IITB-CPU

EE224 : Digital Systems

By

FORAM TRIVEDI
(23B1269)

DEV ARORA
(23B1271)

AMAN RUSSEL
(23B1270)

Under the guidance of
Prof. Virendra Singh.
Electrical Engineering IITB

WORK DISTRIBUTION

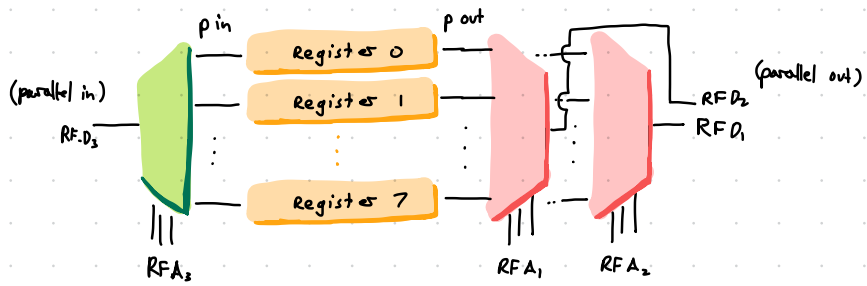
1.) Dev
- (i.) All Jump Statements
- (ii.) Final FSM
- (iii.) Code of ALU, Memory unit
- (iv.) Testing and circuit verification

2.) Aman
- (i.) All load/store Statements
- (ii.) Final Circuit Design
- (iii.) Code of Entire FSM and State outputs
- (iv.) Testing and circuit verfication

3.) Foram
- (i.) All Aritmetic logical Unit statements
- (ii.) Final State Diagram
- (iii.) Code of All other components; i.e. MUX_3, MUX_16, SE_6, SE_9, Register_file, Register 16Bit, CPU Testbench
- (iv.) Testing and circuit verification

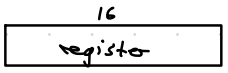
IITB-CPU

Register File (16 bit Reg, 8 Registers);

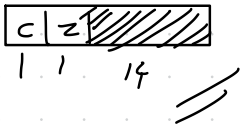


In VHDL, registers can be implemented using easily.

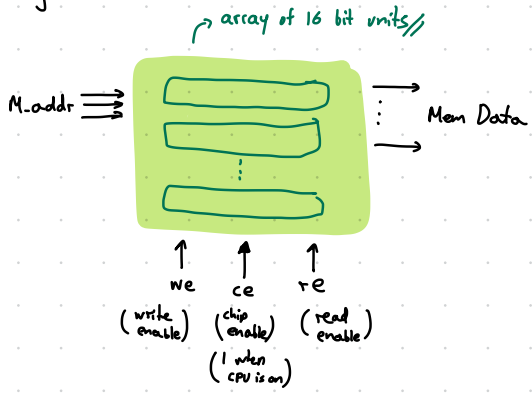
General purpose register:



Condition code Register



Memory unit:



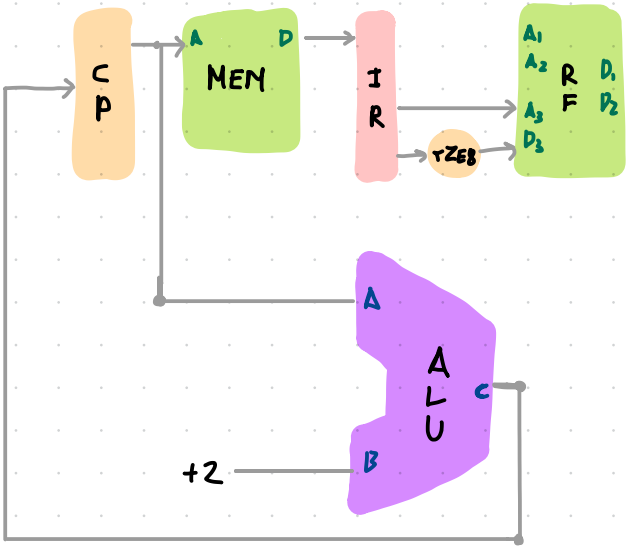
(im proposing 2 units)

(1 for program memory
1 for general memory)

LOAD higher Immediate (LHI): (1000)

M_0 - Fetch

PC → Mem_Address	PC_en
Mem_Data → IR	Mem_read
CP → ALU A	ALU_ADD
+2 → ALU B	IR_en
ALUC → PC	



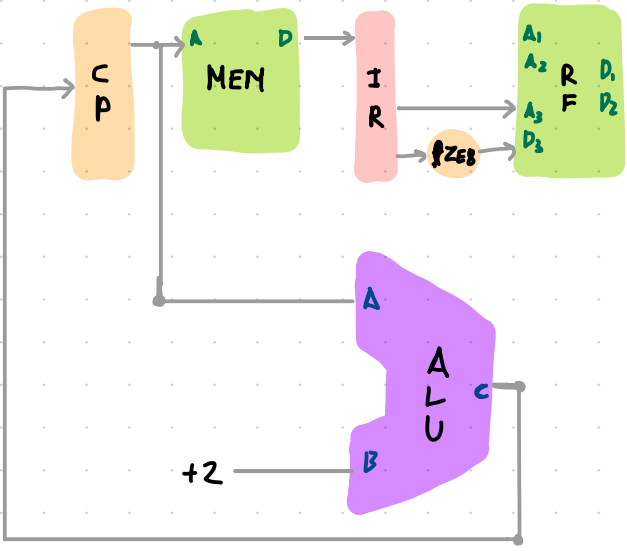
M_1 - Store on MSBs

IR_0_7 → rZE8	RF_Write
rZE8 → RF_D3	
IR_9-11 → RF_A3	

LOAD LOWER IMM : (LLI) (1001)

M_0 – fetch Instr.

PC → Mem_Address	PC_en
Mem_Data → IR	Mem_read
CP → ALU A	ALU_ADD
+2 → ALU B	IR_en
ALUC → PC	



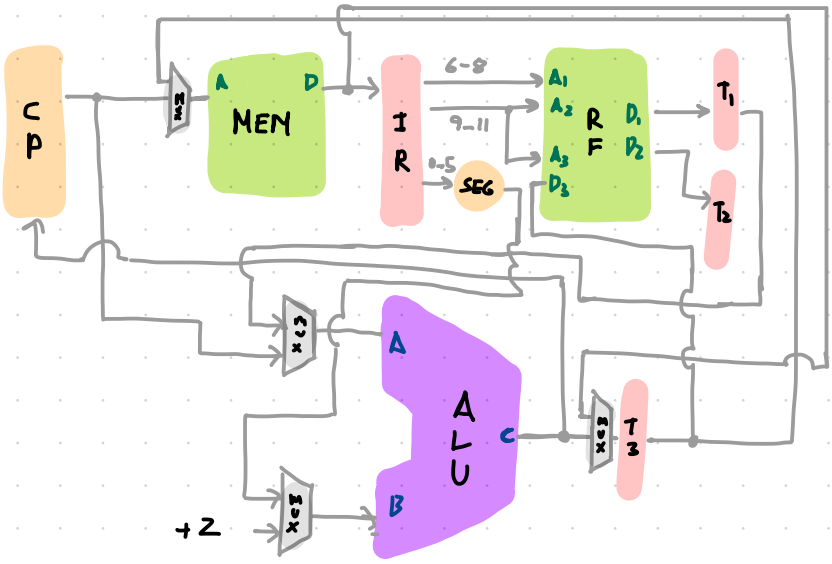
M_2 – Store on LSB's

IR_0-7 → SZE8	RF_write
SZE8 → RF_D3	
IR_9-11 → RF_A3	

LOAD (LW) (1010) :

M_0 – fetch

PC → Mem_Address	PC_en
Mem_Data → IR	Mem_read
CP → ALU A	ALU_ADD
+2 → ALU B	IR_en
ALUC → PC	



M_3 : Understand Opr (I type)

IR_6-8 → RF_A1	T1_write
RF_D1 → T1	T2_write
IR_9-11 → RF_A2	
RF_D2 → T2	

This part is useless but we want to minimize no of states & this is reusable.

M_4 : Compute Address (Add T1 & Imm)

T1 → ALU_A	ALU_ADD
IR_0-5 → SE6	T3_write
SE6 → ALU_B	
ALU_C → T3	

M_5 : Memory Read

T3 → Mem_Addr	Mem_Read
Mem_Data → T3	T3_write

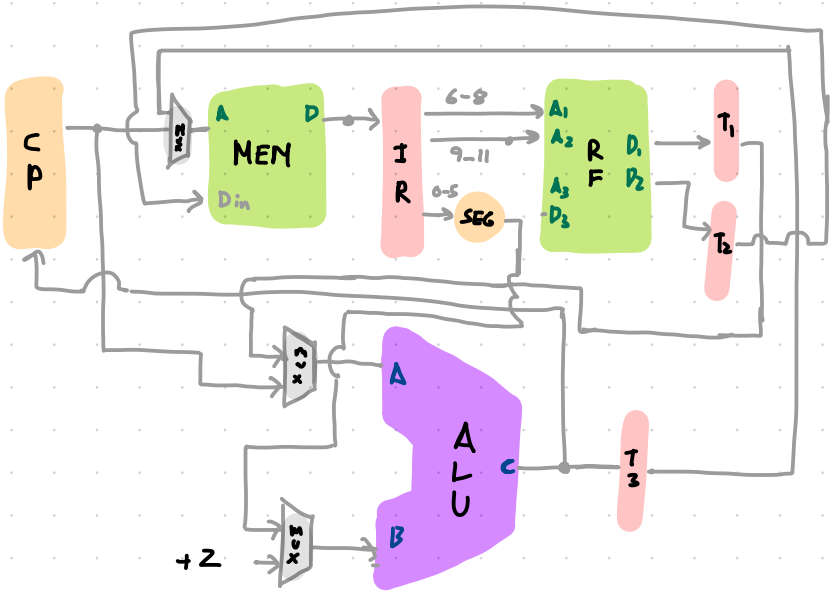
M_6 - Load to Register ($T_3 \rightarrow$ Reg)

$T_3 \rightarrow RF\_D_3$	RF_write
$IR\_9\text{-}11 \rightarrow RF\_A_3$	

STORE (SW) (1011):

M_0 - fetch

$PC \rightarrow Mem\_Address$	PC_en
$Mem\_Data \rightarrow IR$	Mem-read
$CP \rightarrow ALU\ A$	ALU_ADD
$+2 \rightarrow ALU\ B$	
$ALU\ C \rightarrow PC$	



M_3: Understand Opr (I type)

$IR\_6\text{-}8 \rightarrow RF\_A1$	T1_write
$RF\_D1 \rightarrow T1$	T2_write
$IR\_9\text{-}11 \rightarrow RF\_A2$	
$RF\_D2 \rightarrow T2$	

