CHIP.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|       cen_reg       | Flip-flop |   1   |  N  | N  | Y  | N  | N  | N  | N  |
|      stat1_reg      | Flip-flop |   1   |  N  | N  | Y  | N  | N  | N  | N  |
===============================================================================

Inferred memory devices in process
        in routine CHIP line 278 in file
                '/home/raid7_2/user11/b09140/Final_group_7_v1/01_RTL/CHIP.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|       PC_reg        | Flip-flop |  31   |  Y  | N  | Y  | N  | N  | N  | N  |
|       PC_reg        | Flip-flop |   1   |  N  | N  | N  | Y  | N  | N  | N  |
===============================================================================
```



```
Inferred memory devices in process
        in routine Reg_file line 313 in file
                '/home/raid7_2/user11/b09140/Final_group_7_v1/01_RTL/CHIP.v'.
===============================================================================
|    Register Name     |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|       mem_reg        | Flip-flop |  995  |  Y  | N  | Y  | N  | N  | N  | N  |
|       mem_reg        | Flip-flop |  29   |  Y  | N  | N  | Y  | N  | N  | N  |
===============================================================================
```



```
Inferred memory devices in process
        in routine ALUcontrol line 603 in file
                '/home/raid7_2/user11/b09140/Final_group_7_v1/01_RTL/CHIP.v'.
===============================================================================
|    Register Name     |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|       sta_reg        | Flip-flop |   2   |  Y  | N  | N  | N  | N  | N  | N  |
===============================================================================
```



```
Inferred memory devices in process
        in routine ALUfull line 700 in file
                '/home/raid7_2/user11/b09140/Final_group_7_v1/01_RTL/CHIP.v'.
===============================================================================
|    Register Name     |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|        od_reg        | Flip-flop |   1   |  N  | N  | Y  | N  | N  | N  | N  |
|       inpu1_reg      | Flip-flop |  32   |  Y  | N  | N  | N  | N  | N  | N  |
===============================================================================
```

3、 CPU Architecture

Look at figure 1 below.

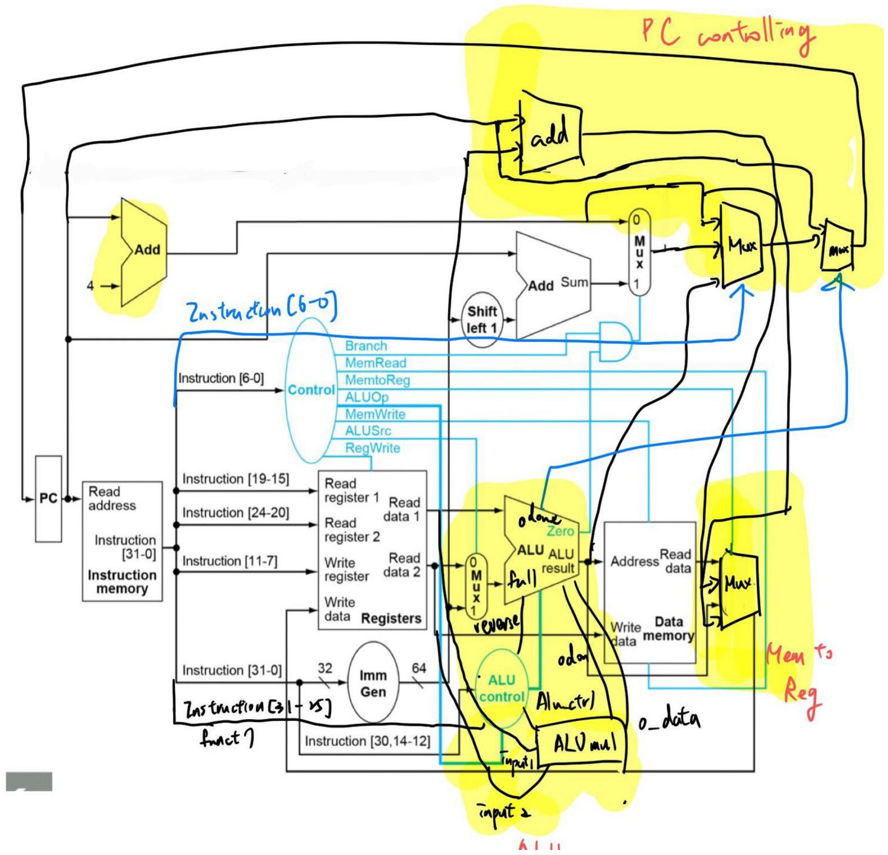


Figure 1. CPU architecture

Besides, lecture slides, in order to accomplish jal, jalr, and auipc function. We modified PC controlling shown in up left yellow colored region. This region controls the calculation of next PC. Also, we modified the multiplexer down right from 2-to-1 to 4-to-1, and we named it “MemtoReg.” This controls the data written back to register. To handle “mul” instruction, we add another alu unit called, “ALU”to calculate multiplication, but to not get confused, in this report we refer to “ALUmul.” Below tables are details about PC controlling and MemtoReg. For ALUmul, see section 5 for more details.

