Computing System Design

Virendra Singh

Professor

Computer Architecture and Dependable Systems Lab

Department of Electrical Engineering

Indian Institute of Technology Bombay

http://www.ee.iitb.ac.in/~viren/

E-mail: viren@ee.iitb.ac.in

CS-226: Digital Logic Design

Lecture 31: 19 April 2021

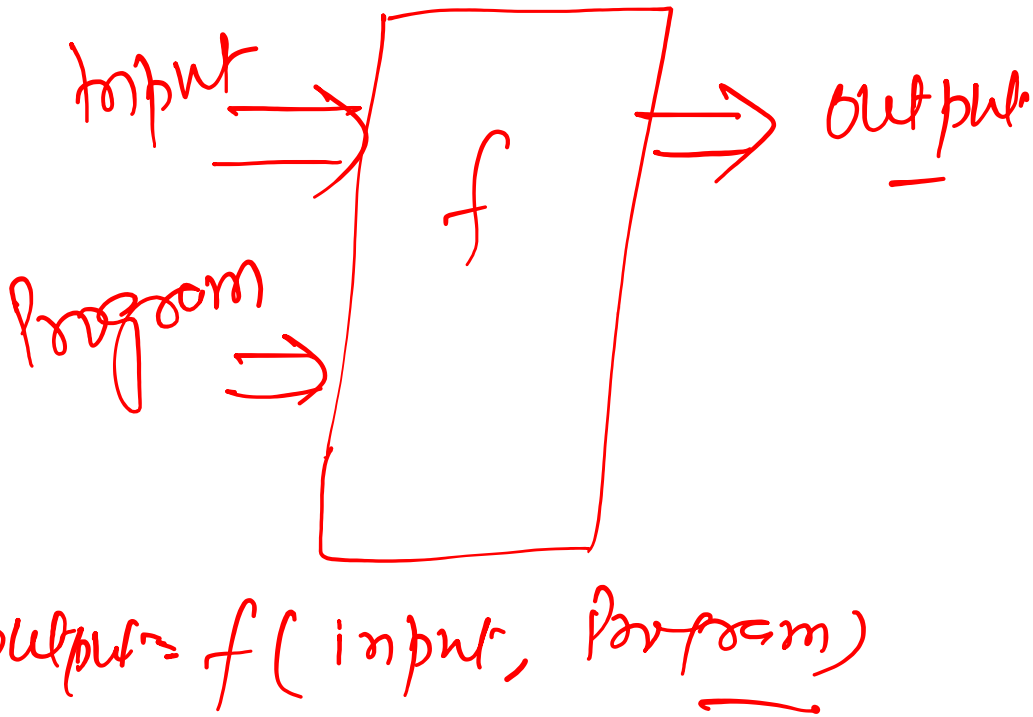
CADSL

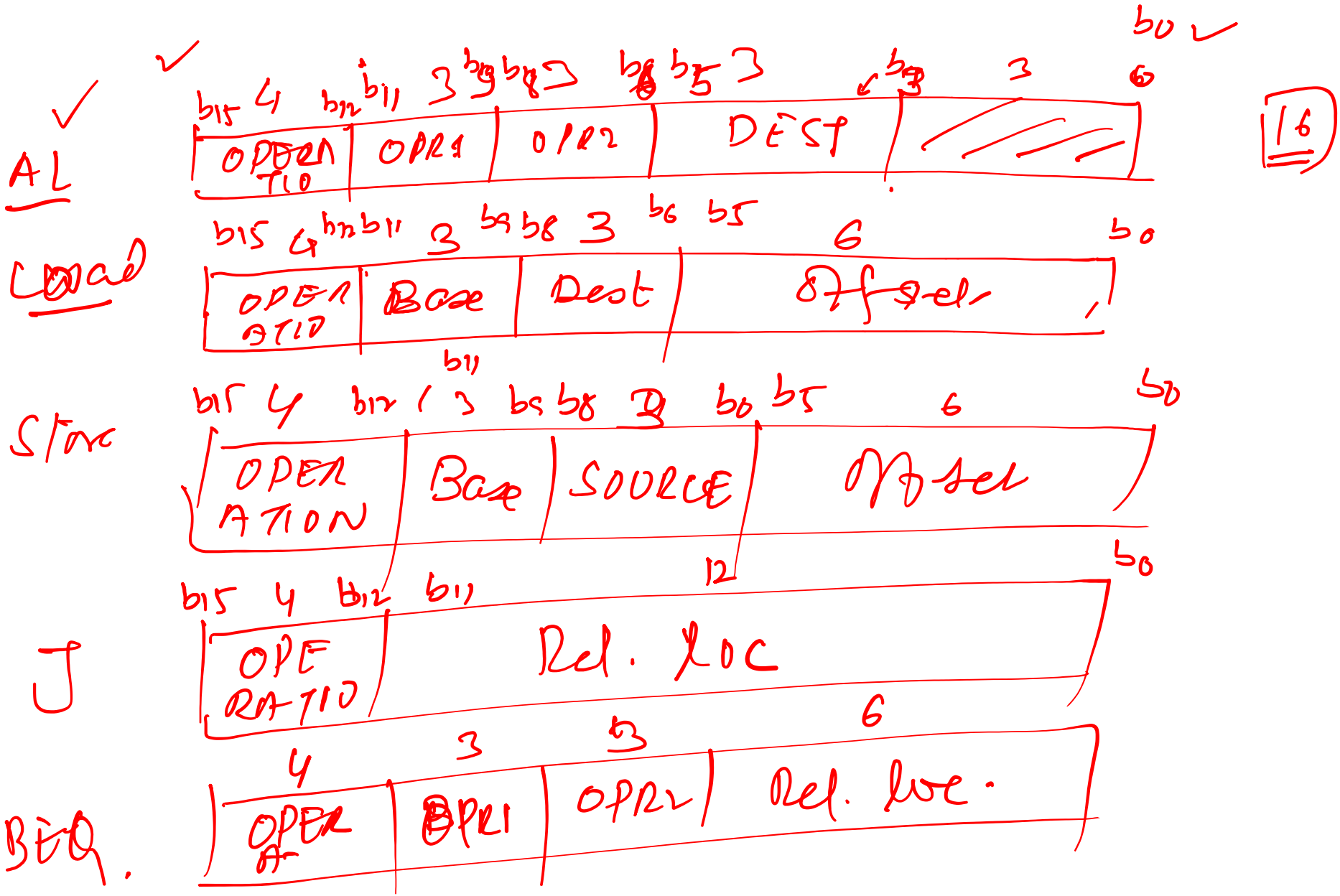
Instruction Set Architecture

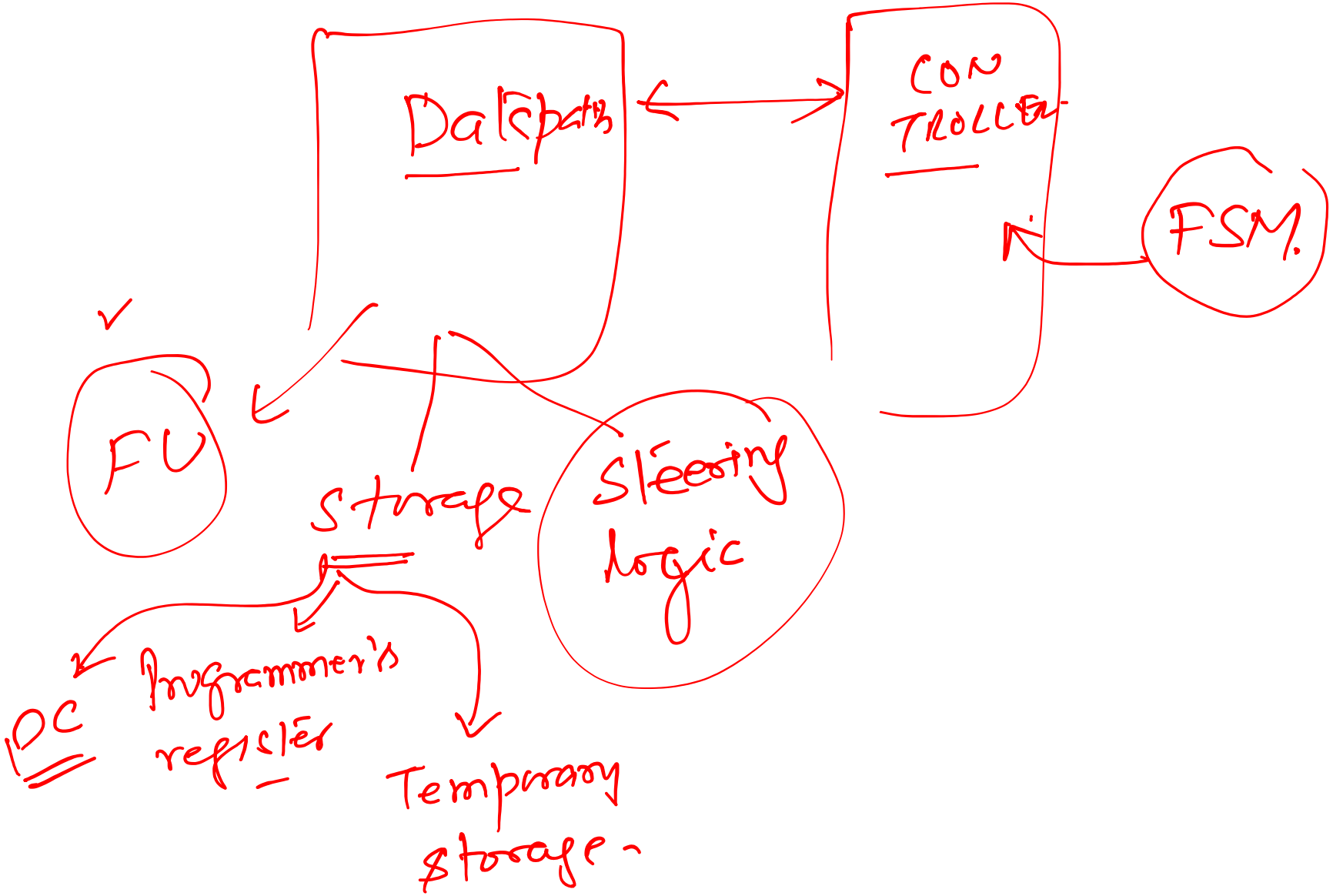
- Instruction set architecture is the structure of a computer that a machine language programmer must understand to write a correct (timing independent) program for that machine.

- The instruction set architecture is also the machine description that a hardware designer must understand to design a correct implementation of the computer.

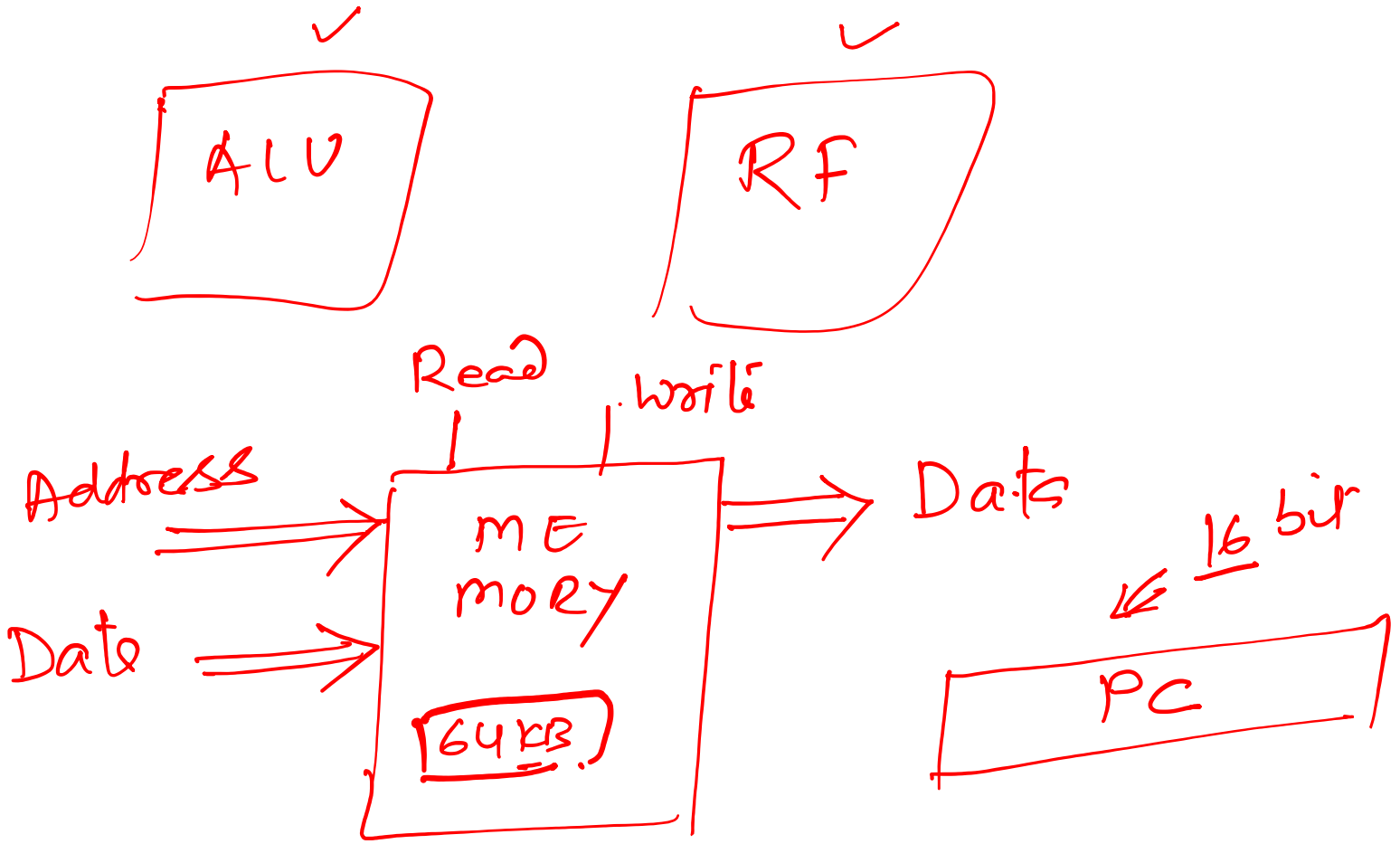
Instruction




AL
OPERATION
OPR1
OPR2
DEST
16
Load
OPERATION
Base
Dest
Offset
Store
OPERATION
Base
SOURCE
Offset
J
OPERATION
Rel. Loc
BEQ.
OPERATION
OPR1
OPR2
Rel. loc.


