IITB-CPU

EE224 : Digital Systems

By

FORAM TRIVEDI
(23B1269)

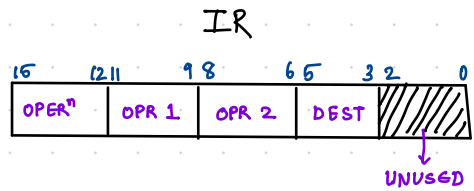
DEV ARORA
(23B1271)

AMAN RUSSEL
(23B1270)

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

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

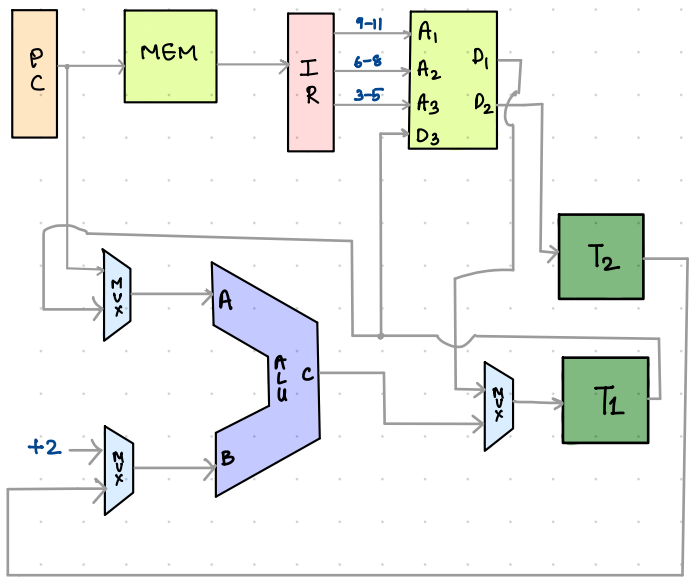
IR_{11-9} → RF_A_1	T_1_en
IR_{8-6} → RF_A_2	T_2_en
RF_D_1 → T_1	
RF_D_2 → T_2	

S_2 - Operation

T_1 → ALU_A	ADD
T_2 → ALU_B	T_1_en
ALU_C → T_1	

S_3 - Update result

T_1 → 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

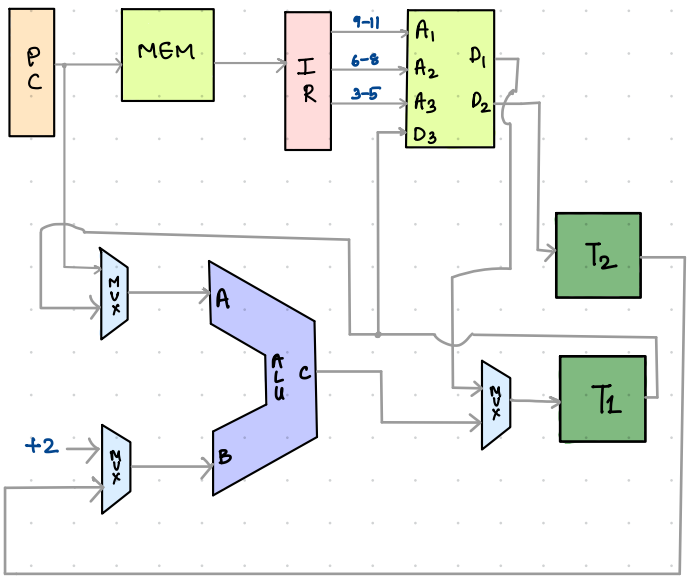
IR_{11-9} → RF_A_1	T_1_en
IR_{8-6} → RF_A_2	T_2_en
RF_D_1 → T_1	
RF_D_2 → T_2	

S_7 - Operation

T_1 → ALU_A	SUB
T_2 → ALU_B	T_1_en
ALU_C → T_1	

S_8 - Update result

T_1 → 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

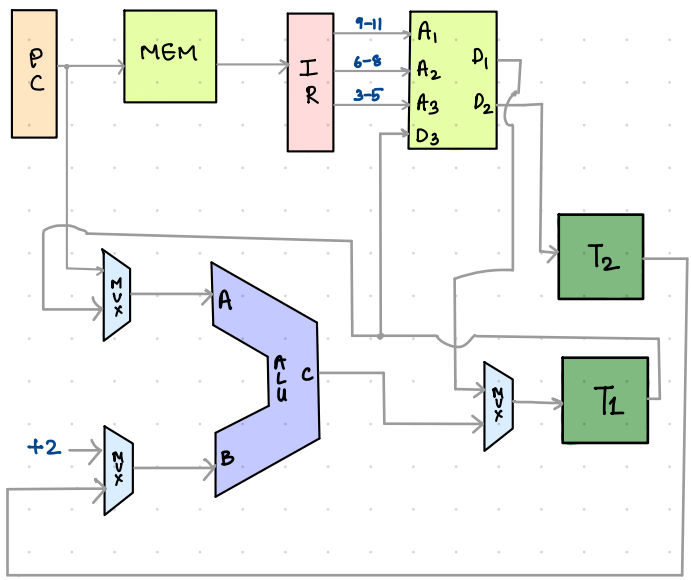
IR_{11-9} → RF_A_1	T_1_en
IR_{8-6} → RF_A_2	T_2_en
RF_D_1 → T_1	
RF_D_2 → T_2	