(1) PC controlling (three-stage multiplexer)

We control the next PC by observing the instruction opcode. The PC is updated while o_done is true.

Instruction type	Next_PC
R	PC+4
I	PC+4
I_load	PC+4
S	PC+4
UJ (jal)	PC + immediate<<1
I_jalr	ALUresult = rs1 + immediate
U (auipc)	PC+4
SB	(Branch && ALU_zero)? PC + immediate<<1: PC+4

(2) MemtoReg (a 4-to-1 multiplexer)

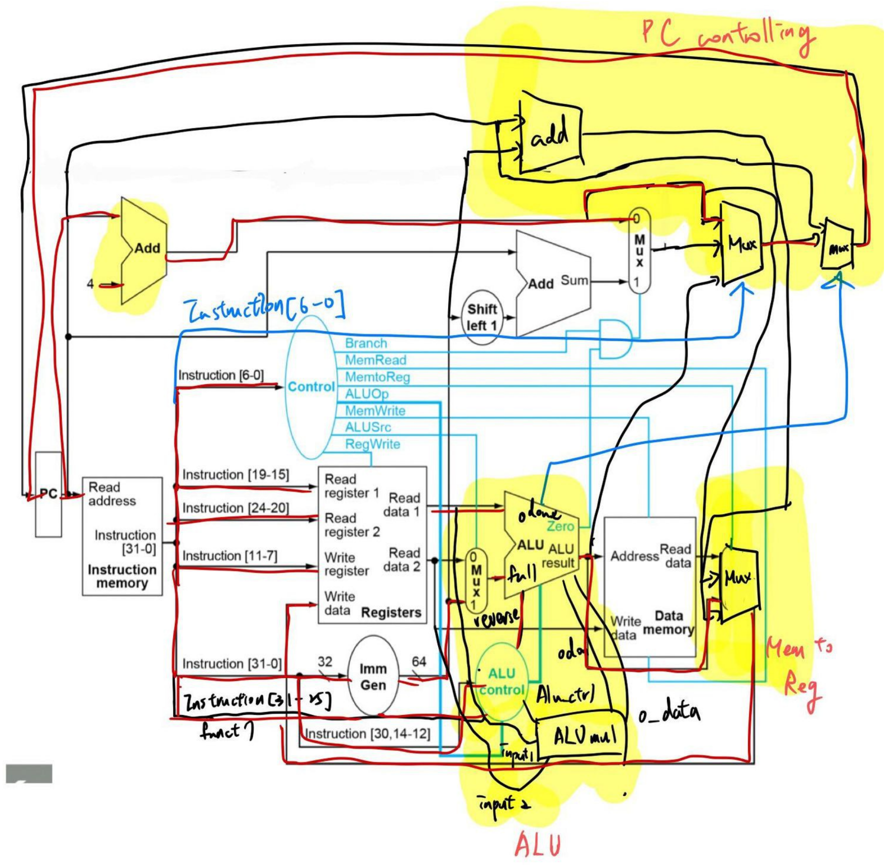
Just as how PC controlling is implemented, we use opcode to control MemtoReg.

Instruction type	MemtoReg
R	ALU_result (result from ALU unit)
I	ALU_result (result from ALU unit)
I_load	i_DMEM_rdata
S	Don’t write back to register
UJ (jal)	PC + 4
I_jalr	PC + 4
U (auipc)	PC + immediate
SB	Don’t write back to register

4、 Data path of Instructions

Since the lecture slides support R-type and load instruction data path, we only show I, S, UJ (jal), I_jalr, U (auipc), and SB type.

