

RISC Design

Introduction

Virendra Singh

Associate 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

EE-739: Processor Design



Lecture 15 (12 Feb 2015)

CADSL

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



Overview of MIPS

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

R	op	rs1	rs2	rd	shmt	funct
I	op	rs1	rd	16 bit address/data		
J	op	26 bit address				

- ❖ rely on compiler to achieve performance

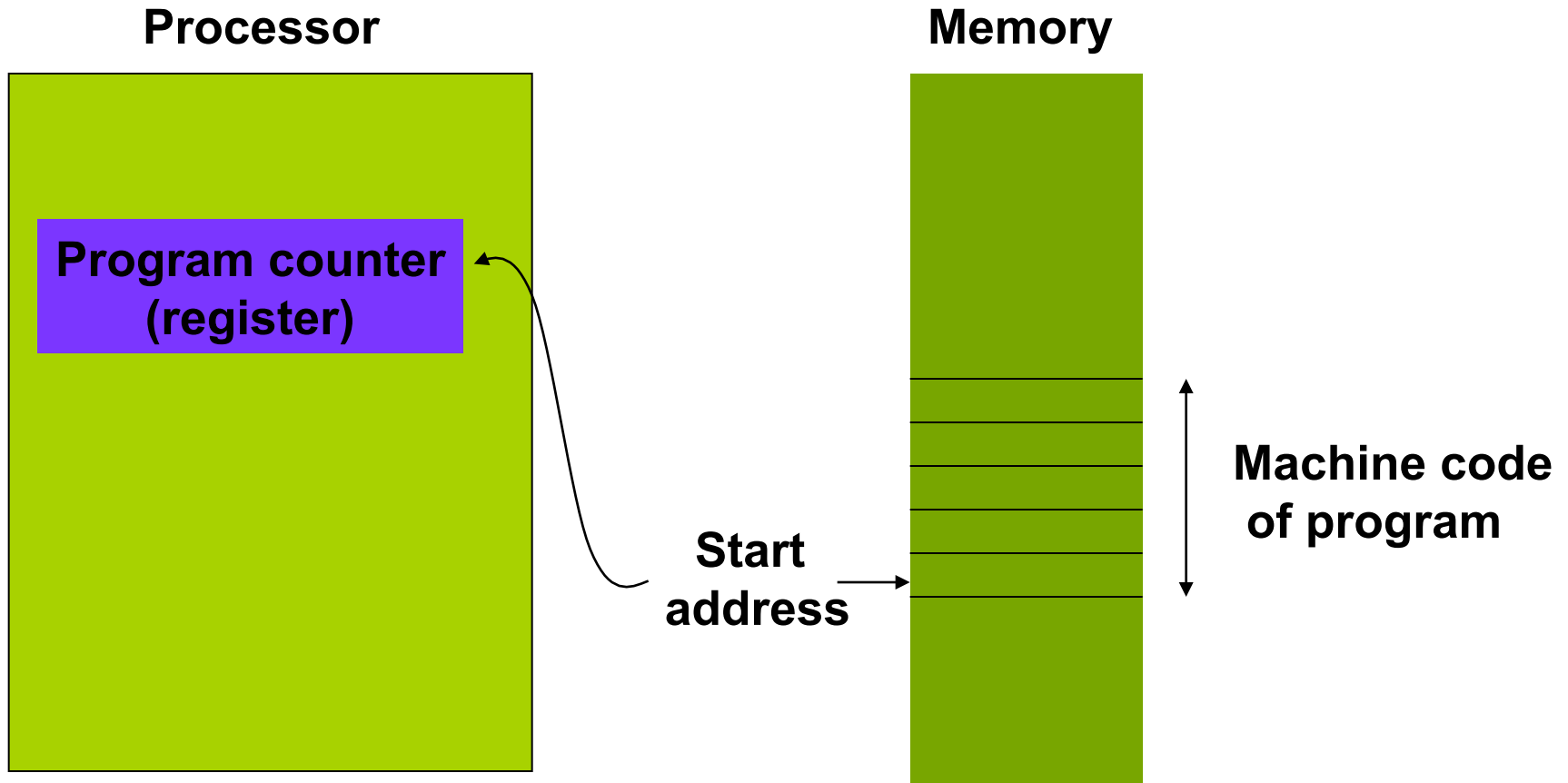


Where Does It All Begin?