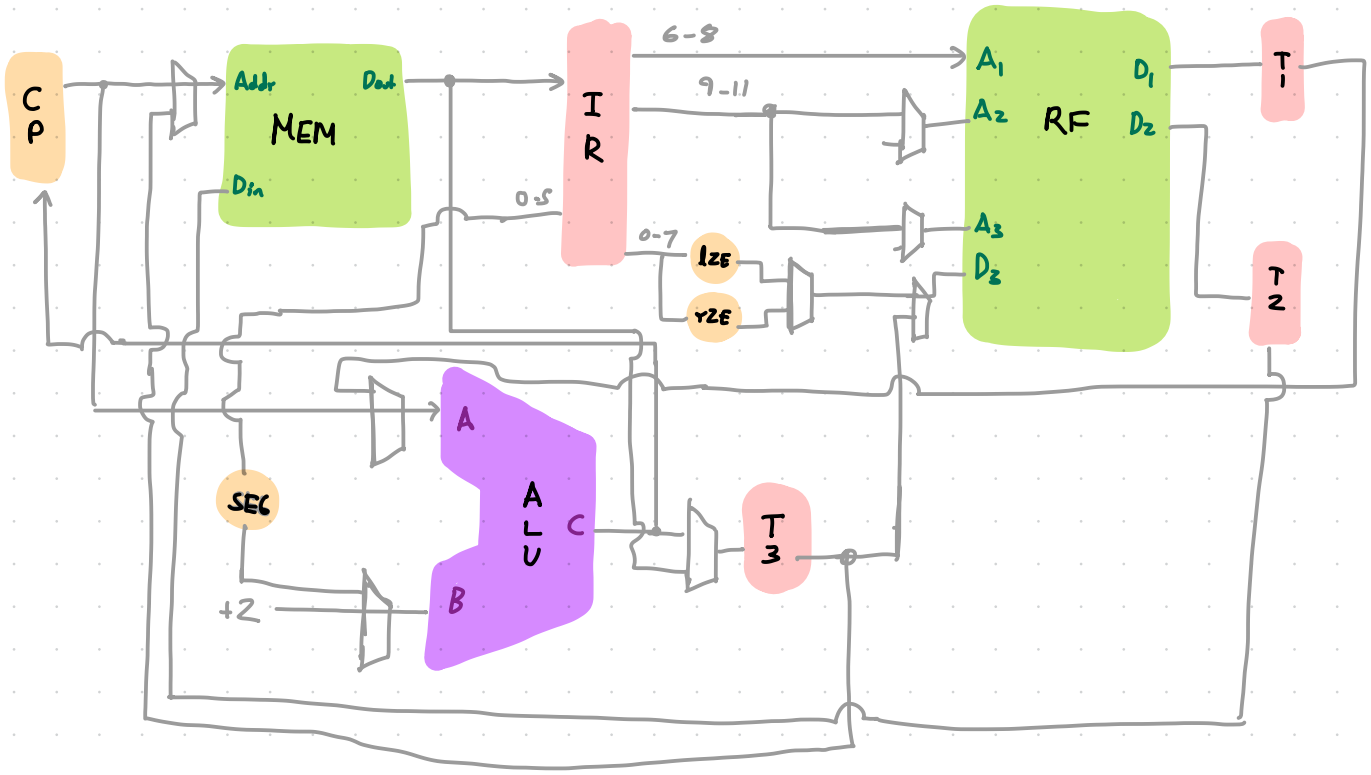
M_4: Compute Address (Add T1 & Imm)

$T1 \rightarrow ALU\_A$	ALU_ADD
$IR\_0\text{-}5 \rightarrow SE6$	T3_write
$SE6 \rightarrow ALU\_B$	
$ALU\_C \rightarrow T3$	

M_7: Memory Write

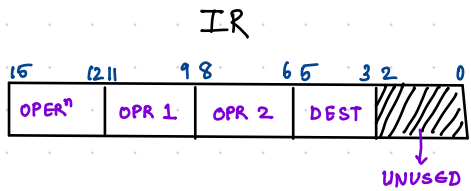
$T_3 \rightarrow Mem\_Addr$	Mem_write
$T_2 \rightarrow Mem\_Data$	

Net diagram with all ~~LOAD~~ Memory statements:

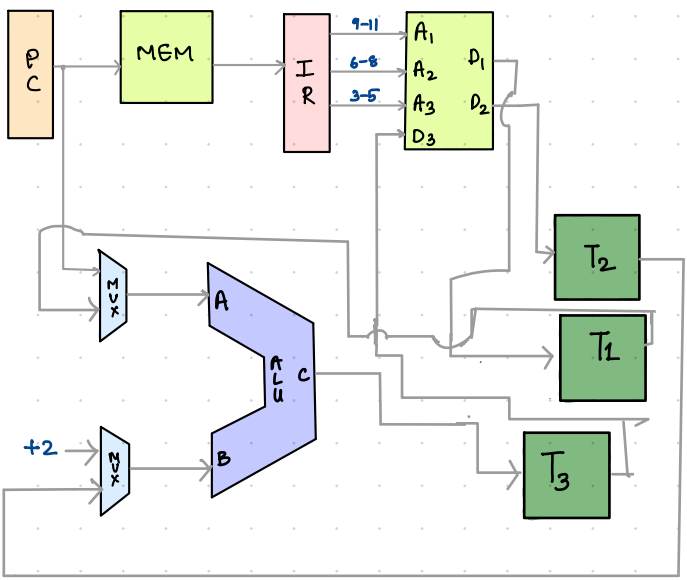


Arithmetic Logical Instructions

1- Bring (Fetch) instructions from memory
2- Understand Instruction
3- Read operands
4- Compute the result
5- Update PC



Addition (ADD): (0000)



S_0 - Fetch

PC → Mem_Address	PC_en
Mem_Data → IR	Mem_read
PC → ALU_A	ALU_ADD
+2 → ALU_B	IR_en
ALU_C → PC	

S_1 - Read Operands

IR_{11-9} → RF_A_1	T_1_en
IR_{8-6} → RF_A_2	T_2_en
RF_D_1 → T_1	
RF_D_2 → T_2	

S_2 - Operation

T_1 → ALU_A	ADD
T_2 → ALU_B	T_1_en
ALU_C → T_3	

S_3 - Update result

T_3 → RF_D_3	RF_en
IR_{5-3} → RF_A_3	

Subtraction (SUB) : (0010)

S_5 - Fetch

PC → Mem_Address	PC_en
Mem_Data → IR	Mem_read
PC → ALU_A	ALU_ADD
+2 → ALU_B	IR_en
ALU_C → PC	

S_6 - Read Operands

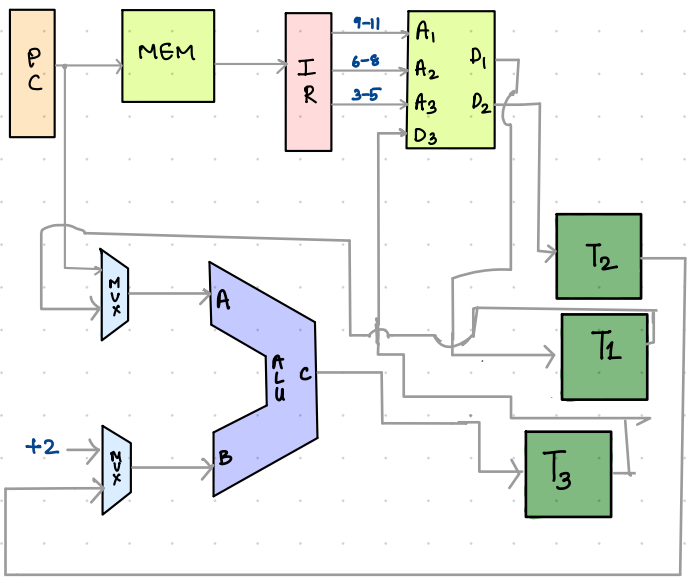
IR_{11-9} → RF_A_1	T_1_en
IR_{8-6} → RF_A_2	T_2_en
RF_D_1 → T_1	
RF_D_2 → T_2	

S_7 - Operation

T_1 → ALU_A	SUB
T_2 → ALU_B	T_3_en
ALU_C → T_3	

S_8 - Update result

T_3 → RF_D_3	RF_en
IR_{5-3} → RF_A_3	



Multiplication (MUL): (0011)

S_9 – Fetch

PC → Mem_Address	PC_en
Mem_Data → IR	Mem_read
PC → ALU_A	ALU_ADD
+2 → ALU_B	IR_en
ALU_C → PC	

S_{10} – Read Operands

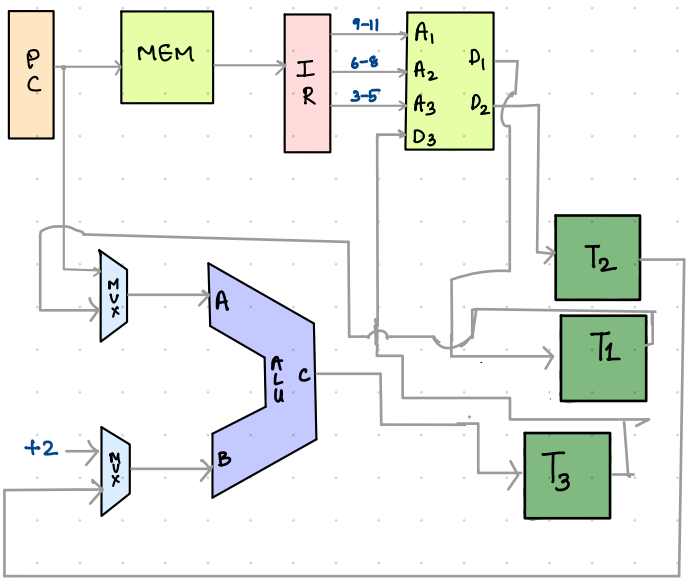
IR_{11-9} → RF_A_1	T_1_en
IR_{8-6} → RF_A_2	T_2_en
RF_D_1 → T_1	
RF_D_2 → T_2	

S_{11} – Operation

T_1 → ALU_A	MUL
T_2 → ALU_B	T_3_en
ALU_C → T_3	

S_{12} – Update result

T_3 → RF_D_3	RF_en
IR_{5-3} → RF_A_3	



Add immediate (ADI): 0001

S_{13} - Fetch

PC → Mem_Address	PC_en
Mem_Data → IR	Mem_read
PC → ALU_A	ALU_ADD
+2 → ALU_B	IR_en
ALU_C → PC	