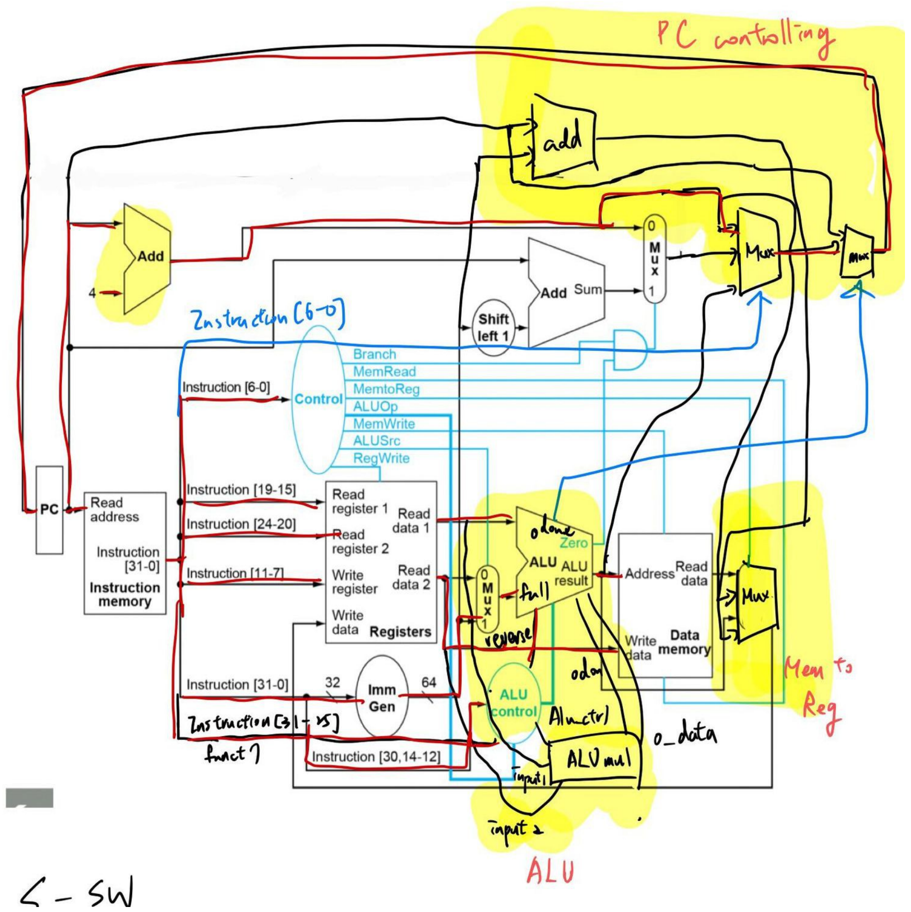
(1) I-type



J-TYPE

rs1 + immediate would be written to rs2, and PC+4 is next PC. The colored (red + blue) is the data path of I-type.