S_{11} – Operation

T_1 → ALU_A	MUL
T_2 → ALU_B	T_1_en
ALU_C → T_1	

S_{12} – Update result

T_1 → RF_D_3	RF_en
IR_{5-3} → RF_A_3	



Add immediate (ADI): 0001

IR

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

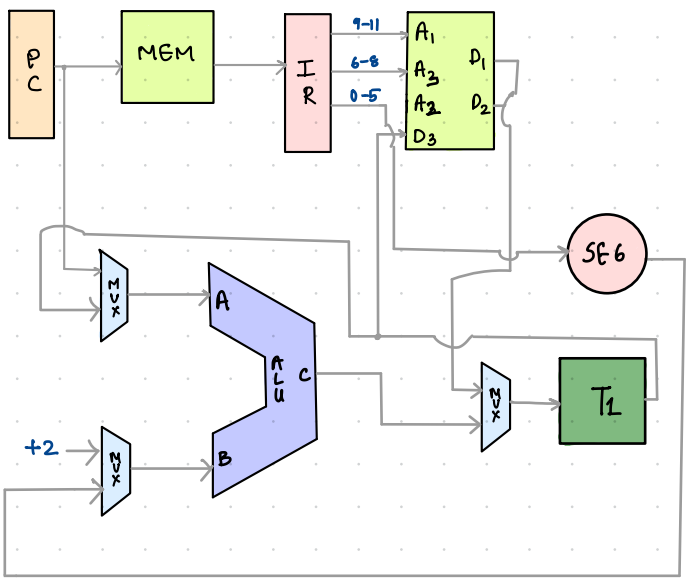
IR_{11-9} → RF_A_1	T_1_en
RF_D_1 → T_1	

S_{15} - Operation

T_1 → ALU_A	ADD
IR_{5-0} → SE6 → ALU_B	T_1_en
ALU_C → T_1	

S_{16} - Update result

T_1 → RF_D_3	RF_en
IR_{6-8} → RF_A_3	



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

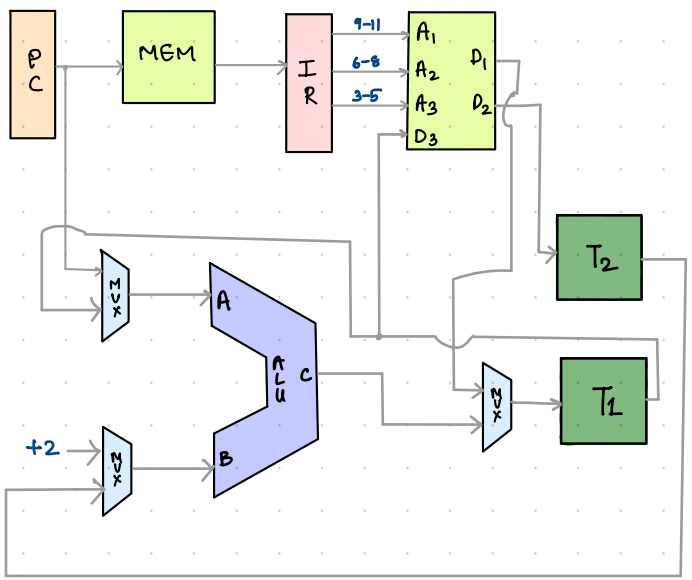
IR_{11-9} → RF_A_1	T_1_en
IR_{8-6} → RF_A_2	T_2_en
RF_D_1 → T_1	
RF_D_2 → T_2	

S_{19} - Operation

T_1 → ALU_A	AND
T_2 → ALU_B	T_1_en
ALU_C → T_1	

S_{20} - Update result

T_1 → 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

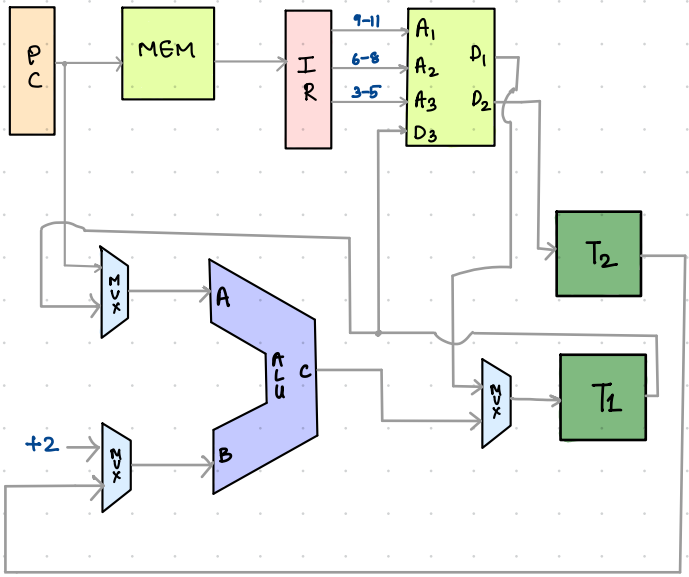
IR_{11-9} → RF_A_1	T_1_en
IR_{8-6} → RF_A_2	T_2_en
RF_D_1 → T_1	
RF_D_2 → T_2	

S_{23} - Operation

T_1 → ALU_A	OR
T_2 → ALU_B	T_1_en
ALU_C → T_1	

S_{24} - Update result

T_1 → RF_D_3	RF_en
IR_{5-3} → RF_A_3	



Logical Implication (IMP): (0110)