S_{14} - Read operands

IR_{11-9} → RF_A_1	T_1_en
RF_D_1 → T_1	

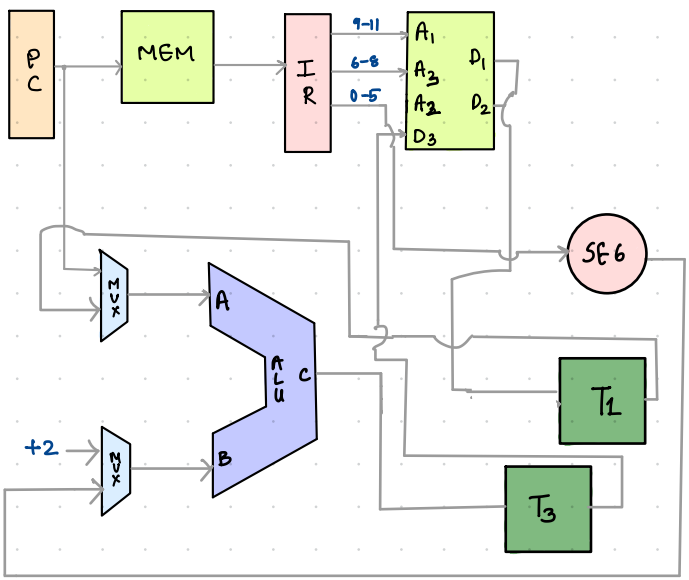
S_{15} - Operation

T_1 → ALU_A	ADD
IR_{5-0} → SE6 → ALU_B	T_3_en
ALU_C → T_3	

S_{16} - Update result

T_3 → RF_D_3	RF_en
IR_{8-6} → RF_A_3	

IR



Logical And (AND) : (0100)

S_{17} - Fetch

PC → Mem_Address	PC_en
Mem_Data → IR	Mem_read
PC → ALU_A	ALU_ADD
+2 → ALU_B	IR_en
ALU_C → PC	

S_{18} - Read Operands

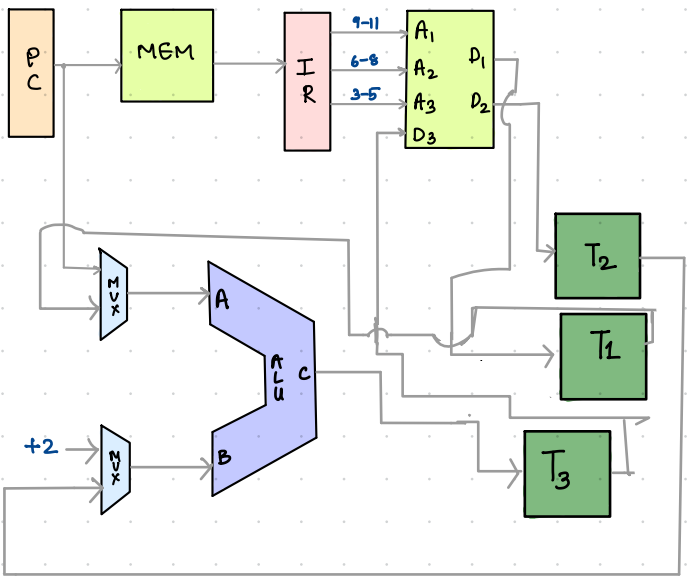
IR_{11-9} → RF_A_1	T_1_en
IR_{8-6} → RF_A_2	T_2_en
RF_D_1 → T_1	
RF_D_2 → T_2	

S_{19} - Operation

T_1 → ALU_A	AND
T_2 → ALU_B	T_3_en
ALU_C → T_3	

S_{20} - Update result

T_3 → RF_D_3	RF_en
IR_{5-3} → RF_A_3	



Logical OR (ORA): (0101)

S_{21} - Fetch

PC → Mem_Address	PC_en
Mem_Data → IR	Mem_read
PC → ALU_A	ALU_ADD
+2 → ALU_B	IR_en
ALU_C → PC	

S_{22} - Read Operands

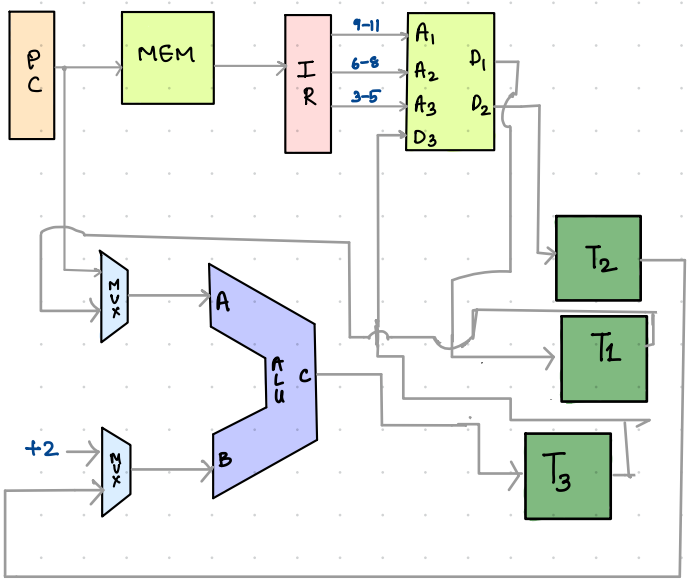
IR_{11-9} → RF_A_1	T_1_en
IR_{8-6} → RF_A_2	T2_en
RF_D_1 → T_1	
RF_D_2 → T_2	

S_{23} - Operation

T_1 → ALU_A	OR
T_2 → ALU_B	T_3_en
ALU_C → T_3	

S_{24} - Update result

T_3 → RF_D_3	RF_en
IR_{5-3} → RF_A_3	



Logical Implication (IMP): (0110)

S_{25} - Fetch

PC → Mem_Address	PC_en
Mem_Data → IR	Mem_read
PC → ALU_A	ALU_ADD
+2 → ALU_B	IR_en
ALU_C → PC	

S_{26} - Read Operands

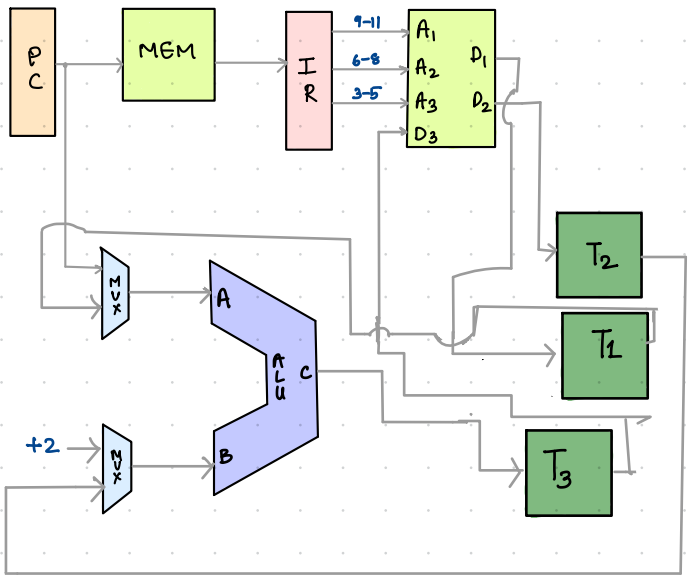
IR_{11-9} → RF_A_1	T_1_en
IR_{8-6} → RF_A_2	T_2_en
RF_D_1 → T_1	
RF_D_2 → T_2	

S_{27} - Operation

T_1 → ALU_A	IMP
T_2 → ALU_B	T_3_en
ALU_C → T_3	

S_{28} - Update result

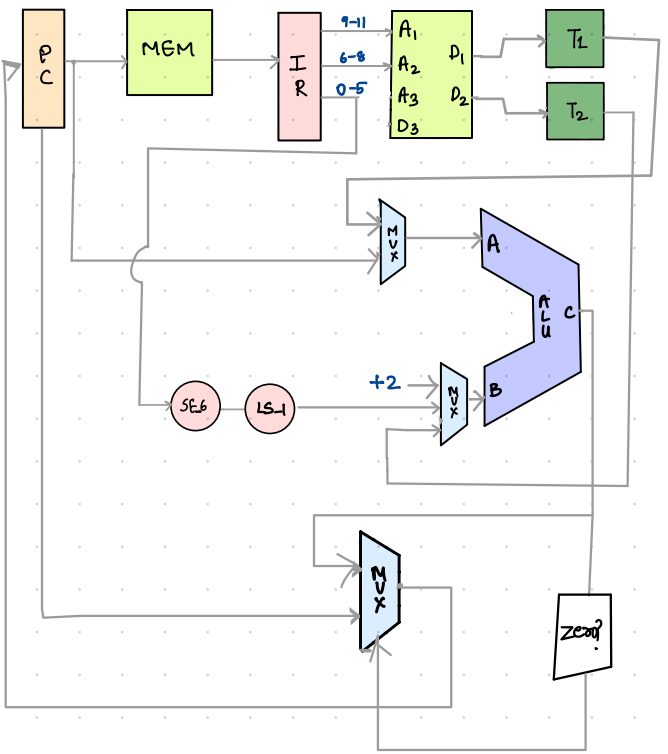
T_3 → RF_D_3	RF_en
IR_{5-3} → RF_A_3	



BEQ (Branch on Equality)

1) Fetch Instruction
2) Update PC
3) Understand Instruction
4) Read Operands
5) Check for equality

IR



D0:-

PC → Mem-A	Mem_Read
PC → ALU_A	ALU-ADD
Mem-D → IR	IR_en
ALU-C → PC	PC_en
+2 → ALU_B	

D1:-

IR_{11-9} → RF_A1	T_1-en
IR_{8-6} → RF_A2	T_2-en
RF_D1 → T_1	
RF-D2 → T_2	

D2:- (Z=1 if ALU_C=0; else=0)

T_1 → ALU-A	ALU_SUB
T_2 → ALU_B	
ALU_C → Z	

D3:- (if Z=1)

PC → ALU_A	ALU_ADD
IR_{5-0} → SE_6	PC-en
SE-6 → LS-1	
LS-1 → ALU-B	
ALU-C → PC	

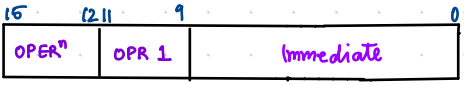
D4:-

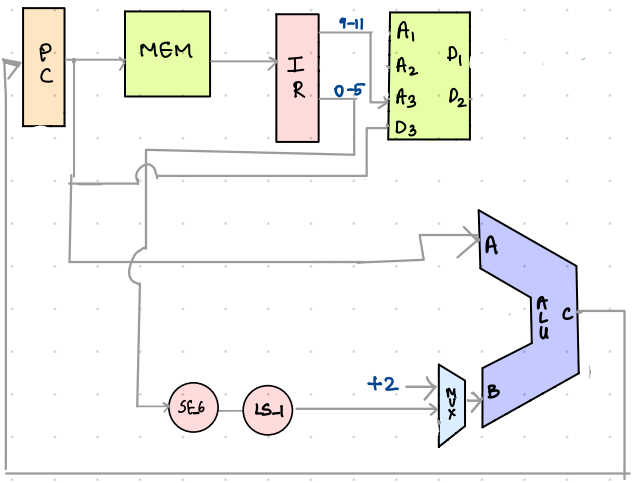
PC → ALU_A	PC-en
+2 → ALU-B	ALU_SUB
ALU-C → PC	

Jump and link

1) Fetch Instruction
2) Update PC
3) Understand Instruction and fetch operand
4) Store PC value
5) Compute PC and update

IR





D0:-

PC → Mem-A	Mem_Read
PC → ALU_A	ALU-ADD
Mem-D → IR	IR_en
ALU-C → PC	PC_en
+2 → ALU-B	