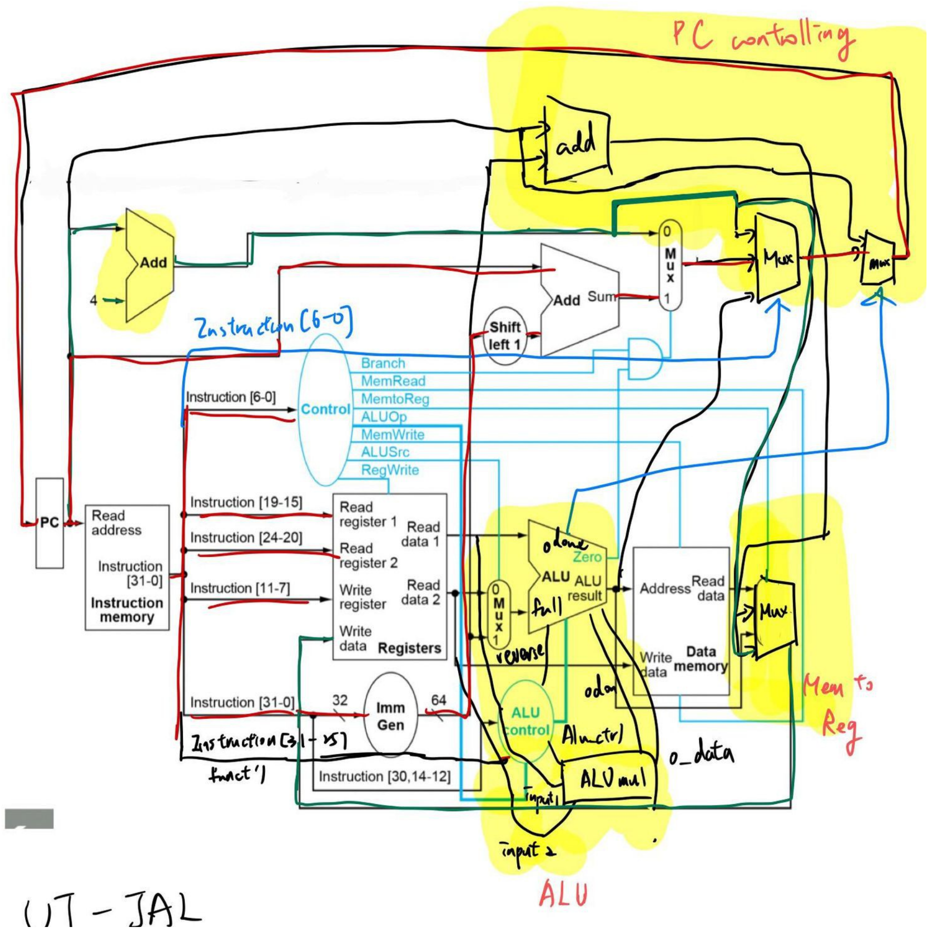
（二）S-type



<-SW

rs2 's data would be written to data memory with address equal to rs1 + immediate. The colored (red + blue) is the data path of S-type.

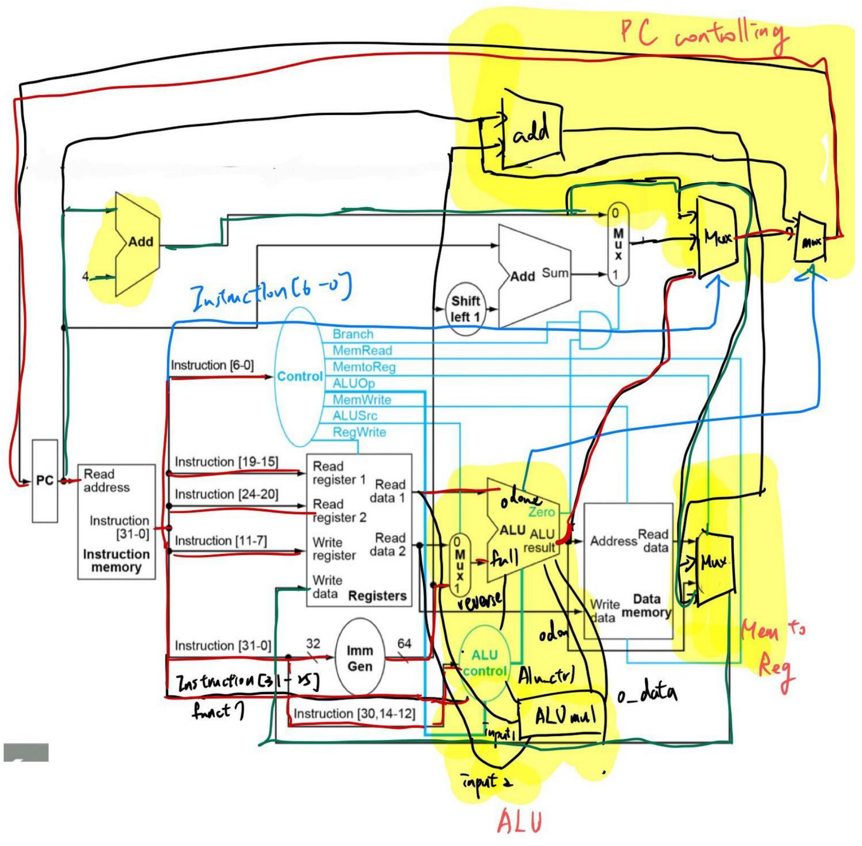
(三) UJ_jal



UJ - JAL

The next PC is PC + immediate <<1, and rd is written to PC + 4. The red line is the calculate of the next PC, and the green line is the written of PC + 4 to rd. The colored (red + blue + green) is the data path of UJ-type.