S_{25} - Fetch

PC → Mem_Address
Mem_Data → IR
PC → ALU_A
+2 → ALU_B
ALU_C → PC

PC_en
Mem_read
ALU_ADD
IR_en

S_{26} - Read Operands

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

T_1_en
T_2_en

S_{27} - Operation

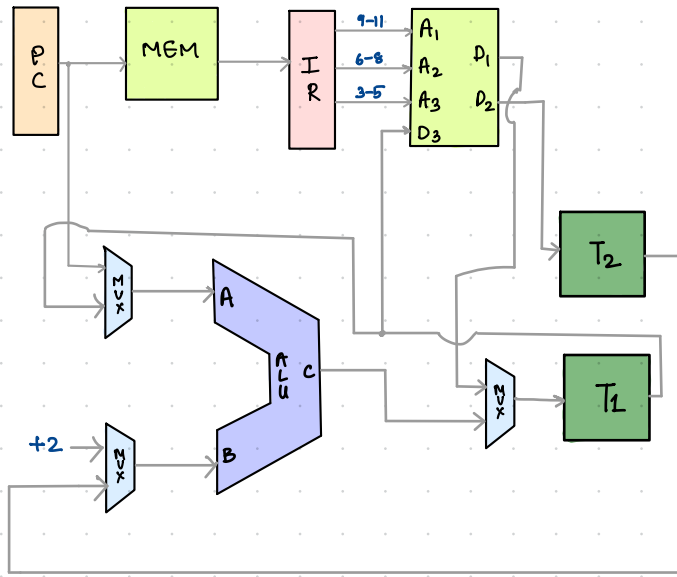
T_1 → ALU_A
T_2 → ALU_B
ALU_C → T_1

IMP
T_1_en

S_{28} - Update result

T_1 → RF_D_3
IR_{5-3} → RF_A_3

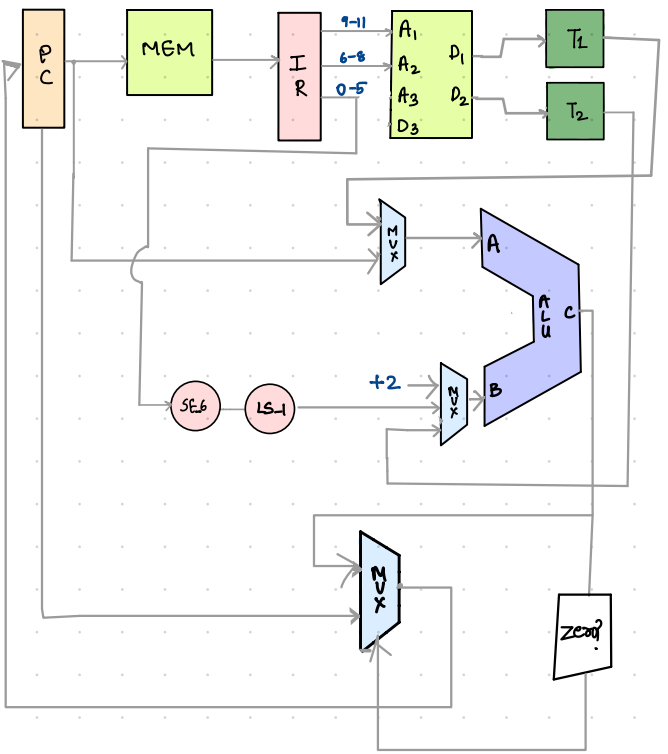
RF_en



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:-

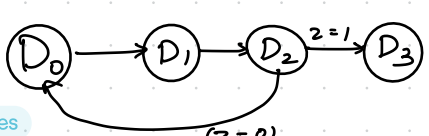
IR_{11-9} → RF_A1	T_1-en
IR_{8-6} → RF_A2	T_2-en
RF_D1 → T_1	
RF-D2 → T_2	

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)

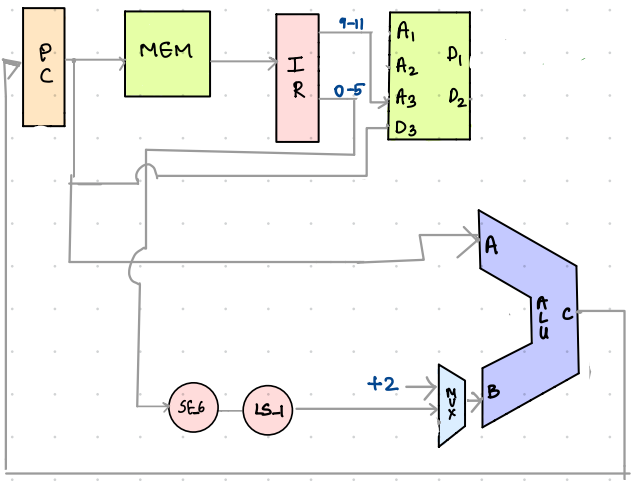
PC → ALU_A	ALU_ADD
IR_{5-0} → SE_6	PC-en
SE-6 → LS-1	
LS-1 → ALU-B	
ALU-C → PC	