D4:-

no data tranfer	Understand Instruction

D5:-

IR_{9-11} → RF_A_3	RF_en
PC → RF-D_3	

D6:-

PC → ALU_A	ALU-ADD
IR_{5-0} → SE_9	PC_en
SE-6 → LS-1	
LS-1 → ALU-B	
ALU-C → PC	

D9:-

PC → ALU_A	PC_en
+2 → ALU-B	ALU_SUB
ALU-C → PC	

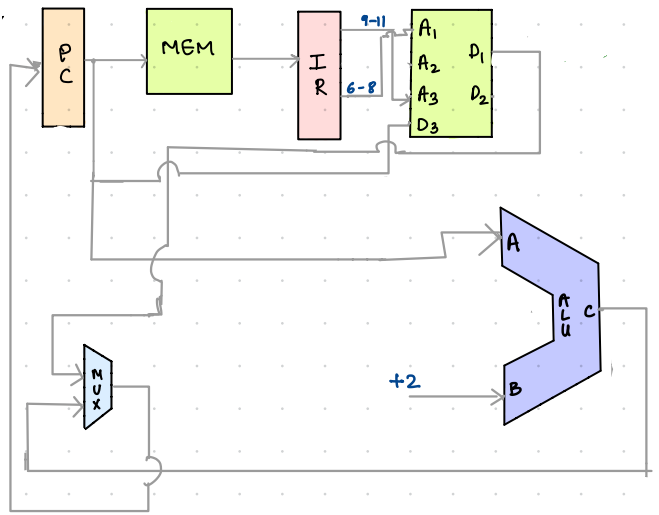
Jump and link to Register

IR

1) Fetch Instruction
2) Update PC
3) Understand Instruction and fetch operand
4) Store PC value
5) Compute PC and update

D0:-

PC → Mem-A	Mem_Read
PC → ALU_A	ALU-ADD
Mem-D → IR	IR_en
ALU-C → PC	PC_en
+2 → ALU-B	



D4:-

no data tranfer	Understand Instruction

D5:-

$IR_{9-11} \rightarrow RF\_A_3$	RF_en
$PC \rightarrow RF\_D_3$	

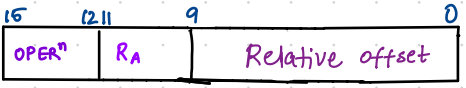
D7:-

$IR_{8-6} \rightarrow RF\_A_1$ PC_en

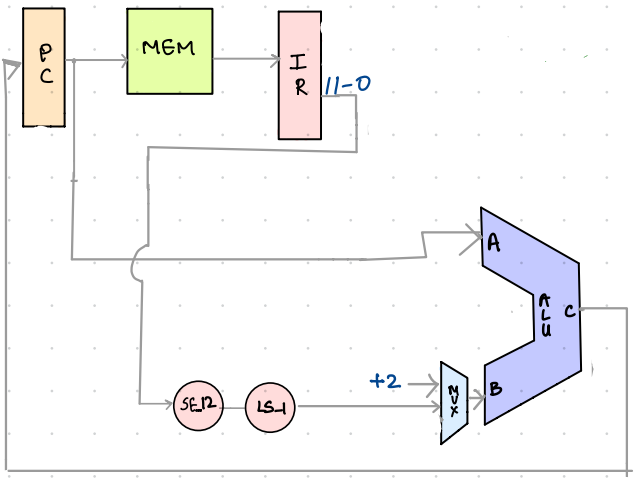
$RF\_D_1 \rightarrow PC$

Jump

IR



1:) Fetch Instruction
2:) Update PC
3:) Understand instruction
4:) Compute PC ; Update PC



D0:-

PC → Mem-A	Mem_Read
PC → ALU_A	ALU-ADD
Mem-D → IR	IR_en
ALU_C → PC	PC_en
+2 → ALU_B	

D4:-

no data tranfer	Understand Instruction

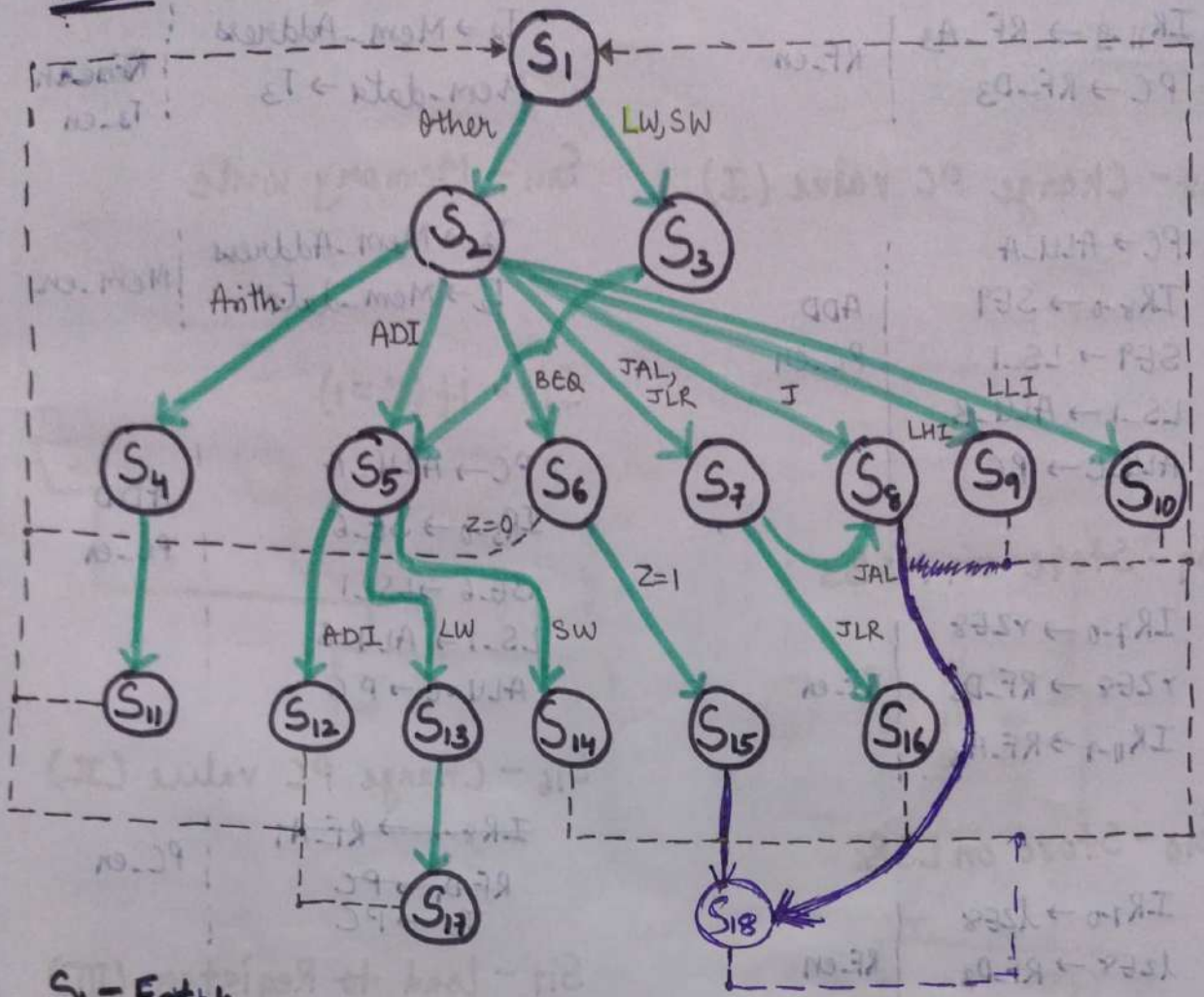
D8:-

PC → ALU_A	ALU_ADD
IR_{11-0} → SE_9	PC_en
SE_12 → LS_1	
LS_1 → ALU_B	
ALU_C → PC	

D9:-

PC → ALU_A	PC_en
+2 → ALU_B	ALU_SUB
ALU_C → PC	

FSM



S_1 - Fetch

PC → Mem_Adress
Mem_data → IR
PC → ALU_A
+2 → ALU_B
ALU_C → PC

PC_en
Mem_read
ADD
IR_en

S_2 - Loading T_1 & T_2 (I)

IR_{11-9} → RF_A_1
RF_D_1 → T_1
IR_{8-6} → RF_A_2
RF_D_2 → T_2

T_1_en
T_2_en

S_3 - Loading T_1 & T_2 (II)

IR_{8-6} → RF_A_1
RF_D_1 → T_1
IR_{11-9} → RF_A_2
RF_D_2 → T_2

T_1_en
T_2_en

S_4 - Operation

T_1 → ALU_A
T_2 → ALU_B
ALU_C → T_3

Operⁿ

S_5 - Addition with Imm.

T_1 → ALU_A
IR_{0-5} → SE6
SE6 → ALU_B
ALU_C → T_3

ADD
T_3_en

S_6 - (Z=1 if ALU_C=0; else =0)

T_1 → ALU_A
T_2 → ALU_B
ALU_C → Z

SUB

S_7 - Store PC value

$IR_{11-9} \rightarrow RF\_A_3$
$PC \rightarrow RF\_D_3$
| RF_en

S_8 - Change PC value (I)

$PC \rightarrow ALU\_A$
$IR_{8-0} \rightarrow SE9$
$SE9 \rightarrow LS\_1$
$LS\_1 \rightarrow ALU\_B$
$ALU\_C \rightarrow PC$
| ADD, PC_en

S_9 - Store on MSBs

$IR_{7-0} \rightarrow rZE8$
$rZE8 \rightarrow RF\_D_3$
$IR_{11-9} \rightarrow RF\_A_3$
| RF_en

S_{10} - Store on LSBs

$IR_{7-0} \rightarrow lZE8$
$lZE8 \rightarrow RF\_D_3$
$IR_{11-9} \rightarrow RF\_A_3$
| RF_en

S_{11} - Load to register (I)

$T_3 \rightarrow RF\_D_3$
$IR_{5-3} \rightarrow RF\_A_3$
| RF_en

S_{12} - Load to register (II)

$T_3 \rightarrow RF\_D_3$
$IR_{6-8} \rightarrow RF\_A_3$
| RF_en

S_{13} - Memory Read

$T_3 \rightarrow Mem\_Address$
$Mem\_data \rightarrow T_3$
| Mem_en, T_3_en

S_{14} - Memory write

$T_3 \rightarrow Mem\_Address$
$T_2 \rightarrow Mem\_data$
| Mem_en

S_{15} - if (Z=1)

$PC \rightarrow ALU\_A$
$IR_{5-0} \rightarrow SE\_6$
$SE\_6 \rightarrow LS\_1$
$LS\_1 \rightarrow ALU\_B$
$ALU\_C \rightarrow PC$
| ADD, PC_en

S_{16} - Change PC value (II)

