RISC Design

Virendra Singh

Professor

Computer Architecture and Dependable Systems Lab

Department of Electrical Engineering, and

Dept. of Computer Science & Engineering

Indian Institute of Technology Bombay

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

E-mail: viren@{ee, cse}.iitb.ac.in

EE-739: Processor Design @ IITB

Lecture 13 (11 February 2022)

CADSL

RISC Design

ISA → Simple instruction

& Fixed encoding ✓

(Boundary)

$S \xrightarrow{I} S'$

State transition

Content of memory + contents of programmer's register

Combinational logic.

O.T + HKT

Instruction set of DLX:

32 bit format

3 formats: R, I, J

A/L, L/S, J

Reg-reg: R

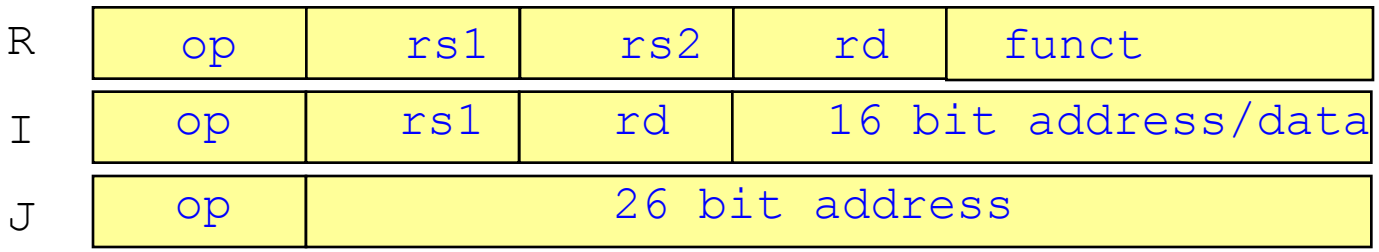
A/L: I

L/S: BtD, BtI

J: cond: I; uncond: J

Overview of DLX ISA

- simple instructions, all 32 bits wide
- very structured, no unnecessary baggage
- only three instruction formats



- rely on compiler to achieve performance

Example Instruction Set: MIPS Subset

MIPS Instruction – Subset

- Arithmetic and Logical Instructions
  - add, sub, or, and, slt
- Memory reference Instructions
  - lw, sw
- Branch
  - beq, j

Data path → FU ← ALU

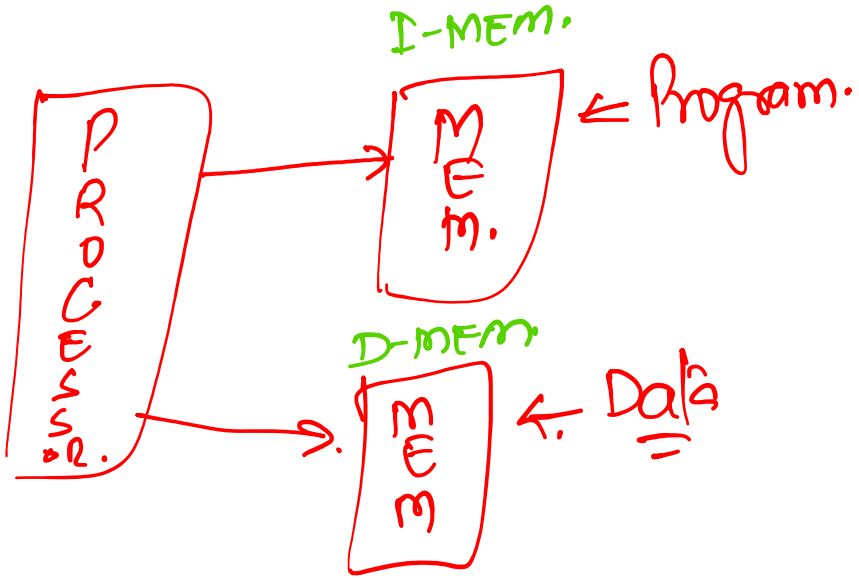
Storage → Programmer's reg.

PC / IP

temporary Registers

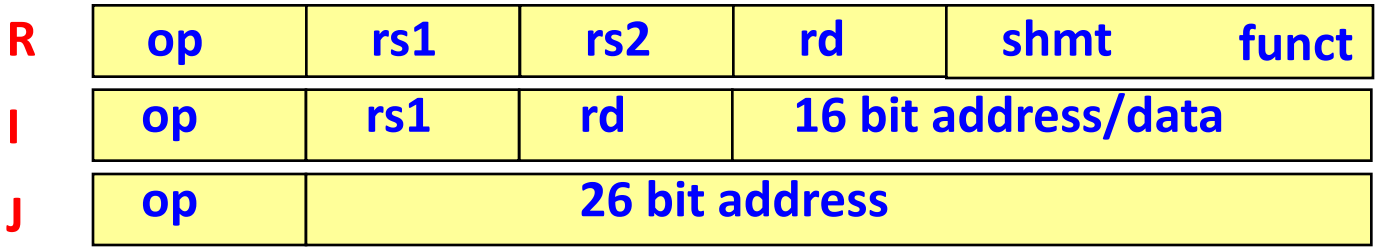
Steering logic → Point-to-point

mux based communi catin.



Overview of MIPS

- simple instructions, all 32 bits wide
- very structured, no unnecessary baggage
- only three instruction formats



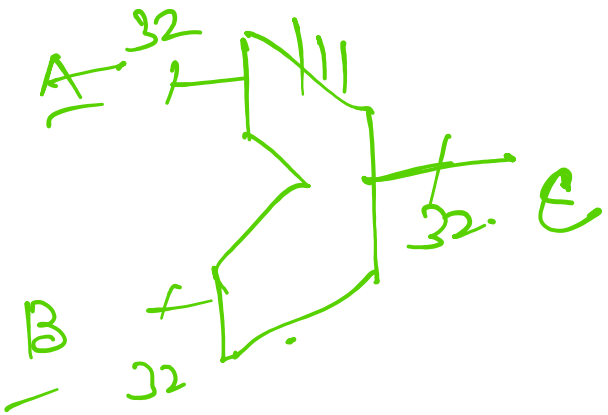
- rely on compiler to achieve performance

Datapath and Control

- Datapath: Memory, registers, adders, ALU, and communication buses. Each step (fetch, decode, execute) requires communication (data transfer) paths between memory, registers and ALU.
- Control: Datapath for each step is set up by control signals that set up dataflow directions on communication buses and select ALU and memory functions. Control signals are generated by a control unit consisting of one or more finite-state machines.

Datapath

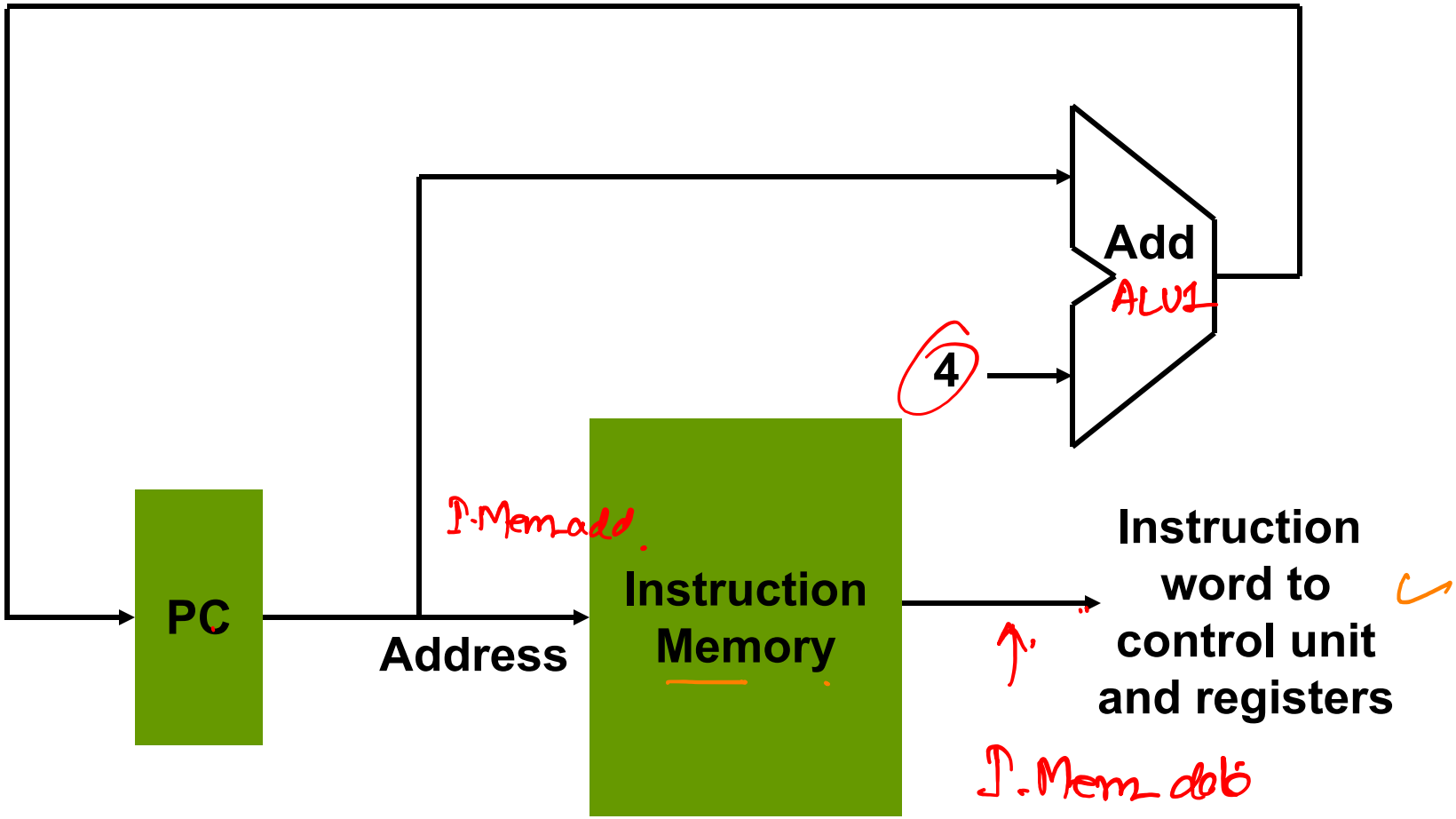
ALU: add/sub/and/or/shift



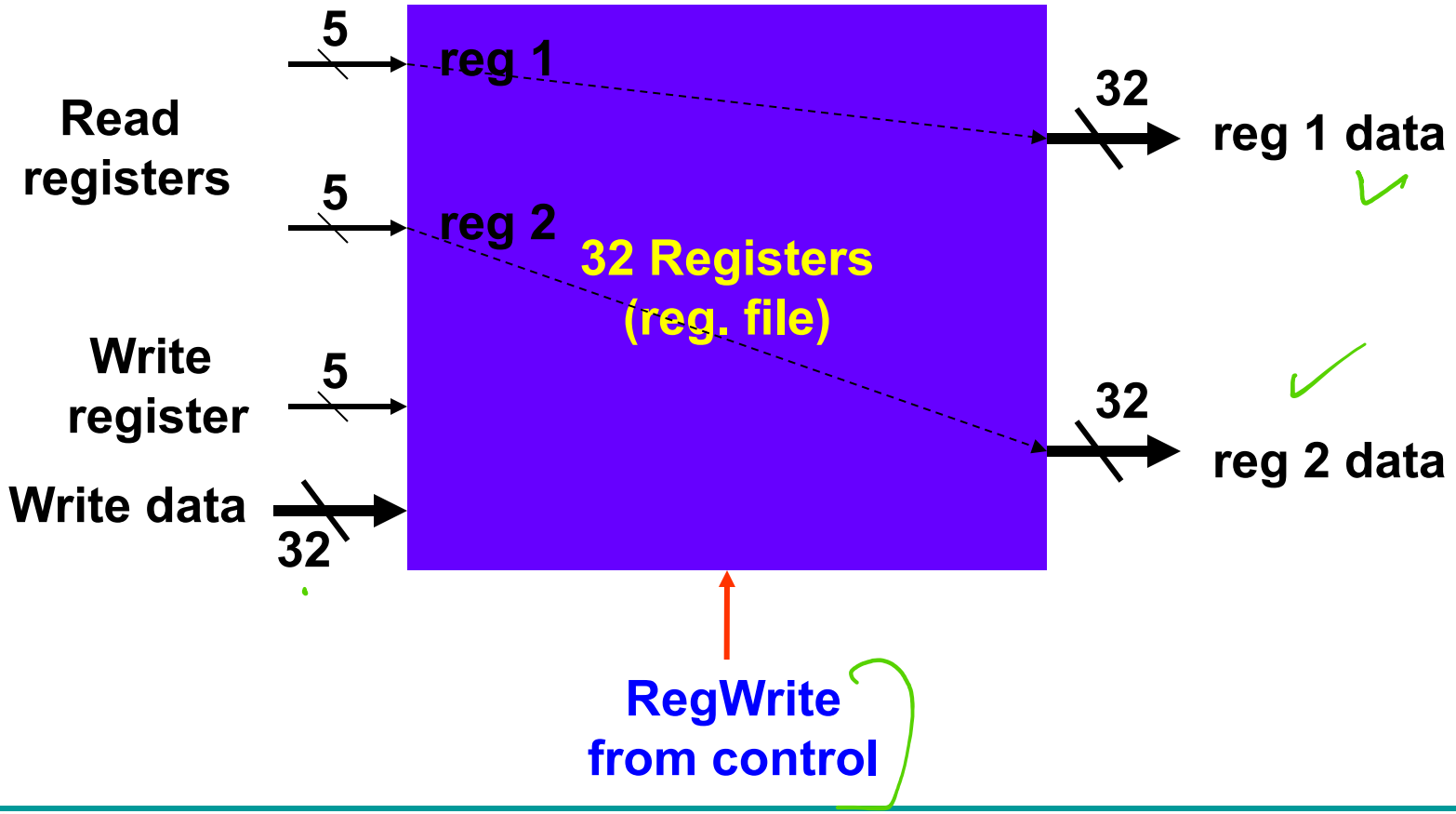
Programmer's Register } ← Set of registers

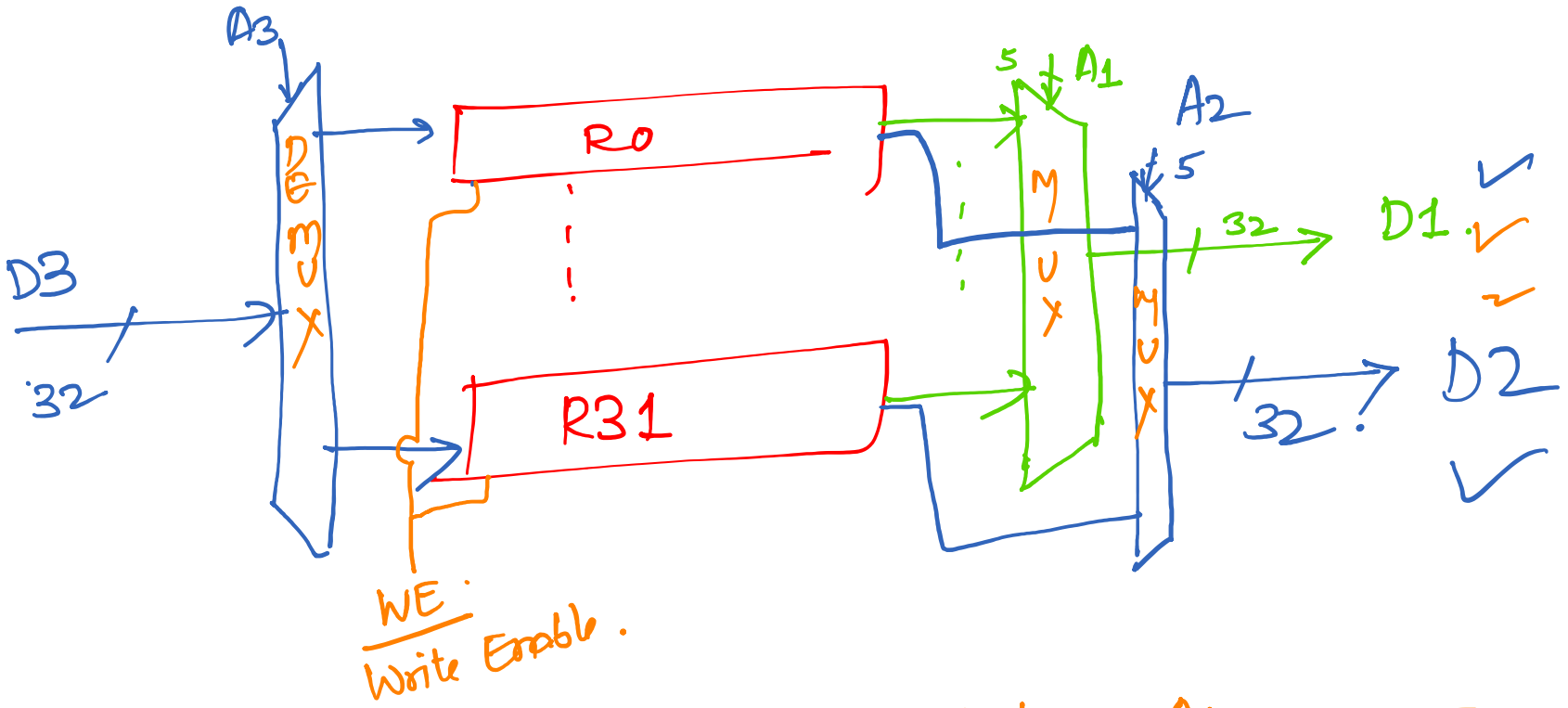
Register File.

Datapath for Instruction Fetch

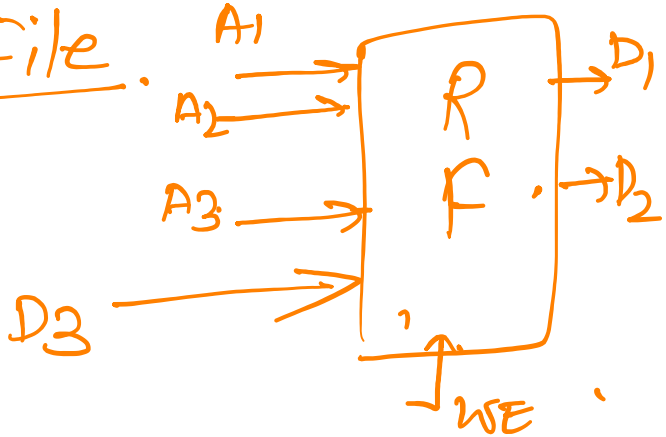


Register File: A Datapath Component



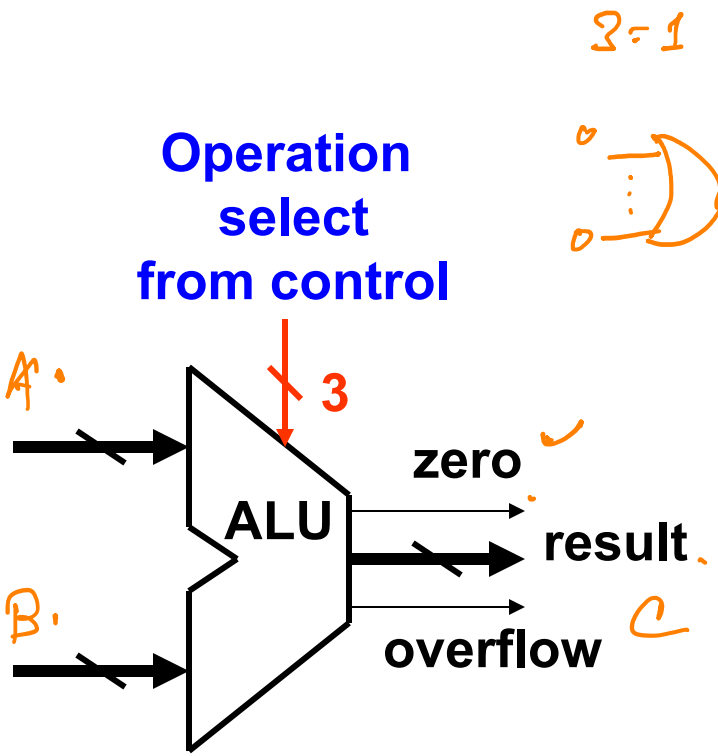


Register File.



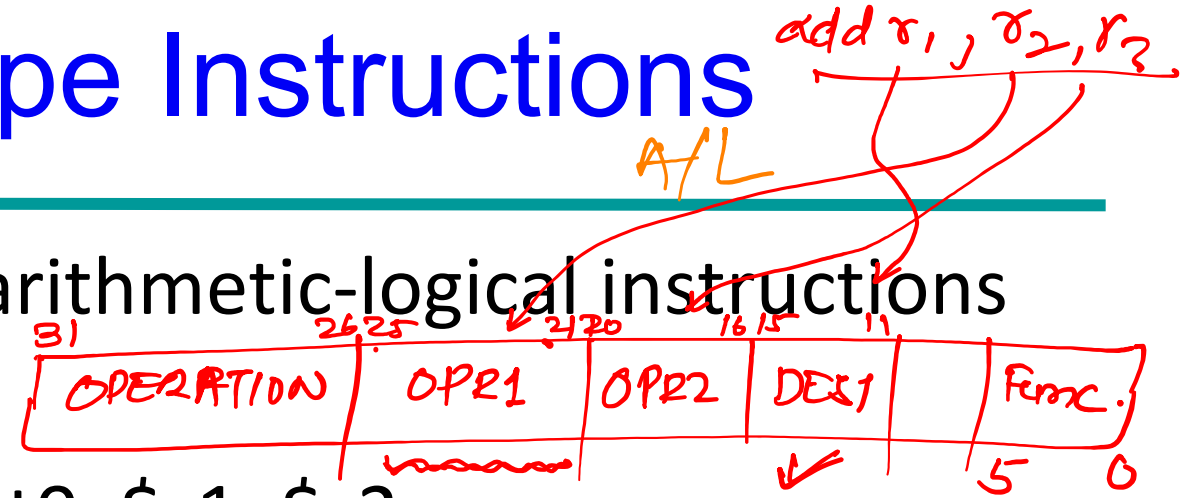
Multi-Operation ALU

Operation select	ALU function
000	AND
001	OR
010	Add
110	Subtract
111	Set on less than



zero = 1, when all bits of result are 0

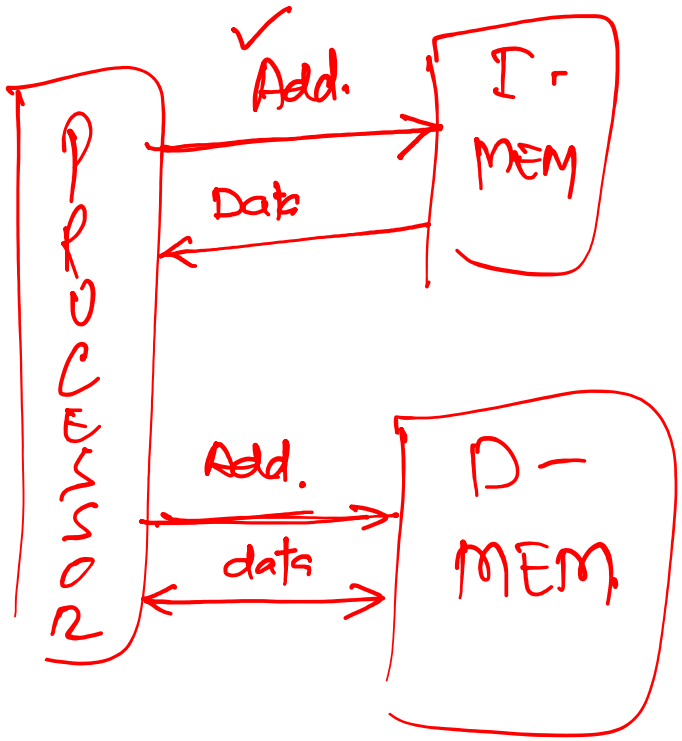
R-Type Instructions



- Also known as arithmetic-logical instructions
- add, sub, slt ⇒
- Example: add $t0, $s1, $s2
  - Machine instruction word

    000000 10001 10010 01000 00000 100000

    opcode $s1 $s2 $t0 function
  - Read two registers
  - Write one register
  - Opcode and function code go to control unit that generates RegWrite and ALU operation code.



HKT + OT

Instruction should be fetch

understand

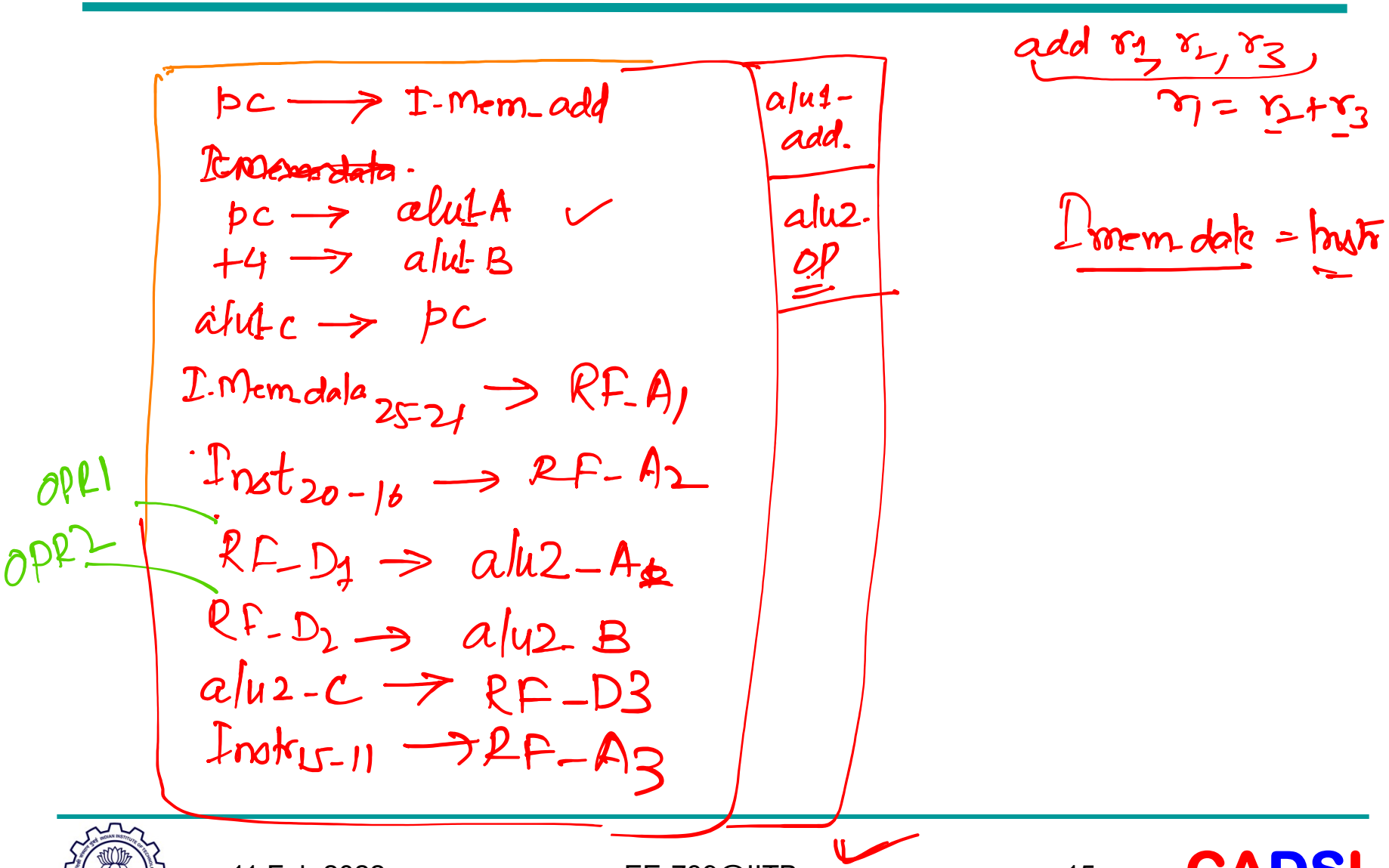
Read operands

Execute

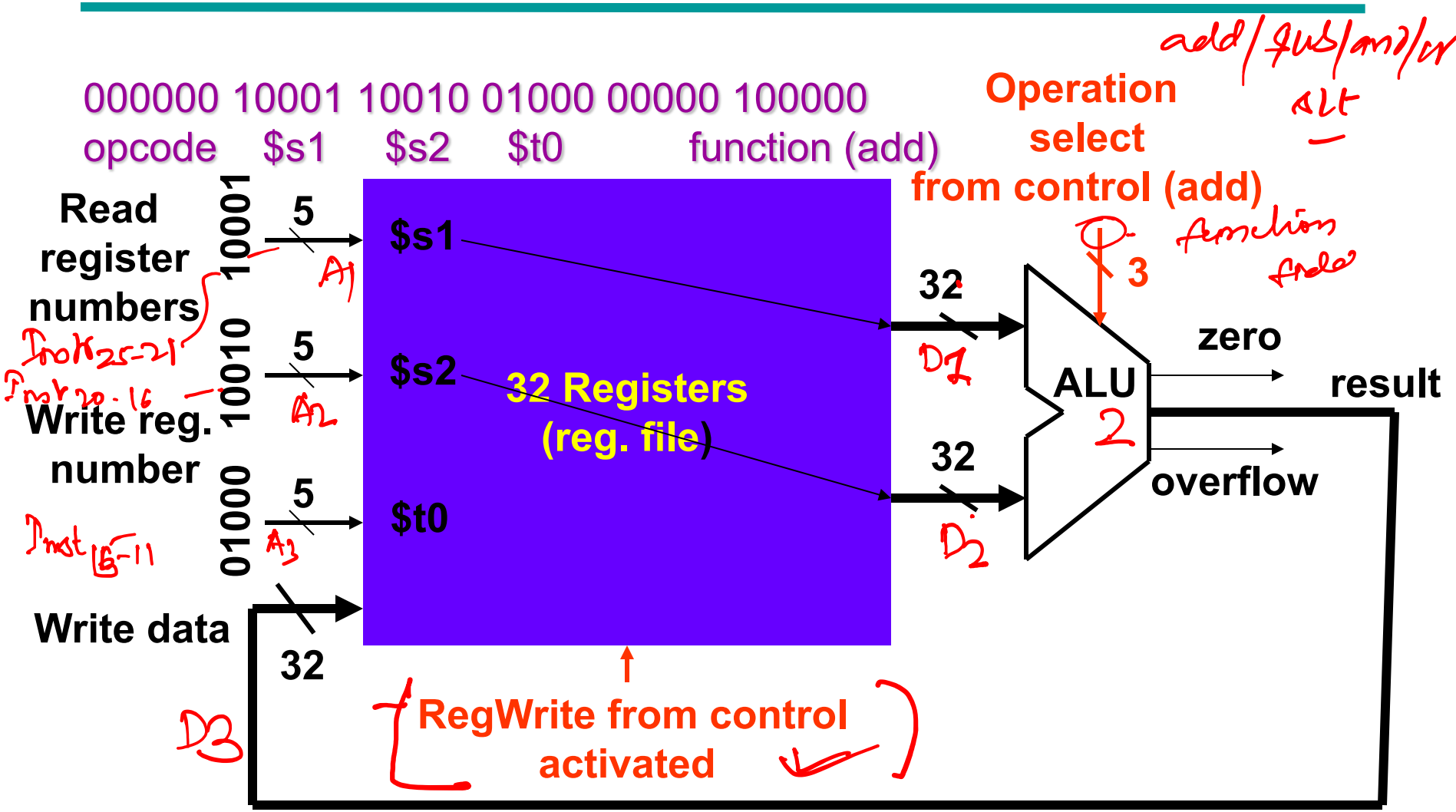
Update the state

R-Type Instructions

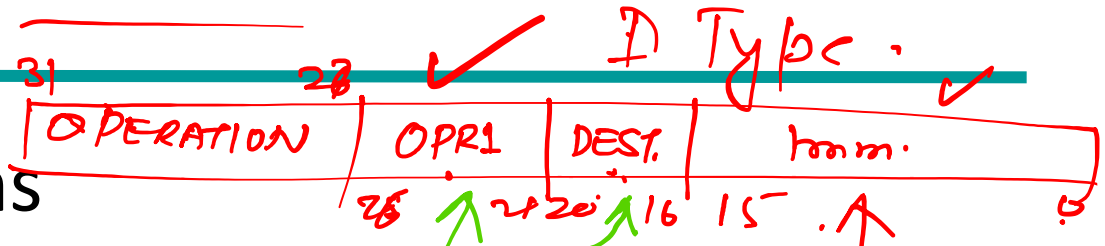
HFC



Datapath for R-Type Instruction



Load and Store Instructions



- I-type instructions
- lw $t0, 1200 ($t1) # incr. in bytes

  100011 01001 01000 0000 0100 1011 0000

  opcode $t1 $t0 1200

- sw $t0, 1200 ($t1) # incr. in bytes

  101011 01001 01000 0000 0100 1011 0000

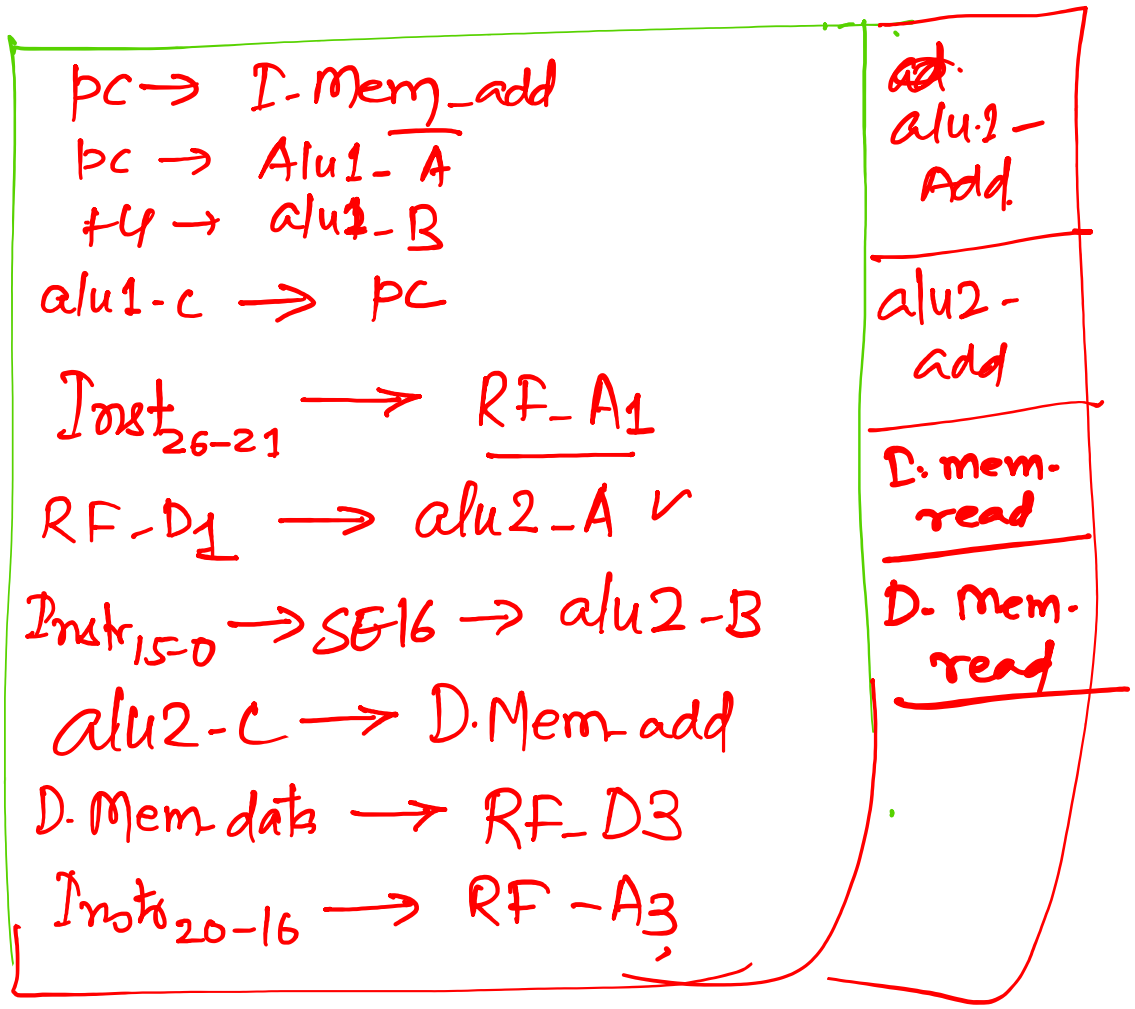
  opcode $t1 $t0 1200

$R_0 = M[r_1 + 200]$

load r0, (r1) 200

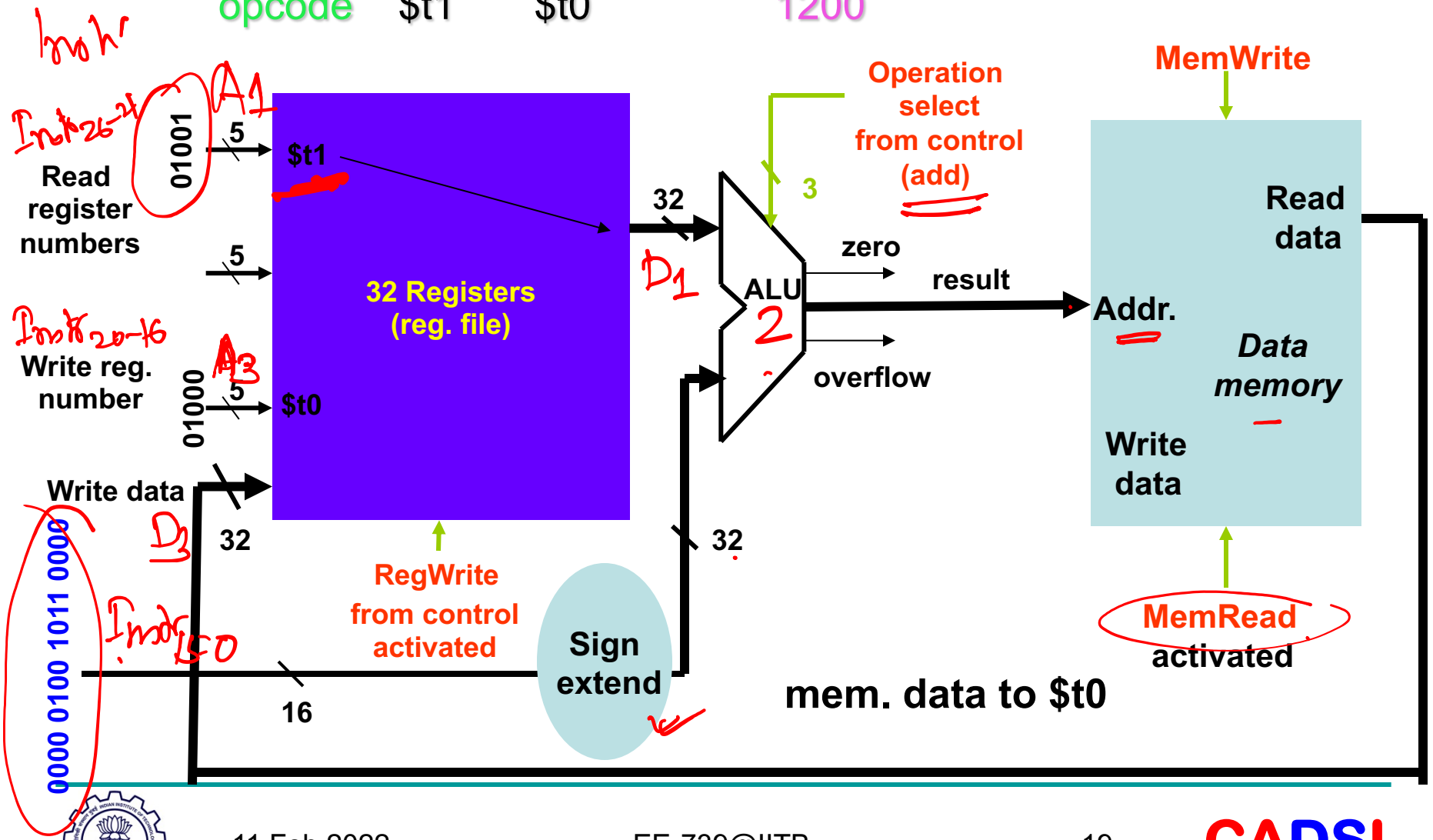
Load and Store Instructions

(OA1)

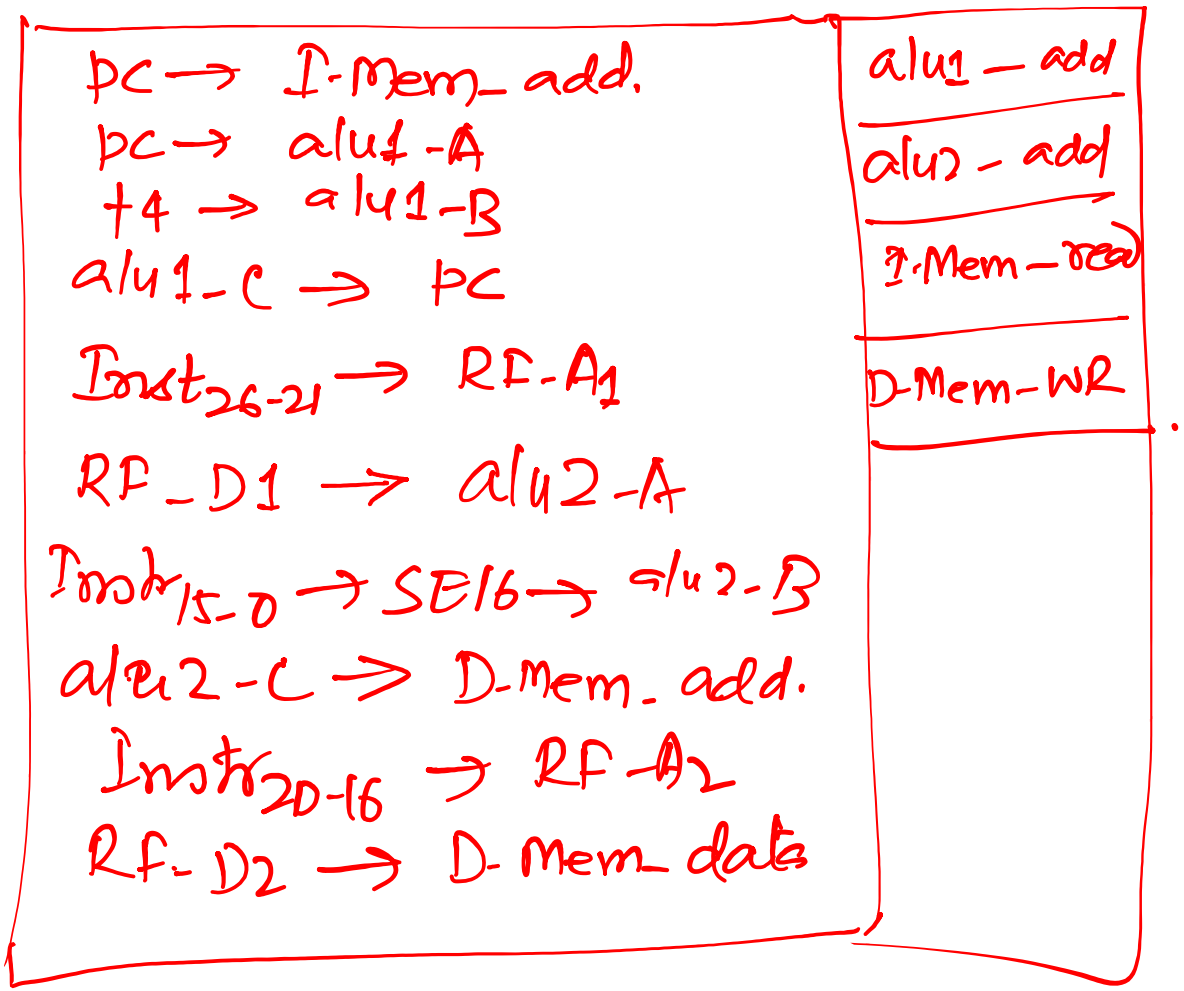


Datapath for lw Instruction

100011 01001 01000 0000 0100 1011 0000
opcode $t1 $t0 1200

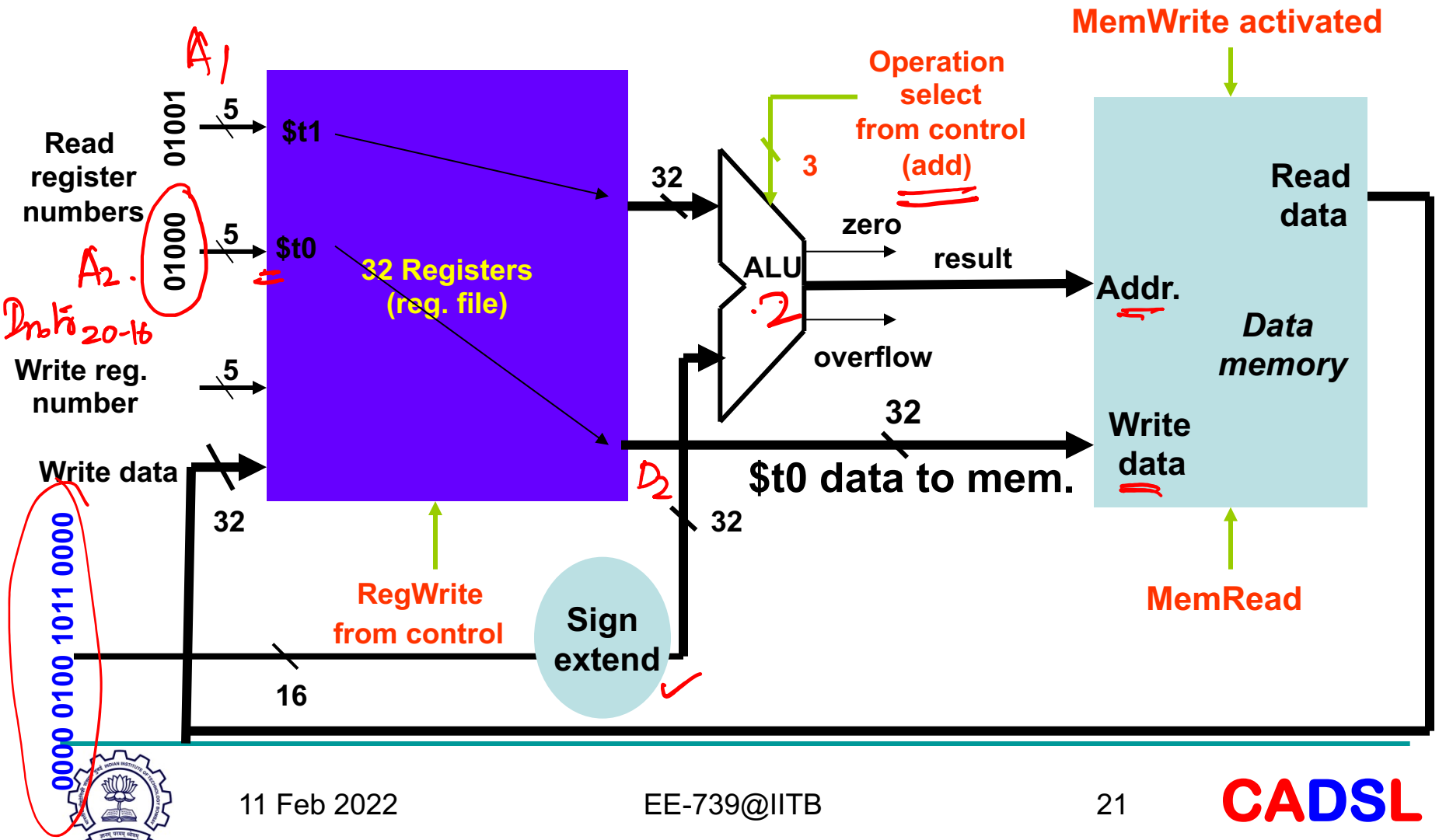


Load and Store Instructions



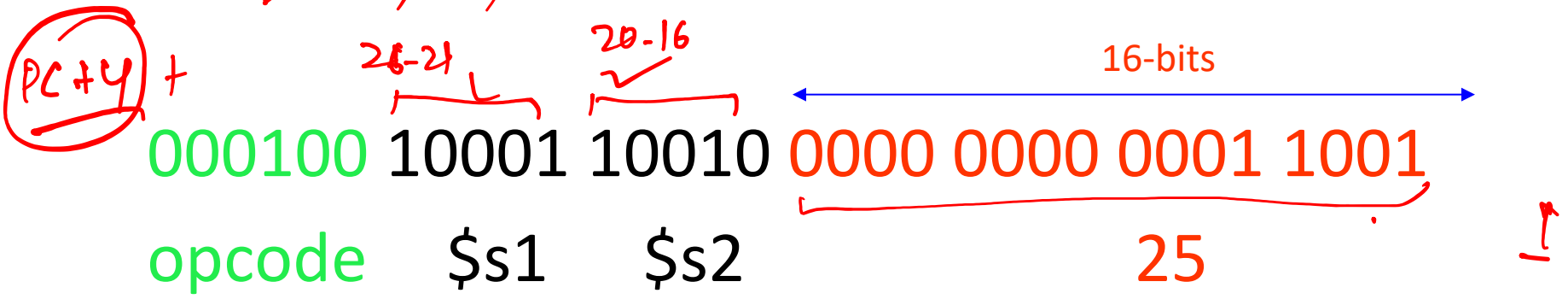
Datapath for sw Instruction

101011 01001 01000 0000 0100 1011 0000
opcode $t1 $t0 1200



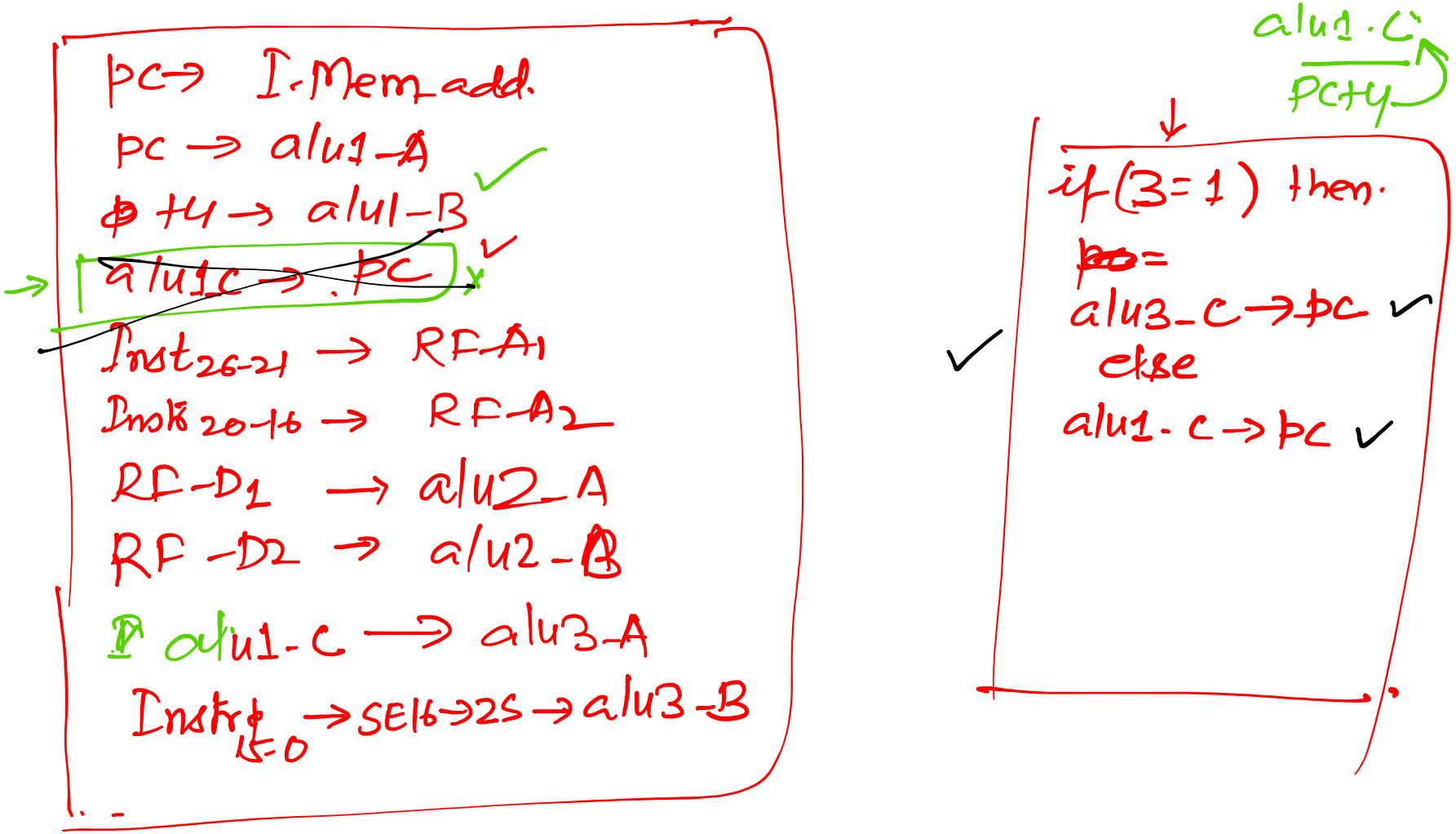
Branch Instruction (I-Type)

- beq $s1, $s2, 25 # if $s1 = $s2, advance PC through 25 instructions

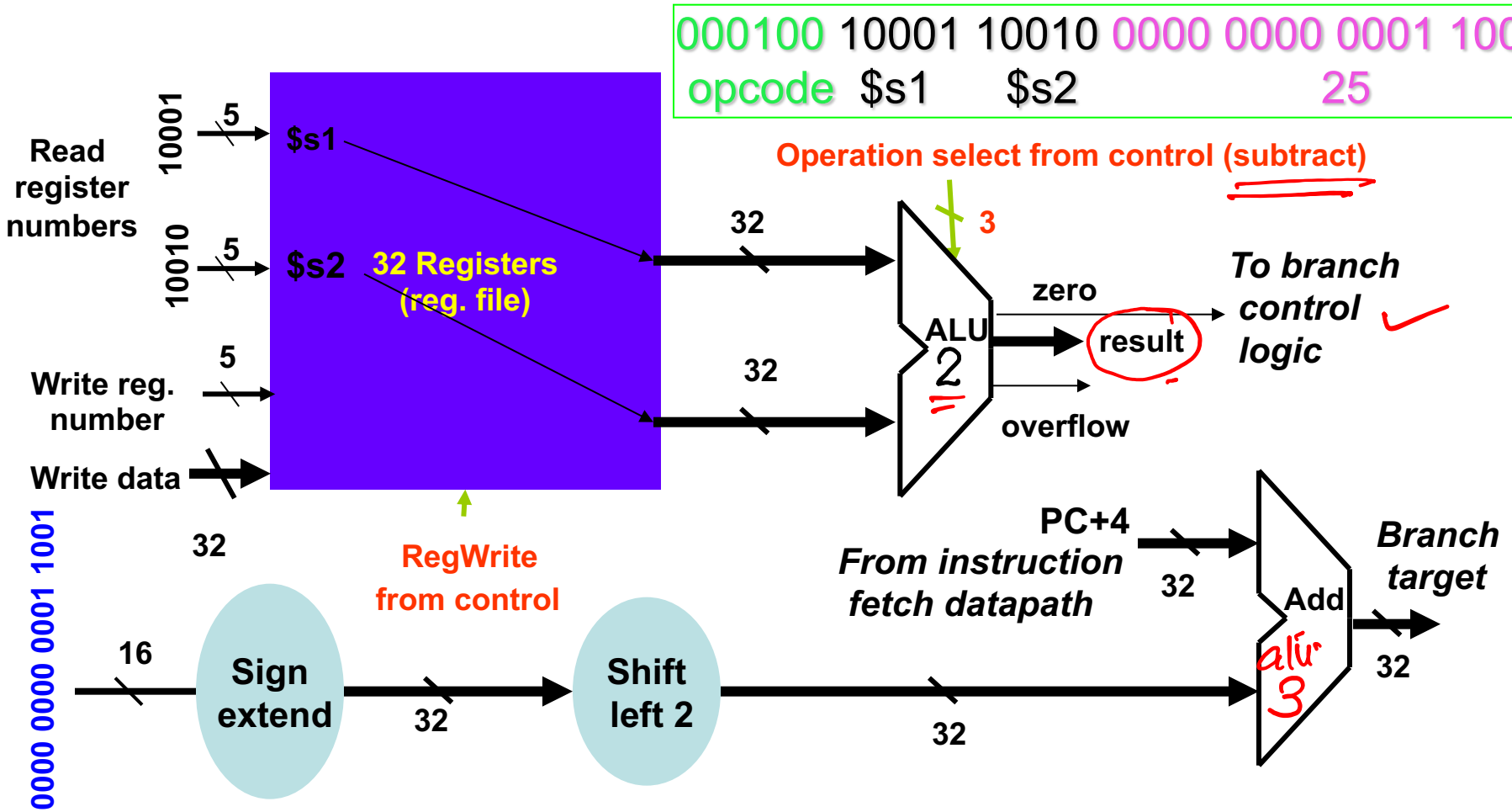


Note: Can branch within ± 2^{15} words from the current instruction address in PC.

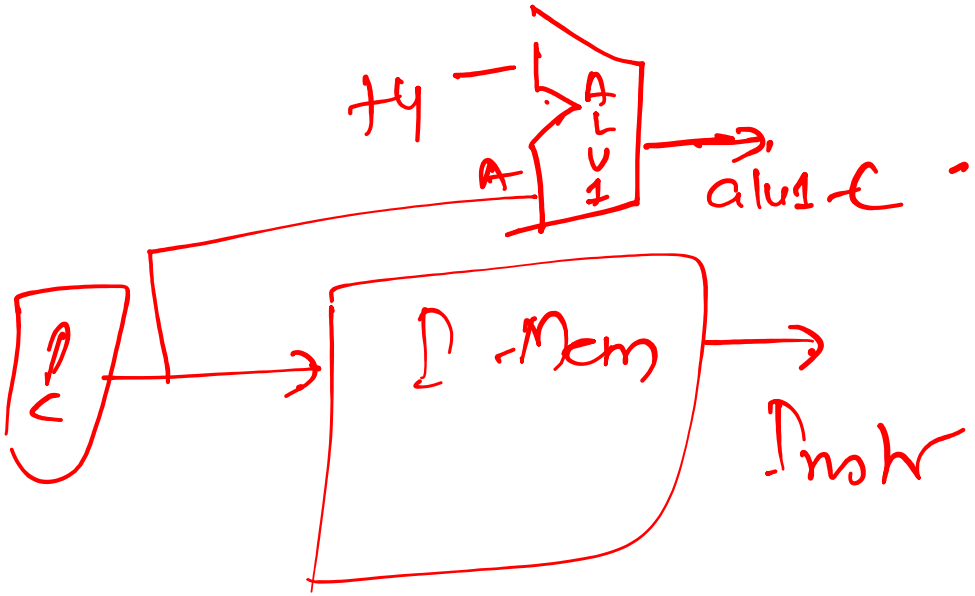
Branch Instructions



Datapath for beq Instruction

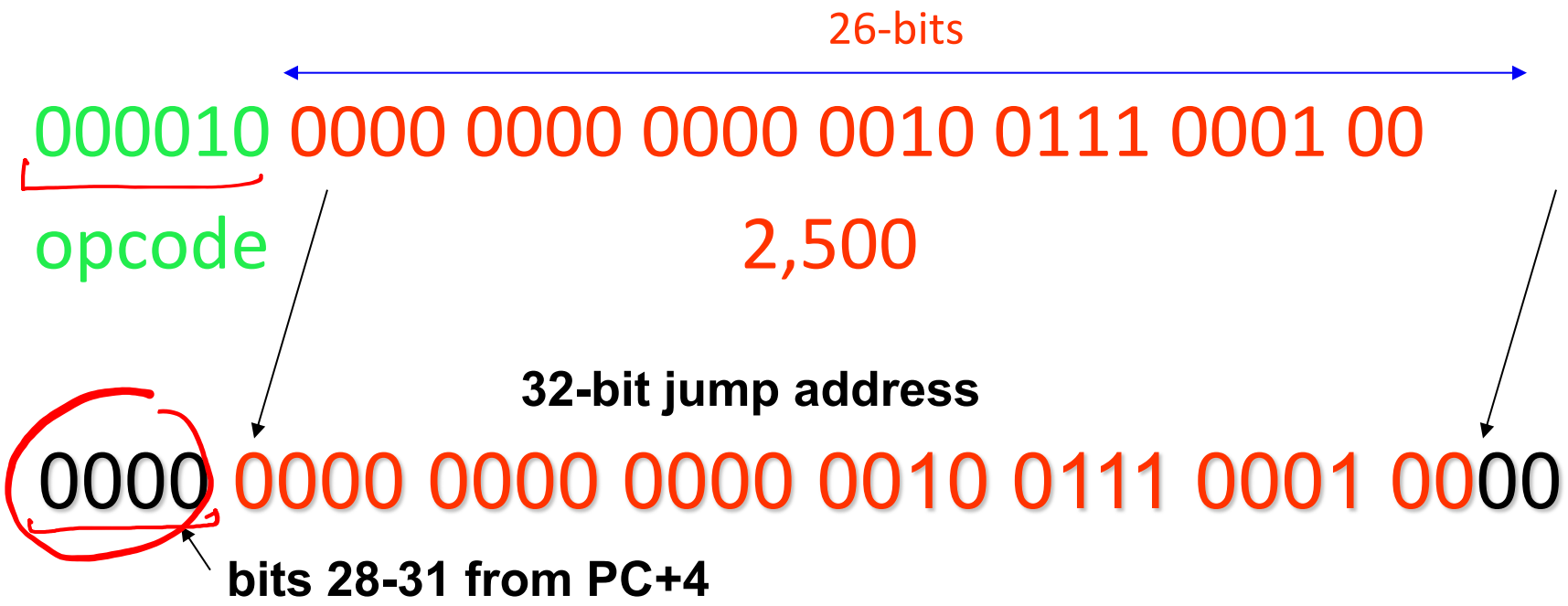


Branch Instructions



J-Type Instruction

- j 2500 # jump to instruction 2,500



pc → I-mem-add

pc → alu1-A

Φ i+4 → alu1-B

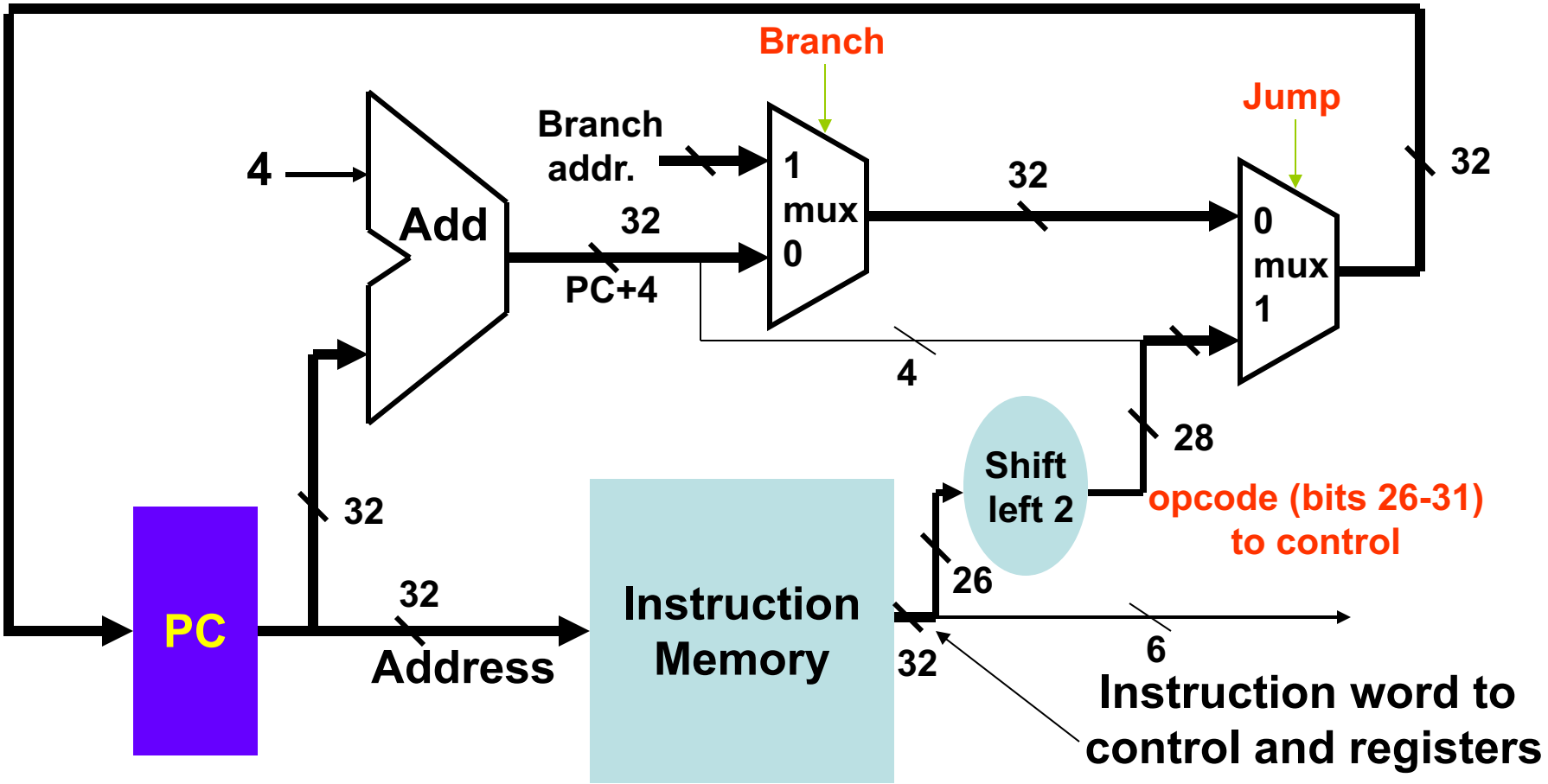
~~Inst.~~

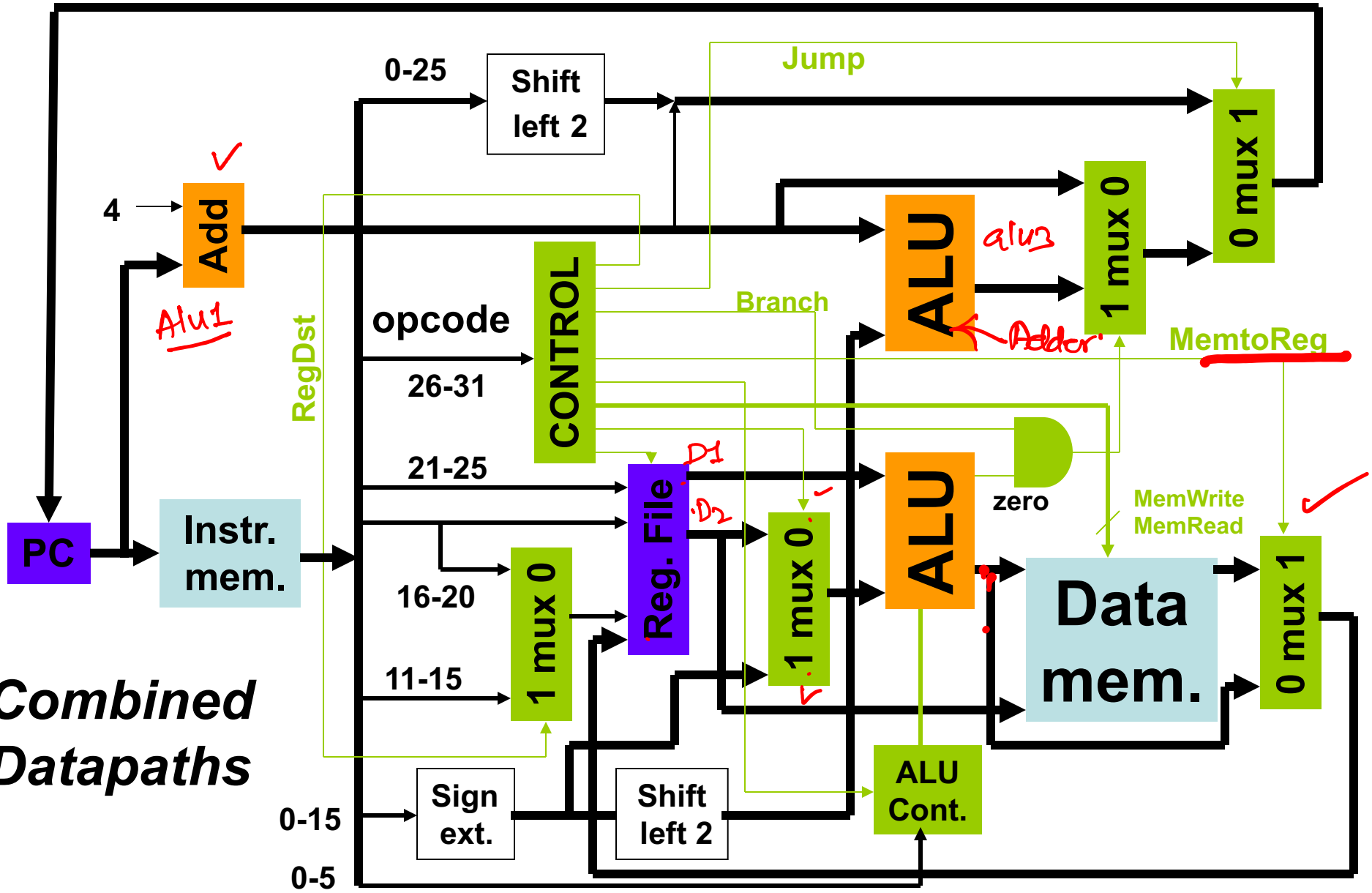
alu1-C → alu3-A

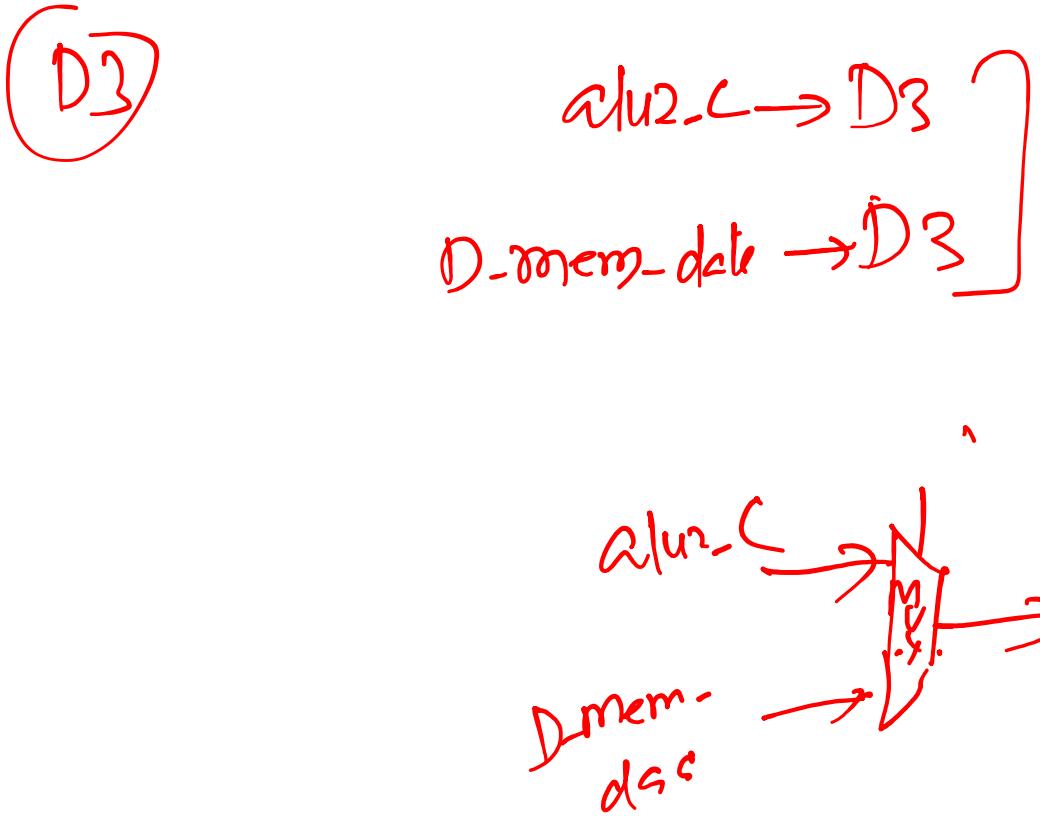
$Instr_{25-0}$ → SE26 $\xrightarrow{-25}$ alu3-B.

alu3-C → PC

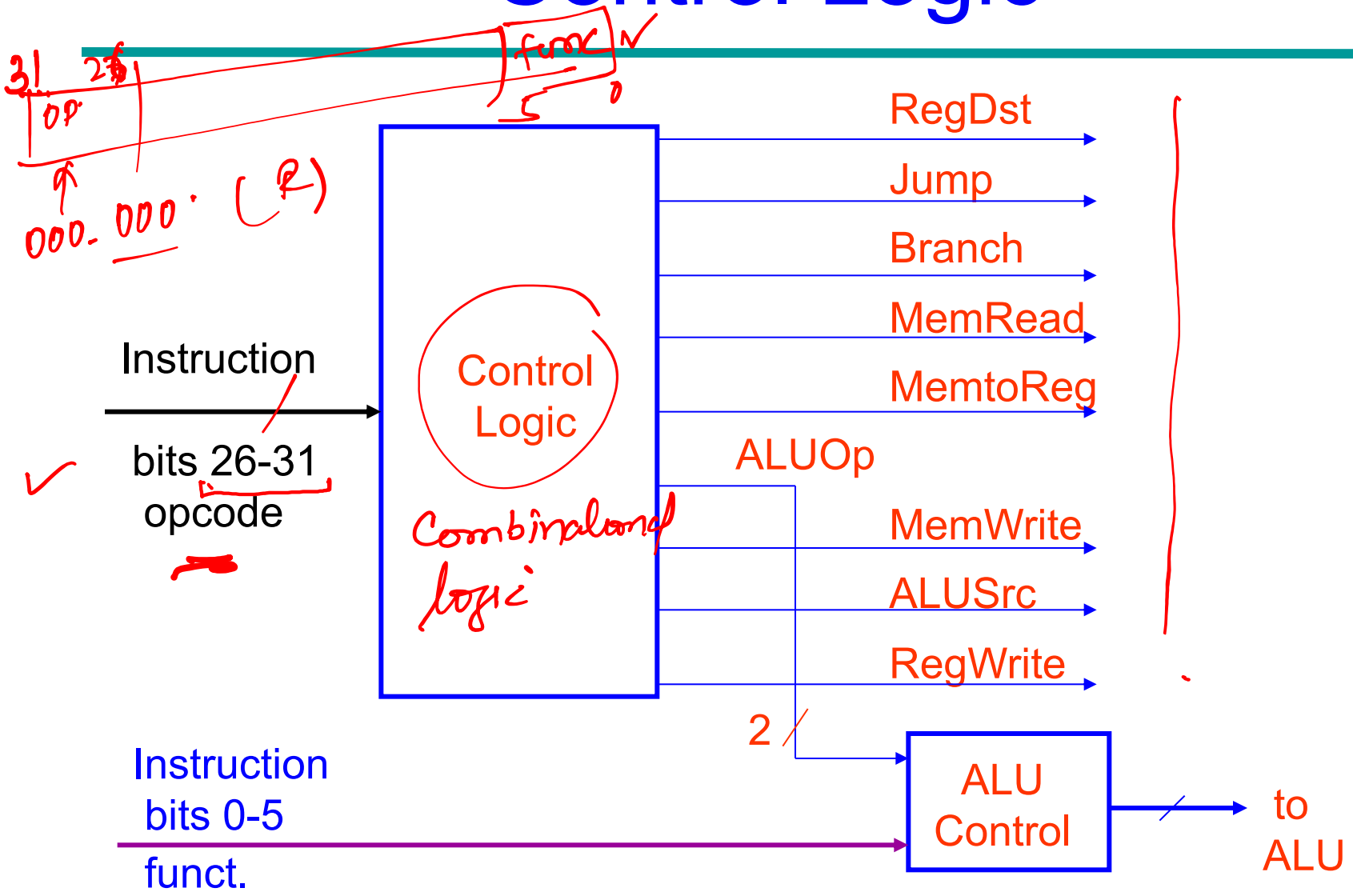
Datapath for Jump Instruction




Combined Datapaths
PC
Instr. mem.
Add
4
Alu1
RegDst
0-25
Shift left 2
Jump
opcode
26-31
CONTROL
Branch
21-25
16-20
11-15
1 mux 0
Reg. File
D1
D2
ALU
alu3
Adder
1 mux 0
0 mux 1
MemtoReg
zero
MemWrite
MemRead
Data mem.
0 mux 1
1 mux 0
ALU Cont.
0-15
Sign ext.
Shift left 2
0-5


D3
alu2_C → D3
D_mem_data → D3
alu_C
D_mem_data

Control Logic

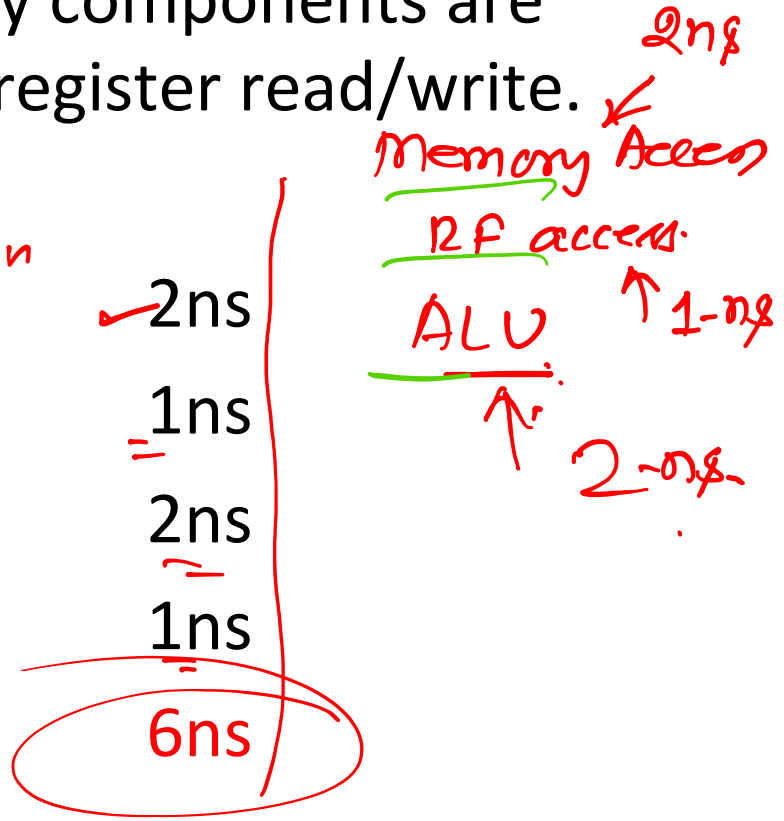


Control Logic: Truth Table

Instr type	Inputs: instr. opcode bits						Outputs: control signals									
	31	30	29	28	27	26	RegDst	Jump	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp2
R	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
lw	1	0	0	0	1	1	0	0	1	1	1	1	0	0	0	0
sw	1	0	1	0	1	1	X	0	1	X	0	0	1	0	0	0
beq	0	0	0	1	0	0	X	0	0	X	0	0	0	1	0	1
j	0	0	0	0	1	0	X	1	X	X	X	X	X	X	X	X

How Long Does It Take?

- Assume control logic is fast and does not affect the critical timing. Major time delay components are ALU, memory read/write, and register read/write.
- Arithmetic-type (R-type)
  - Fetch (memory read) 2ns
  - Register read 1ns
  - ALU operation 2ns
  - Register write 1ns
  - Total 6ns



Time for lw and sw (I-Types)

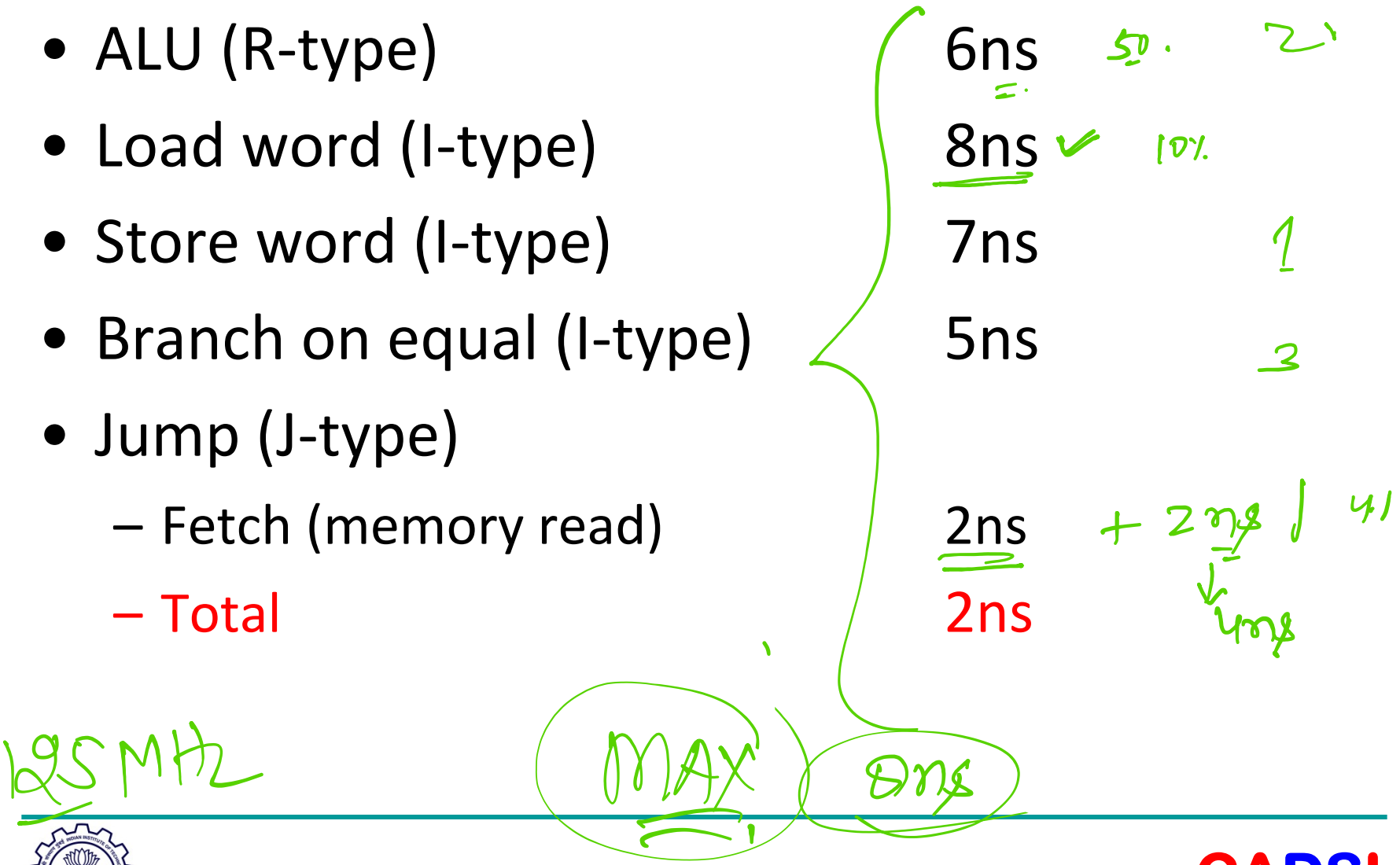
- ALU (R-type) 6ns
- Load word (I-type)
  - Fetch (memory read) 2ns
  - Register read 1ns
  - ALU operation 2ns
  - Get data (mem. Read) 2ns
  - Register write 1ns
  - Total 8ns
- Store word (no register write) 7ns

Time for beq (I-Type)

- ALU (R-type) 6ns
- Load word (I-type) 8ns
- Store word (I-type) 7ns
- Branch on equal (I-type)
  - Fetch (memory read) 2ns
  - Register read 1ns
  - ALU operation 2ns
  - Total 5ns

Time for Jump (J-Type)

- ALU (R-type) — 6ns
- Load word (I-type) — 8ns
- Store word (I-type) — 7ns
- Branch on equal (I-type) — 5ns
- Jump (J-type)
  - Fetch (memory read) — 2ns
  - Total — 2ns



How Fast Can the Clock Be?

- If every instruction is executed in one clock cycle, then:
  - Clock period must be at least 8ns to perform the longest instruction, i.e., *lw*.
  - This is a single cycle machine (CPI = 1)
  - It is slower because many instructions take less than 8ns but are still allowed that much time.
- Method of speeding up: Use multicycle datapath.

Thank You