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:-

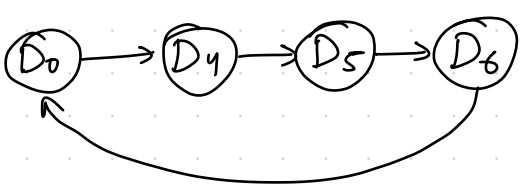
no data tranfer	Understand Instruction

D5:-

IR_{9-11} → RF_A_3	RF_en
PC → RF-D_3	

D6:-

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



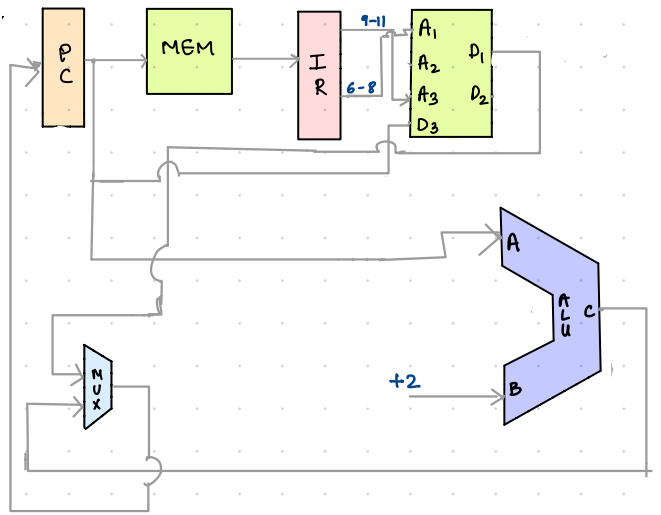
Jump and Link to Register

IR

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

D_0 :-

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



D_4 :-

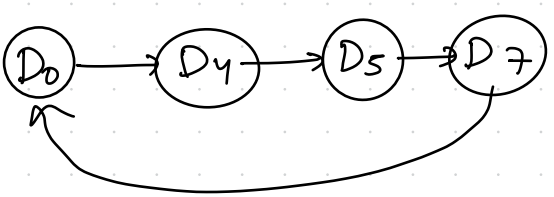
no data tranfer	Understand Instruction

D_5 :-

IR_{9-11} → RF_A_3	RF_en
PC → RF_D_3	

D_7 :-

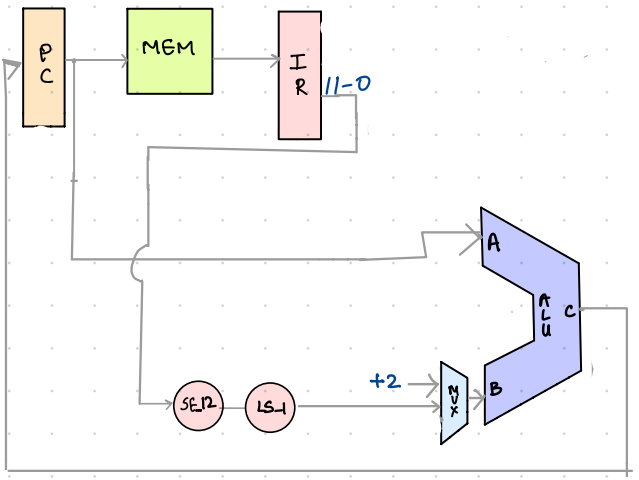
IR_{8-6} → RF_A_1	PC_en
RF_D_1 → PC	



Jump

IR

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



D0 :-

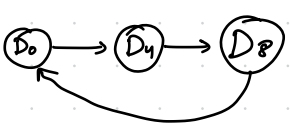
PC → Mem-A	Mem_Read
PC → ALU_A	ALU-ADD
Mem-D → IR	IR_en
ALU_C → PC	PC_en
+2 → ALU_B	

D4 :-

no data tranfer	Understand Instruction

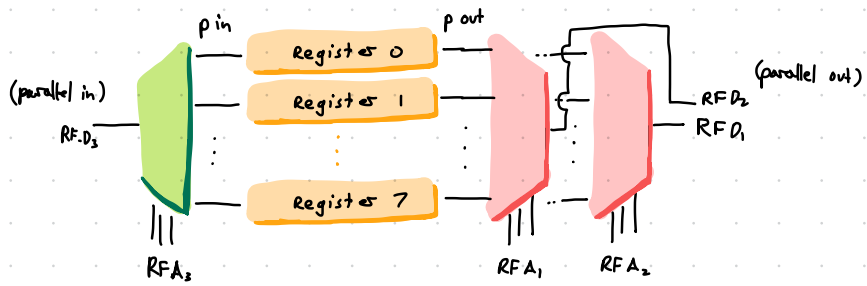
D8 :-

PC → ALU-A	ALU_ADD
IR_{11-0} → SE_12	PC-en
SE_12 → LS-1	
LS_1 → ALU_B	
ALU-C → PC	



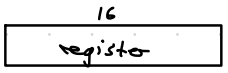
IITB-CPU

Register File (16 bit Reg, 8 Registers);

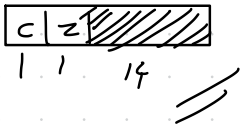


In VHDL, registers can be implemented using easily.

