EE-224 (Project)

Little Computer Architecture

Sarvadnya Desai	-	210040138
Ishaan Manhar	-	210070033
Harsh Raj Chaudhari	-	210040060
Akshat Taparia	-	210110014

Specifications!

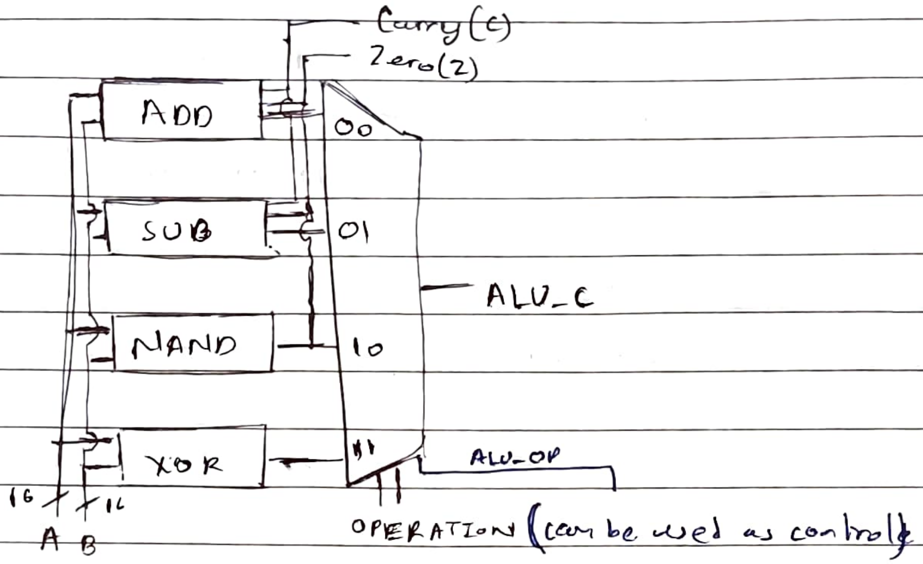
- ALU: Functions ADD, SUB, NAND, XOR along with Carry and Zero flags.

- Programmer Register (Register File): Contains 8 general purpose registers (R0 to R7) with R7 as PC (program counter)

- Multiple instanciations of Temporary Registers (T)

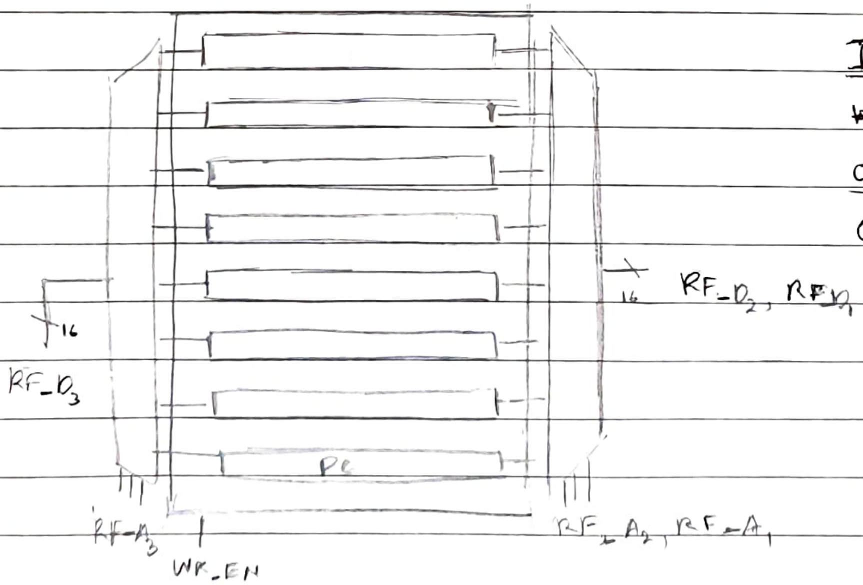
- Sign extender: A 16-bit extention to add leading zeros to numbers less than 16-bits.

- ALU DESIGN:



Input : ALU_A, ALU_B (16 bit)
Control : Operation function code -(2 bit), ALU_OP
Outputs : ALU_C and Carry and Zero flags
(Operations Are functional.)

