Computer Architecture

Final Project Report

2020/7/3

B05901168 陳冠豪　　D07921016 王世全

1. CPU Architecture

The signal ALUState and PCControl are added in the CPU. ALUState is added for forming a FSM in ALU that operates under single cycle instructions (SCYCLE) or muti-cycle instructions (MCYCLE). The PCControl is added to control PC operations among PC+4, PC+ImmOut, and ALU's output that named ALUresult. A multiplexer Mux4 is re-designed for integrating the mux needs of PC states and that between memory data and ALUresult. In addition, the branch AND gate output is sent back to Control for determining what state of PCControl will be issued for B-TYPE instructions and others. A complete architecture is shown in Figure 1 below.

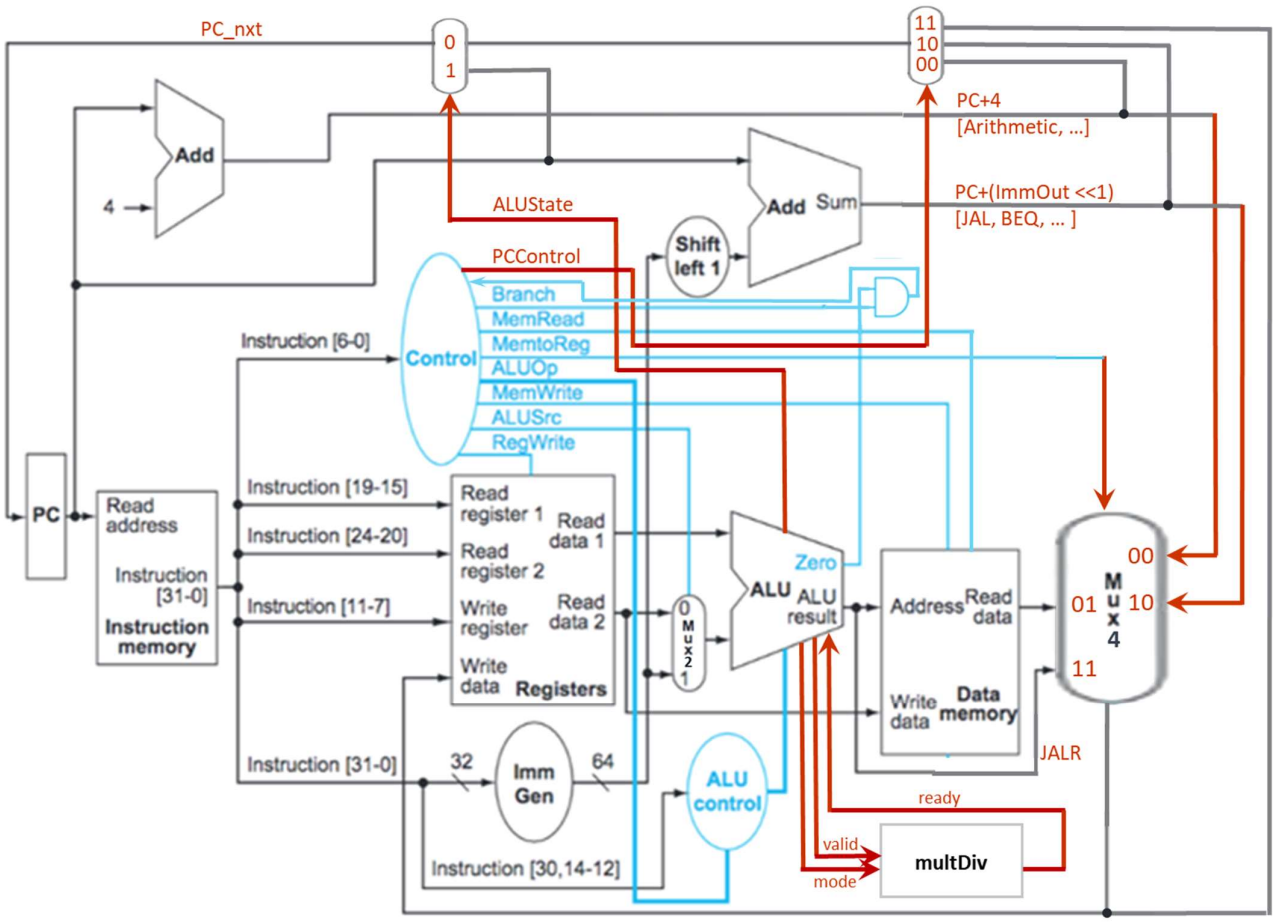


Figure 1　The CPU architecture

2. Data path of instructions

1) R-TYPE

The R-TYPE instructions include ADD, SUB, SLL, SLT, SLTU, XOR, SRL, SRA, OR, and AND. Their control signal are listed in Table 2.1. The ALUOp is determined by the instruction's "func" field and the state "000" controls ALU doing arithmetic operations. The ALUSignal controls the type of arithmetic operation in ALU as shown in Table 2.1. The instruction path is shown in the Figure 2.1 below.

Table 2.1 R-TYPE instructions' control signals

Operation	PCControl	Branch	MemRead	MemroReg	ALUOp	MemWrite	ALUSrc	RegWrite	ALUSignal	ALUState	valid	mode
ADD	00	0	0	00	000	0	0	1	0000	0	0	x
SUB	00	0	0	00	000	0	0	1	0001	0	0	x
SLL	00	0	0	00	000	0	0	1	0010	0	0	x
SLT	00	0	0	00	000	0	0	1	0011	0	0	x
SLTU	00	0	0	00	000	0	0	1	0100	0	0	x
XOR	00	0	0	00	000	0	0	1	0101	0	0	x
SRL	00	0	0	00	000	0	0	1	0110	0	0	x
SRA	00	0	0	00	000	0	0	1	0111	0	0	x
OR	00	0	0	00	000	0	0	1	1000	0	0	x
AND	00	0	0	00	000	0	0	1	1001	0	0	x