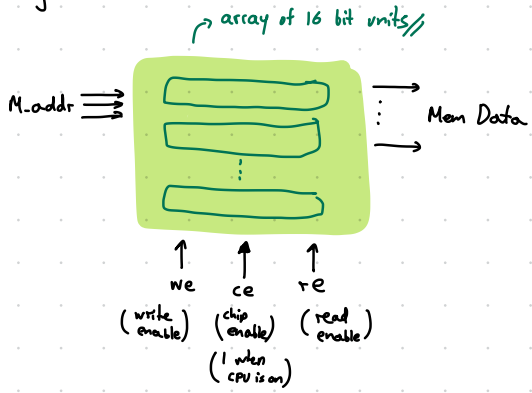
General purpose register:



Condition code Register



Memory unit:



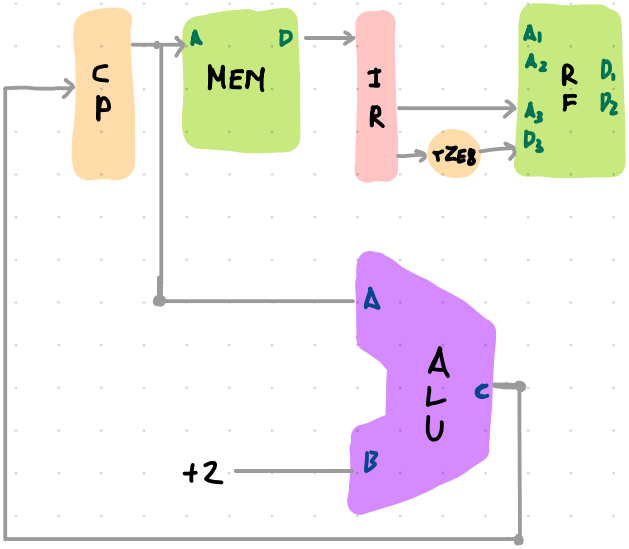
(im proposing 2 units)

(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	



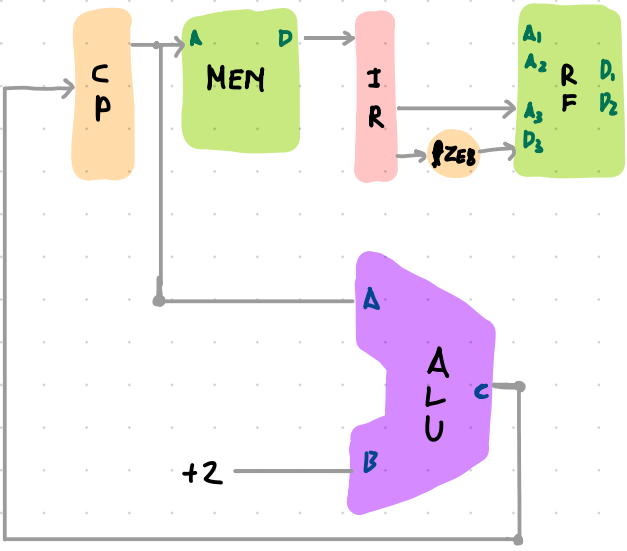
M_1 - Store on MSBs

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	



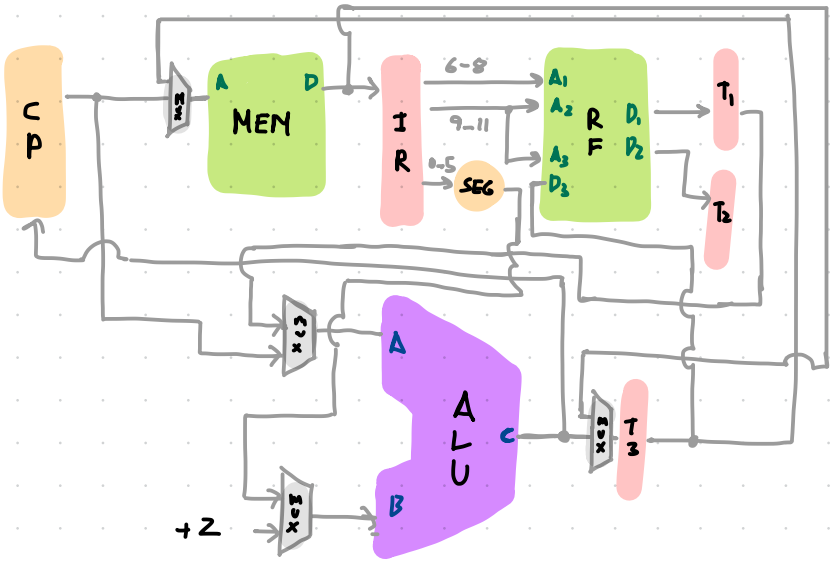
M_2 - Store on LSB's

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_en
RF_D1 → T1	T2_en
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_en
SE6 → ALU_B	
ALU_C → T3	