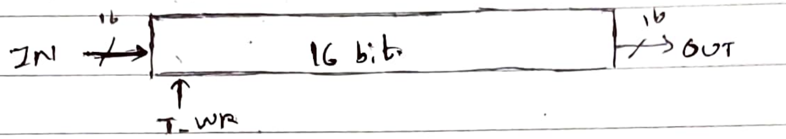
- Register File (8 registers):



INPUTS: IN (16-bit), ~~WR_ADD~~ RF_D3, RF_A1, RF_A2, RF_A3, ~~RD_ADD~~
OUTPUTS: ~~Out~~ RF_D2, RF_D1 (16-bit)
CONTROL: WR_EN,

We have 1 input demultiplexer and 2 output (read) multiplexers connected to the RF.

Temporary Register (T):



Sign Extention:

To be performed 'via bit shifting using behavioural modeling.

Concatenate leading zeroes / (or 1s) as required.

LW(LOAD):

15 12	11 9	8 6	5 0
0100	RA	RB	Imm (6 bit)

① Read Instructions :

PC → Mem_add	
Mem_data → T1	T1_WR
PC → ALU_A	
+2 → ALU_B	ALU_OP
ALU_C → PC	WR_EN

② Understand / Read operands

T1(8-6) → RF_A1	
RF_D1 → T2	T2_WR

③ Compute Address :

T2 → ALU_A	
Sign_Ext(T1(5-0)) → ALU_B	ALU_OP
ALU_C → T2	T2_WR

④ Memory Reading :

T_2 → Mem_Add	
Mem_Data → T_2	T2_WR

⑤ Update :

T1(11-9) → RF_A3	
T2 → RF_D3	WR_EN

SW (STORE):

15 12	11 9	8 6	5 0
0101	RA	RB	IMM

① Read ~~0101~~ Instruction:

PC → Mem_add	
Mem_data → T1	T1_WR
PC → ALU_A	
+2 → ALU_B	
ALU_C → PC	WR_EN

② Read Operands :

T1 (8-6) → RF_A1	
RF_D1 → T2	T2_WR
T2(11-9) → RF_A2	
RF_D2 → T3	T3_WR

③ Compute Address:

T2 → ALU_A	
Sign Ext (T1 (5-0)) → ALU_B	
ALU_C → T2	T2_WR

④ Transfer Data to memory:

T2 → Mem_add	
T3 → Mem_data	MEM_WR

LM (Load Multiple):

15	12 11	9	7 0
0110	RA	0	Bits

① Read Instruction:

PC → Mem_add	
Mem_data → T1	T1_WR
PC → ALU_A	
+2 → ALU_B	ALU_OP
ALU_C → PC	WR_EN

② Store the next instruction, Read Operands:

PC → T2	T2_WR
T1(11-9) → RF_A1	
RF_D1 → T3	T3_WR

③ SETTING UP FOR LOOPING:

T3 → MEM_ADD	
MEM_DATA → T4	T4_WR
T3 → ALU_A	
+2 → ALU_B	ALU_OP
ALU_C → T3	T3_WR
000 → T5(2-0)	T5_WR

④ The Looping State:

T5(2-0) → RF_A3	
T4 → RF_D3	T1(0) → WR_EN
T5 → ALU_A	
+1 → ALU_B	ALU_OP
ALU_C → T5	T5_WR
T3 → MEM_ADD	
MEM_DATA → T4	T1(0) → T4_WR
T3 → ALU_A	
+2 → ALU_B	T1(0) → ALU_OP
ALU_C → T3	T1(0) → T3_WR

Repeat ④ for control T1(1), T1(2),... T1(6)

⑤ Writing R7:

T5(2-0) → RF_A3	T1(7) → WR_EN
~~RF_D3~~ T4 → RF_D3	

STORE MULTIPLE (SM):

15	12 11	9	7 0
0101	RA	0	Bits

① Read Instruction:

PC → Mem_Add	
Mem_data → T1	T1_WR
PC → ALU_A	
+2 → ALU_B	ALU_OP
ALU_C → PC	WR_EN

② Initiate counter, Read Operand:

000 → T2(2-0)	T2_WR
T1(11-9) → RF_A1	
RF_D1 → T3	T3_WR

~~T3 →~~

~~③~~

T4 = content

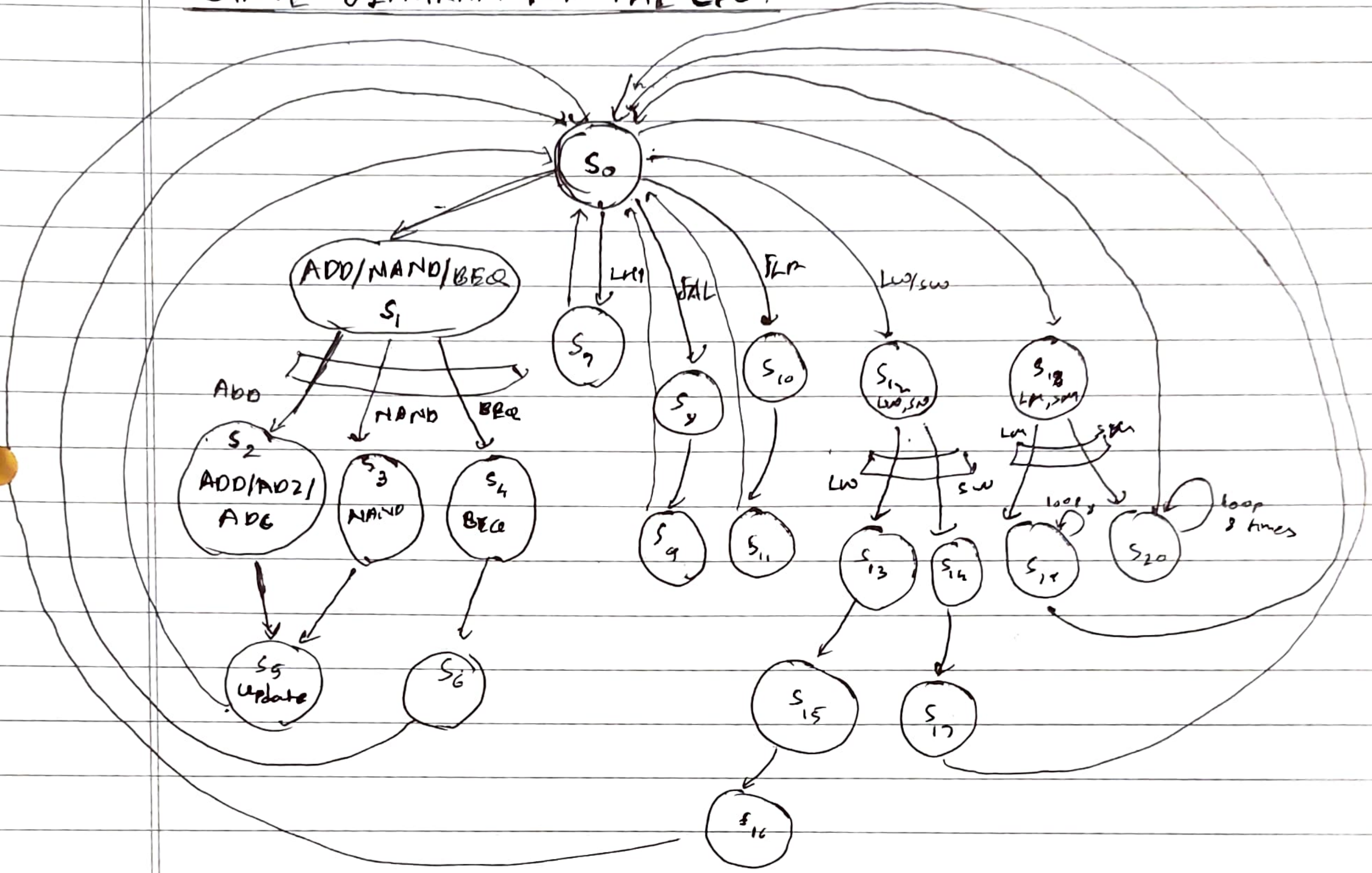
~~T3 → MEM_ADD~~	
~~MEM_DATA → T4~~	~~T4_WR~~
~~T3 → ALU_A~~	
~~+2 → ALU_B~~	~~T1(0) → ALU-OP~~
~~ALU_C → T3~~	~~T1(0) → T3_WR~~
~~T2(2-0) → RF_A3~~	
~~T4 → RF-D3~~	~~T1(0) → ~~RR~~-EN~~
~~T2 → ~~RF-Y~~ ALU_A~~	
~~+1 → ALU_B~~	
~~ALU_C → T2~~	~~T2_WR~~

③ Control Loop:

T2(2-0) → RF_A1	
RF_D1 → T4	T1(0) → T4_WR
T3 → MEM-ADD	
T4_WR ⟹ MEM-DATA	T1(0) → MEM-WR
T3 → ALU_A	
+2 → ALU_B	T1(0) → ALU_OP
ALU_C → T3	T1(0) → T3_WR
T2 → ALU_A	
+1 → ALU_B	ALU_OP
ALU_E → T2	T2_WR

- Repeat step ③ for T1(0), T1(1), T1(2) ...... T1(7).