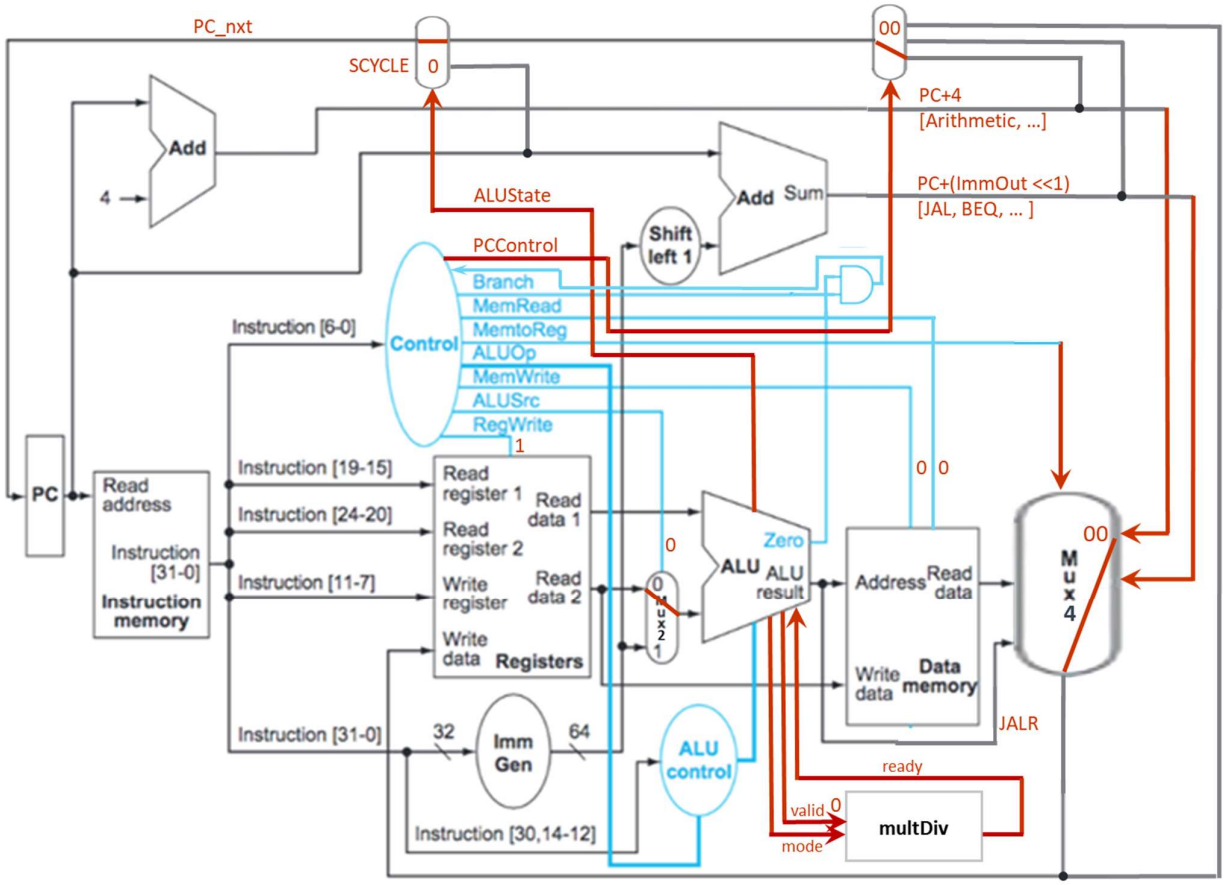


Figure 2.1 R-TYPE instructions' data path

2) I-TYPE

The I-TYPE instructions include ADDI, SLTI, SRLI, SRAI, etc. The ALUSignal controls the type of operation in ALU and the corresponding control signals are shown in Table 2.2. The instruction path is shown in the Figure 2.2 below.

Table 2.2 I-TYPE instructions' control signals

Operation	PCControl	Branch	MemRead	MemroReg	ALUOp	MemWrite	ALUSrc	RegWrite	ALUSignal	ALUState	valid	mode
ADDI	00	0	0	00	001	0	1	1	0000	0	0	x
SLLI	00	0	0	00	001	0	1	1	0010	0	0	x
SLTI	00	0	0	00	001	0	1	1	0011	0	0	x
SLTIU	00	0	0	00	001	0	1	1	0100	0	0	x
XORI	00	0	0	00	001	0	1	1	0101	0	0	x
SRLI	00	0	0	00	001	0	1	1	0110	0	0	x
SRAI	00	0	0	00	001	0	1	1	0111	0	0	x
ORI	00	0	0	00	001	0	1	1	1000	0	0	x
ANDI	00	0	0	00	001	0	1	1	1001	0	0	x