M_5 : Memory Read

T3 → Mem_Addr	Mem_Read
Mem_Data → T3	T3_en

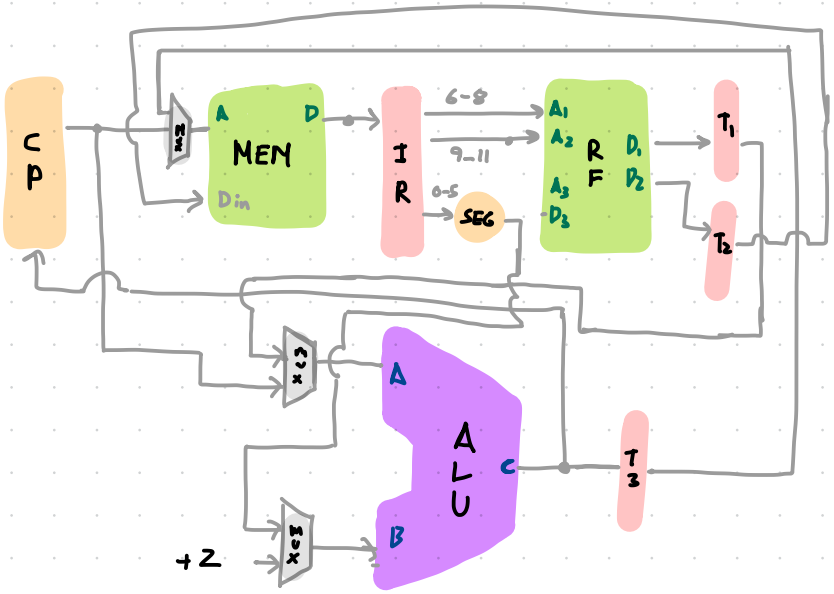
M_6 - Load to Register ($T_3 \rightarrow$ Reg)

$T_3 \rightarrow RF\_D_3$	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	
ALUC $\rightarrow$ PC	



M_3: Understand Opr (I type)

$IR\_6\text{-}8 \rightarrow RF\_A1$	T1_en
$RF\_D1 \rightarrow T1$	T2_en
$IR\_9\text{-}11 \rightarrow RF\_A2$	
$RF\_D2 \rightarrow T2$	

M_4: Compute Address (Add T1 & Imm)

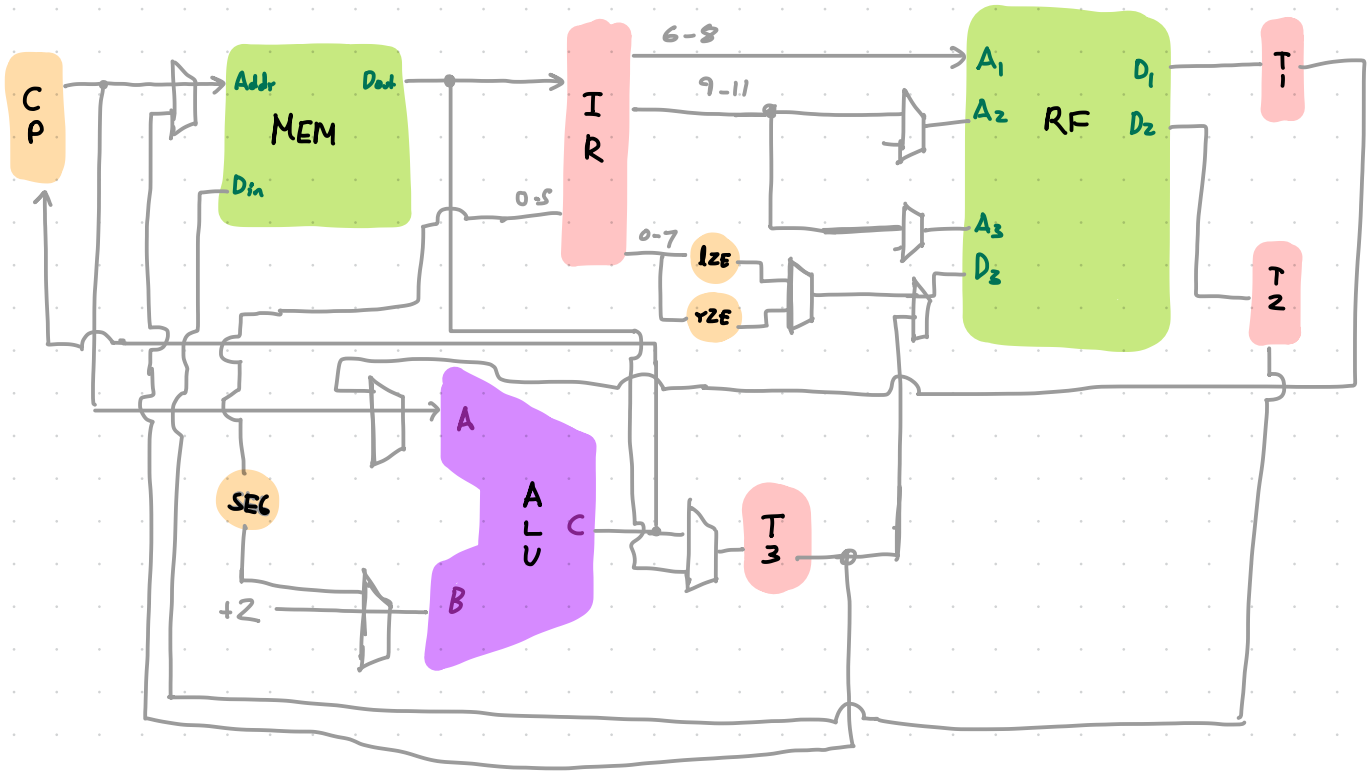
$T1 \rightarrow ALU\_A$	ALU_ADD
$IR\_0\text{-}5 \rightarrow SE6$	T3_e
$SE6 \rightarrow ALU\_B$	
$ALU\_C \rightarrow T3$	

M_7: Memory Write

$T_3 \rightarrow$ Mem_Addr	Mem_write
$T_2 \rightarrow$ Mem_Data	

Memory

Net diagram with all ~~LOAD~~ statements :