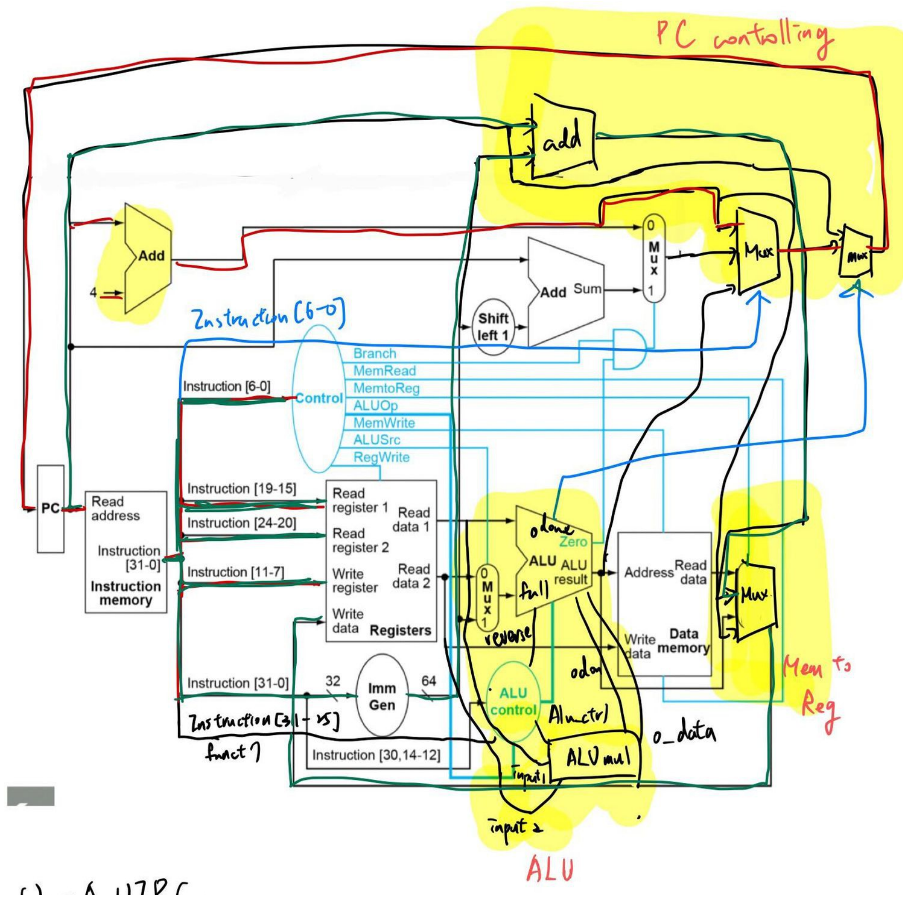
（四）I_jalr



I_JALR

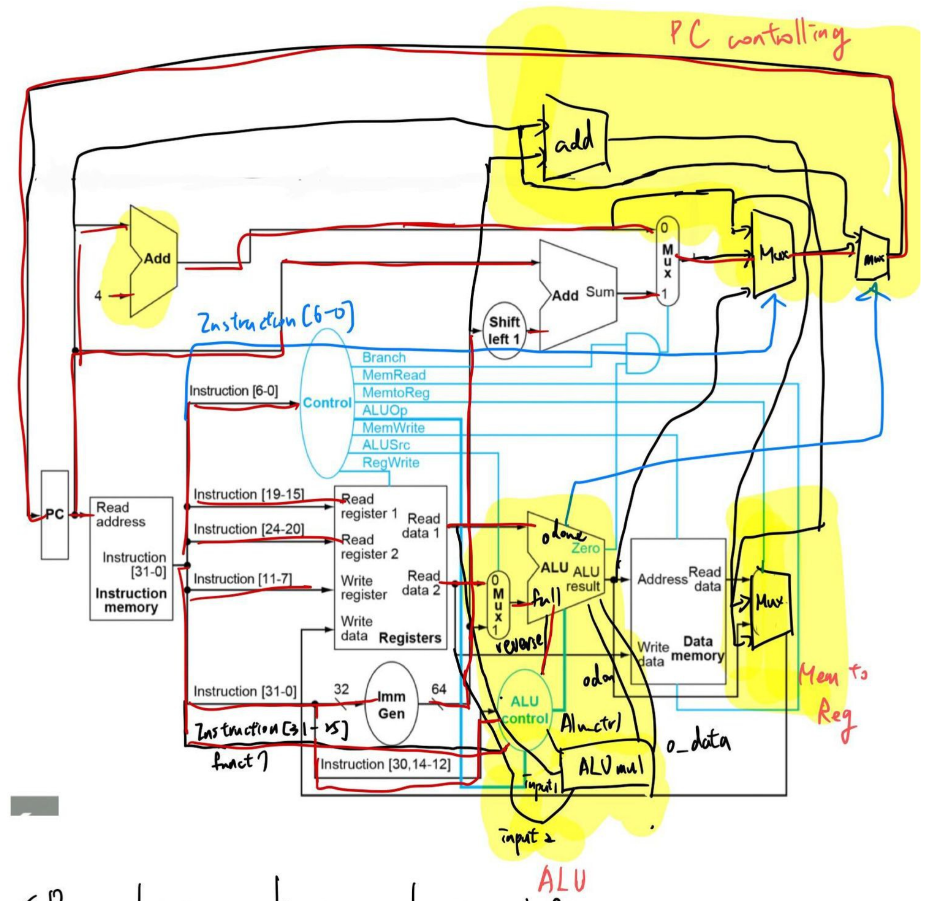
The next PC is rs1 + immediate, and rd is written to PC + 4. The red line is the calculate of the next PC, and the green line is the written of PC + 4 to rd. The colored (red + blue + green) is the data path of I-jalr.

（五）U (auipc)



The next PC is PC+4, and rd is written to PC + immediate. The red line is the calculate of the next PC, and the green line is the written of PC + immediate