Datapath
CONTROLLER
FSM
FU
storage
Sleeping logic
PC
Programmer's register
Temporary storage

Datapath Components



Instruction

AL Instruction

ADD R_1, R_2, R_3

1. Bring instruction from memory (PC) ⇐ S1
2. Understand instruction ⇐ S2

OPERATION	OPR1	OPR2	DEST	///
4	3	3	3	

3. Read operand1 & operand 2 ⇐ S3
4. Compute the instruction (Addition Oper. ⇐ S4)
5. Write the result in R- Dest. reg ⇐ S5
6. update PC PC = PC+2

Instruction

State machine (TASK)

S1

PC → Mem_Add

Mem_Data → ir

PC → alu-A

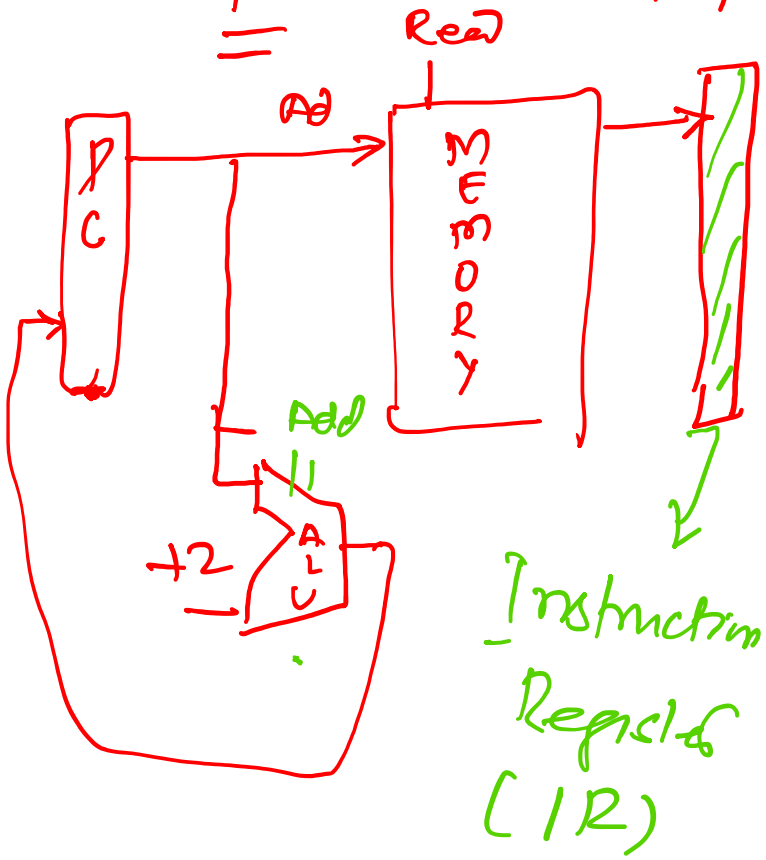
+2 → alu-B

alu-C → PC

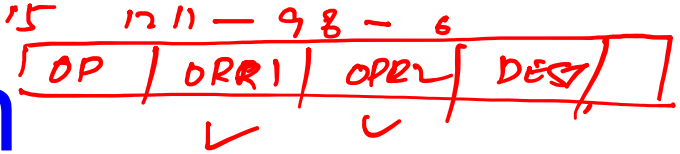
• MR
• PC_W
• IR_W
• alu-add

PC

Memory



Instruction



S_2^{-1}

t_1 write signal

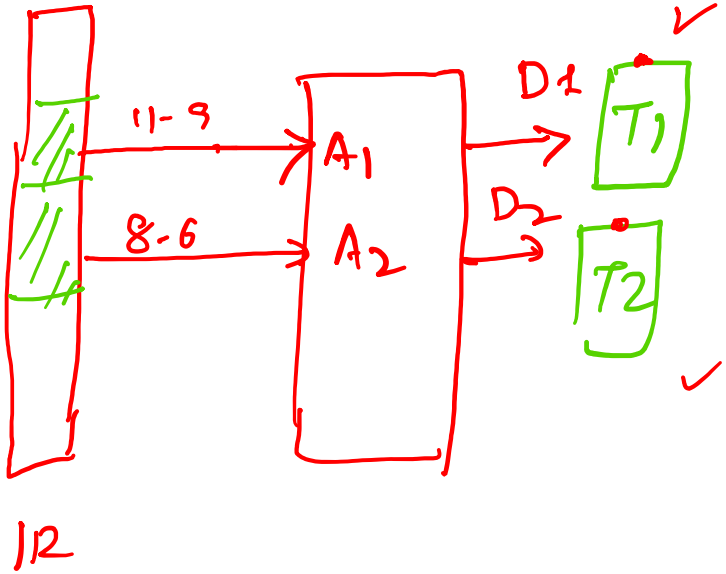
$ir_{11-9} \rightarrow rf\text{-}A_1$

$ir_{8-6} \rightarrow rf\text{-}A_2$

$rf\text{-}D_1 \rightarrow t1$

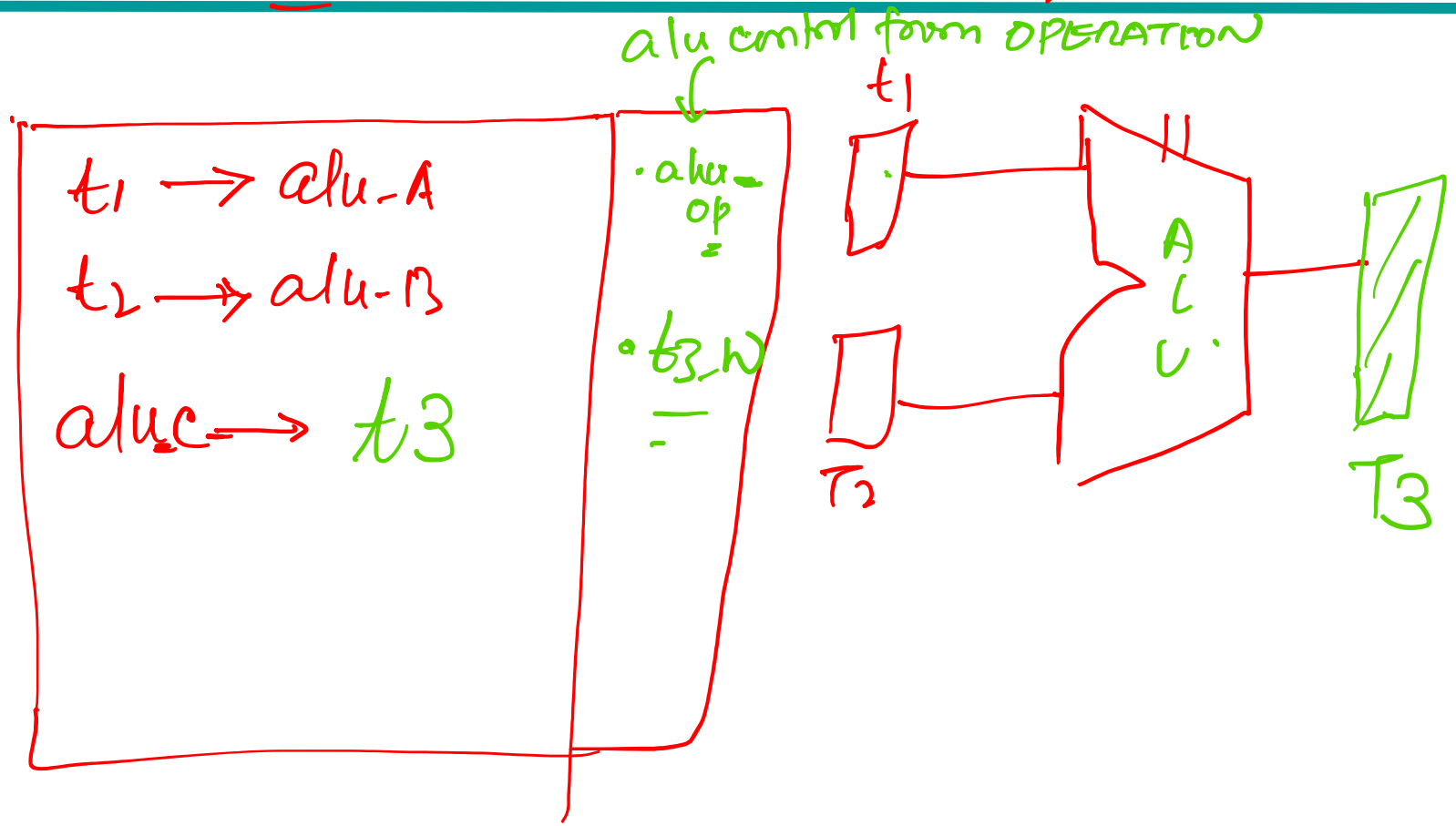
$rf\text{-}D_2 \rightarrow t2$

- t_1-w
- t_2-w



Instruction

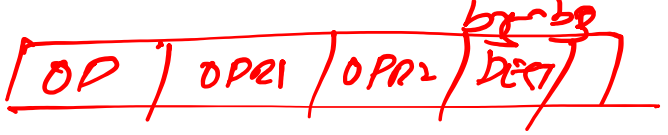
S3 S4 (Execute instruction)