MUX SIGNALS :

MUX 1: 6

P_{11}	P_{10}	out
0	0	Disconnect
0	1	IP
1	0	T3

MUX 3: 10

P_3	Out
0	IR 6_8
1	IR 9_11

MUX 6: 5

P_{62}	P_{61}	P_{60}	Out
0	0	0	Disconnect
0	0	1	+2
0	1	0	T2
0	1	1	SE6
1	0	0	LS1
1	0	1	SE9

MUX 2: 9

P_2	Out
0	IR 9-11
1	IR 6-8

MUX 4 2

P_{41}	P_{40}	Out
0	0	Disconnect
0	1	IR 3_5
1	0	IR 6_8
1	1	IR 9_11

MUX 5: 4

P_{51}	P_{50}	Out
0	0	Disconnect
0	1	PC
1	0	T1

MUX 7: 3

P_{72}	P_{71}	P_{70}	Out
0	0	0	Disconnect
0	0	1	T3
0	1	0	rZE8
0	1	1	IZE8
1	0	0	CP

MUX 8 7

P_{81}	P_{80}	Out
0	0	Disconnect
0	1	ALU_C
1	0	Mem_Data

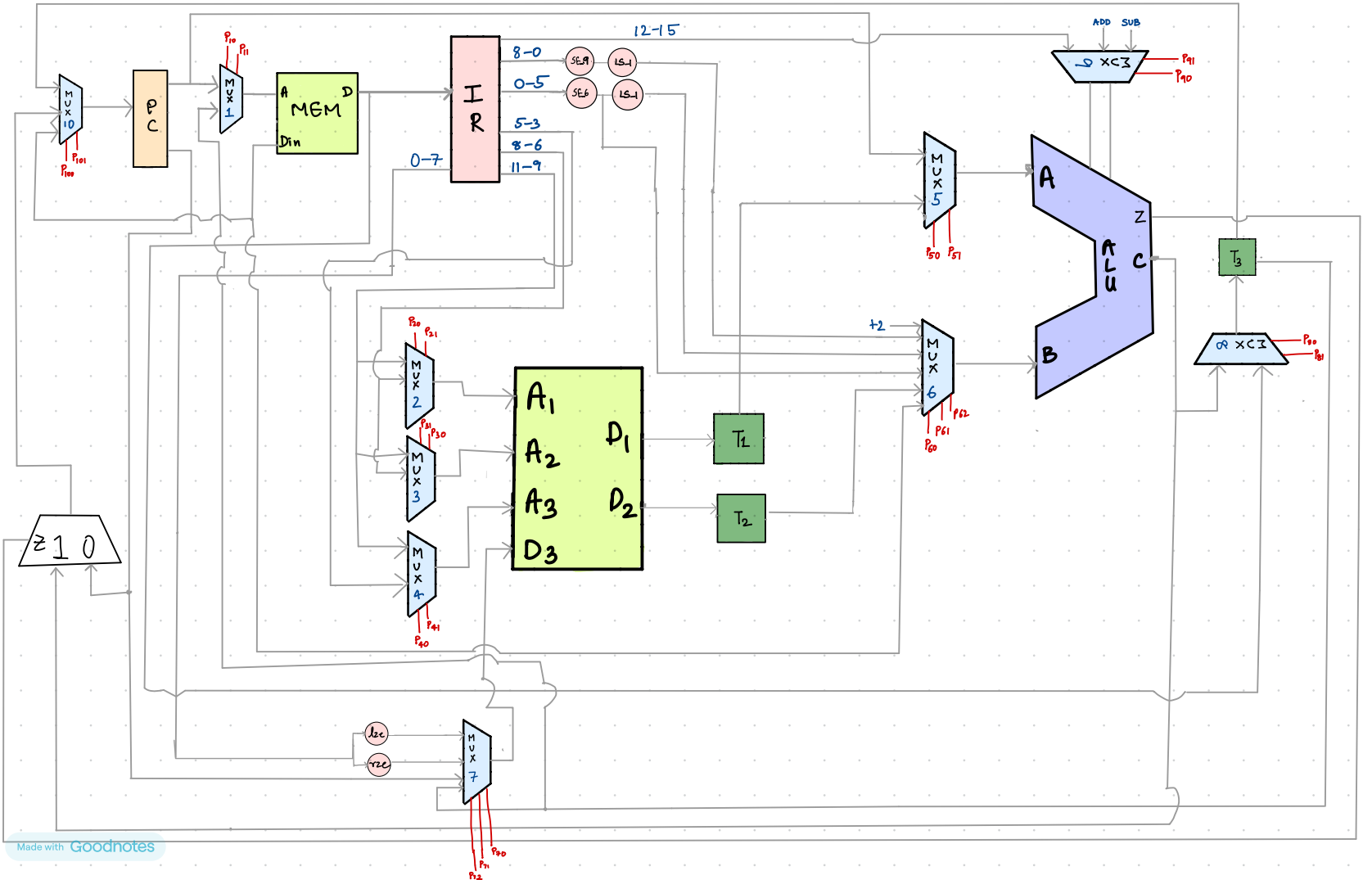
MUX 9: (1)