STATE DIAGRAM FOR THE CPU:



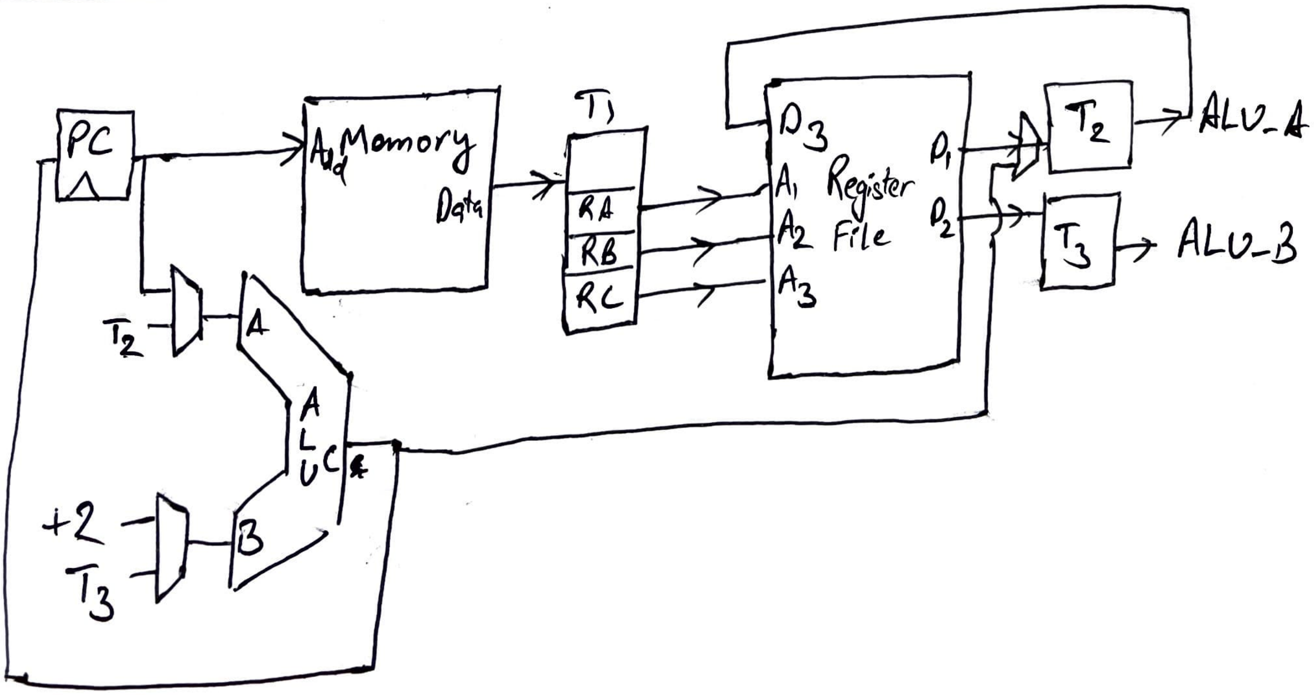
ADD rc, ra, rb

S1	PC → Mem_Add Mem_Data → T_1 PC → ALU_A +2 → ALU_B ALU_C → PC	MRD PCE ADD T_1:E

S2	T_1(11-9) → RF_A_1 T_1(8-6) → RF_A_2 RF_D_1 → T_2 RF_D_2 → T_3	T_2:E T_3:E

S3	T_2 → ALU_A T_3 → ALU_B ALU_C → T_2	ADD T_2:E

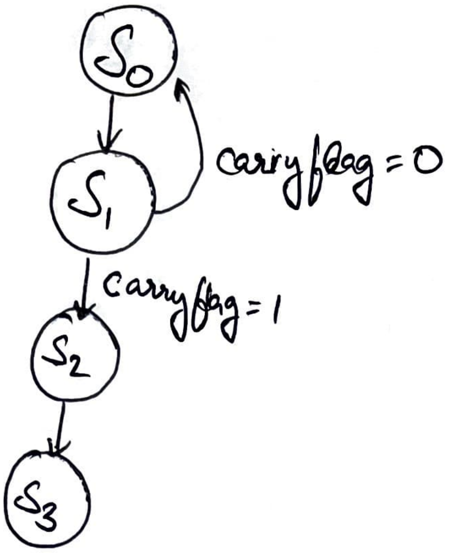
S4	T_2 → RF_D_3 $T_{1(5-3)}$ → RF_A_3	RFWE