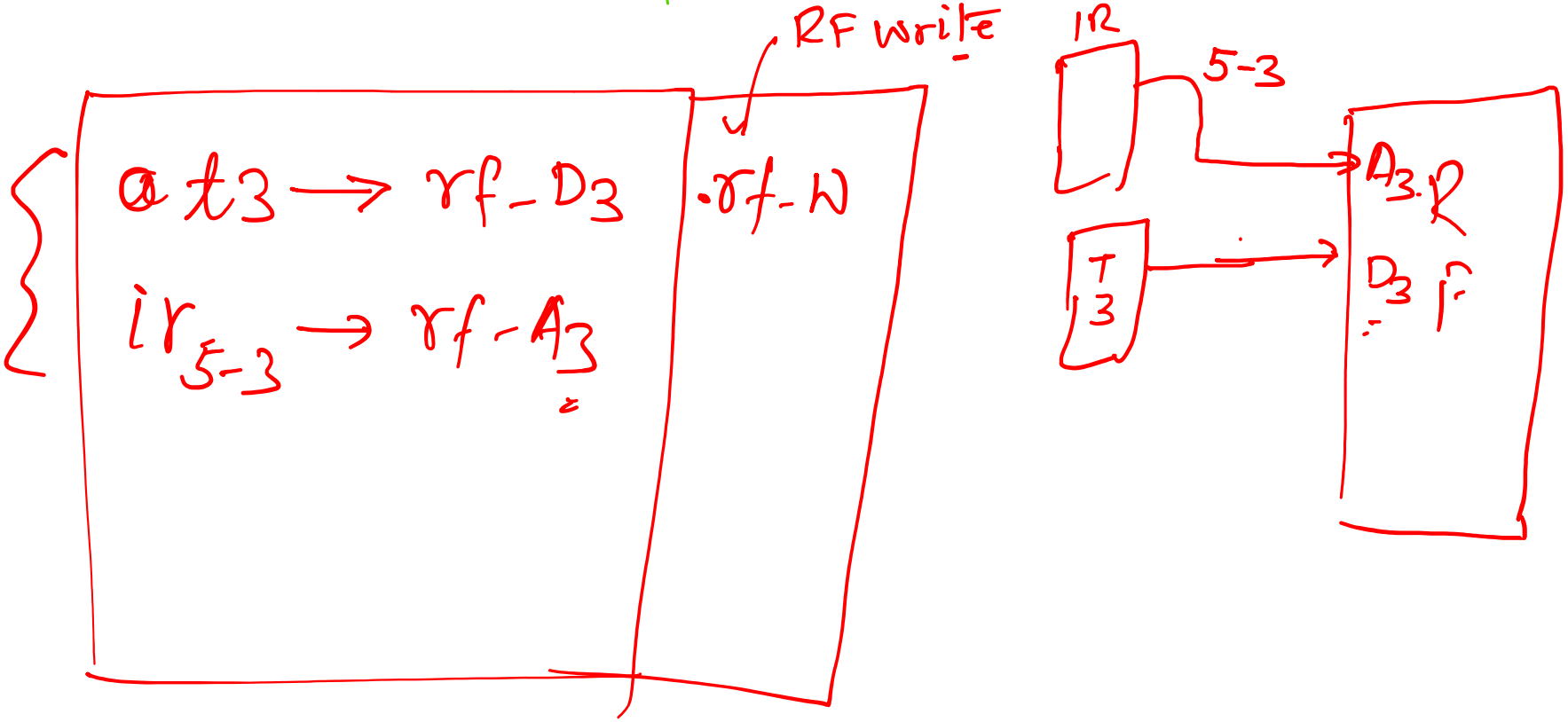


Instruction

S5

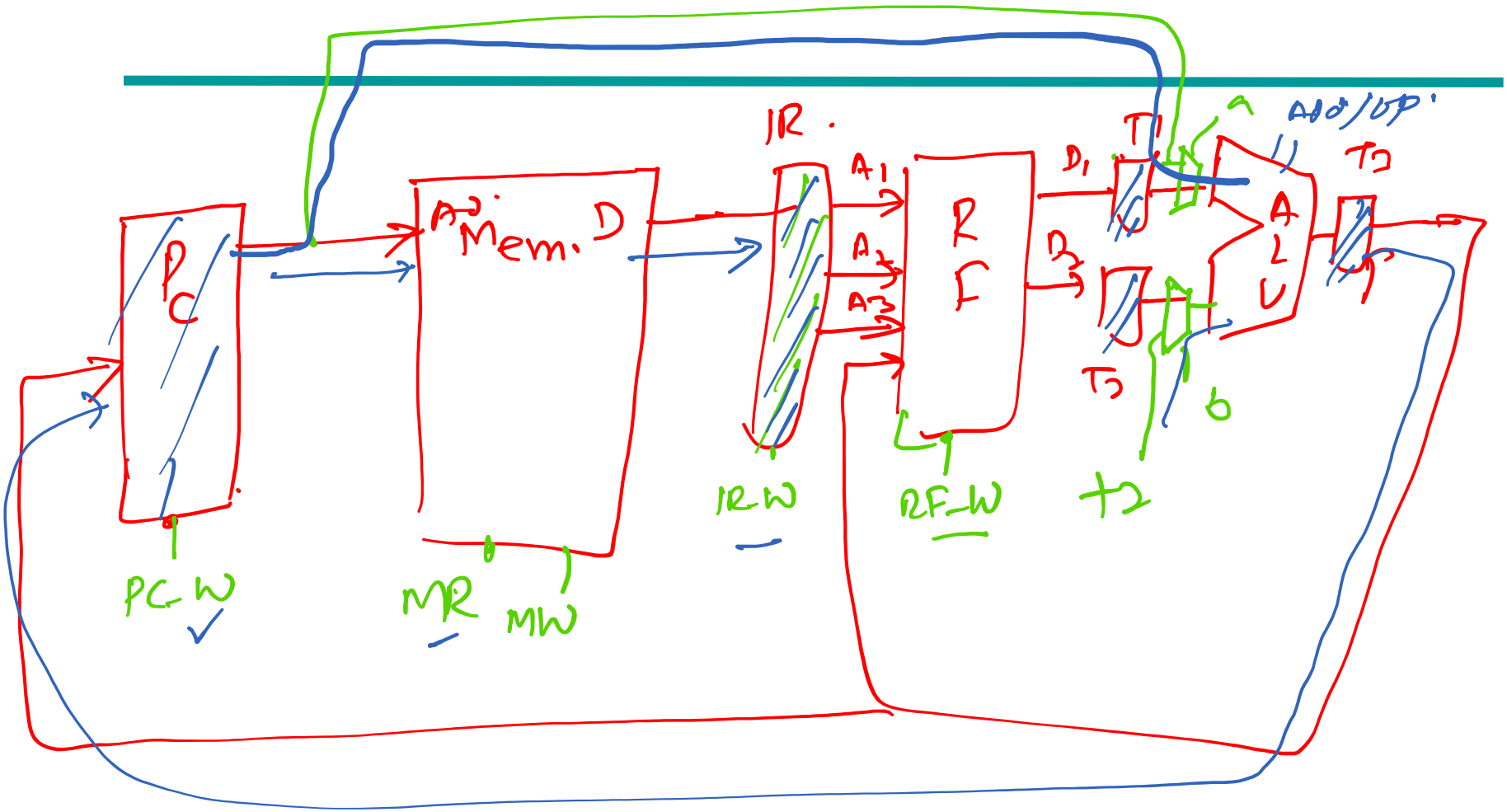
Update the state





$S_1 \rightarrow S_2' \rightarrow S_4 \rightarrow S_5$

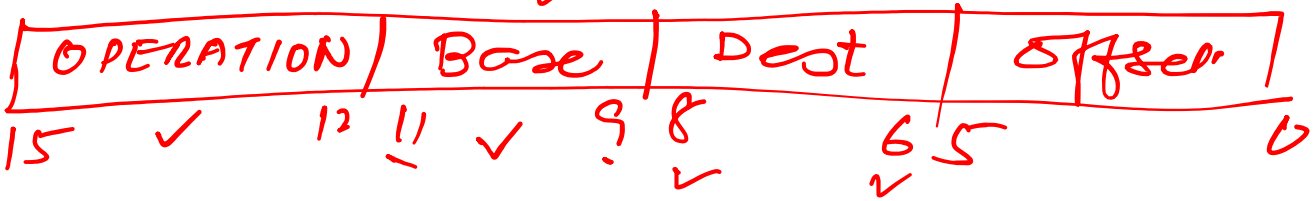
Instruction



Arithmetic & Logical Units

Instruction

Load Instruction

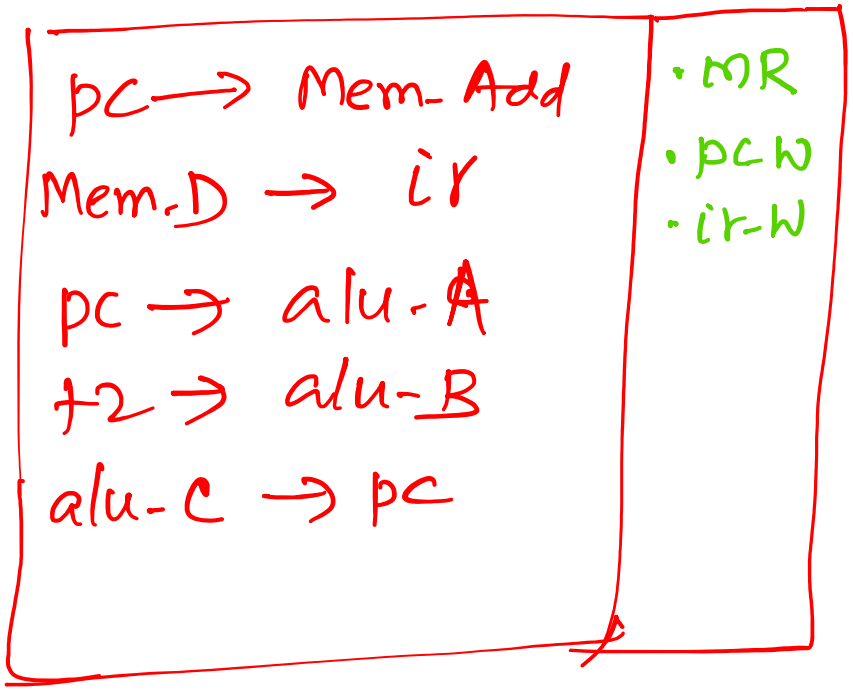


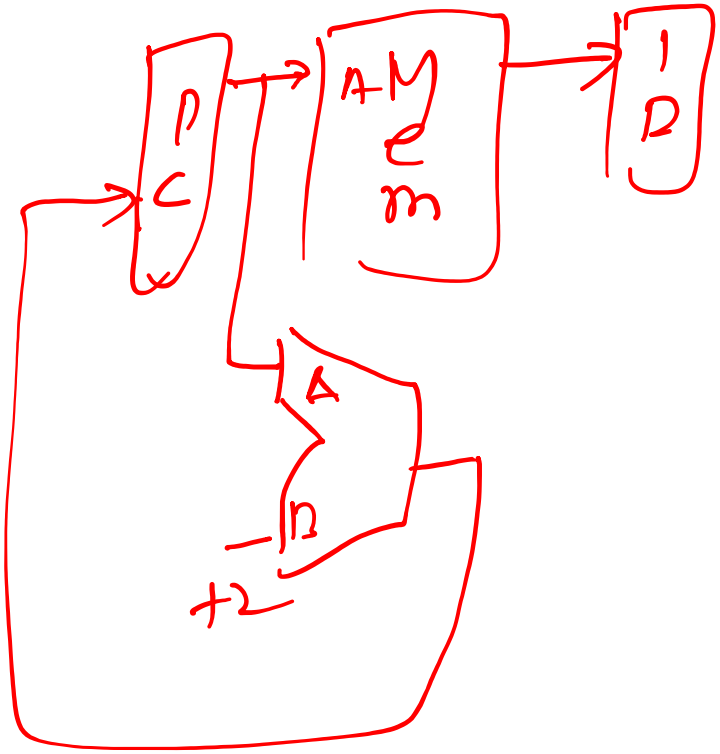
SG OPERATIONS

1. Read instruction from memory (PC) & update PC.
2. Understand and read the base address of memory
3. Compute the address of memory (Base add + offset)
4. Read memory
5. ~~Write~~ Update the reg.