P_{91}	P_{90}	out
0	0	disconnect
0	1	0000 (add)
1	0	0010 (sub)
1	1	IR 12_15

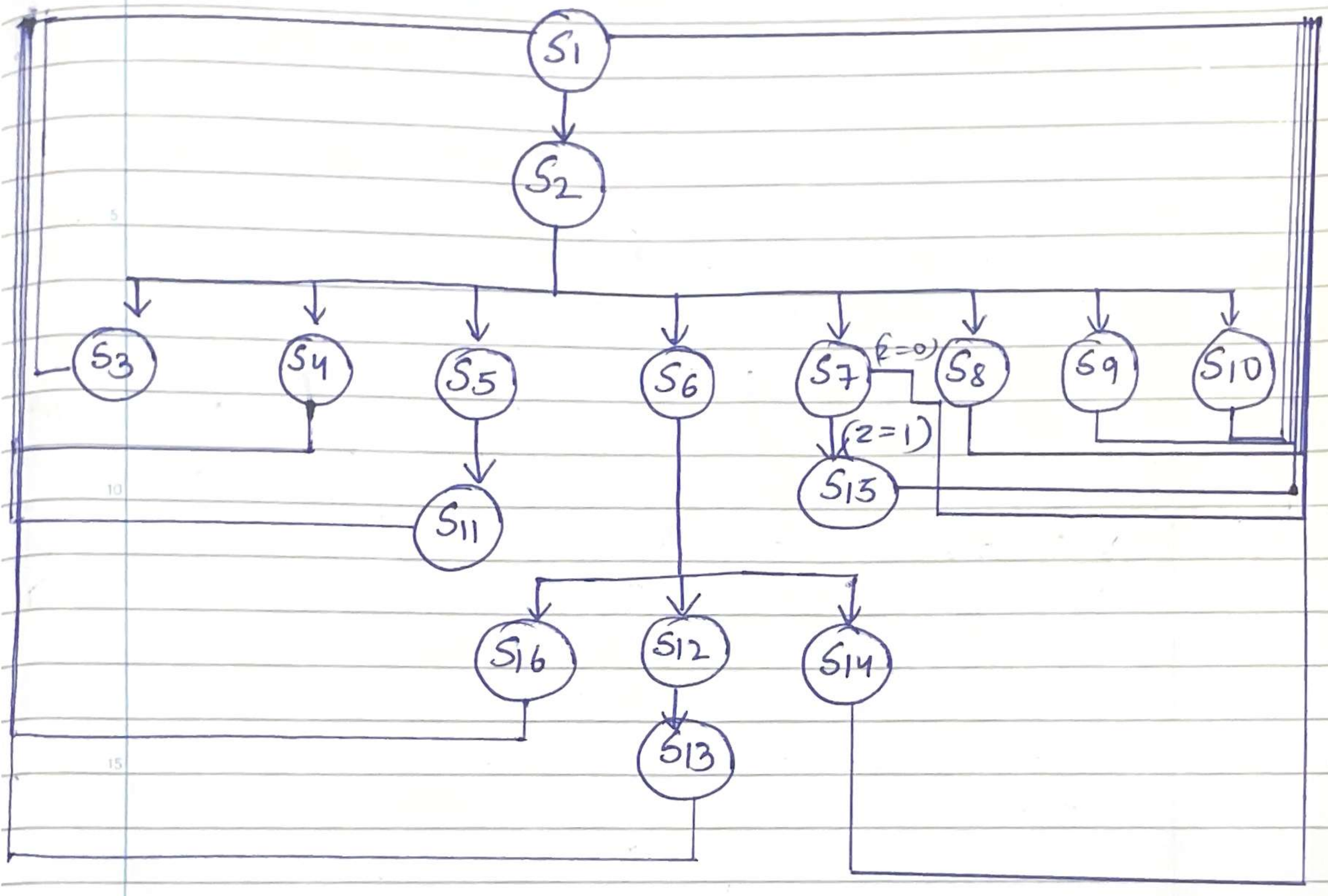
MUX 10: 8

$P_{10.1}$	$P_{10.0}$	Out
0	0	Disconnect
0	1	ALU-C
1	0	BEQ
1	1	T2

	8		1		7		3			5			4		2		10	9	6	
State	P_{10-1}	P_{10-0}	P_{9-1}	P_{9-0}	P_{8-1}	P_{8-0}	P_{7-2}	P_{7-1}	P_{7-0}	P_{6-2}	P_{6-1}	P_{6-0}	P_{5-1}	P_{5-0}	P_{4-1}	P_{4-0}	P_3	P_2	P_{1-1}	P_{1-0}
S_1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1
S_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S_3	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0
S_4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0
S_5	0	0	1	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0
S_6	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0
S_7	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
S_8	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0
S_9	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0
S_{10}	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S_{11}	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
S_{12}	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
S_{13}	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
S_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
S_{15}	1	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0



Date :



S_1: Fetch Information and Update Instruction
S_2: Understand and Fetch Operand
S_3: Store required value (LLI)
S_4: Store required value (LHI)
S_5: Execute operation
S_6: Execute memory addition operation
S_7: Compute (Z)
S_8: Store current instruction pointer
S_9: Compute and update PC (JAL)
S_{10}: Update PC (JLR)
S_{11}: Update Result

S_{12}: Read Memory
S_{13}: Update Register
S_{14}: Read Memory
S_{15}: Compute and update PC (if z==1)
S_{16}: Update Result