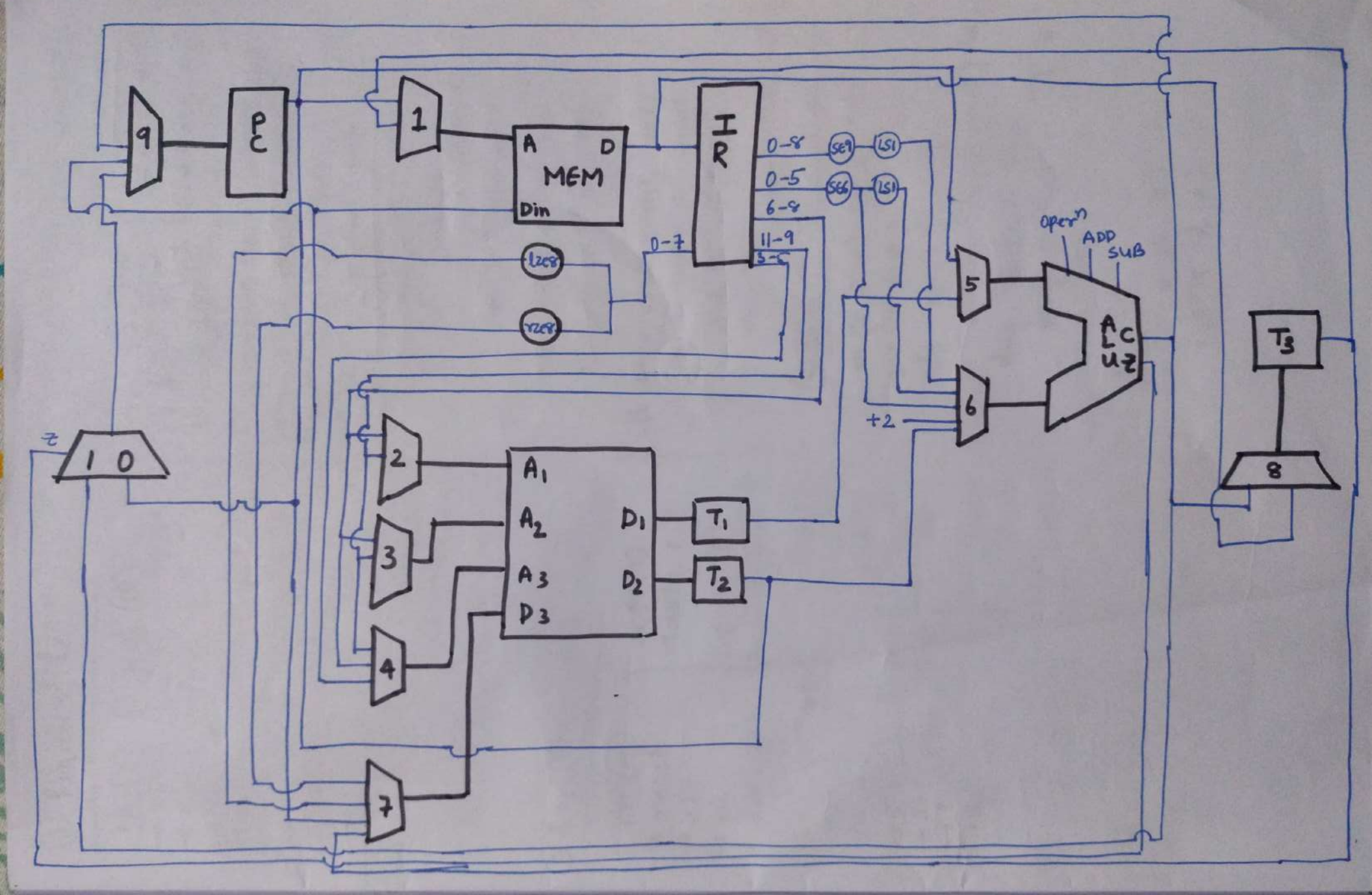
~~$IR_{8-6} \rightarrow RF\_A_3$~~
~~$RF\_D_3 \rightarrow PC$~~
$T_2 \rightarrow PC$
| PC_en

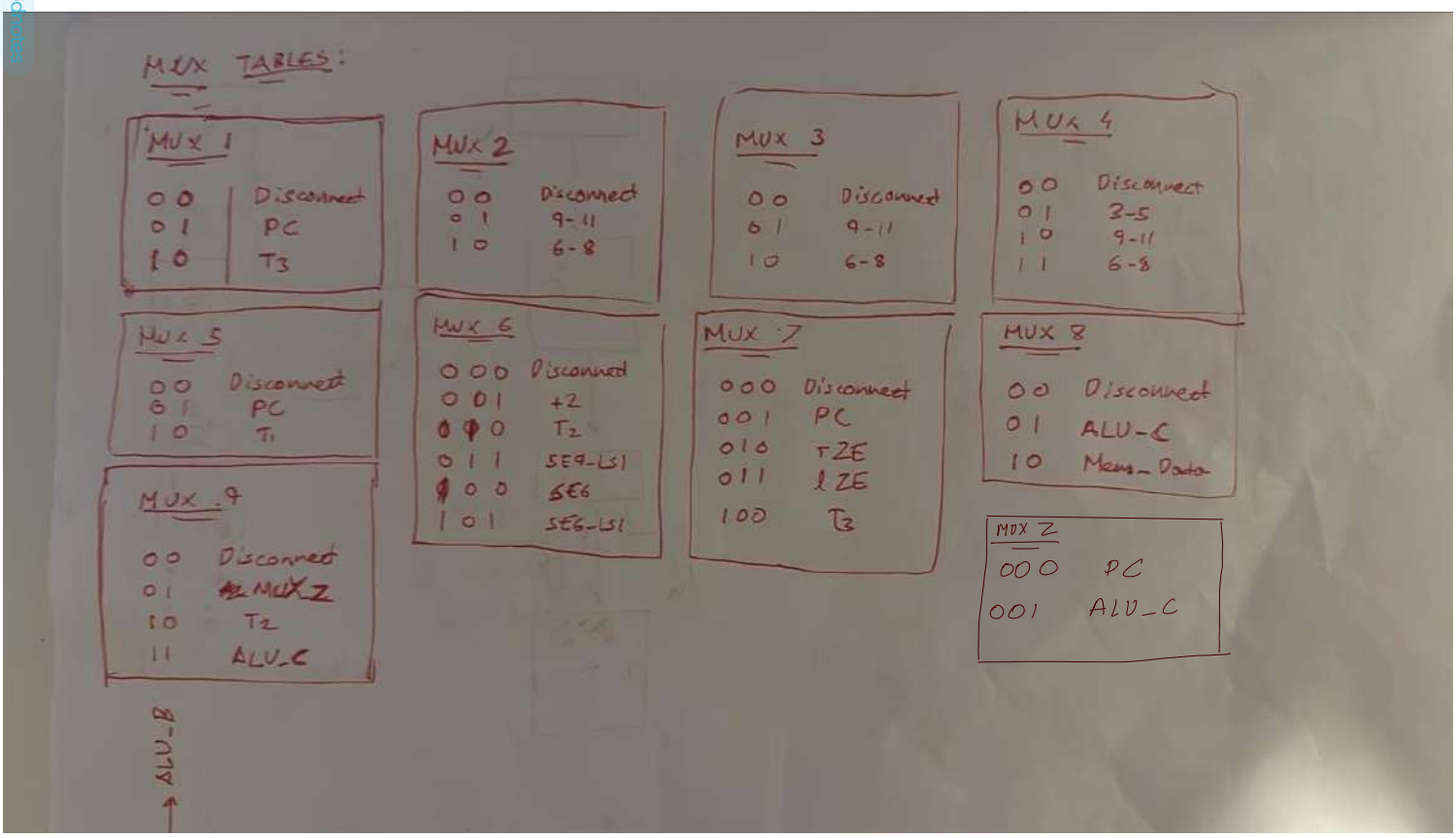
S_{17} - Load to Register (III)

$T_3 \rightarrow RF\_D_3$
$IR_{11-9} \rightarrow RF\_A_3$
| RF_en

S_{18} - Subtraction of PC

$PC \rightarrow ALU\_A$
$+2 \rightarrow ALU\_B$
$ALU\_C \rightarrow PC$
| SUB, PC_en

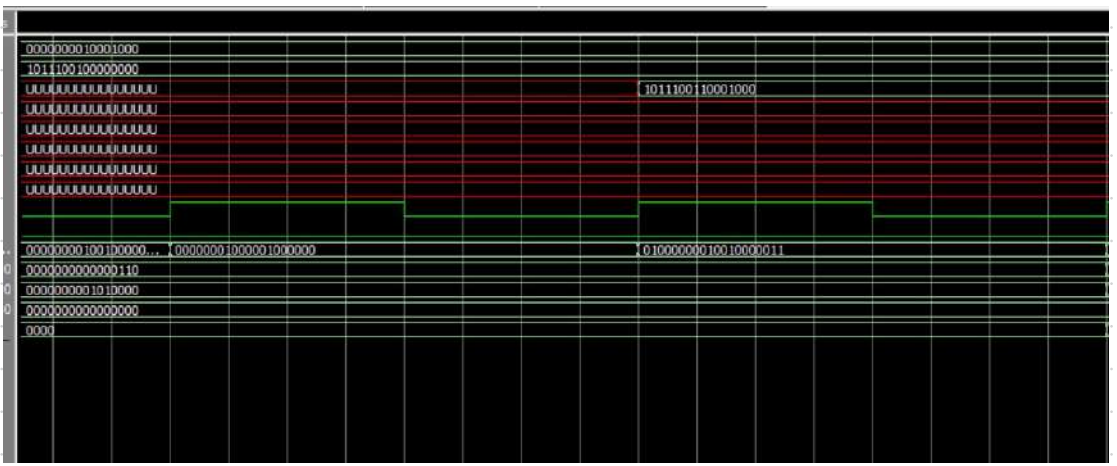

9
PC
1
A
D
MEM
Din
IR
0-8
0-5
6-8
11-9
0-7
SE9
SE6
LS1
LS1
LD8
RD8
5
6
Operⁿ
ADD
SUB
ALC
U Z
+2
T3
8
2
3
4
7
A1
A2
A3
D3
D1
D2
T1
T2
Z
1 0


MUX TABLES:
MUX 1
00 Disconnect
01 PC
10 T3
MUX 2
00 Disconnect
01 9-11
10 6-8
MUX 3
00 Disconnect
01 9-11
10 6-8
MUX 4
00 Disconnect
01 3-5
10 9-11
11 6-8
MUX 5
00 Disconnect
01 PC
10 T1
MUX 6
000 Disconnect
001 +2
010 T2
011 SE9-15
100 SE6
101 SE6-15
MUX 7
000 Disconnect
001 PC
010 TZE
011 IZE
100 T3
MUX 8
00 Disconnect
01 ALU-C
10 Mem_Data
MUX 9
00 Disconnect
01 MUX 2
10 T2
11 ALU_C
MUX 2
000 PC
001 ALU_C
ALU_B