Instruction

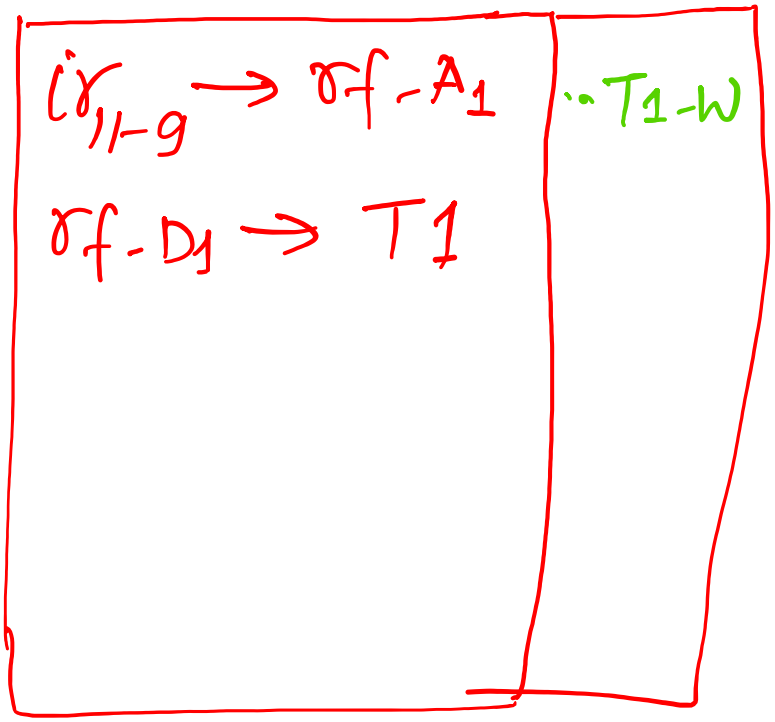
SG

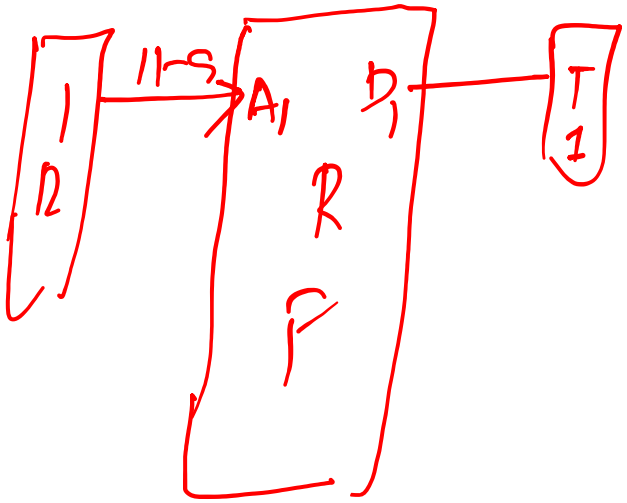




Instruction

S7

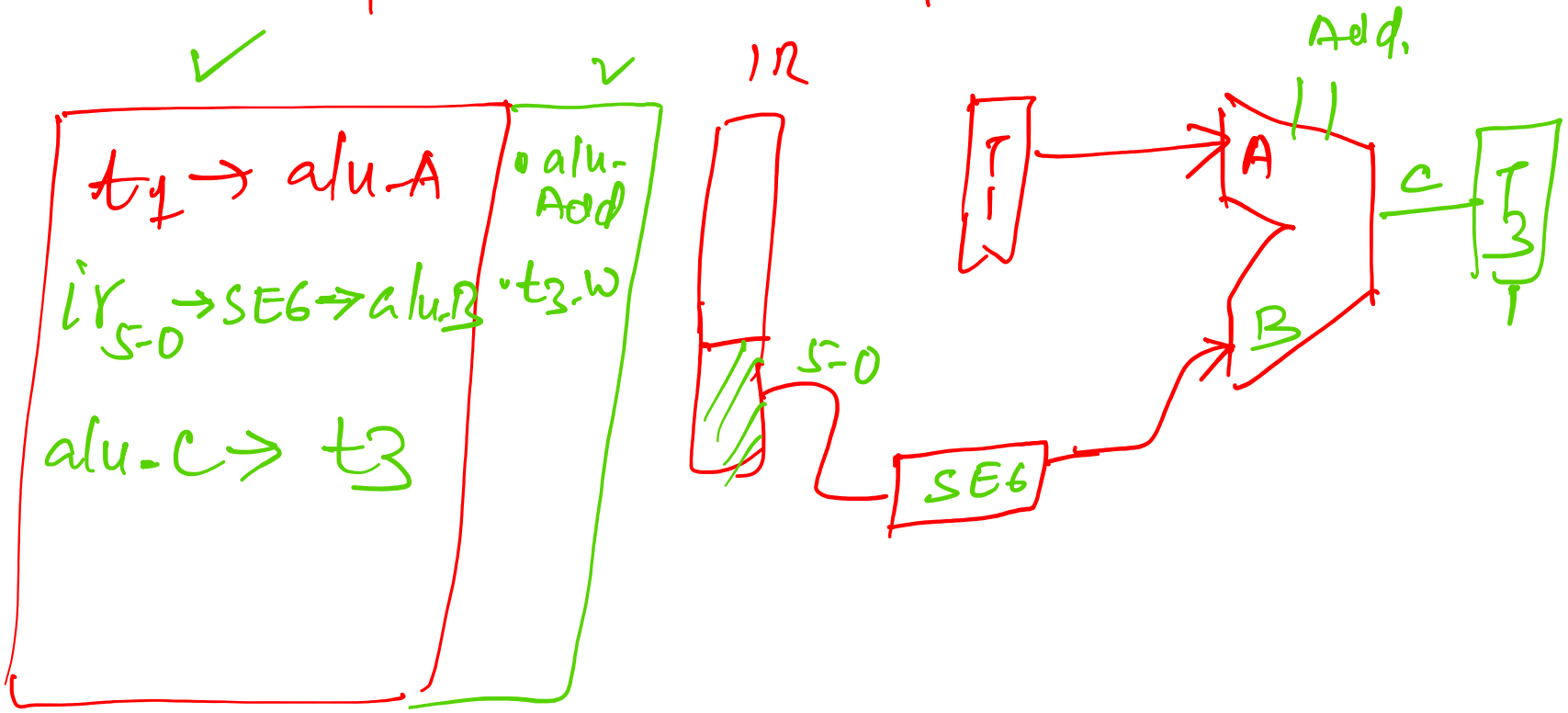




Instruction

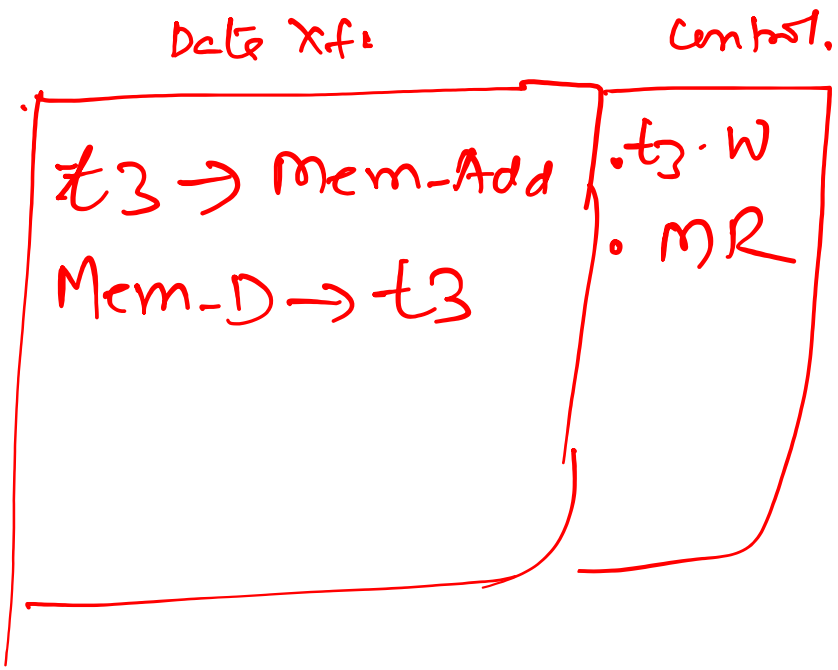
SD

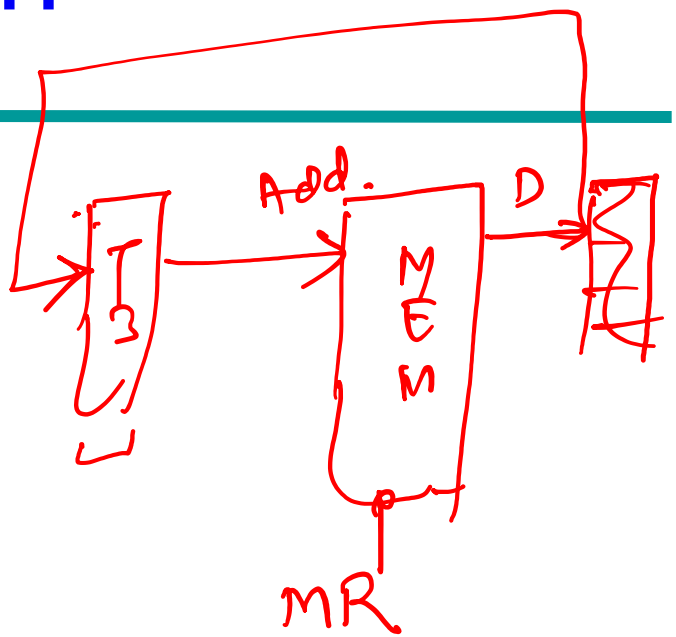
Compute the address of memory



Instruction

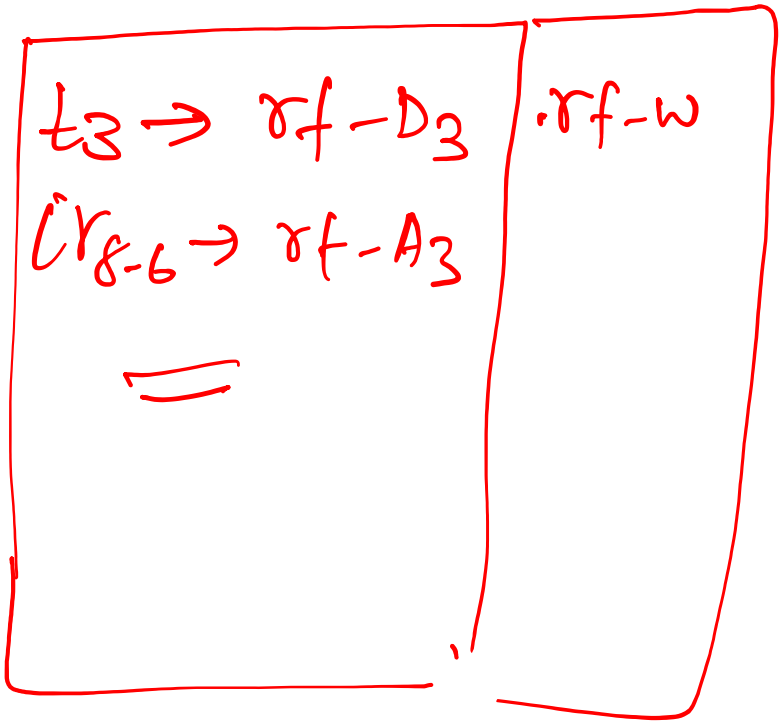
Eg Memory Read.