to rd. The colored (red + blue + green) is the data path of U-type (auipc).

（六）SB

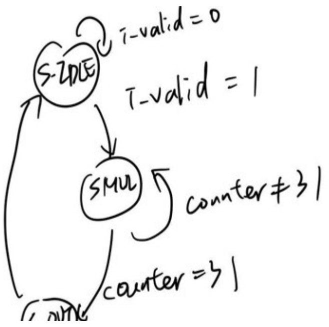


SB beq, bne, bge, blt

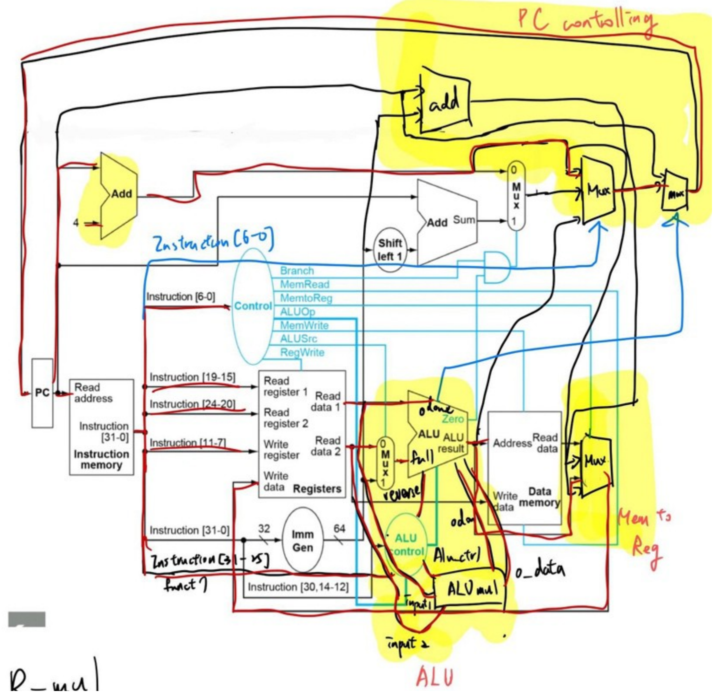
rs1_data and rs2_data is compared, if ALU_zero return true, next PC is updated to PC + immediate, else PC + 4. Since beq and bne are opposite instruction, so do bge and blt, thus "reverse" is the variable to control whether the result should be inverse. The colored (red + blue) is the data path of I-type.