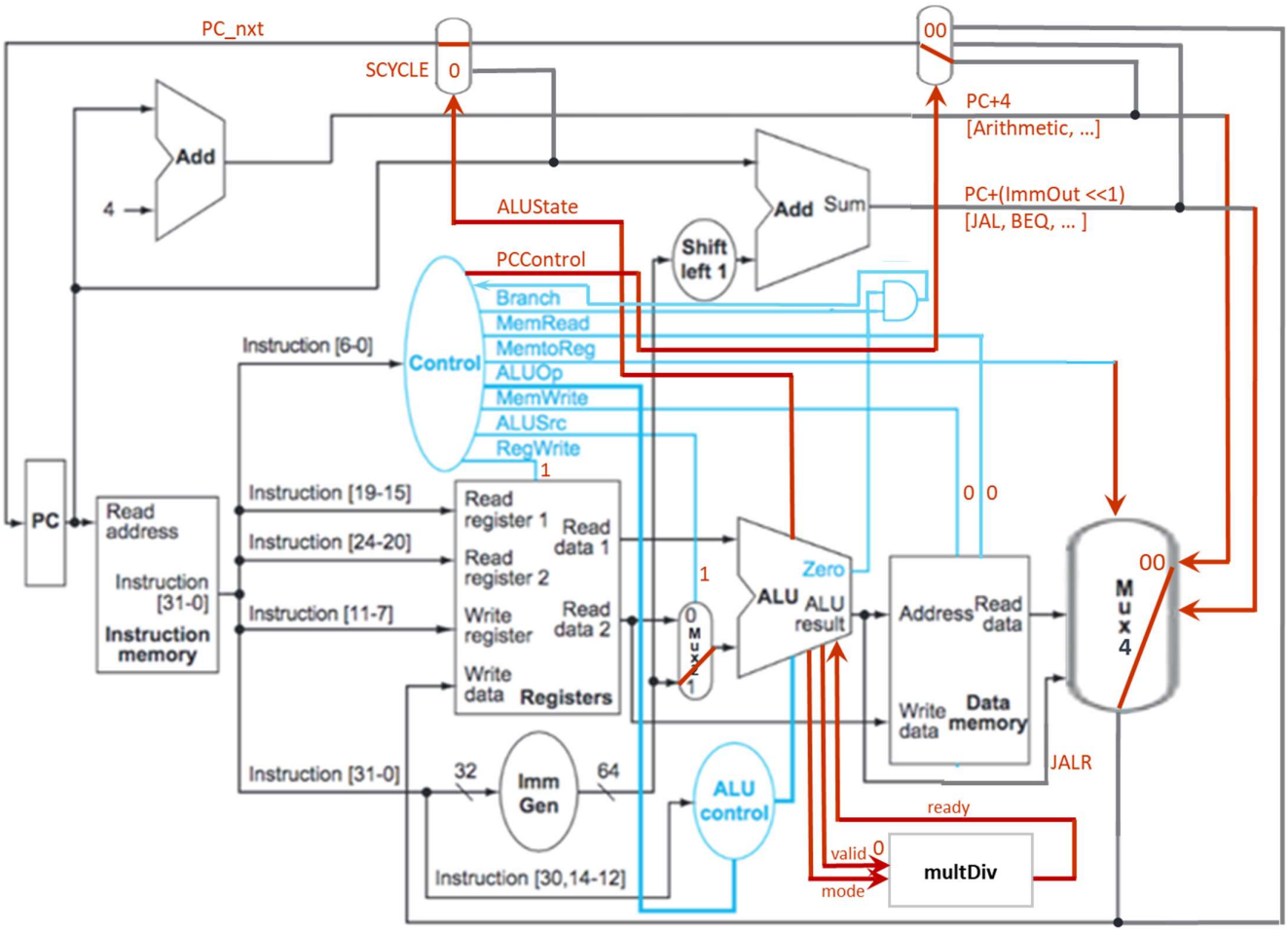


Figure 2.2 I-TYPE instructions' data path

3) B-TYPE

The B-TYPE instructions include BEQ and BNE, etc. The ALUSignal controls the type of operation in ALU and the corresponding control signals are shown in Table 2.3. The instruction path is shown in the Figure 2.3 below.

Table 2.3 B-TYPE instructions' control signals

Operation	PCControl	Branch	MemRead	MemroReg	ALUOp	MemWrite	ALUSrc	RegWrite	ALUSignal	ALUState	valid	mode
BEQ	01	1	0	00	011	0	0	1	0001	0	0	x
BNE	01	1	0	00	011	0	0	1	0001	0	0	x

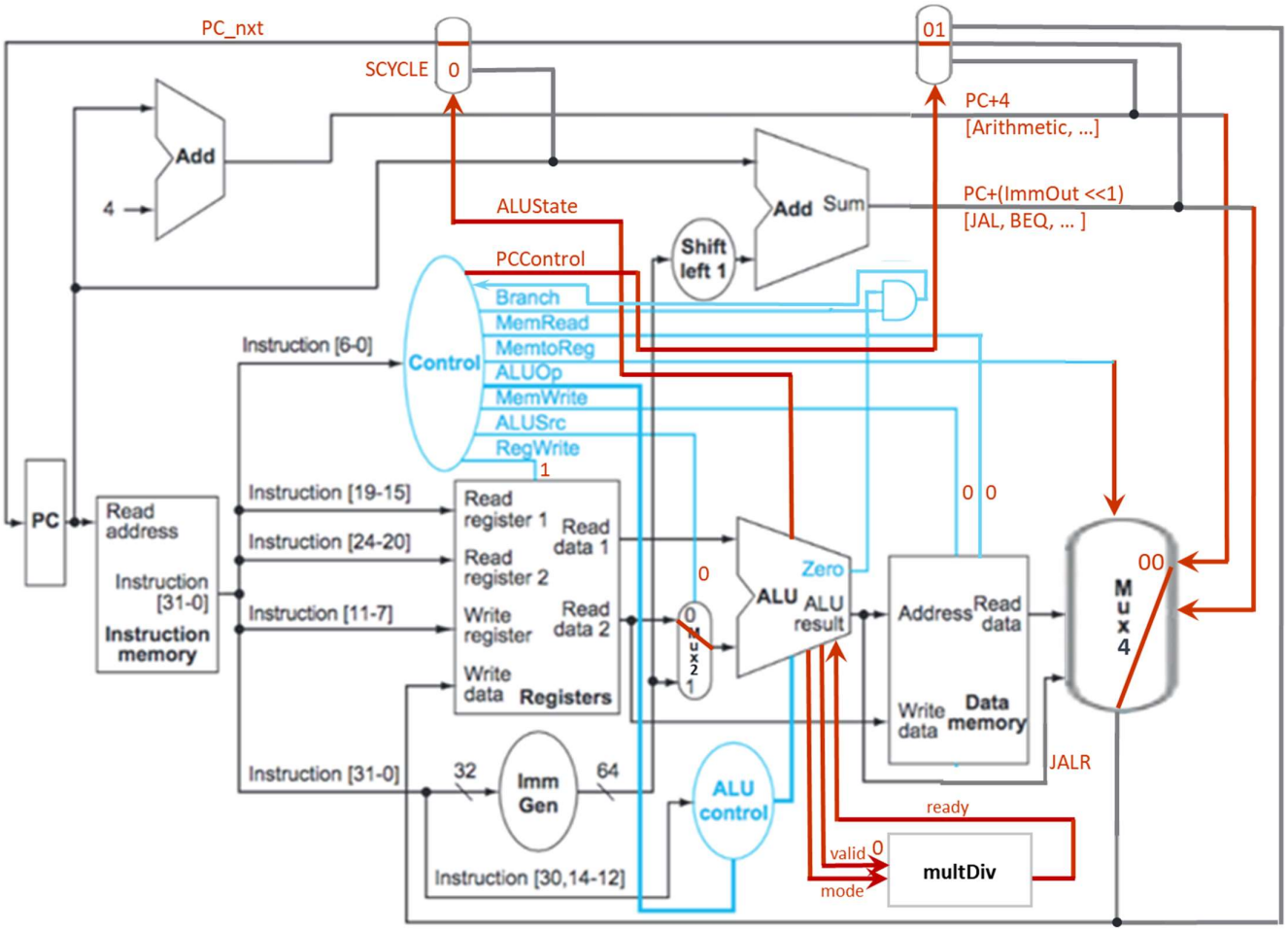


Figure 2.3 B-TYPE instructions' data path

4) LI-TYPE

The LI-TYPE instructions are memory data load instructions such as LW, etc. The ALUSignal controls the type of operation in ALU and the corresponding control signals are shown in Table 2.4. The instruction path is shown in the Figure 2.4 below.