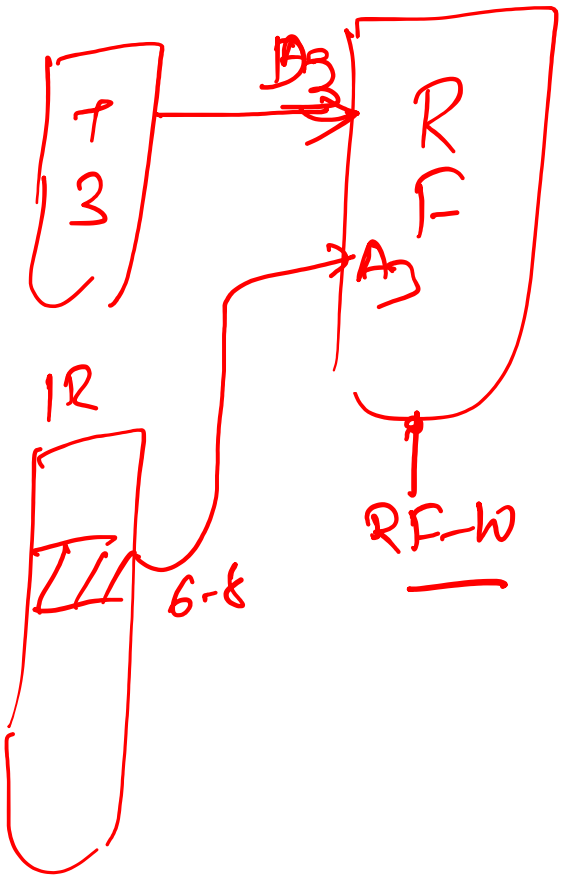




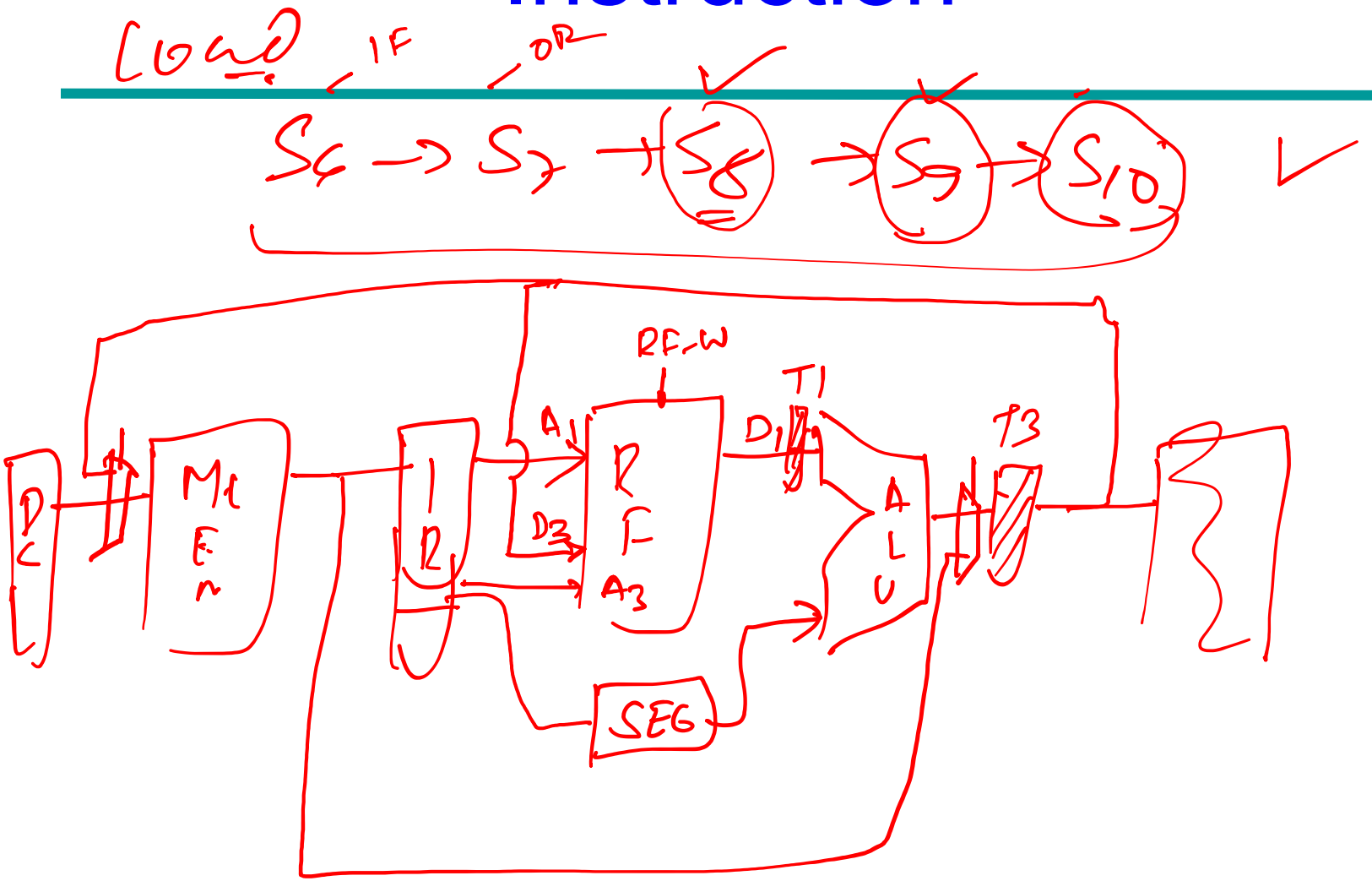
Instruction

S10 Update RF



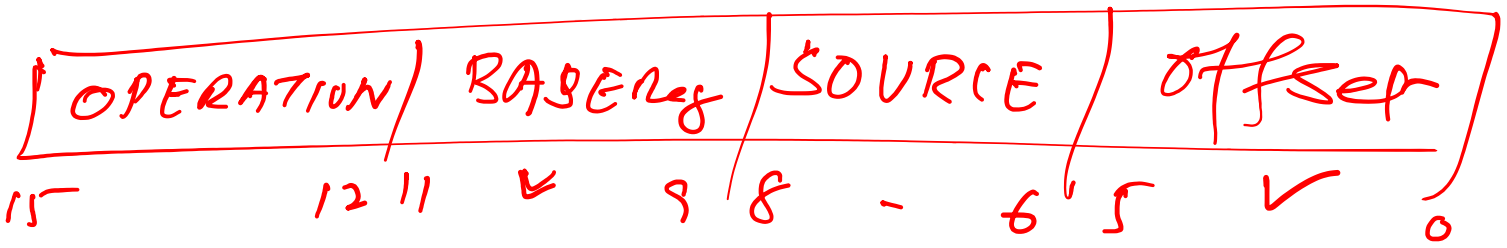


Instruction



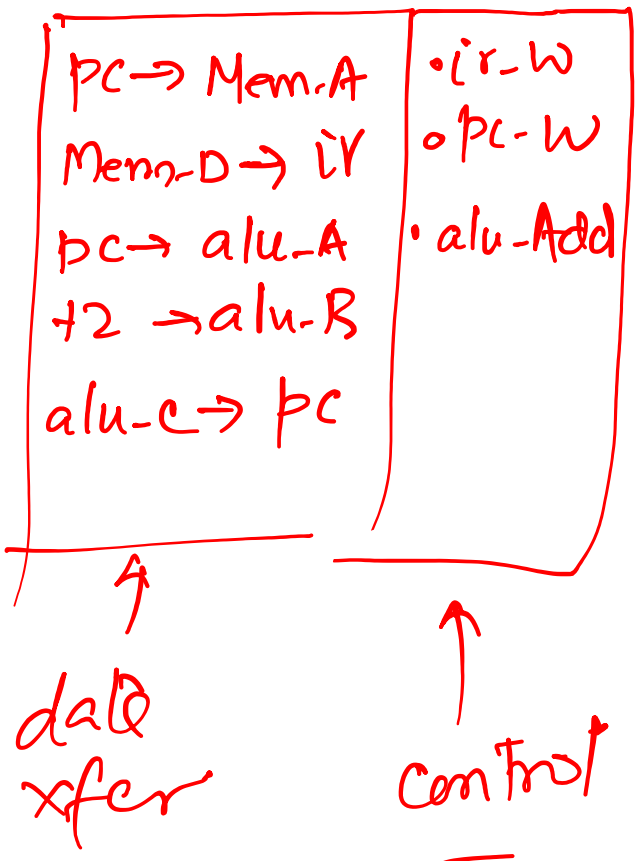
Store Instruction

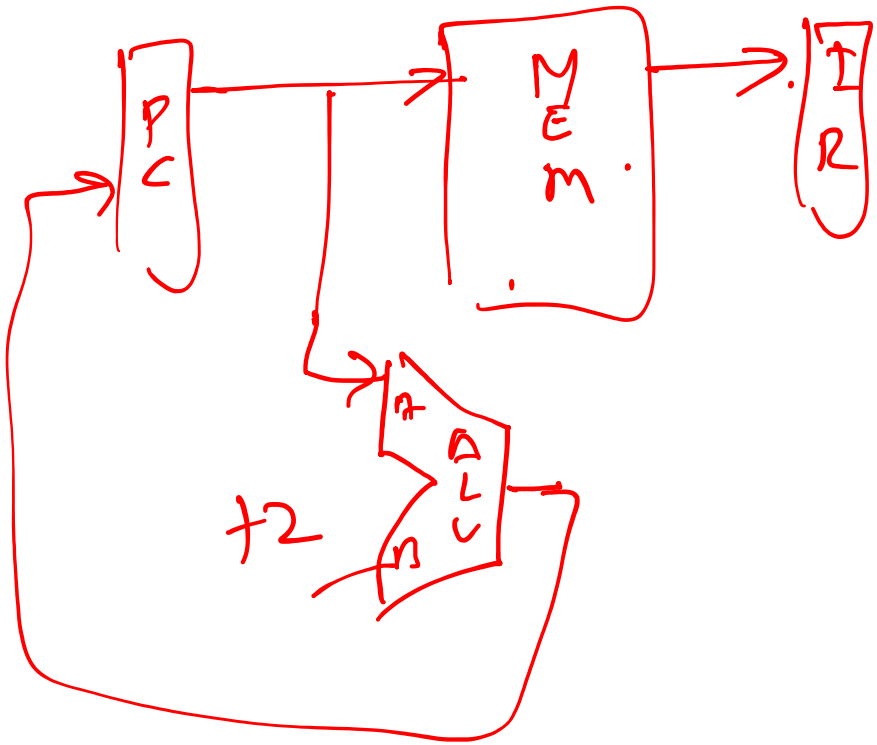
1. Read instruction from memory (fetch) <= S11
2. Understand & Read operands <= S2
3. Compute the address <= S13
4. Write to memory <= S14



Store Instruction

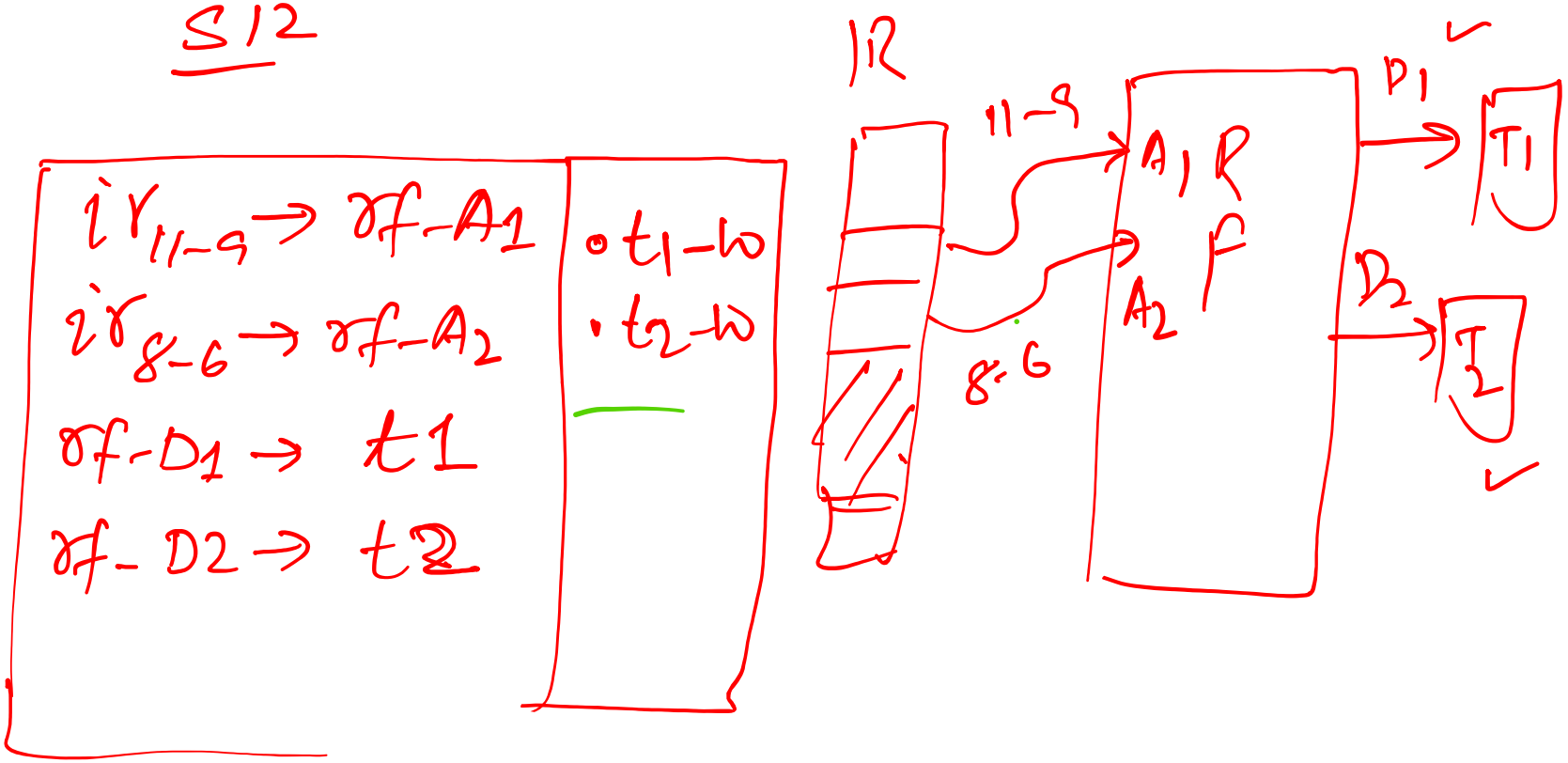
S11



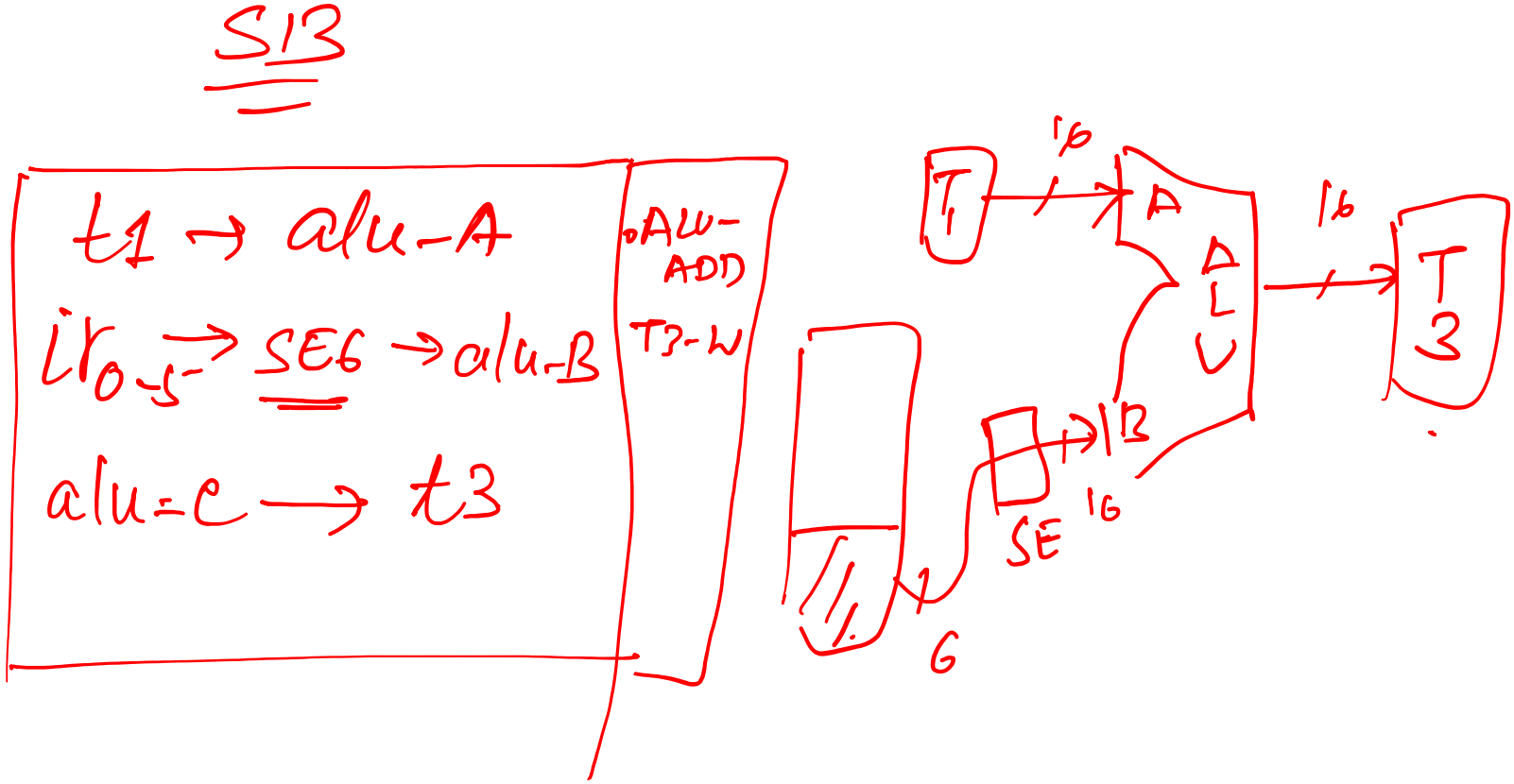


Store Instruction

S12

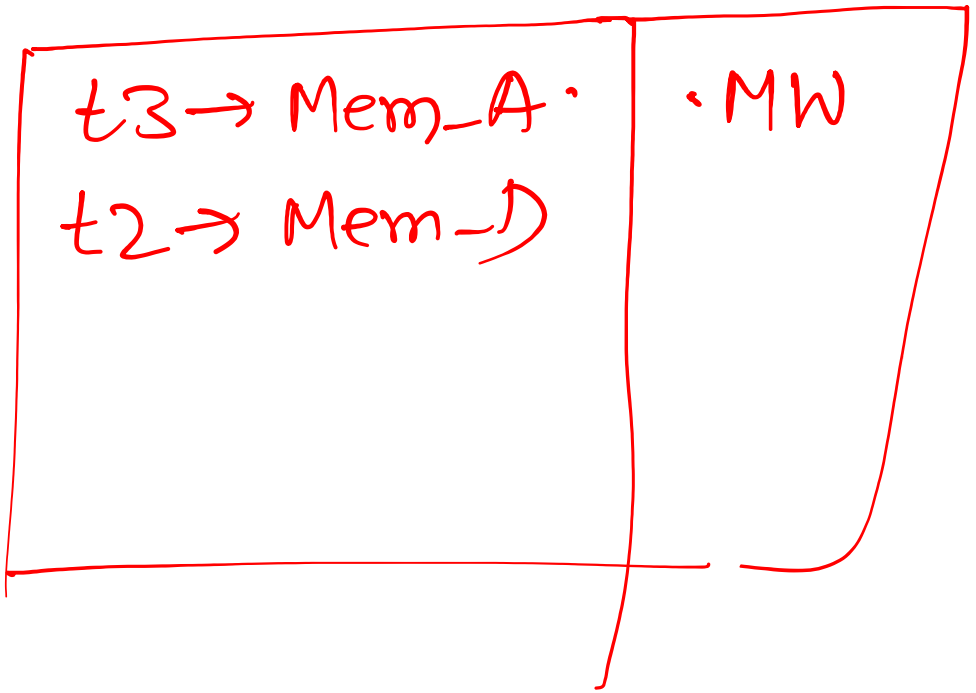


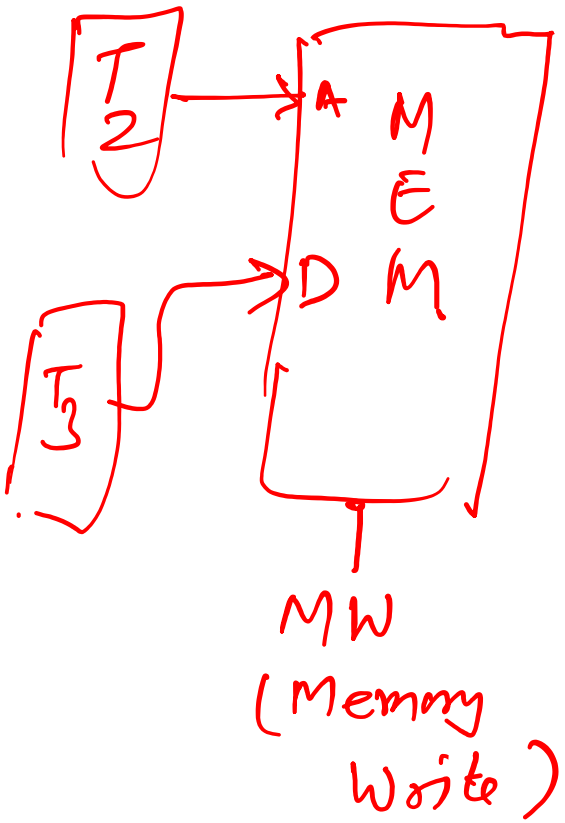
Store Instruction



Store Instruction

S14





S11 → S12 - S13 → S14

BEQ Instruction

BEQ R1, R2, loc.

(1) Read instruction (fetch) update PC ⟸ S15

(2) Understand & read operands. Compute Address for taken. ⟸ S16

(3) Execute (Compare operands & make a decision) ⟸ S17

if equal PC = PC+2+offset X2.

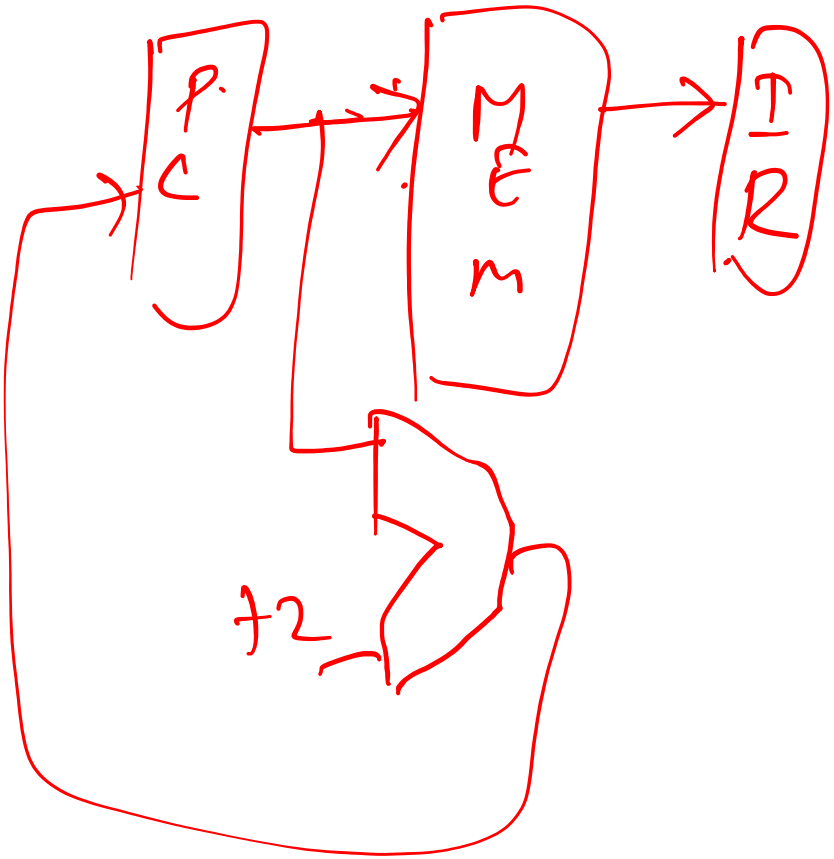
else PC = PC+2

OPERATION	OPR1	OPR2	offset
15 — 12	11 — 9	8 — 6	5 — 0

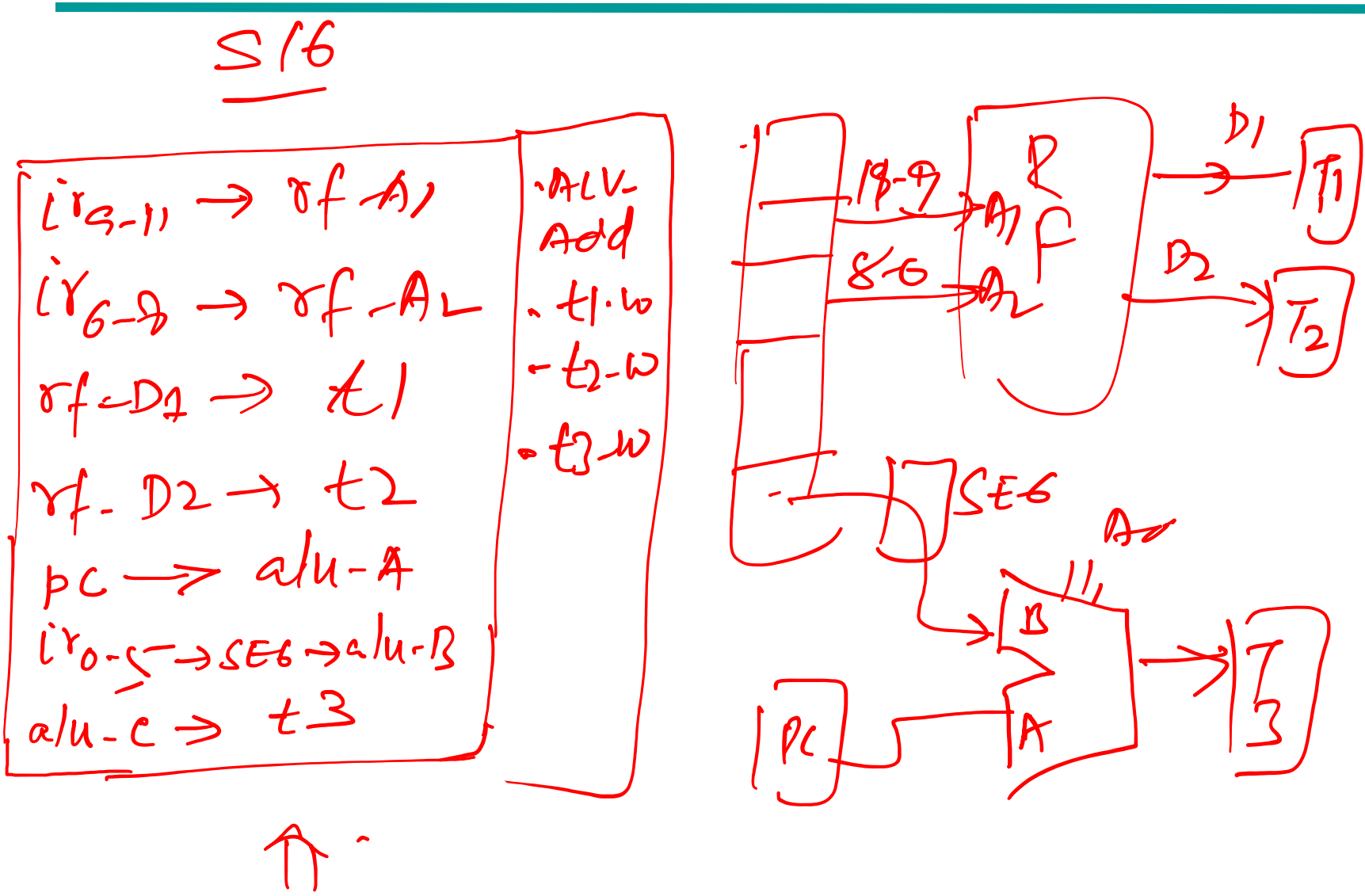
BEQ Instruction

S1S

pc → Mem-A
Mem-D → ir
pc → alu-A
+2 → alu-B
alu-C → pc



BEQ Instruction



BEQ Instruction

S17

Z=1 ($t_1 - t_2 == 0$)

$t_1 \rightarrow$ alu-A

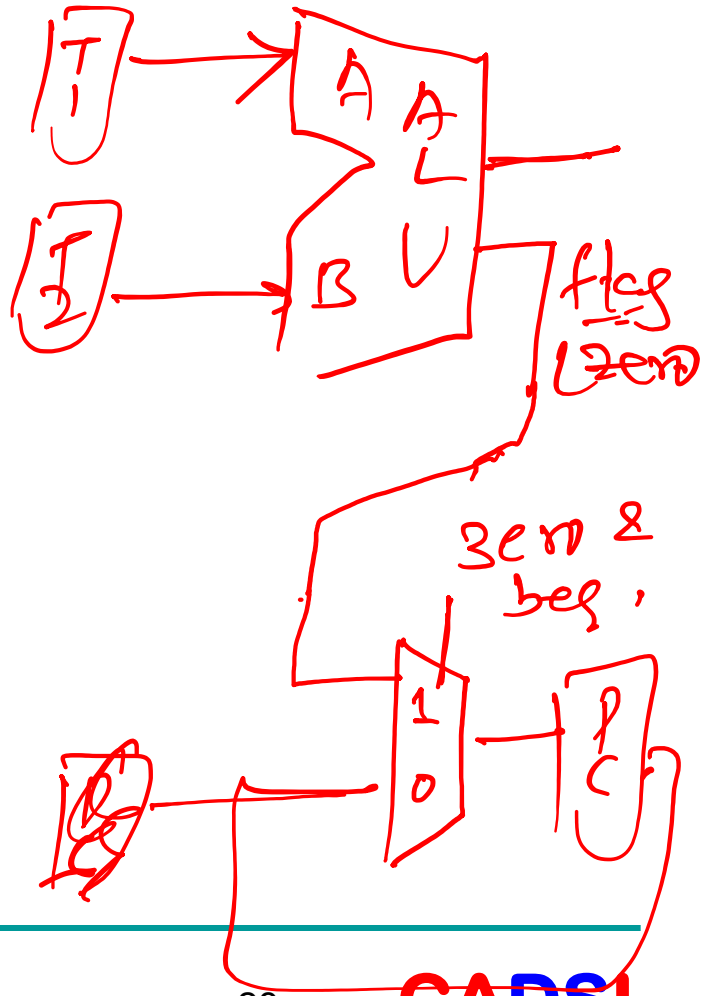
$t_2 \rightarrow$ alu-B

if (Zero) then

~~PC~~

$t_3 \Rightarrow$ PC

• Alu-Sub

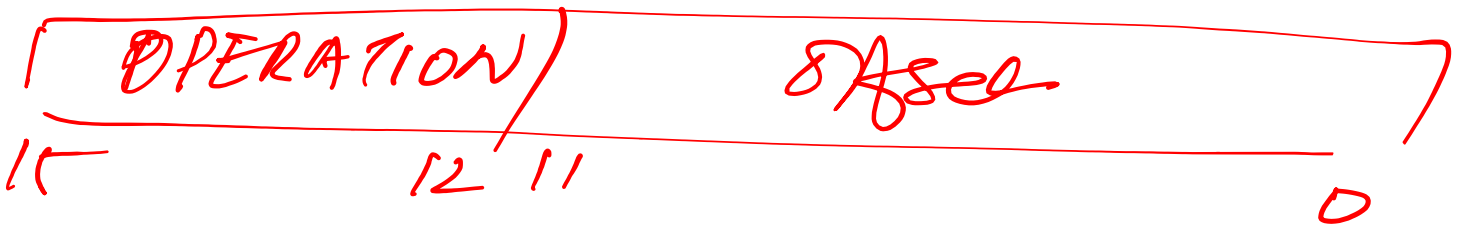


BEQ Instruction

S15-S14 → S12

Jump

PC = PC+2 + Offset*2

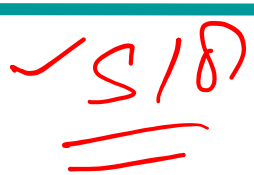


① Fetch instruction ⇐ S18

② Compute the address ⇐ S19

③ Xfer to the add. ⇐ S20

Jump



Jump

S19

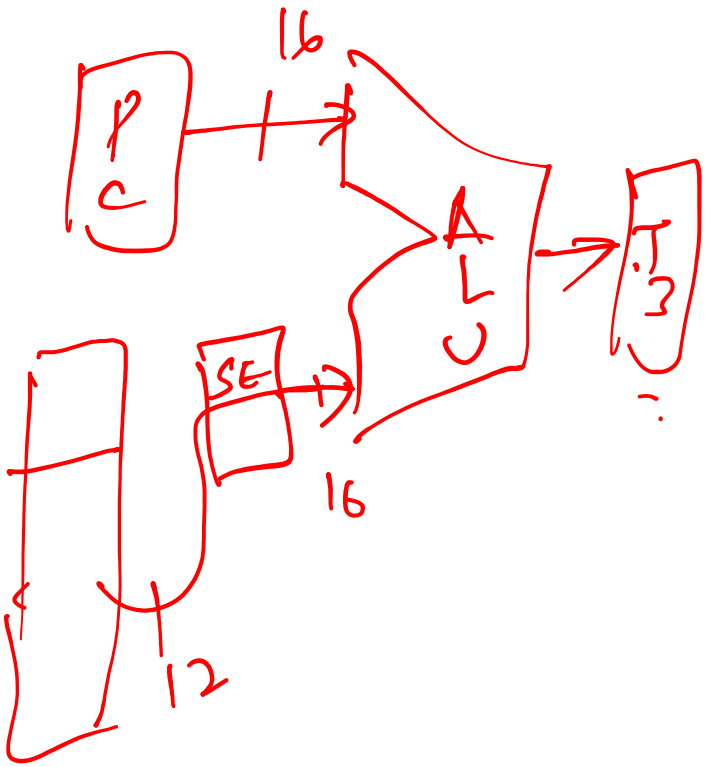
PC → alu-A

IR_{11-0} → SE12 → alu-B

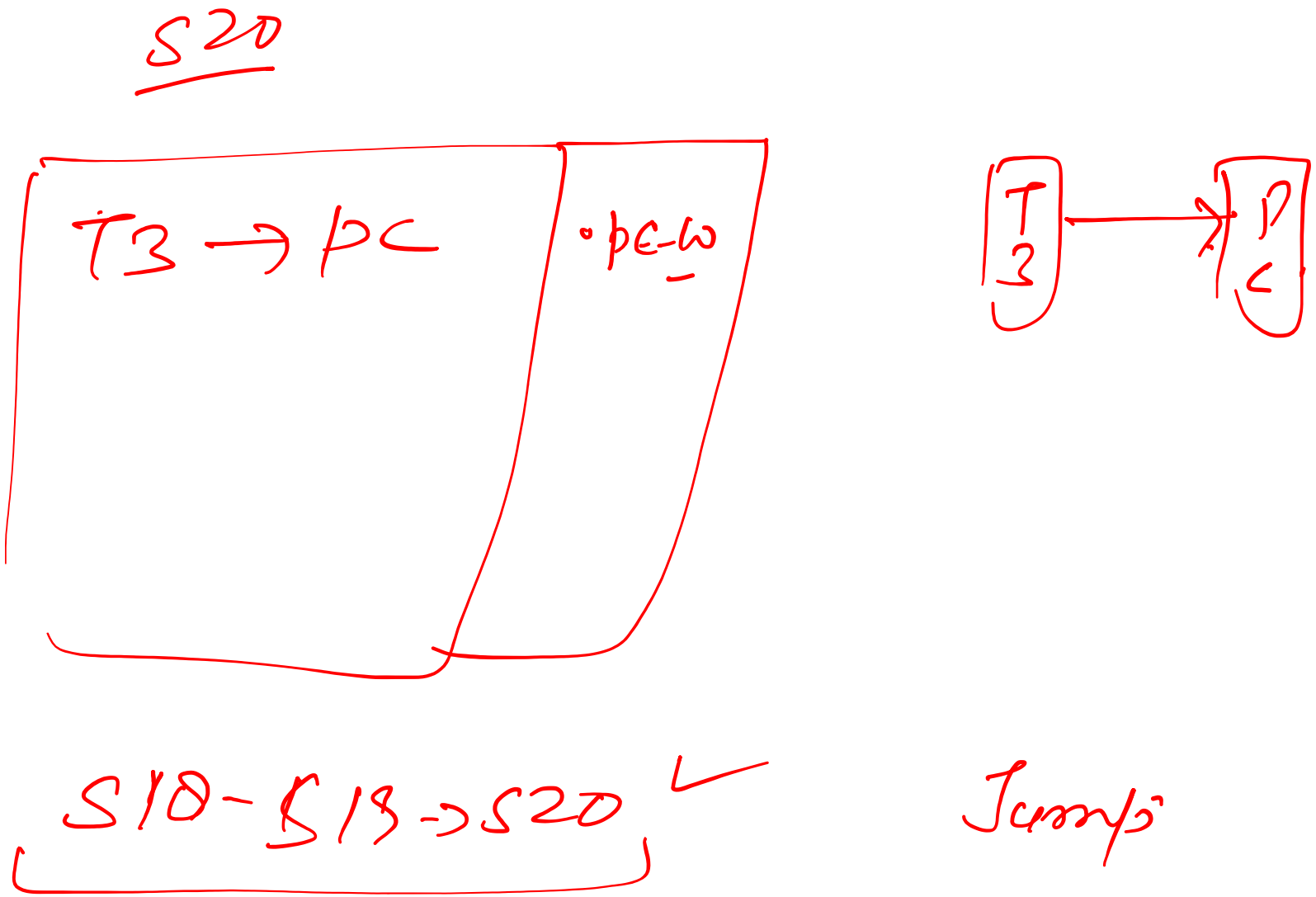
alu-c → t3

.

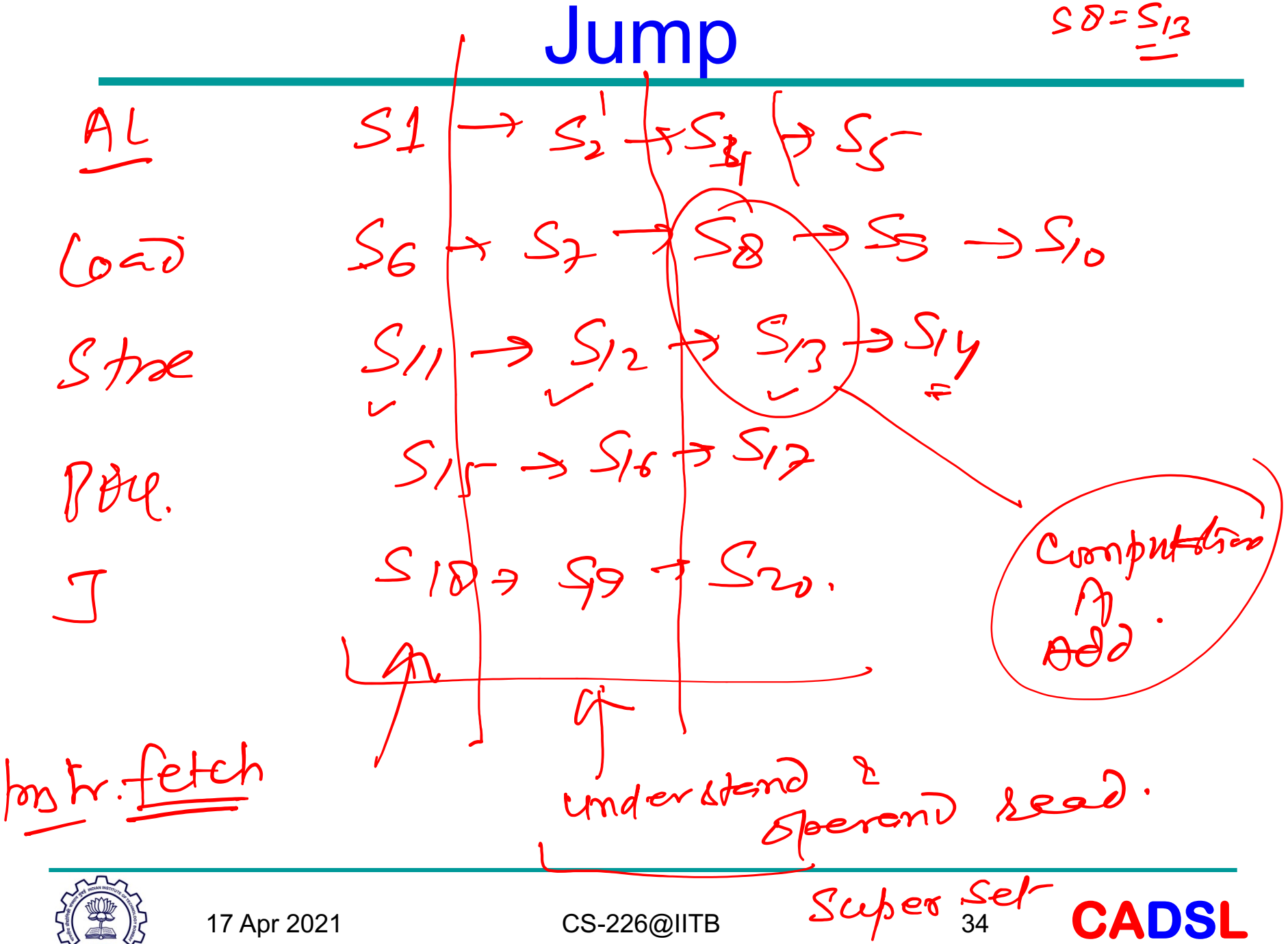
alu-Add



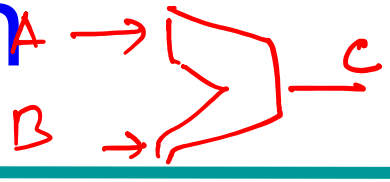
Jump



Jump


SD = S13
AL
S1 → S2' → S4 → S5
Load
S6 → S7 → S8 → S9 → S10
Store
S11 → S12 → S13 → S14
PUSH
S15 → S16 → S17
J
S18 → S19 → S20.
Computation in ALU.
Instr. fetch
understand & operand read.
Super set

State S1: Fetch



S1 → S2 → S3 → S4 → S5

State machine (TASK)

S1

PC → Mem_Add

Mem_Data → ir

PC → alu-A

+2 → alu-B

alu-C → PC

- MR
- PC_W
- IR_W
- alu-add

PC

Read

Add

Memory

PC

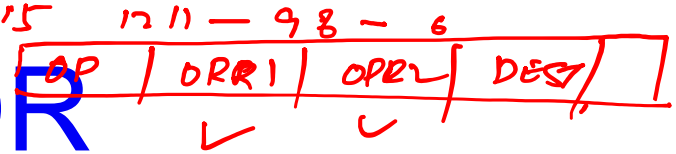
Add

+2

ALU

Instruction Register (IR)

State S2: OR



Mode

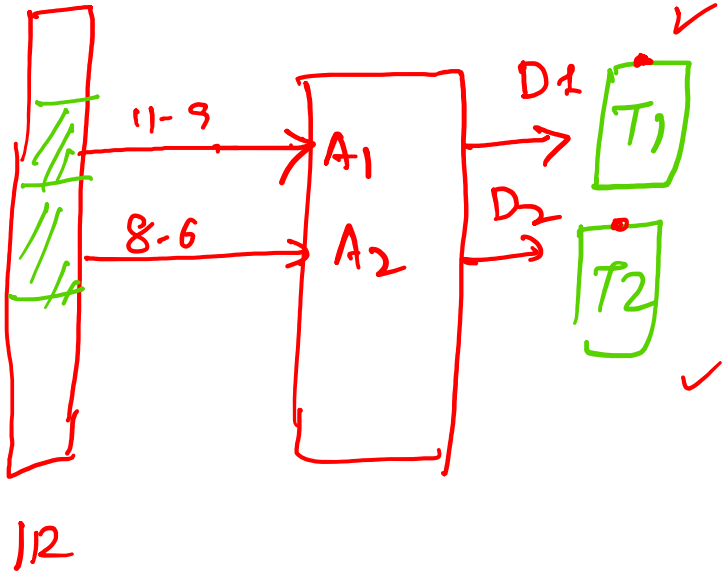
S_2 1 S_2 → S_2'

S_2^{-1}

t_1 write signal

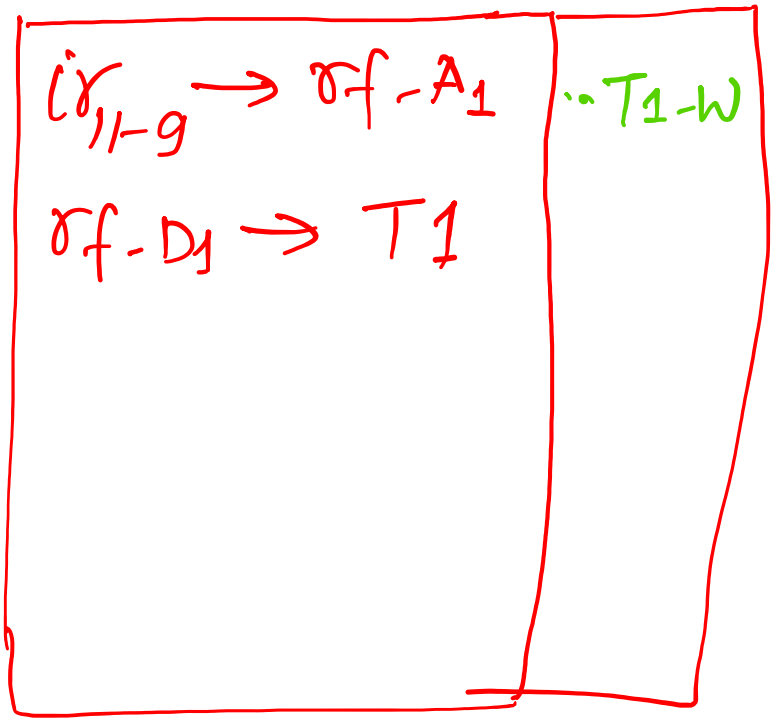
- $ir_{11-9} \rightarrow rf\text{-}A_1$
- $ir_{8-6} \rightarrow rf\text{-}A_2$
- $rf\text{-}D_1 \rightarrow t1$
- $rf\text{-}D_2 \rightarrow t2$

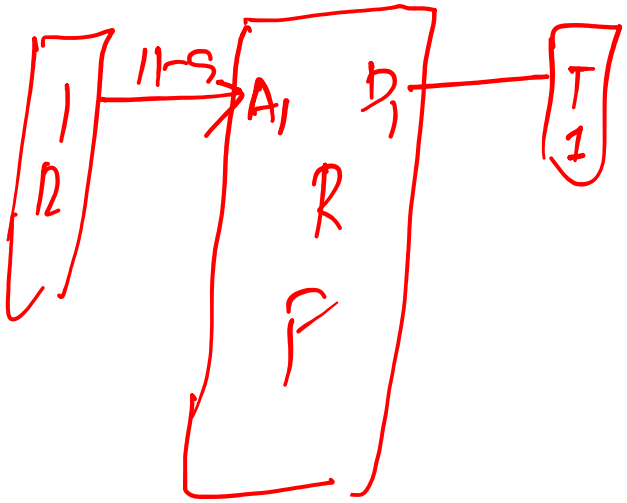
- t_1_w
- t_2_w



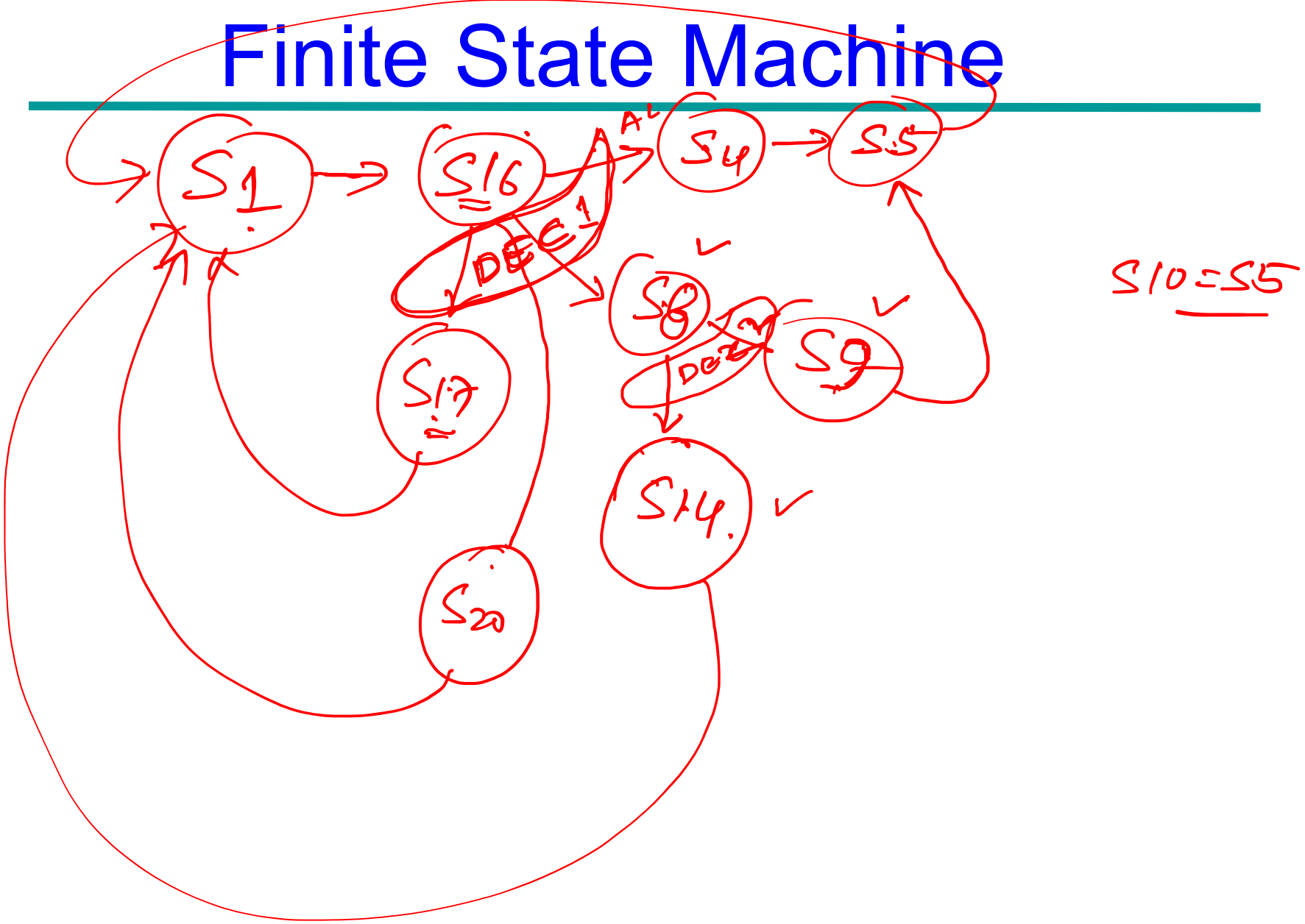
Instruction

S7

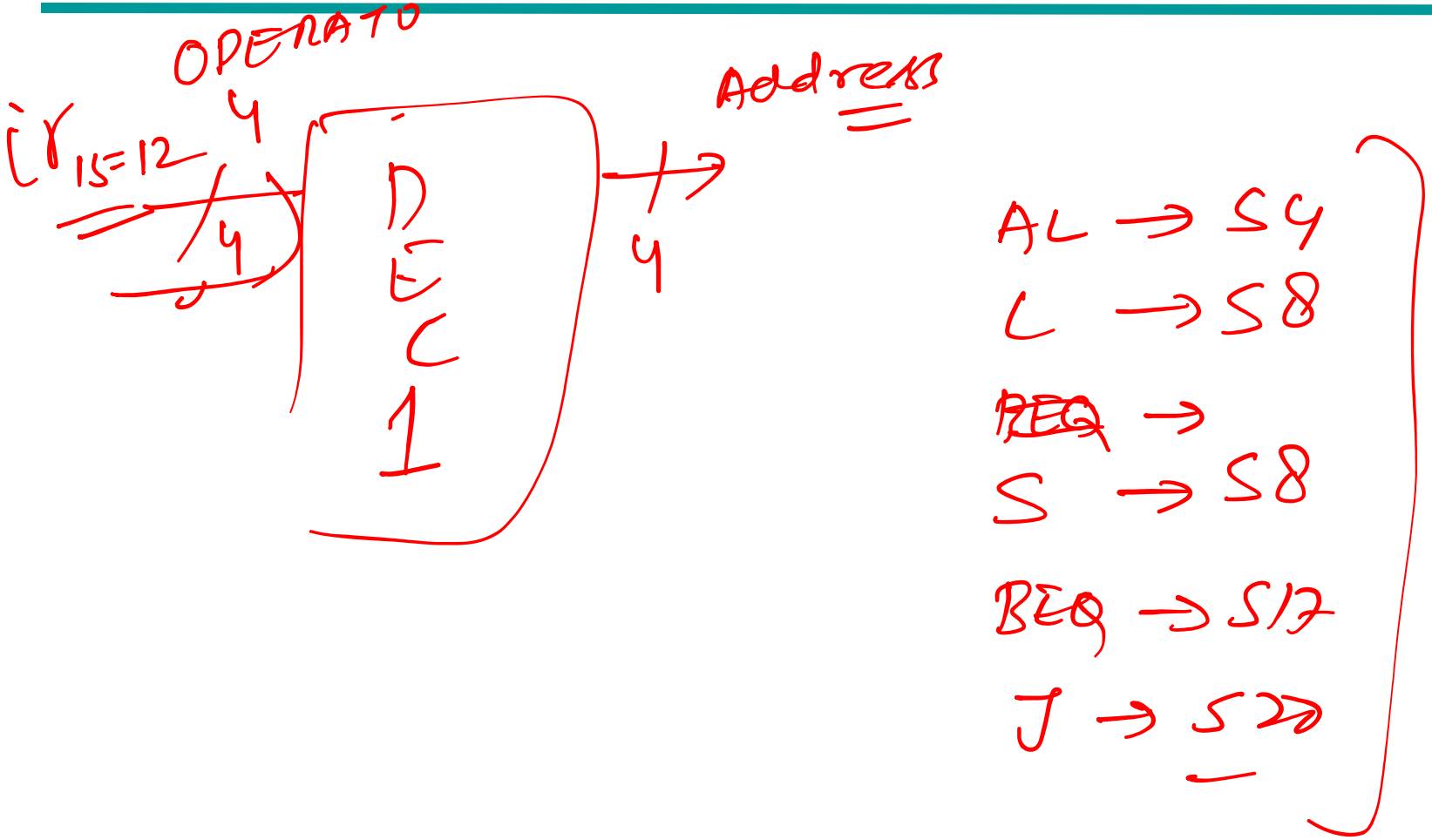




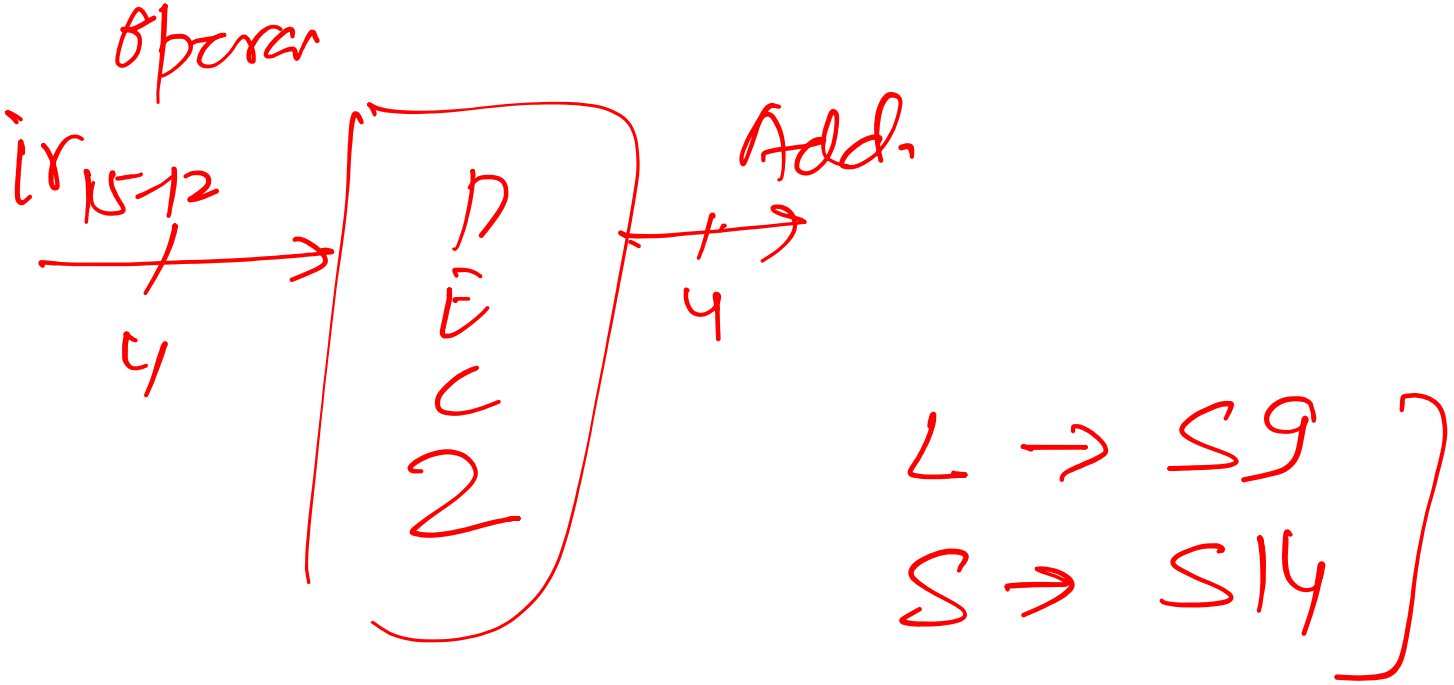
Finite State Machine



Finite State Machine



Finite State Machine

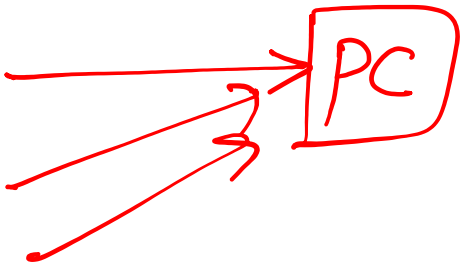


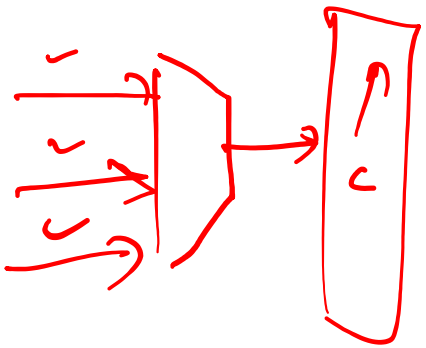
Finite State Machine

Control points

Write signal to Reg

Memory Read & Write





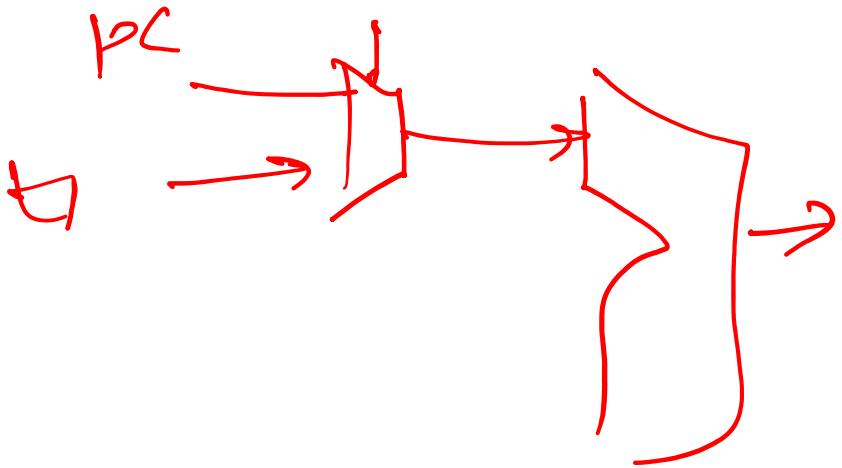
→ PC

Finite State Machine

ALU-A

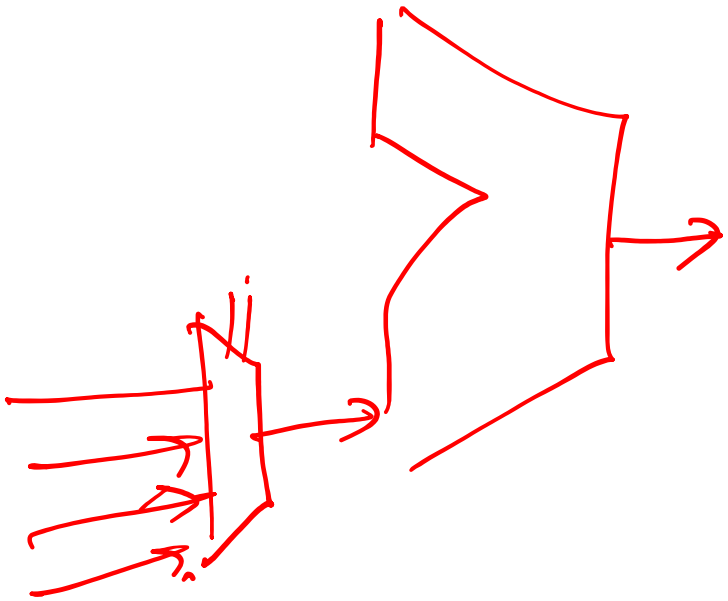
pc → alu-A

t1 → alu-A



Finite State Machine

alu-B → alu-B



t_2 → alu-B

+2 → alu-B

SE6 → alu-B

SE12 → alu-B

Finite State Machine

CONTROL POINTS

State machine

1. Next State
2. Control points

Thank You