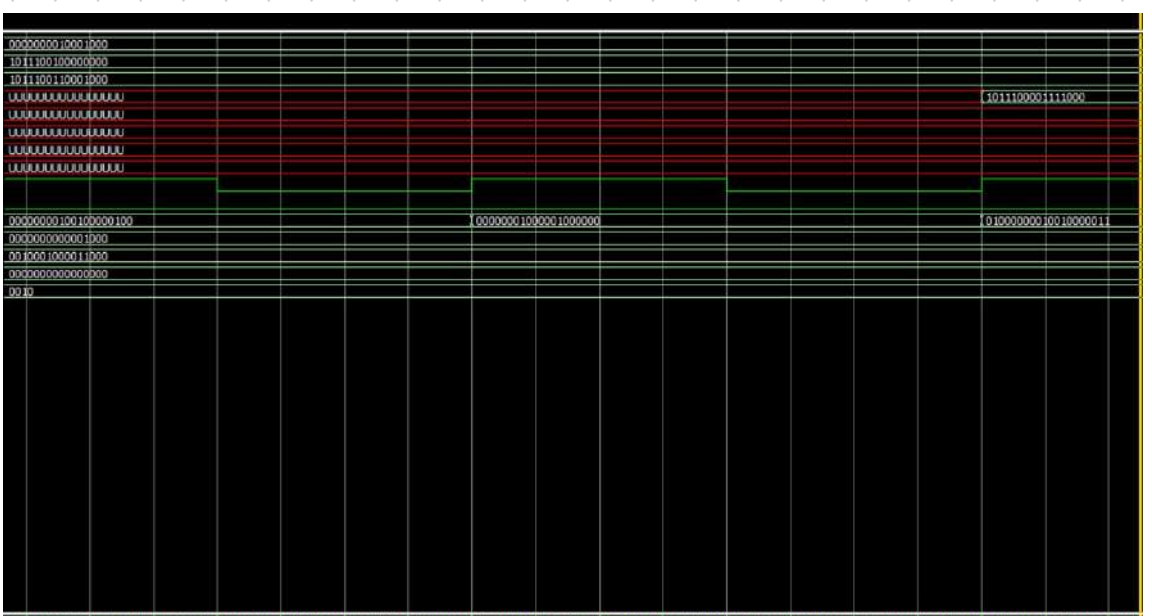
Mux signals / State	19 18 MUX1	17 16 MUX2	15 14 MUX3	13 12 MUX4	11 10 MUX5	9 7 MUX6	6 4 MUX7	3 2 MUX8	1 0 MUX9	ALU Mode
S_1	01	00	00	00	01	001	000	00	11	0000
S_2	00	01	10	00	00	000	000	00	00	ZZZZ
S_3	00	00	00	00	10	010	000	01	00	S_3_add, S_3_sub, S_3_mul, S_3_ora, S_3_and, S_3_xnor 0000 0010 0011 0101 0100 0110
S_4	00	00	00	00	10	010	000	00	00	0010
S_5	00	00	00	10	00	000	001	00	00	ZZZZ
S_6	00	00	00	00	01	011	000	00	11	0000
S_7	00	00	00	10	00	000	010	00	00	ZZZZ
S_8	00	00	00	10	00	000	011	00	00	ZZZZ
S_9	00	00	00	00	10	100	000	01	00	0000
S_{10}	00	00	00	01	00	000	100	00	00	ZZZZ
S_{10}'	00	00	00	10	00	000	100	00	00	ZZZZ
S_2'	00	10	01	00	00	000	000	00	00	ZZZZ
S_{11}	00	00	00	00	01	101	000	00	01	0000
S_{12}	00	00	00	00	00	000	000	00	10	ZZZZ
S_{13}	10	00	00	00	00	000	000	10	00	ZZZZ
S_{14}	10	00	00	00	00	000	000	00	00	ZZZZ
S_{15}	00	00	00	11	00	000	100	00	00	ZZZZ
S_{16}	01	00	00	00	01	001	000	00	11	0010