Table 2.4 LI-TYPE instructions' control signals

Operation	PCControl	Branch	MemRead	MemroReg	ALUOp	MemWrite	ALUSrc	RegWrite	ALUSignal	ALUState	valid	mode
LW	00	0	1	01	110	0	1	1	0000	0	0	x

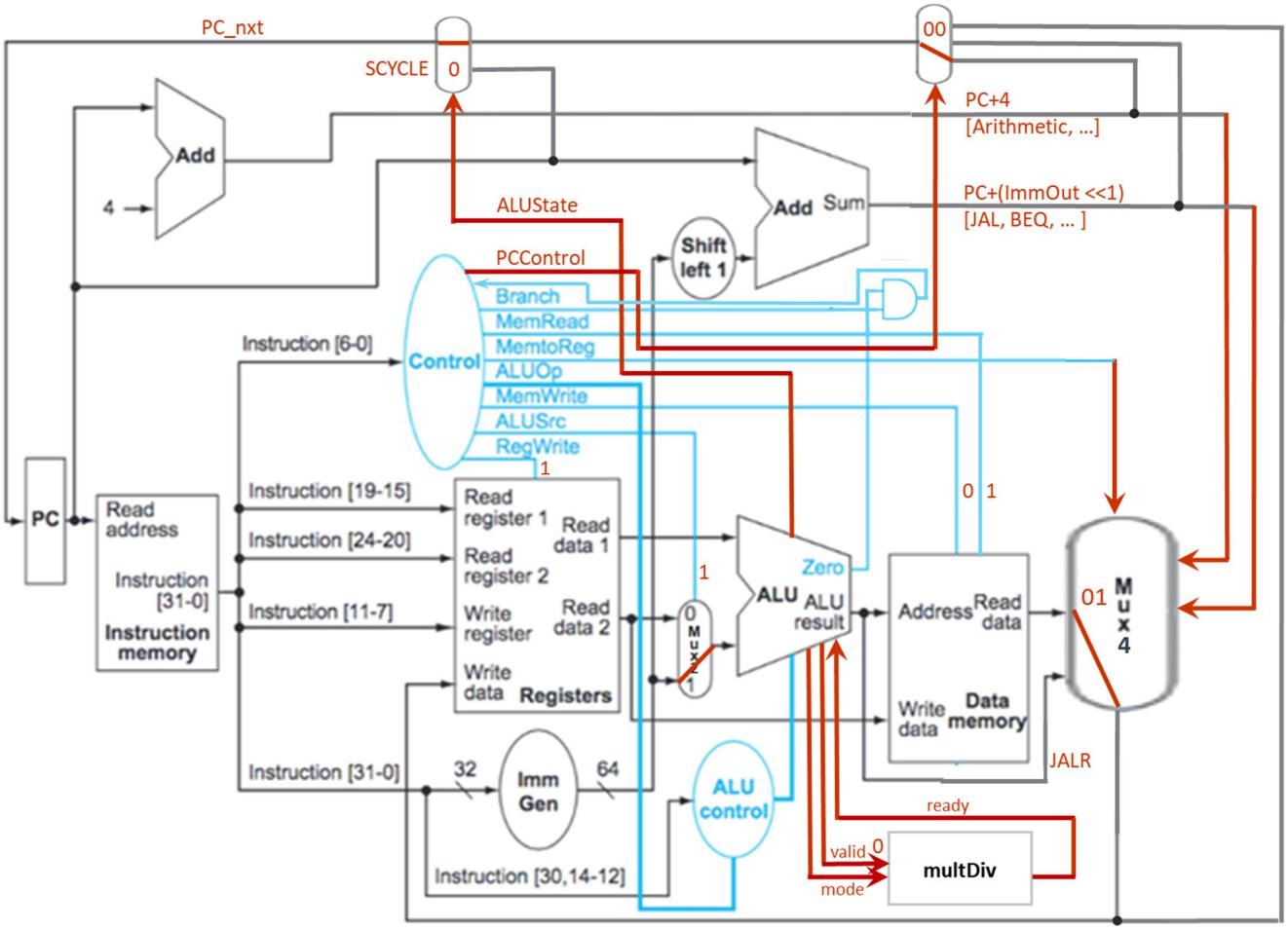


Figure 2.4 LI-TYPE instructions' data path

5) S-TYPE

The S-TYPE instructions are memory data store instructions such as SW, etc. The ALUSignal controls the type of operation in ALU and the corresponding control signals are shown in Table 2.5. The instruction path is shown in the Figure 2.5 below.

Table 2.5 S-TYPE instructions' control signals

Operation	PCControl	Branch	MemRead	MemroReg	ALUOp	MemWrite	ALUSrc	RegWrite	ALUSignal	ALUState	valid	mode
SW	00	0	0	00	010	1	1	0	0000	0	0	x