5、 Mul instruction implementation

For mul operation, we use ALU_mul to handle calculation, clock and rst_n is required so that every calculation occurred on clock rising edge. Input1 and input2 is the two numbers need to multiply. ALU_ctrl is mul so that ALU_mul knows it is multiplication. i_valid is the control signal to set that the calculation is ready. To handle mul, we need state machine, which is below.

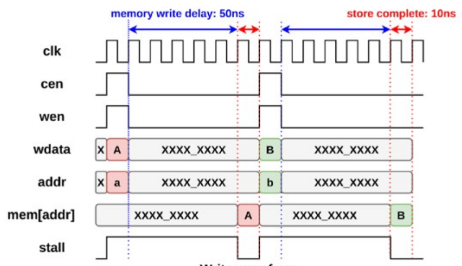


Initial state is S_IDLE, then when i_valid is true, the state goes to S_MUL. When counter doesn't reach 31, the next state is still S_MUL. After counter = 31, the next state is S_OUT which means the calculation is done. So, the output is o_data which is our multiply result, and o_don is set true to mean that the calculation is finish. After the calculation is finished, the next PC is updated to PC + 4. The below is the data path of mul operation.

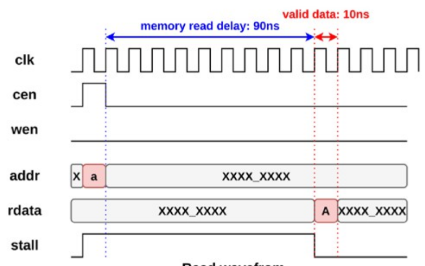


6、 Observation

1. To debug on Verilog, nWave is necessary and we need to choose which signals to display on nWave. If the simulation time exceeds, it means that maybe jal, jalr, or SB-type instruction doesn't jump to desired destination.
2. If the lw doesn't load word (ZZZZ appears in register file) it means that we don't wait for the i_DMEM_stall to load data. In order to let i_DMEM_stall load data o_DMEM_cen should be high. When i_DMEM_stall is low, this means that the data is available for us to read.
3. For sw, not only o_DMEM_cen need to be high, but also o_DMEM_wen need to be high too. When i_DMEM_stall is low, it is available for us to write data in memory.
4. The waveform of 2 & 3 is below. To accomplish this, we need to set state machine. We set state machine for cen and o_done.



Write wavefrom



Read wavefrom

5. For multicycle operation like “mul”, we need to be careful since we need to stall PC until the calculation is finished. So, we use control signal “o_done” to control whether PC can be updated or not. As mentioned in 4, since o_done is strongly relative to stall (when stall is low, sw and lw is finished) and mul (should be in S_OUT state to set o_done true), so it is better to set o_done as state machine.
6. For hw1 test case, we need to check where the data is stored in memory, if the data is stored lower in actual memory than in the address run in risc-v Jupiter, we need to reduce codes. Also, since the result is stored in a0 (x10), so we need to slightly modify the code so that the result is placed in x10.
7. It is a big challenge for us to implement CPU when we are not very familiar to Verilog. Before writing the code, we should better draw state machine to check our result is correct or not. However, it is still a good experience since it enhances our Verilog coding skill and review what we learn in class (CPU).

7、 Work distribution

Item	劉碩	于瑄
Implement CPU	v	
Implement module ALU	v	
Implement module ALUfull	v	
Implement module control		v
Implement module IMMGEN		v
Implement MemToReg	v	
Debug CHIP.v	v	v
Test I0	v (check waveform and modify code for every instruction)	v (calculate correct PC and value stored in register for every instruction)
Test I1	v(同上)	v（同上）
Implement I2 risc v		v
Debug I2 risc v and Test I2	v	
Test I3	v（同 Test I0）	v（同 Test I0）
Debug latch	v (比較多)	v（比較少）

report		v