ADC and ADZ

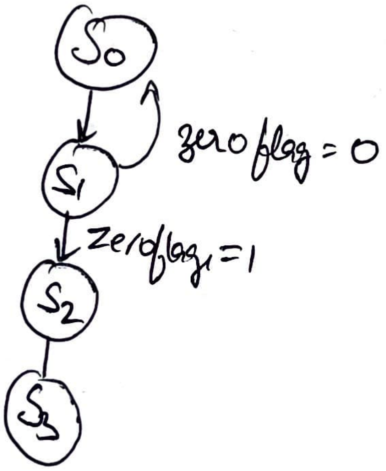
ADC rc, ra, rb

All the states are same as ADD, but there is one change in state diagram.



ADZ rc, ra, rb

Simillarly states are same but change in state diagram.



Clubbing the three State diagrams (ADD, ADC, ADZ)

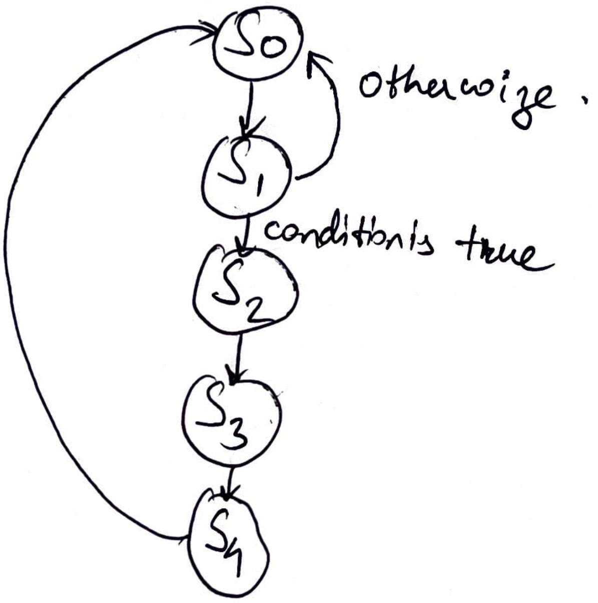
C.Z → 2 bits that we have in instruction.

C_F, Z_F → carryflag & zero flag generated in ALU & stored in a d-flip flop.

Truth Table when we perform Addition (go to S_2 from S_1)

C	Z	C_F	Z_F
0	0	X	X
1	0	1	X
0	1	X	1

condition = $\bar{C}\bar{Z} + C.C\_F + Z.Z\_F$

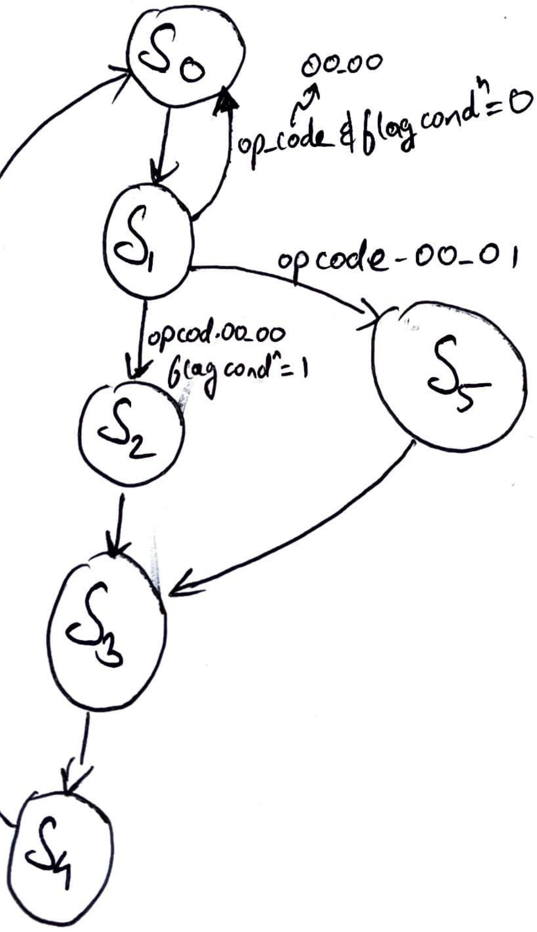


ADI rb, ra, imm6

All the states are the same except it ~~f~~ S_2.