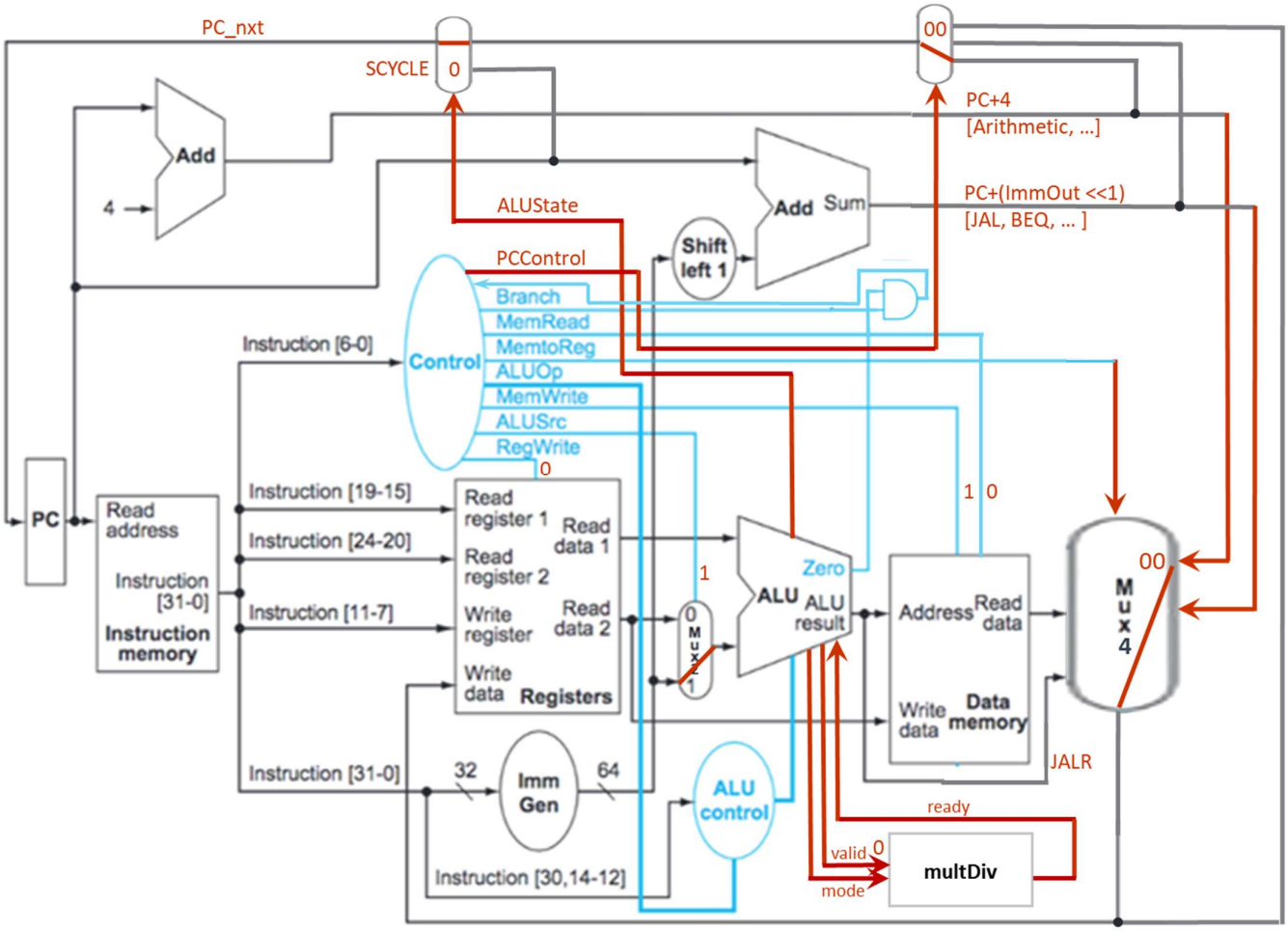


Figure 2.5 S-TYPE instructions' data path

6) U-TYPE

The U-TYPE instructions adds a 20-bit upper immediate to the PC such as AUIPC. The ALUSignal controls the type of operation in ALU and the corresponding control signals are shown in Table 2.6. The instruction path is shown in the Figure 2.6 below.

Table 2.6 U-TYPE instructions' control signals

Operation	PCControl	Branch	MemRead	MemroReg	ALUOp	MemWrite	ALUSrc	RegWrite	ALUSignal	ALUState	valid	mode
AUIPC	00	0	0	10	100	0	1	1	0000	0	0	x

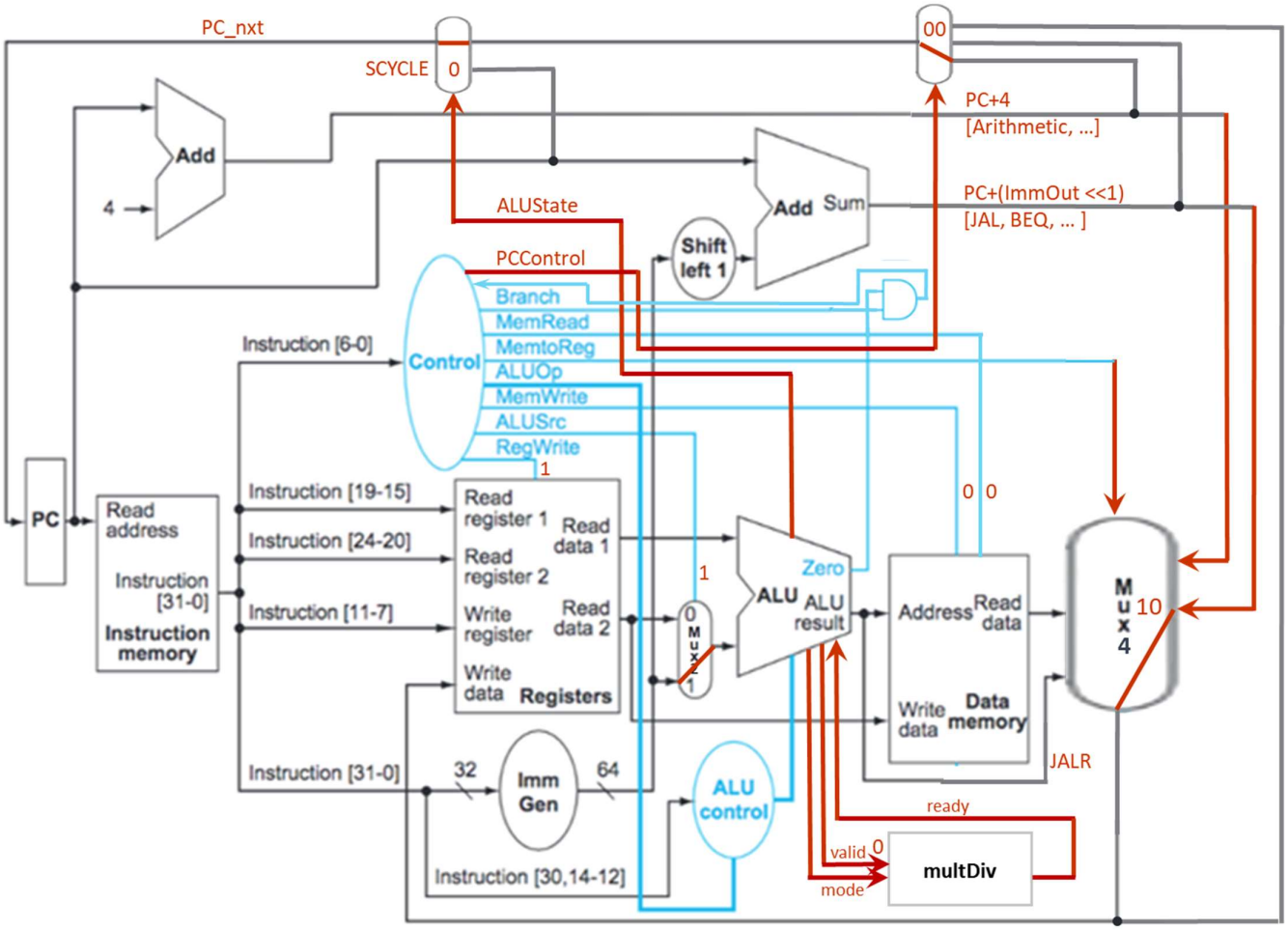


Figure 2.6 U-TYPE instructions' data path

7) J-TYPE

The J-TYPE instructions performs procedure call such as JAL. The ALUSignal controls the type of operation in ALU and the corresponding control signals are shown in Table 2.7. The instruction path is shown in the Figure 2.7 below.

Table 2.7 J-TYPE instructions' control signals

Operation	PCControl	Branch	MemRead	MemroReg	ALUOp	MemWrite	ALUSrc	RegWrite	ALUSignal	ALUState	valid	mode
JAL	10	0	0	11	101	0	1	1	0000	0	0	x

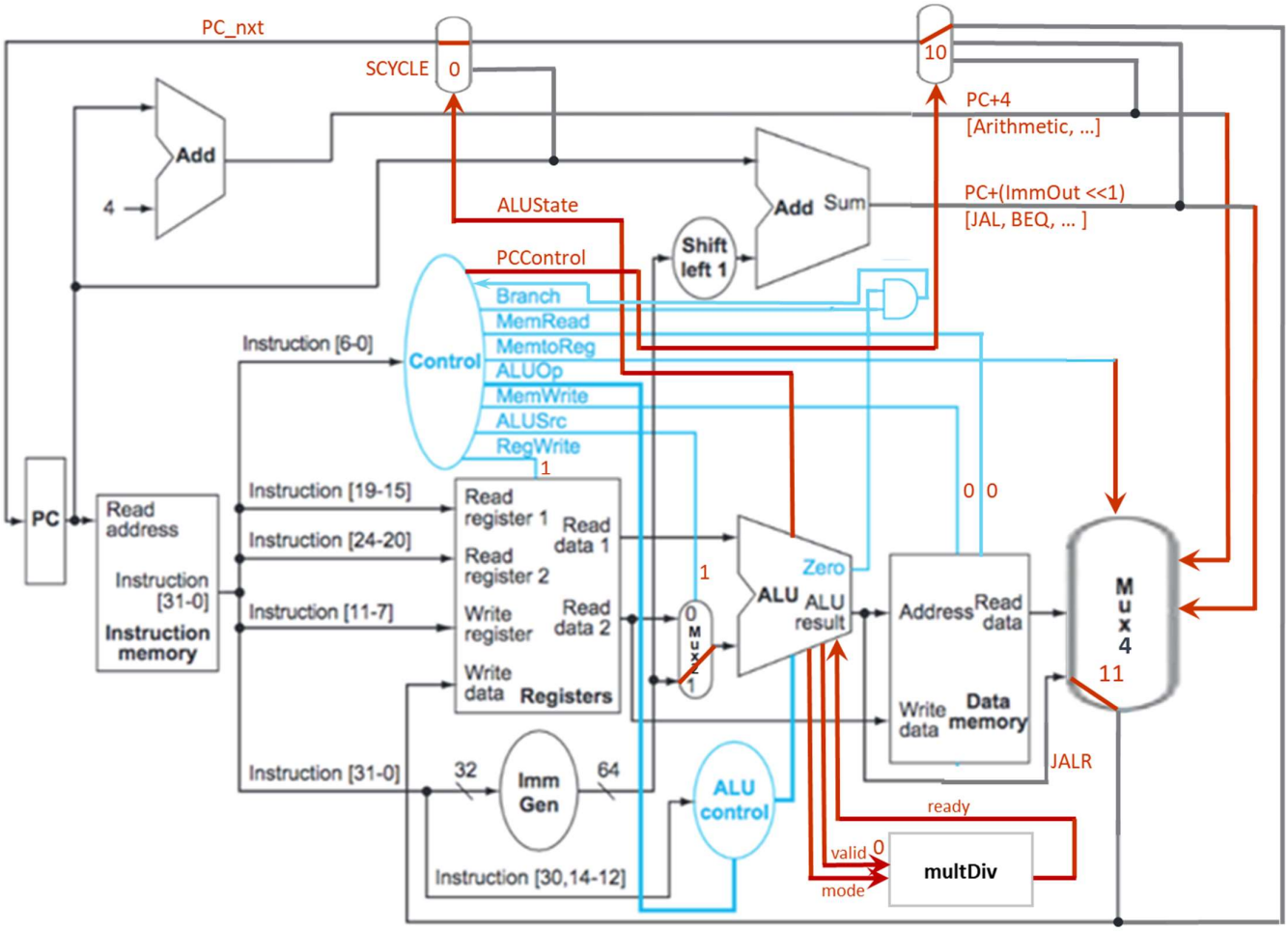


Figure 2.7 J-TYPE instructions' data path

8) JI-TYPE

The JI-TYPE instructions performs a procedure return such as JALR. The ALUSignal controls the type of operation in ALU and the corresponding control signals are shown in Table 2.8. The instruction path is shown in the Figure 2.8 below.

Table 2.8 JI-TYPE instructions' control signals

Operation	PCControl	Branch	MemRead	MemroReg	ALUOp	MemWrite	ALUSrc	RegWrite	ALUSignal	ALUState	valid	mode
JALR	11	0	0	11	111	0	1	1	0000	0	0	x

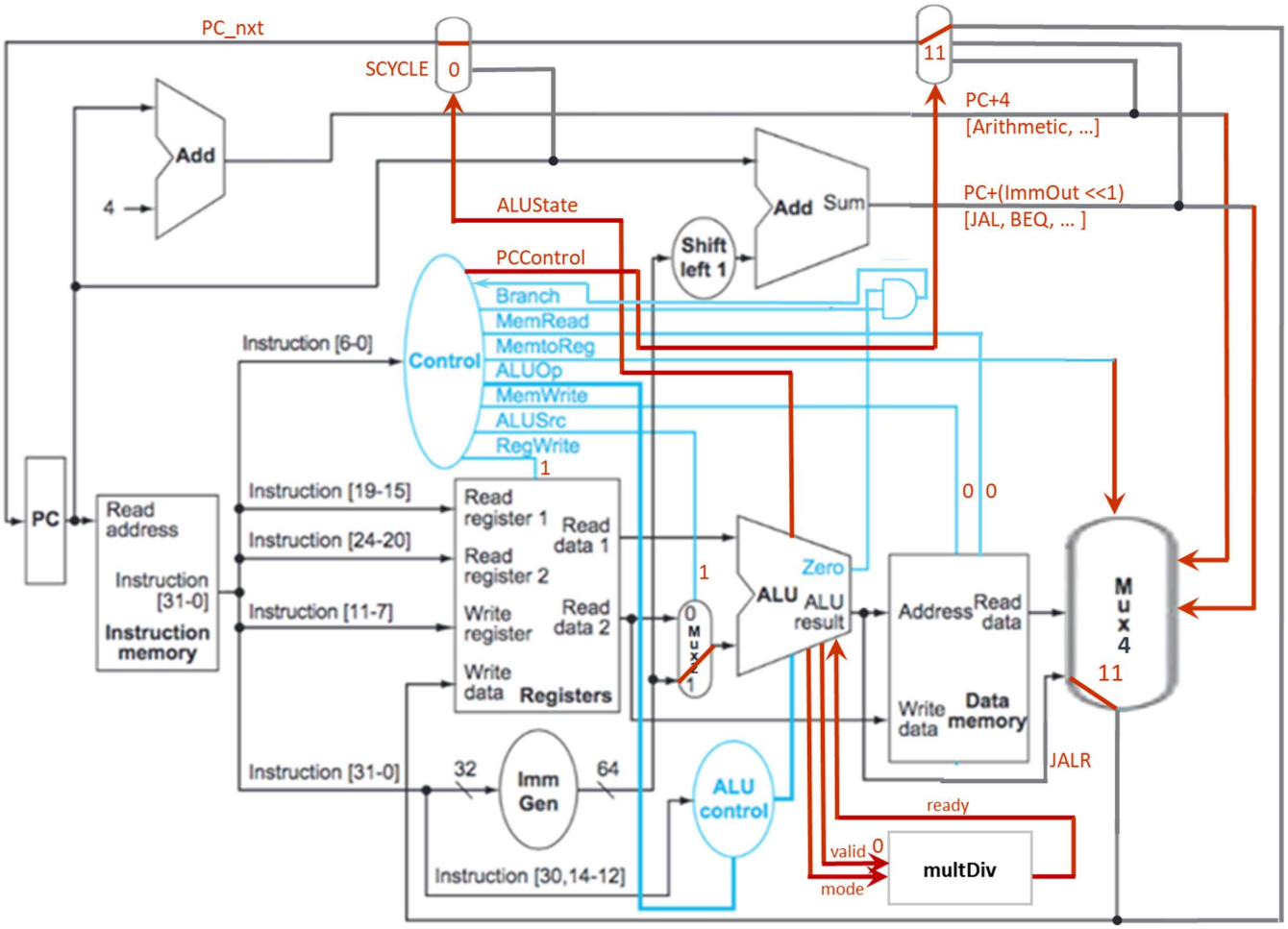


Figure 2.8 JI-TYPE instructions' data path

3. Multi-cycle instructions

The Multi-cycle instructions, hereinafter MCYCLE instructions, are those requesting for arithmetic multiplication and division such as MUL and DIV. The CPU contains a multDiv unit specially designed for either of the arithmetic operations. It is composed of an operand register, ALU, and a left/right shift register and execute multiplication and division base on input control signals valid and mode. While operating, the state signal ALUState is raise up to 1 so no other instructions will be executed under the state as PC_nxt is set to PC during the period. After 32 cycles, after either multiplication or division result appear, the multDiv raises ready state signal to 1 which ends the MCYCLE operation while turning the ALUState back to 0. Then the CPU will continue to the next instruction. The ALUSignal informs ALU for initiating signal valid to 1 and turn the CPU goes to MCYCLE instruction mode.

The corresponding control signals are shown in Table 3 and the instruction path is shown in the Figure 3 below.

Table 3 MCYCLE instructions' control signals

Operation	PCControl	Branch	MemRead	MemroReg	ALUOp	MemWrite	ALUSrc	RegWrite	ALUSignal	ALUState	valid	mode
MUL	xx	0	0	00	000	0	0	1	1010	1	1	0
DIV	xx	0	0	00	000	0	0	1	1011	1	1	1

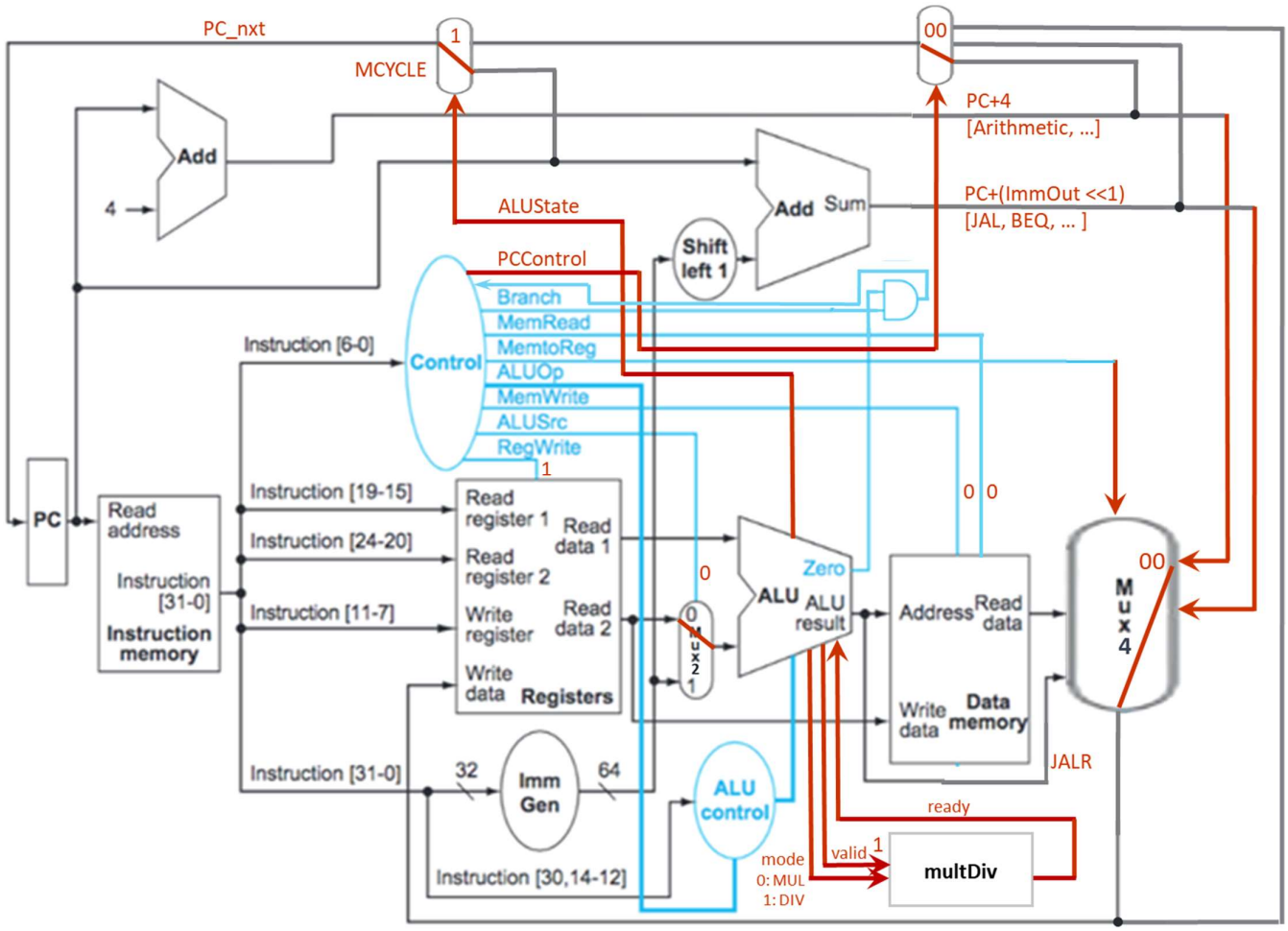


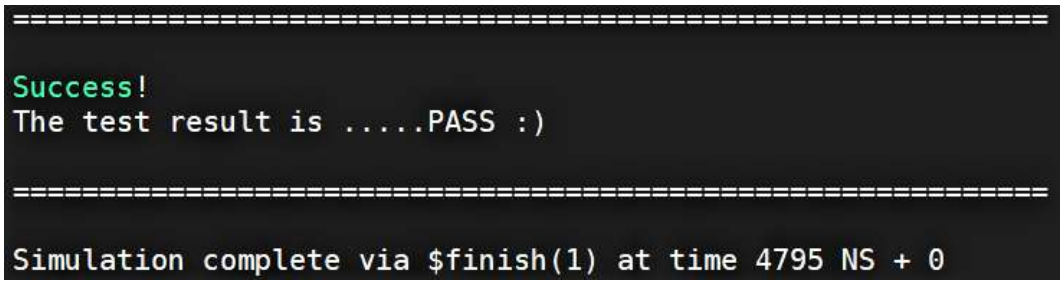
Figure 3 MCYCLE instructions' data path

4. Total simulation time

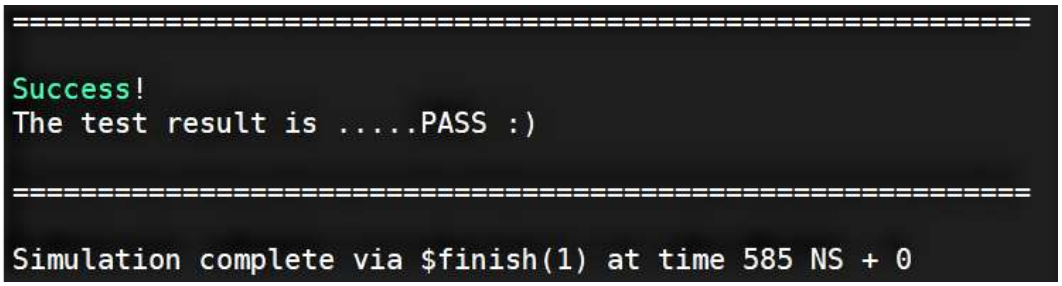
1) Leaf: a = 1, b = 9, c = 2, d = 2

```
==================================================================

Success!
The test result is .....PASS :)

==================================================================

Simulation complete via $finish(1) at time 255 NS + 0
```

2) Fact: n = 10



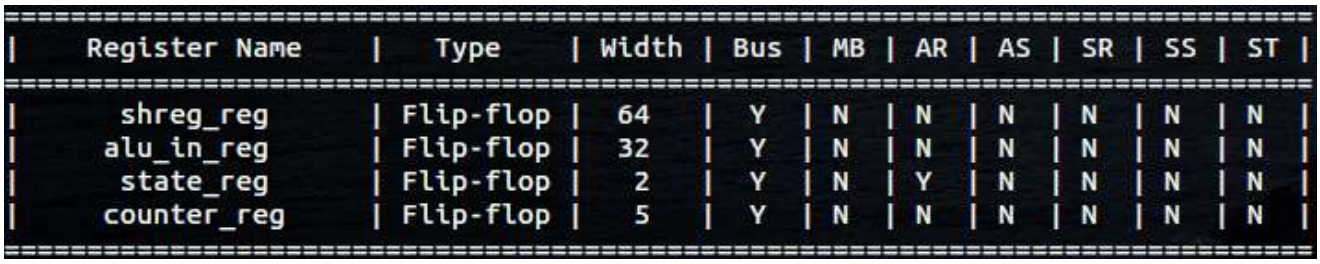
3) (Bonus) HW1: n = 10



5. Observation

1) Our CPU design passes the two required tests, leaf and fact, with spent time meeting the requirement. The results are shown above. We did also apply different values on a, b, c, d for leaf and n for fact for inspecting correctness of the design and the results were positive.
2) Our CPU design passes the additional bonus test, hw1, while implementing SLL and SRL instructions in our design. The result is shown above. We did also apply different value on n in this test for inspecting correctness of the design. The results were all positive.
3) A better process for such a CPU design incorporating several states and multiple instructions is to architect a sound data path design before code implementation so as to avoid the possible workloads back-and-forth between updating the architecture and debugging the Verilog code if the architecture was not made completely in beginning.
4) Adding a FSM to ALU making the CPU easier handling SCYCLE and MCYCLE instructions with the integrated multDiv unit. The state signal ALUState is updated following the state change in the FSM in the ALU of the CPU.
5) It would be interested in converting the multDiv into a “Faster” architecture applyied with parallel architecture. Such a Faster multDiv can finish both multiplication and division in a single cycle, which would remove the handling mechanism currently in this CPU for SCYCLE and MCYCLE instructions. However it would result in cost adder in real world applications.

6. Snapshot of the “Register Table”



Register Name	Type	Width	Bus	MB	AR	AS	SR	SS	ST
shreg_reg	Flip-flop	64	Y	N	N	N	N	N	N
alu_in_reg	Flip-flop	32	Y	N	N	N	N	N	N
state_reg	Flip-flop	2	Y	N	Y	N	N	N	N
counter_reg	Flip-flop	5	Y	N	N	N	N	N	N

7. Work distribution

	陳冠豪	王世全
CHIP.v	V	V
Control.v	V	V
Alu_Control.v	V	V
Alu.v	V	V
Imm_Gen.v	V	
Mux.v	V	
Required test: Leaf	V	V
Required test: fact	V	V
Bonus test: hw1	V	V
Report		V