SCREENSHOTS OF WORKING SIMULATIONS

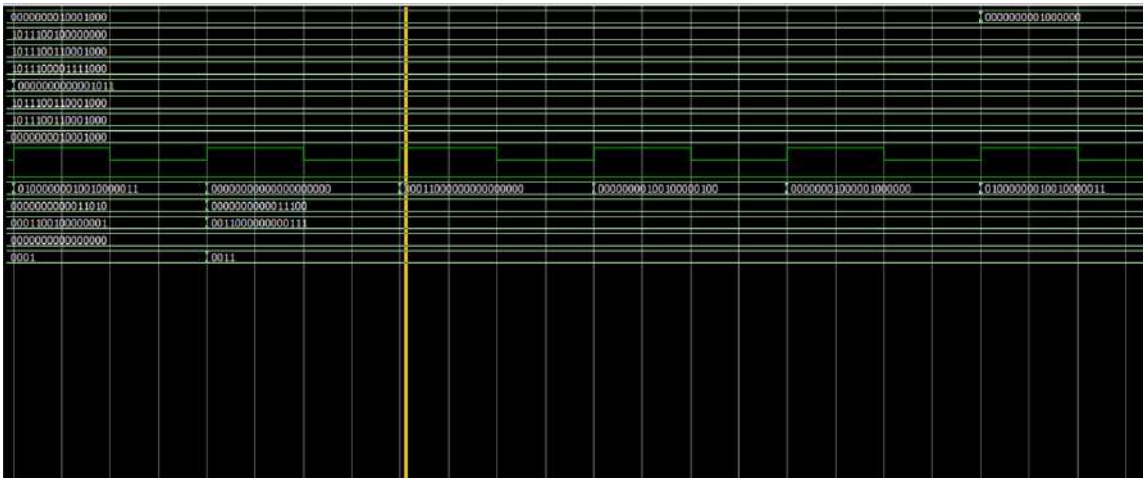
1) ***ADD***



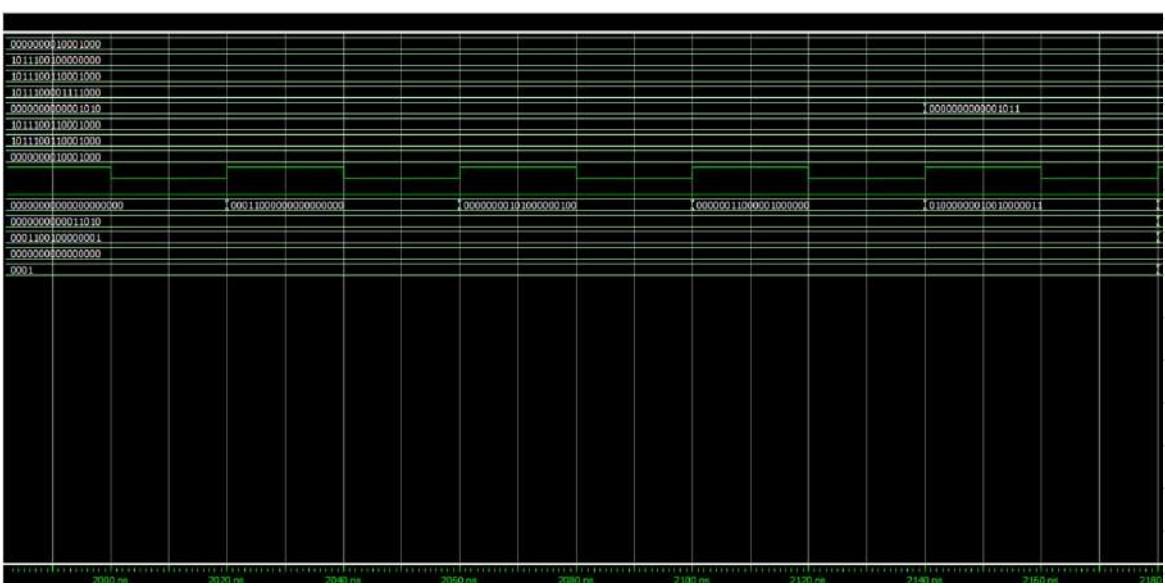
2) SUBTRACT



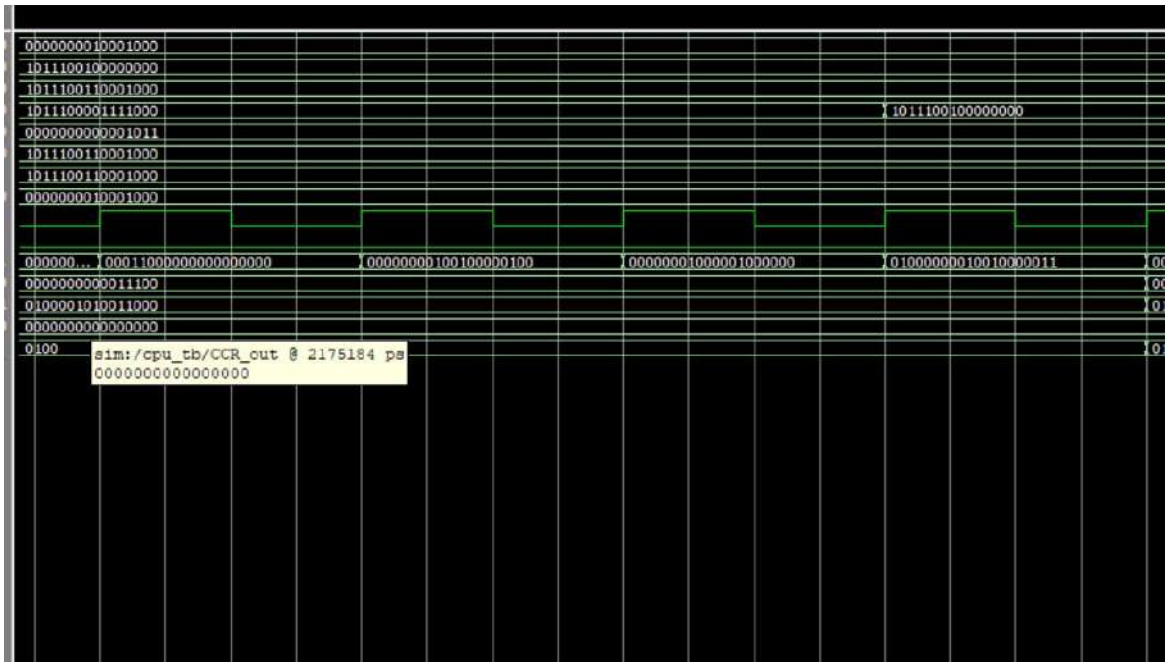
3) ***MUL***



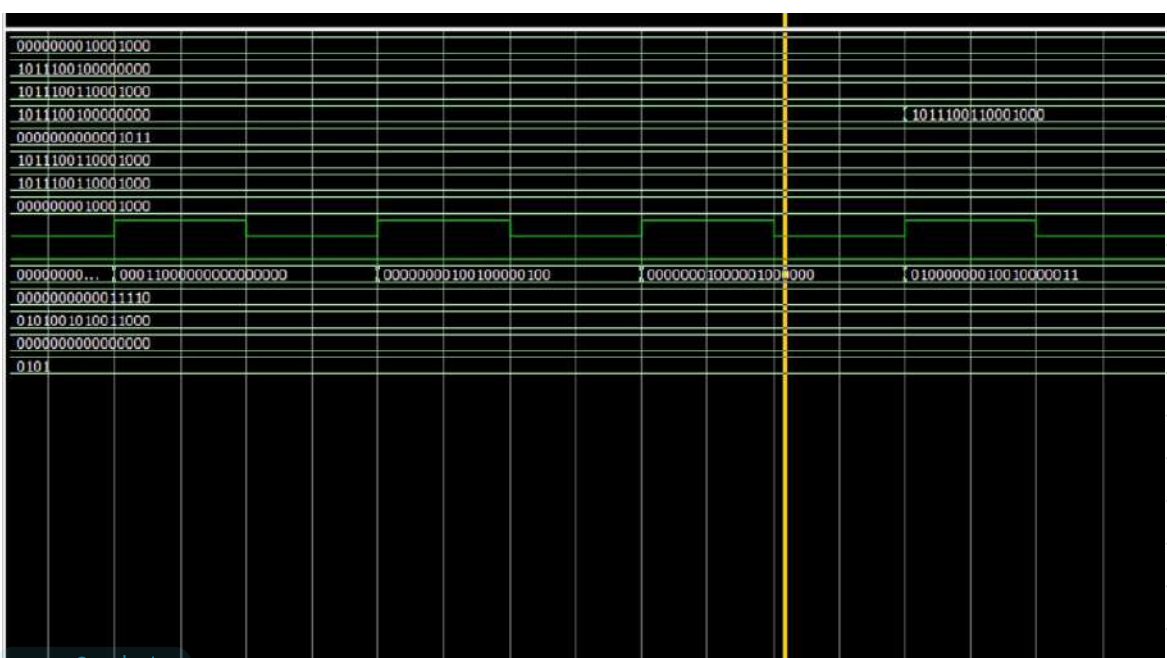
4) ADI



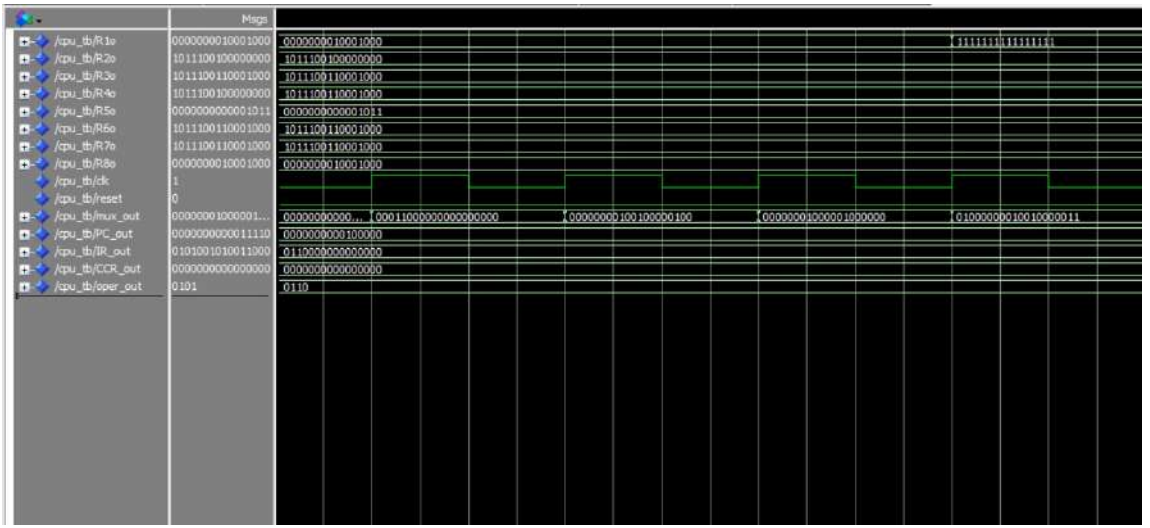
5) AND



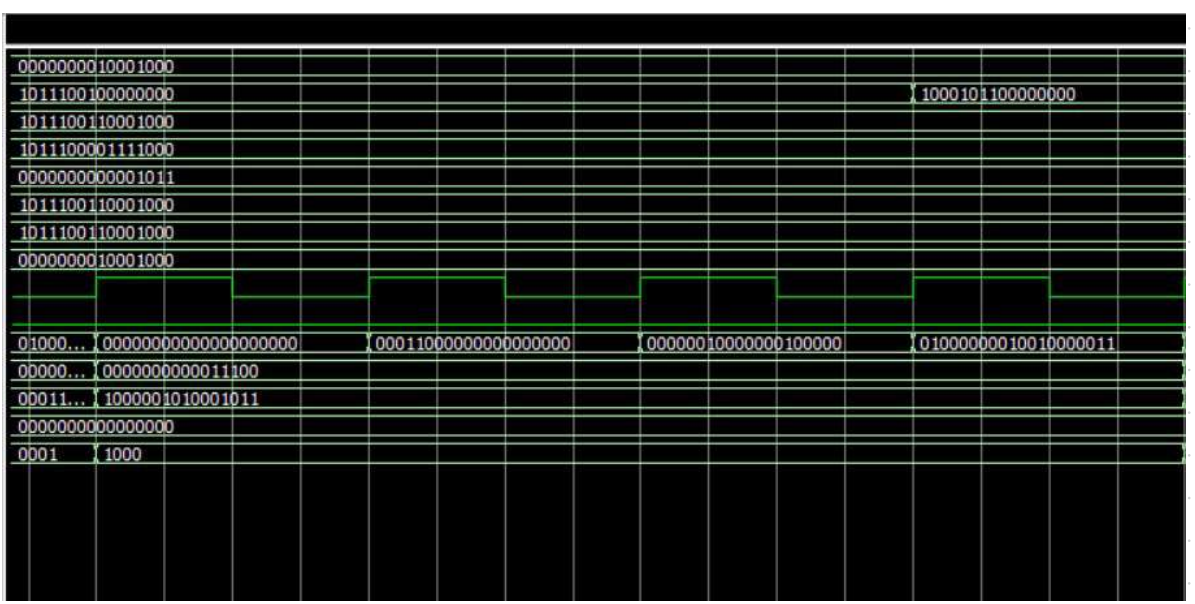
6) ORA