S_5 T_1(11-9) → RF_A_1 T_1(5-0) → SE6_IN SE6_OUT → T_2 RF_D_3 → T_3	T_2 : E T_3 : E

State diagram



NDU | NDC | NDZ

S_1:

PC → Mem_Add Mem_Add → T_1 (16-bits)	T1·WR
PC → ALU A +2 → ALU B	ADD
ALU_C → PC	WR_EN

S_2:

$T_{1(11-9)}$ → RF_A_1 $T_{1(8-6)}$ → RF_A_2 RF_D_1 → T_2 RF_D_2 → T_3	T2_WR T3_WR

S_3:

T_2 → ALU_A T_3 → ALU_B ALU_C → T_2	NAND

S_4:

T_2 → RF_A_3 $T_{1(5-3)}$ → RF_D_3

to incorporate NDZ and NDC we check condition: $z \cdot z1 + c \cdot c1 + c'z' = 1$ before entering state 2.

LHI:	00_11	RA	Imm
	15-12	11→9	8-0

S_1:	PC → Mem_Add	T1_WR
	Mem_Data → T1	
	PC → ALU_A	
	+2 → ALU_B	ADD
	ALU_C → PC	WR_EN

S_2:	$T_{1(8\to0)} \rightarrow \boxed{7S}$	WR_EN
	$\boxed{7S} \rightarrow RF\_D_3$	
	$T_{1(11\to9)} \rightarrow RF\_A_3$	

⇒ Shifts the 9 bits Imm by 7 bits
⇓
Multipling by 10000000

BEQ:	11_00	RA	RB	Imm
	15→12	11→9	8→6	5→0

S_1:	PC → Mem_Add	T1_WR
	Mem Data → T1	
	PC → ALU_A	
	+2 → ALU_B	ADD
	ALU_C → PC	WR_EN

S_2	$T_1(11\to9) \rightarrow RF\_A_1$	T2_WR
	$T_1(8\to6) \rightarrow RF\_A_2$	T3_WR
	$RF\_D_1 \rightarrow T_2$	
	$RF\_D_2 \rightarrow T_3$	
	$T_1(5\text{-}0) \xrightarrow{SE16} ALU\_A$	SUB
	2 → ALU_B	
	ALU_C → T4	T4_WR

⇒ Storing value from RA and RB in T2 and T4 and computing Imm-2

S_3:	$T_2 \rightarrow ALU\_A$	SUB
	$T_3 \rightarrow ALU\_B$	
	ALU_Z → Z	Z_WR

⇒ Checking Cond^n

S_4:	T4 → ALU_A	ADD
	If (Z=1) then	
	$T_1(5\to0) \rightarrow ALU.B$	
	Else	
	0 → ALU.B	
	ALU_C → PC	

JAL : 10.00 (15-12) RA (11-9) Imm (8 → 0)

S1:

PC → Mem_Add	T1_WR
Mem Data → T1	
PC → T_2	
PC → ALU-A	ADD
+2 → ALU-B	WR.EN
ALU-C → PC	

S2:

T2 → RF.D_3	WR_EN
T1(11-9) → RF.A_3	
T1(8-0) $\xrightarrow{SE16}$ ALU_A	SUB
2 → ALU_B	
ALU_C → T2	T2 WR

⇒ ① Storing PC in RA
② Computing Imm-2 and Storing in T2

S3:

T_2 → ALU_A	
PC → ALU_B	WR.EN
ALU.C → PC	

JALR

10 01 (15-12) RA (11-9) RB (8-6) Imm

S1: Same as JAL

S2:

T2 → RF.D_3	WR.EN
T_1(11-9) → RF.A_3	
T_1(8-6) → RF.A_1	
RF.D_1 → T2	T2.WR

⇒ ① Storing PC in RA
② Storing Address in RB to T2

S3:

T_2 → PC	T2.WR