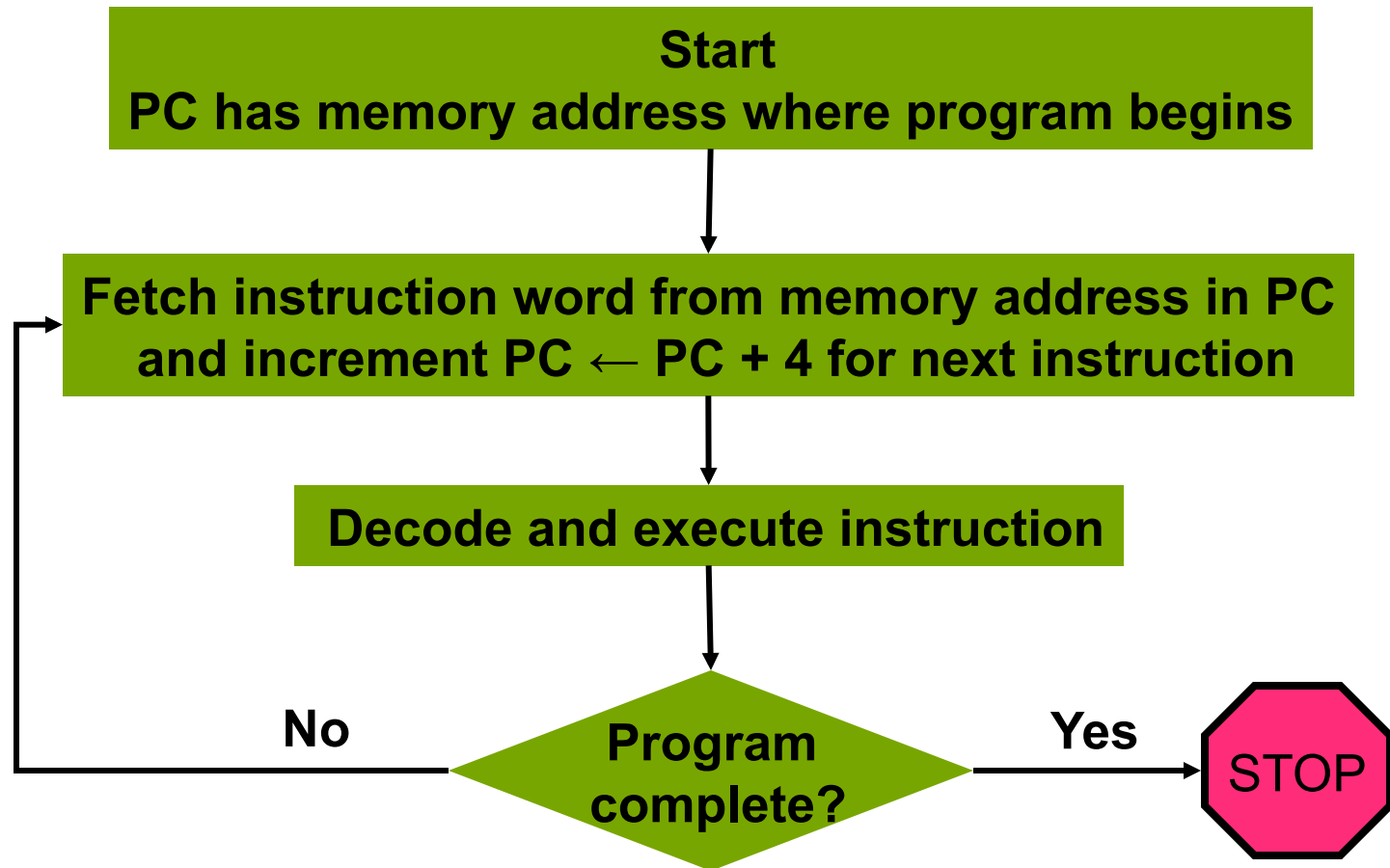
- In a register called *program counter (PC)*.
- PC contains the memory address of the next instruction to be executed.
- In the beginning, PC contains the address of the memory location where the program begins.



Where is the Program?



How Does It Run?

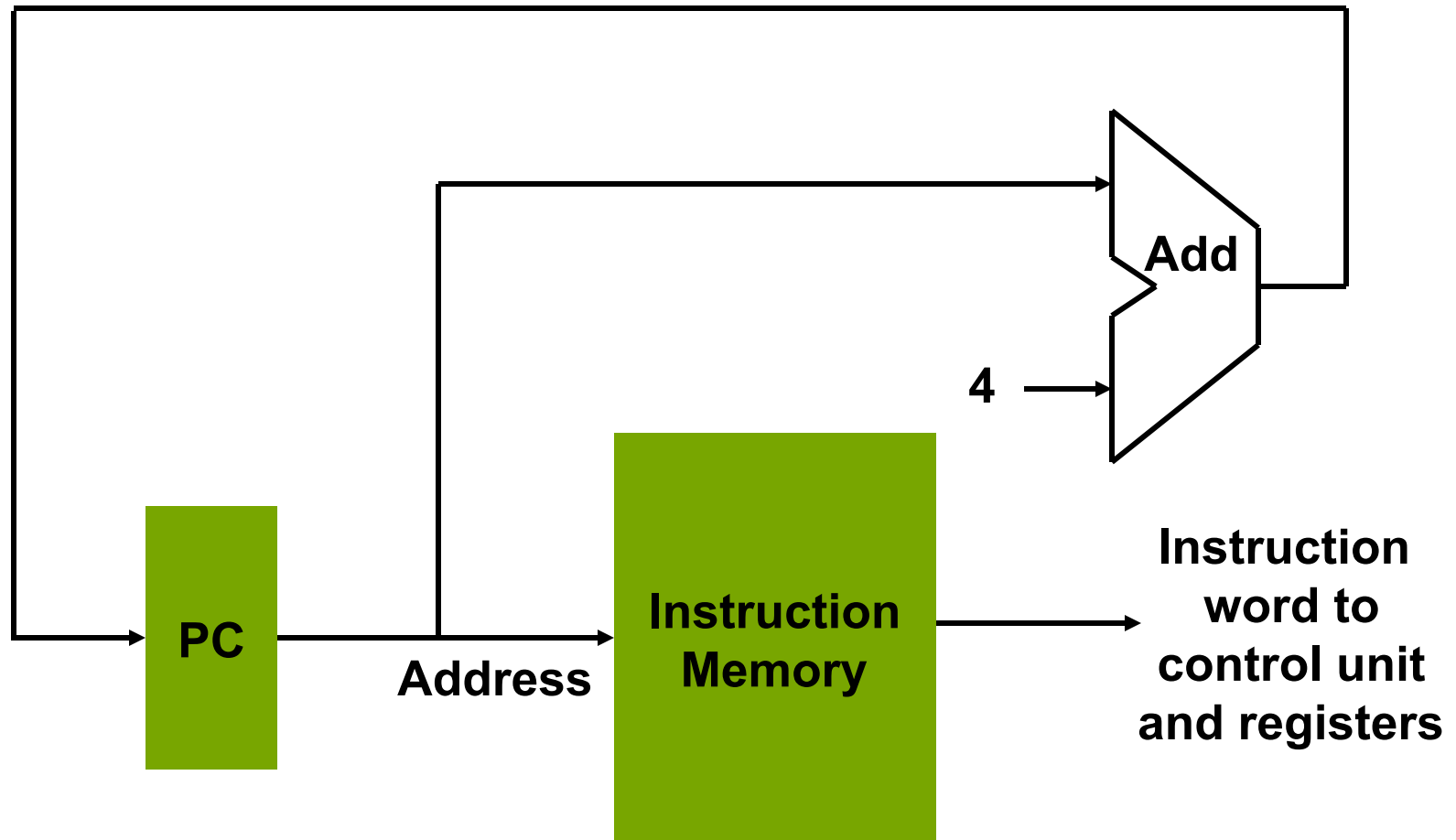


Datapath and Control

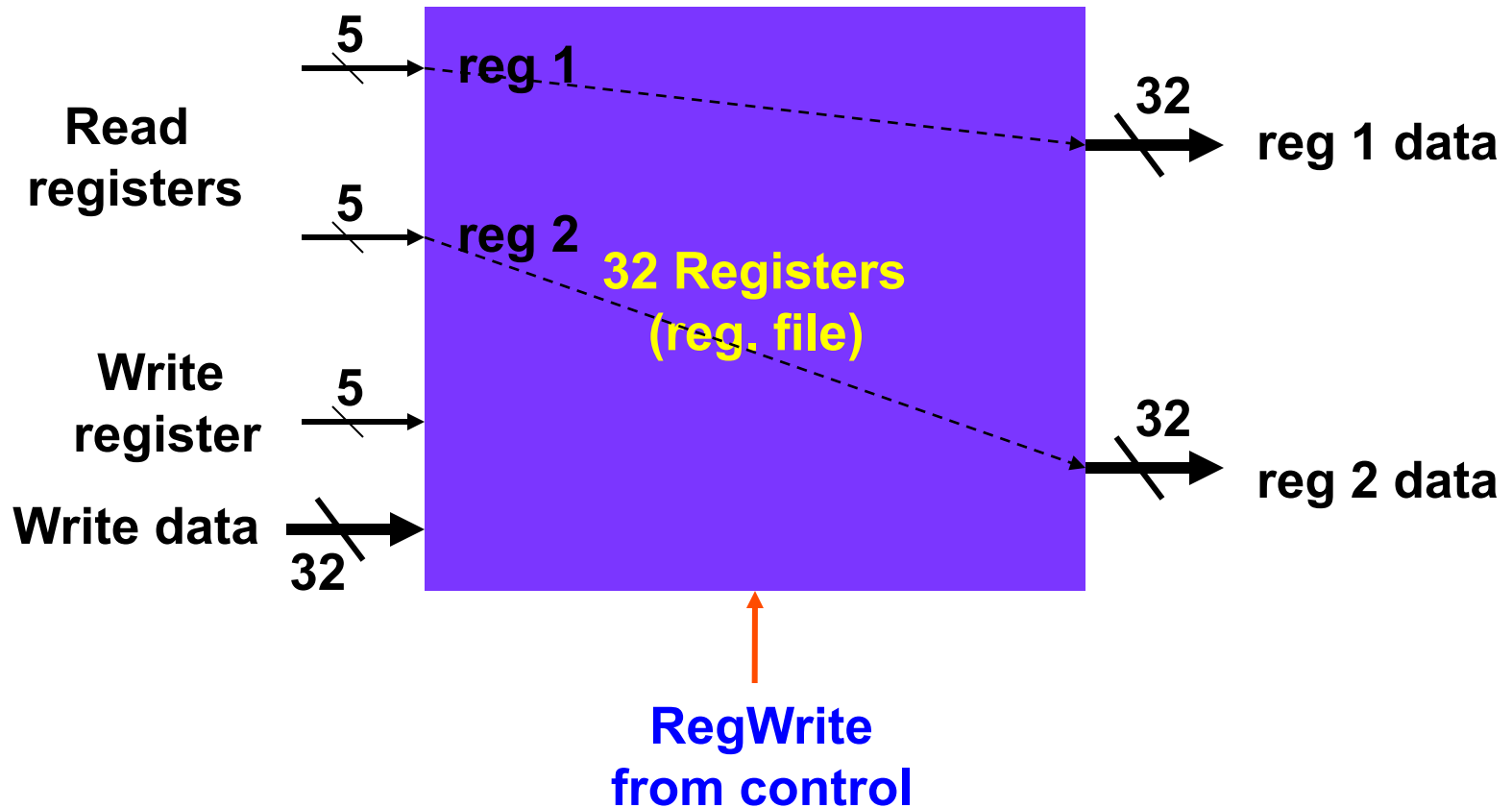
- **Datapath**: Memory, registers, adders, ALU, and communication buses. Each step (fetch, decode, execute) requires communication (data transfer) paths between memory, registers and ALU.
- **Control**: Datapath for each step is set up by control signals that set up dataflow directions on communication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



Datapath for Instruction Fetch



Register File: A Datapath Component



Multi-Operation ALU

Operation
select

ALU function

000

AND

001

OR

010

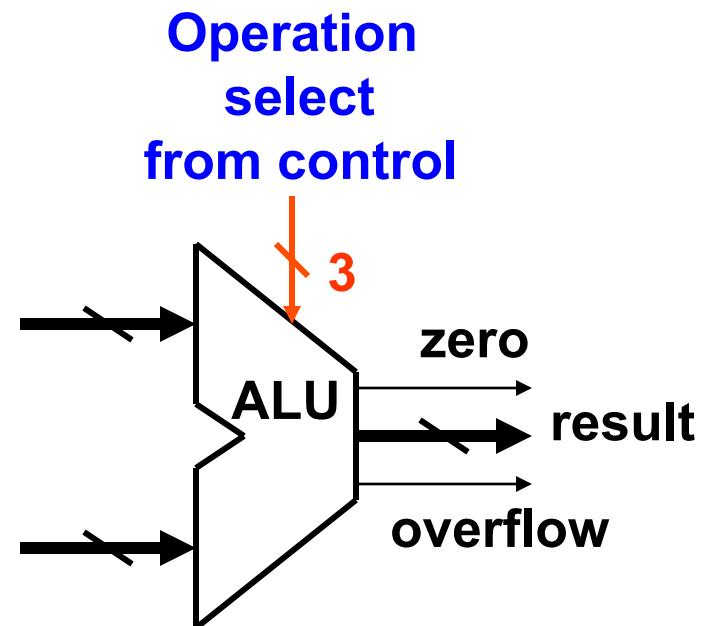
Add

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

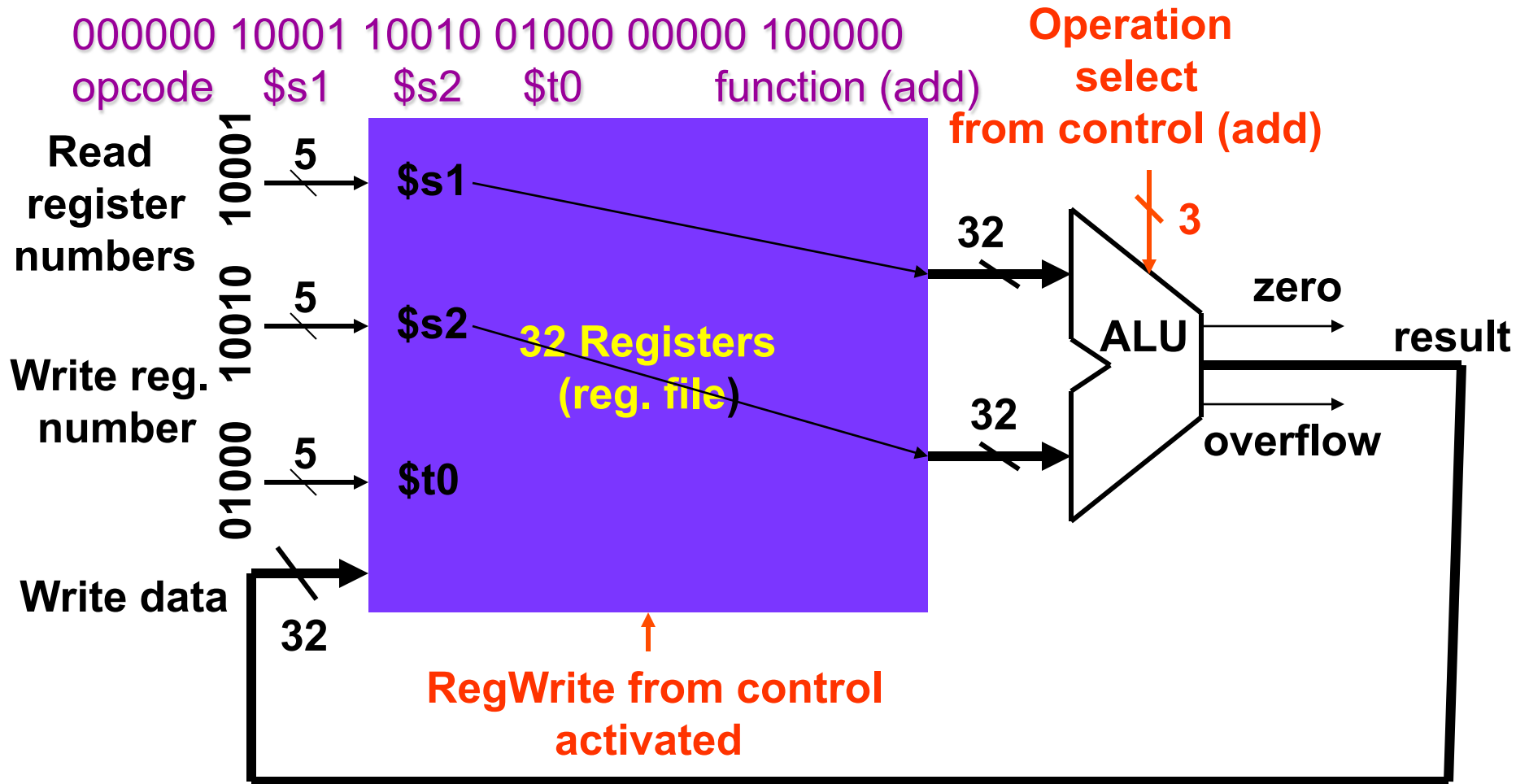
000000 10001 10010 01000 00000 100000

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.



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

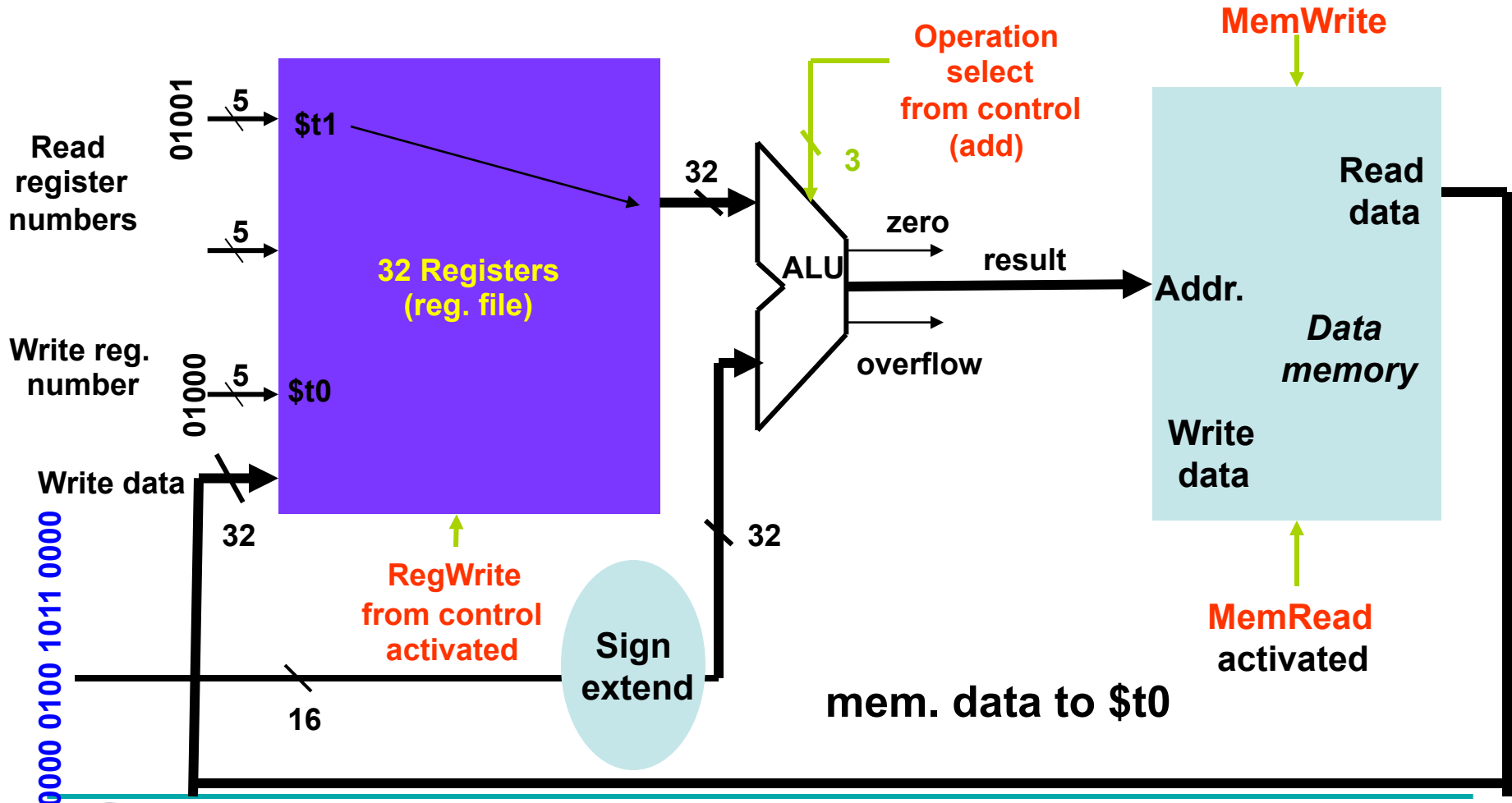
101011 01001 01000 0000 0100 1011 0000

opcode \$t1 \$t0 1200



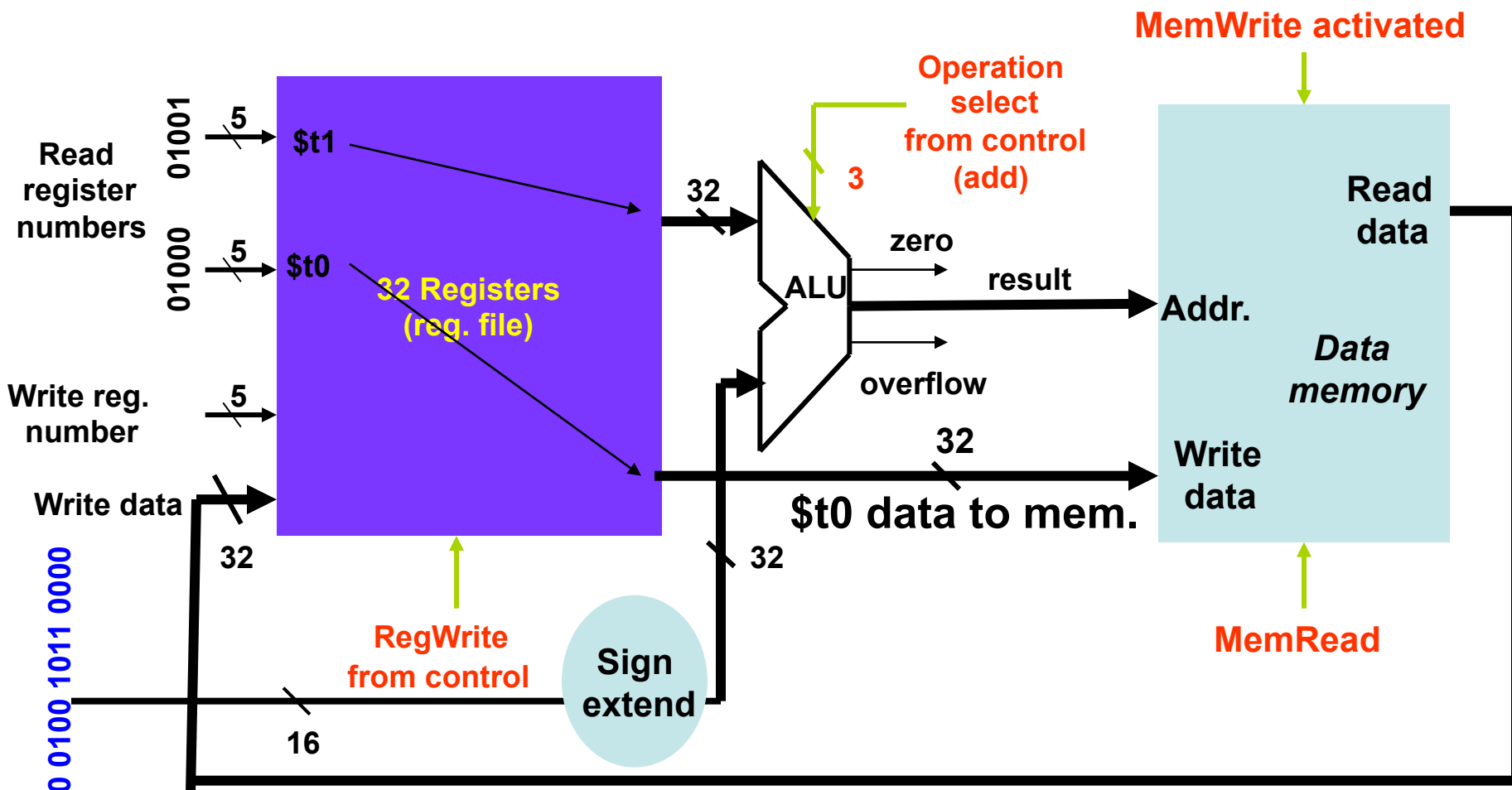
Datapath for lw Instruction

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



Datapath for sw Instruction

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



0000 0100 1011 0000



Branch Instruction (I-Type)

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



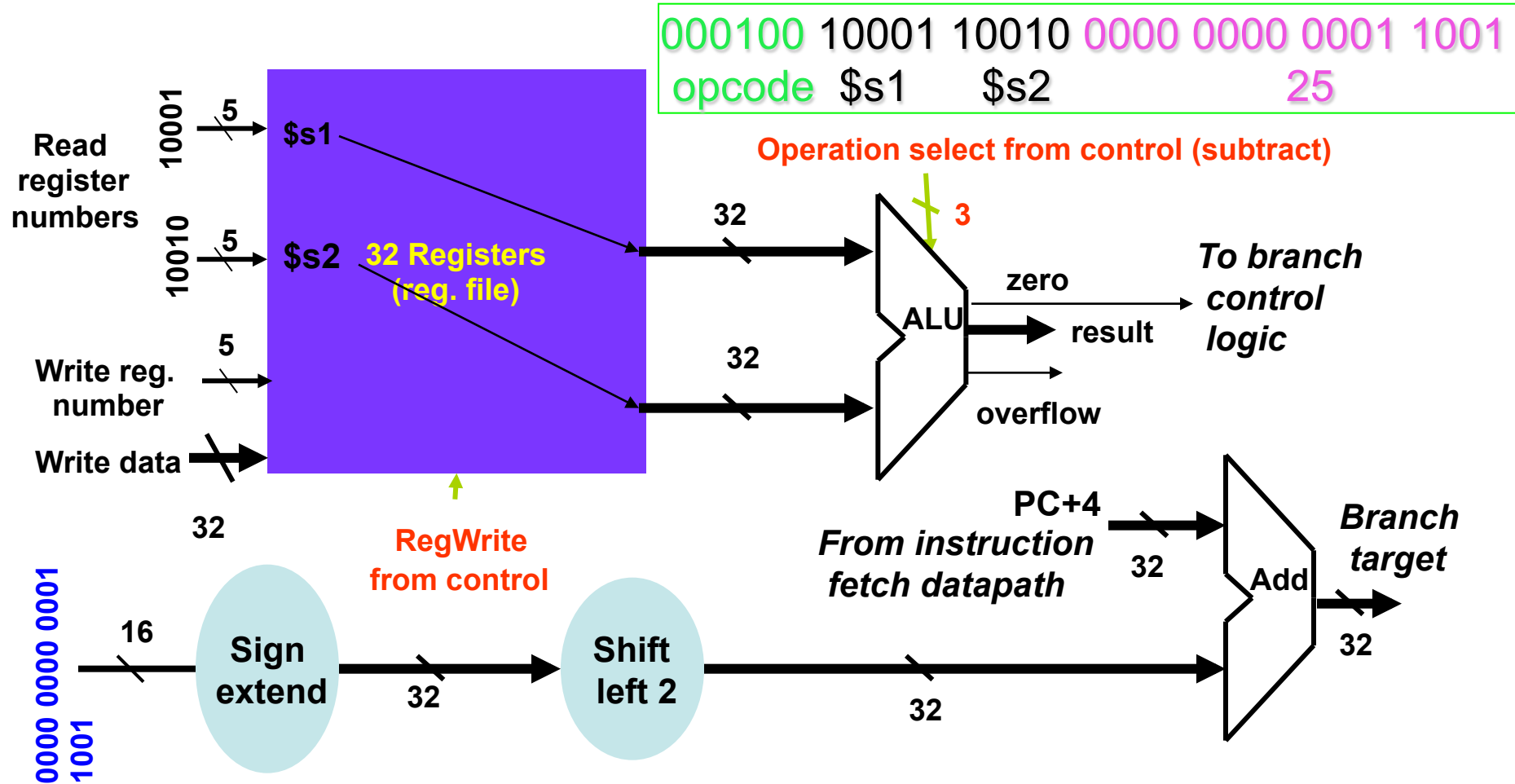
16-bits

000100 10001 10010 0000 0000 0001 1001

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

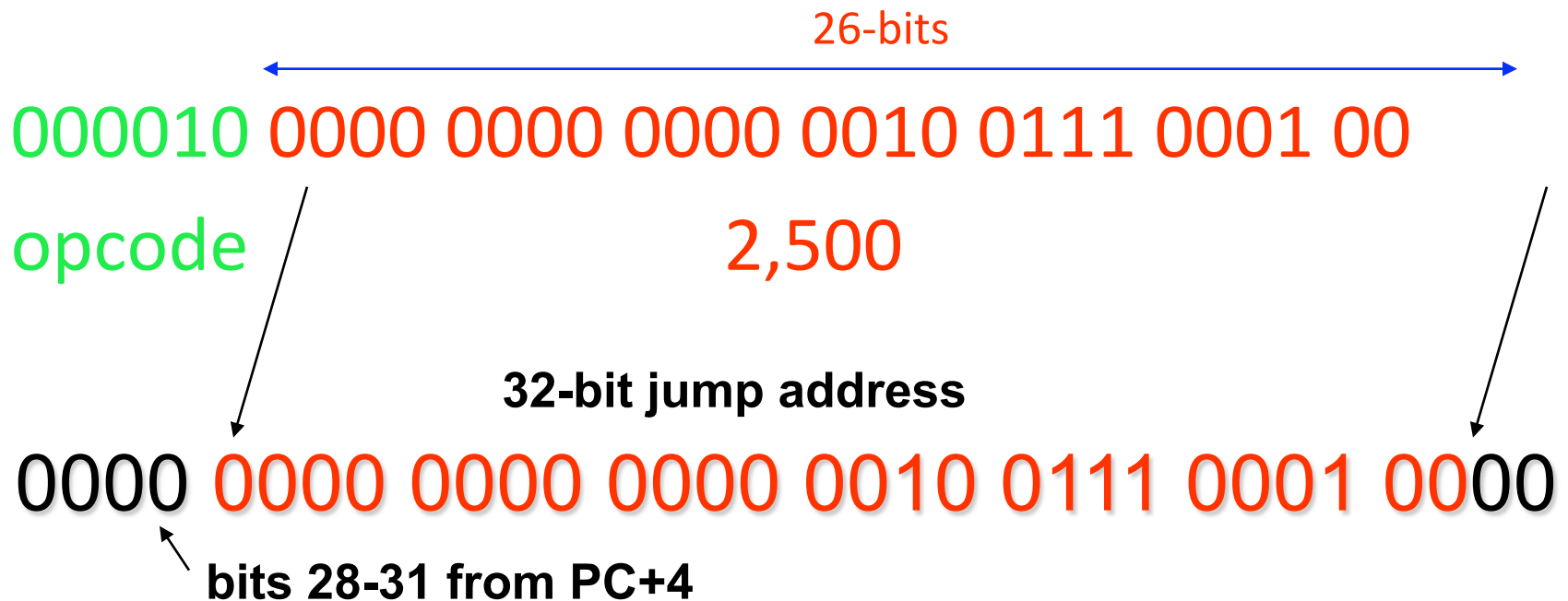


Datapath for beq Instruction

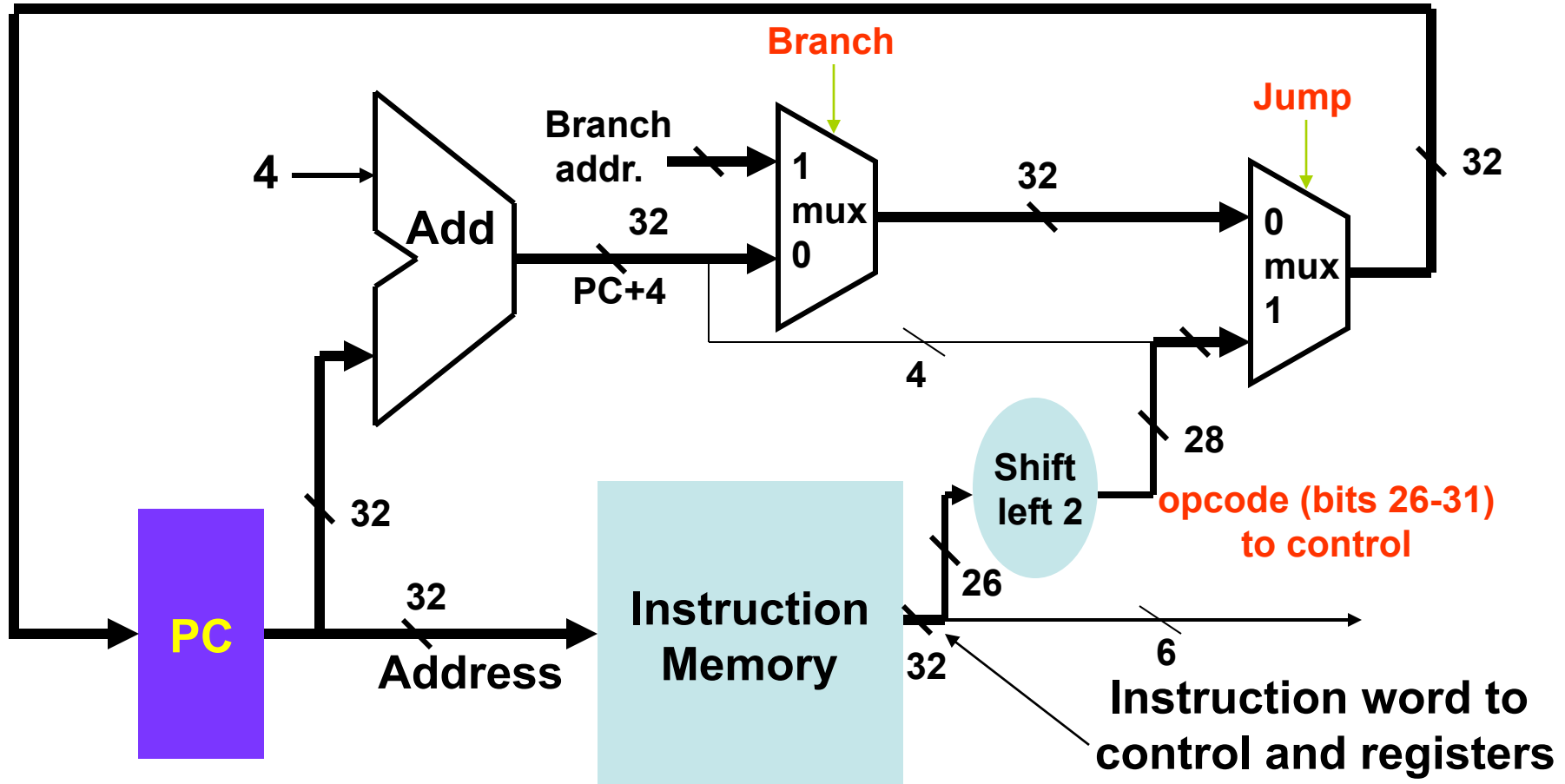


J-Type Instruction

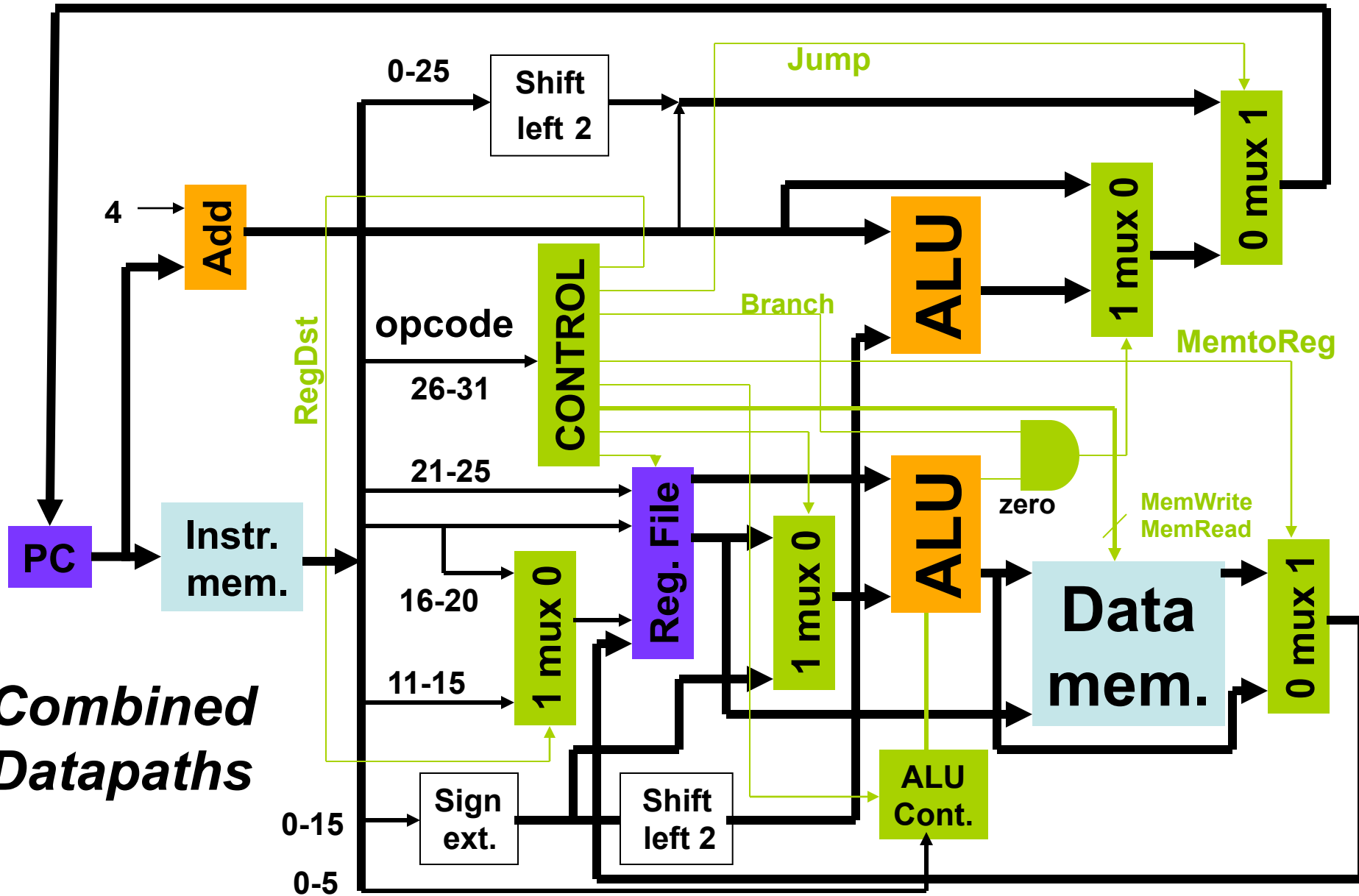
- j 2500 # jump to instruction 2,500



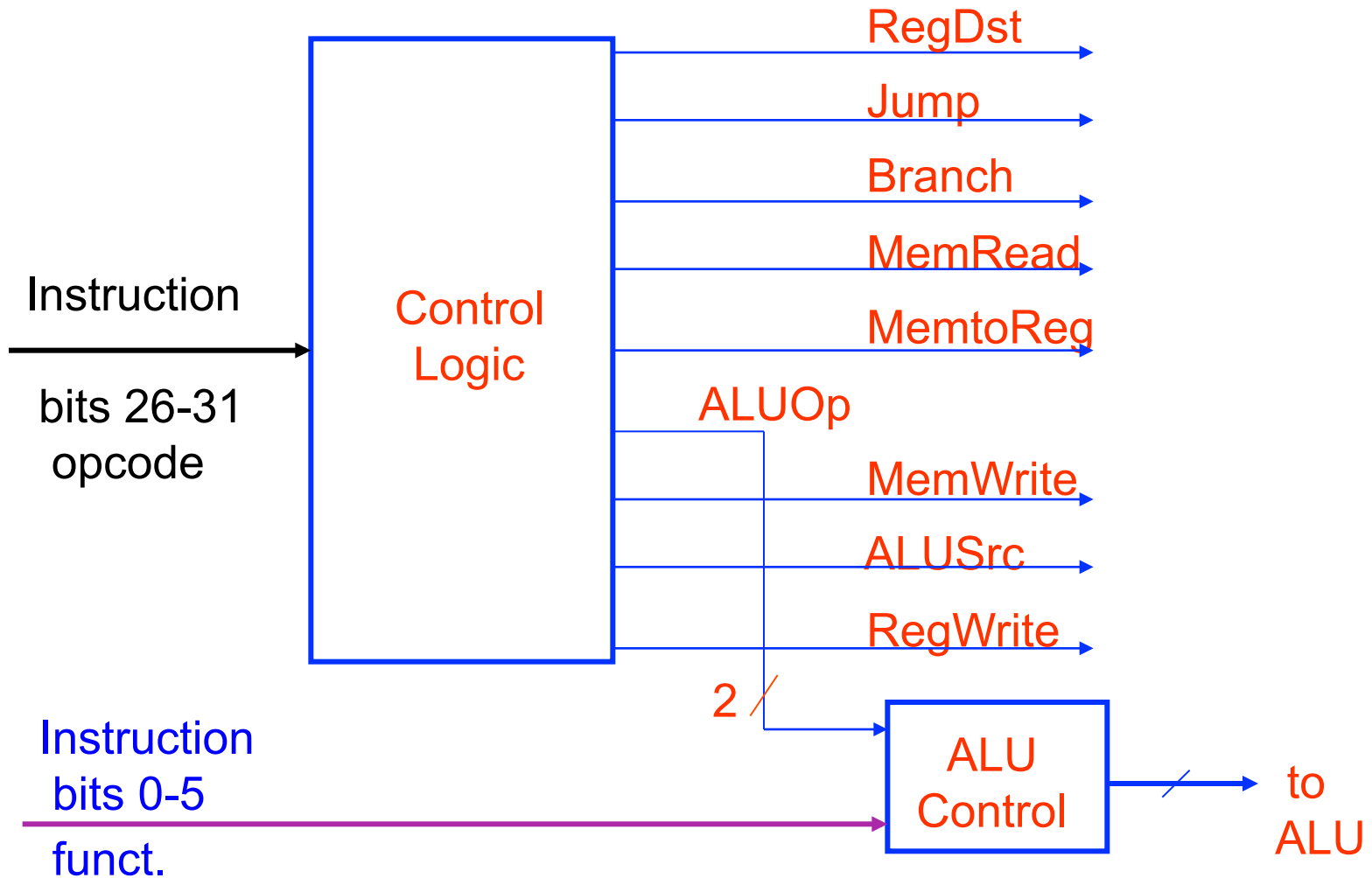
Datapath for Jump Instruction



Combined Datapaths



Control Logic



Control Logic: Truth Table

Instr type	Inputs: instr. opcode bits						Outputs: control signals									
	31	30	29	28	27	26	RegDst	Jump	ALUSrc	MemoReg	RegWrite	MemRead	MemWrite	Branch	ALOp1	ALOp2
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., $1/w$.
 - 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.



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.
 - 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