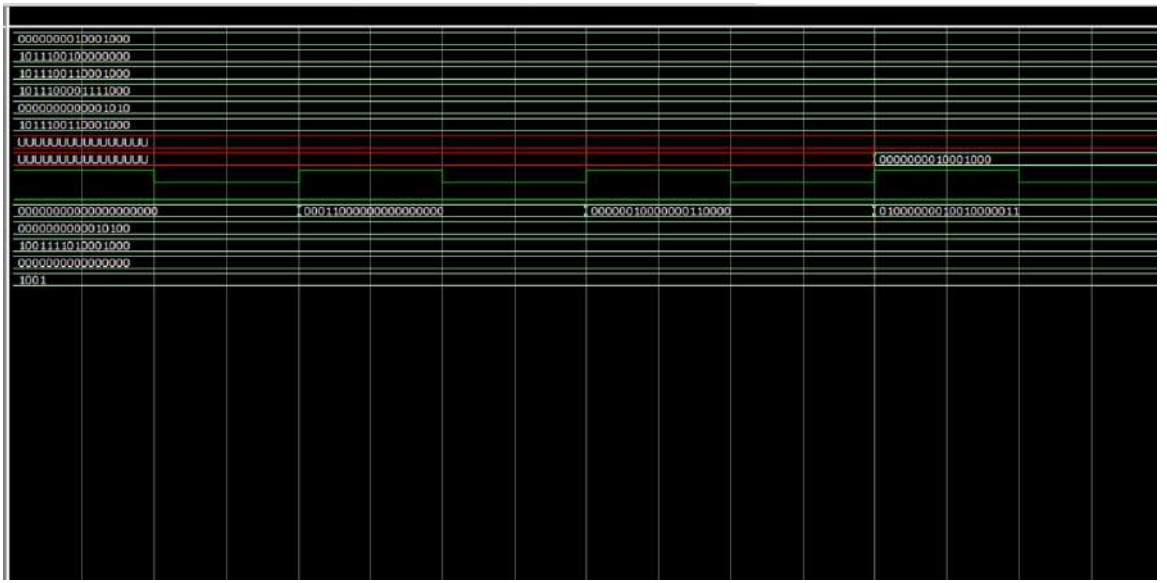
7) IMP



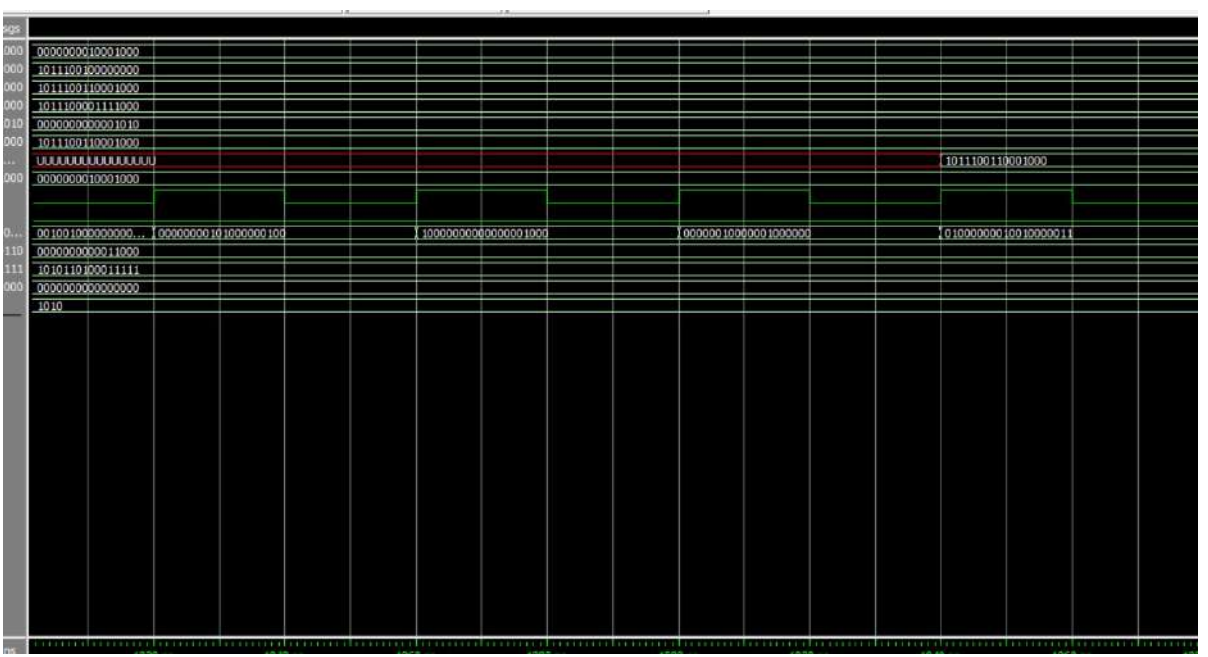
8) LHI



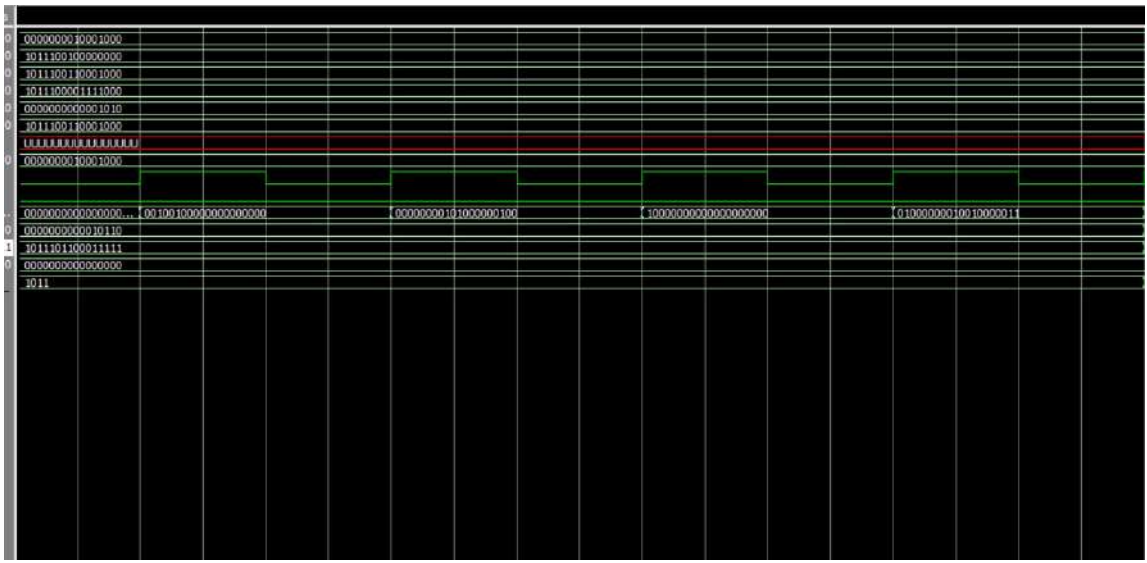
9) LLI



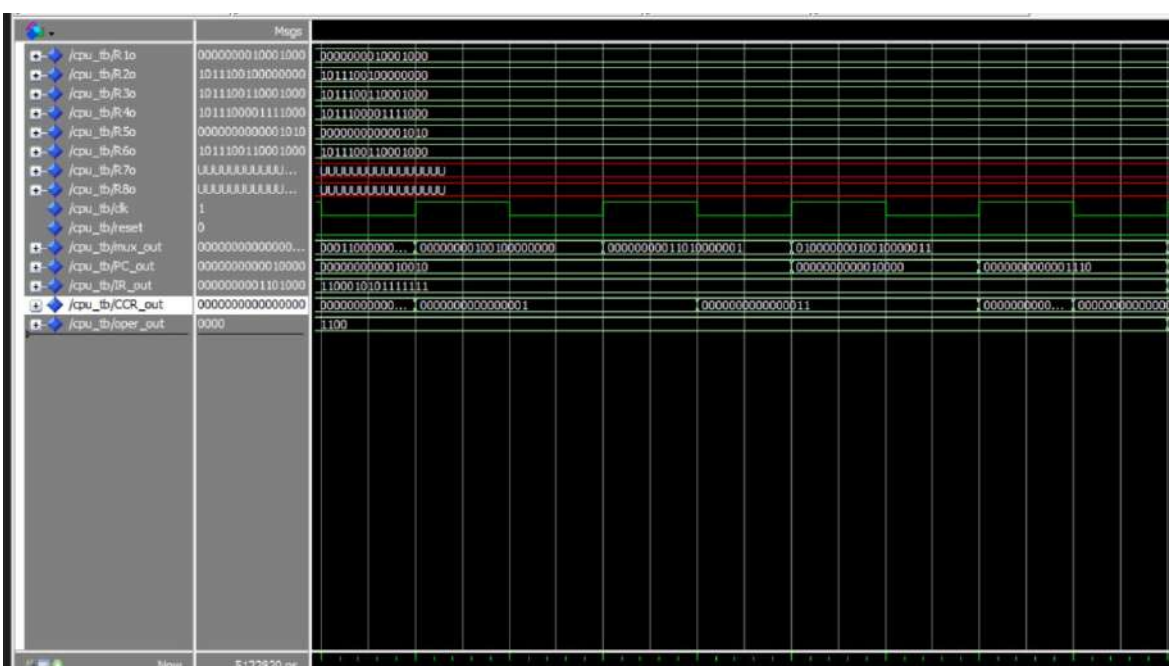
10) LW



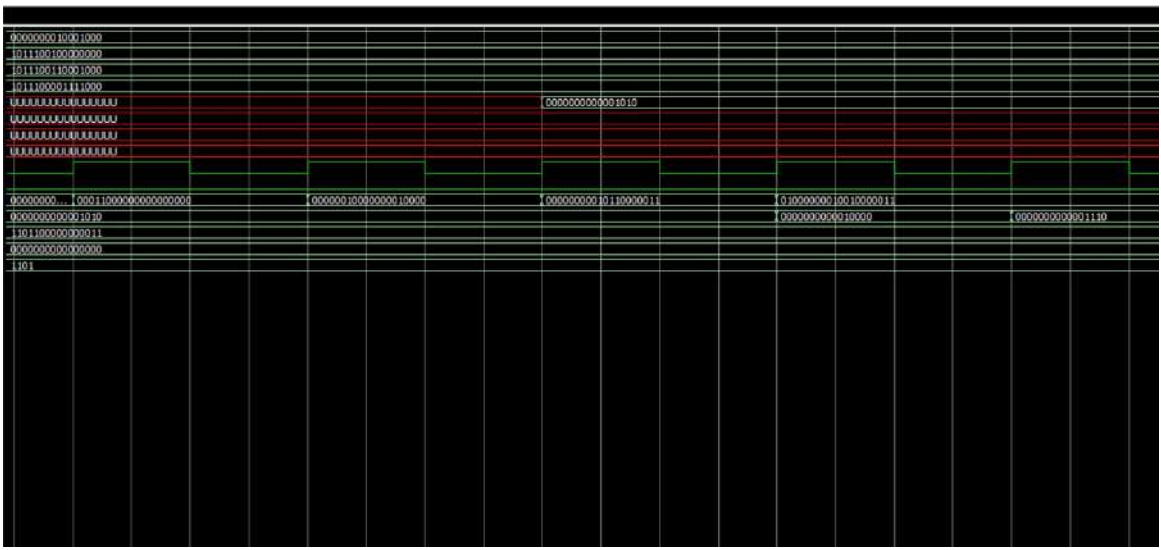
11) SW



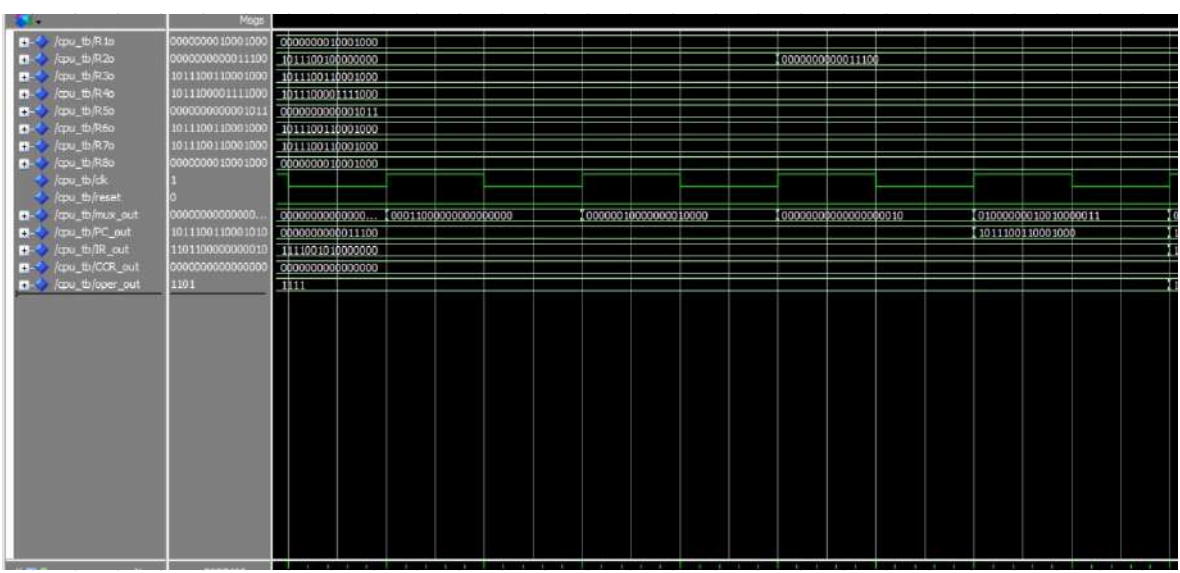
12) BEQ



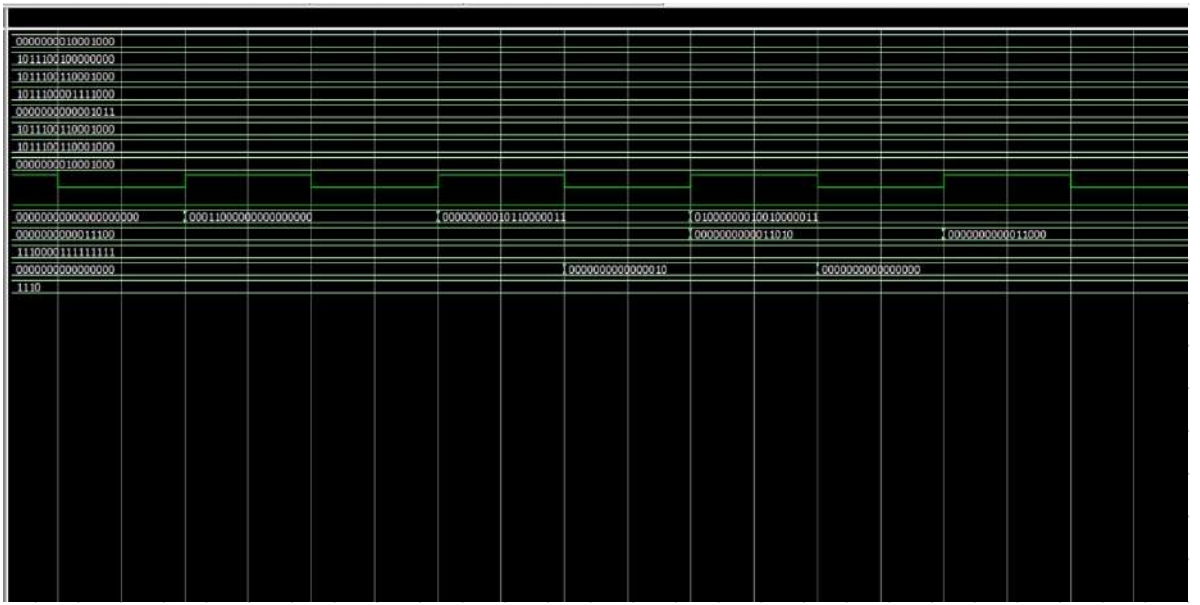
13) JAL



14) JLR



15) Jump



Reference: