RISC Design

Single Cycle Implementation

Virendra Singh
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

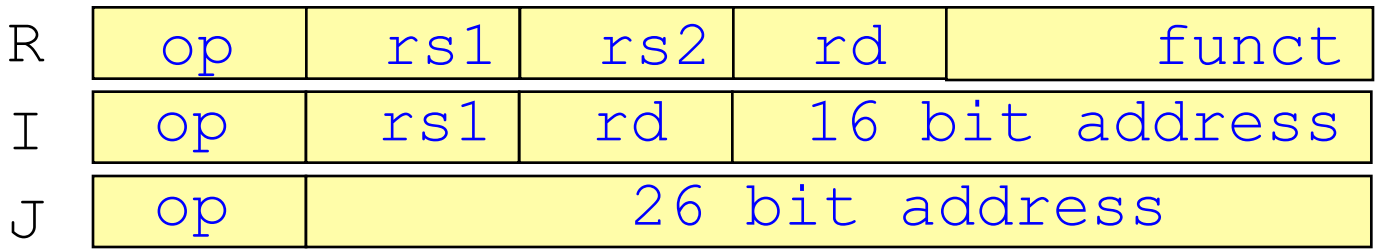
EE-309: Microprocessors

Lecture 30 (05 Oct 2015)

CADSL

Overview of DLX

- Simple instructions, all 32 bits wide
- Very structured, no unnecessary baggage
- Only three instruction formats

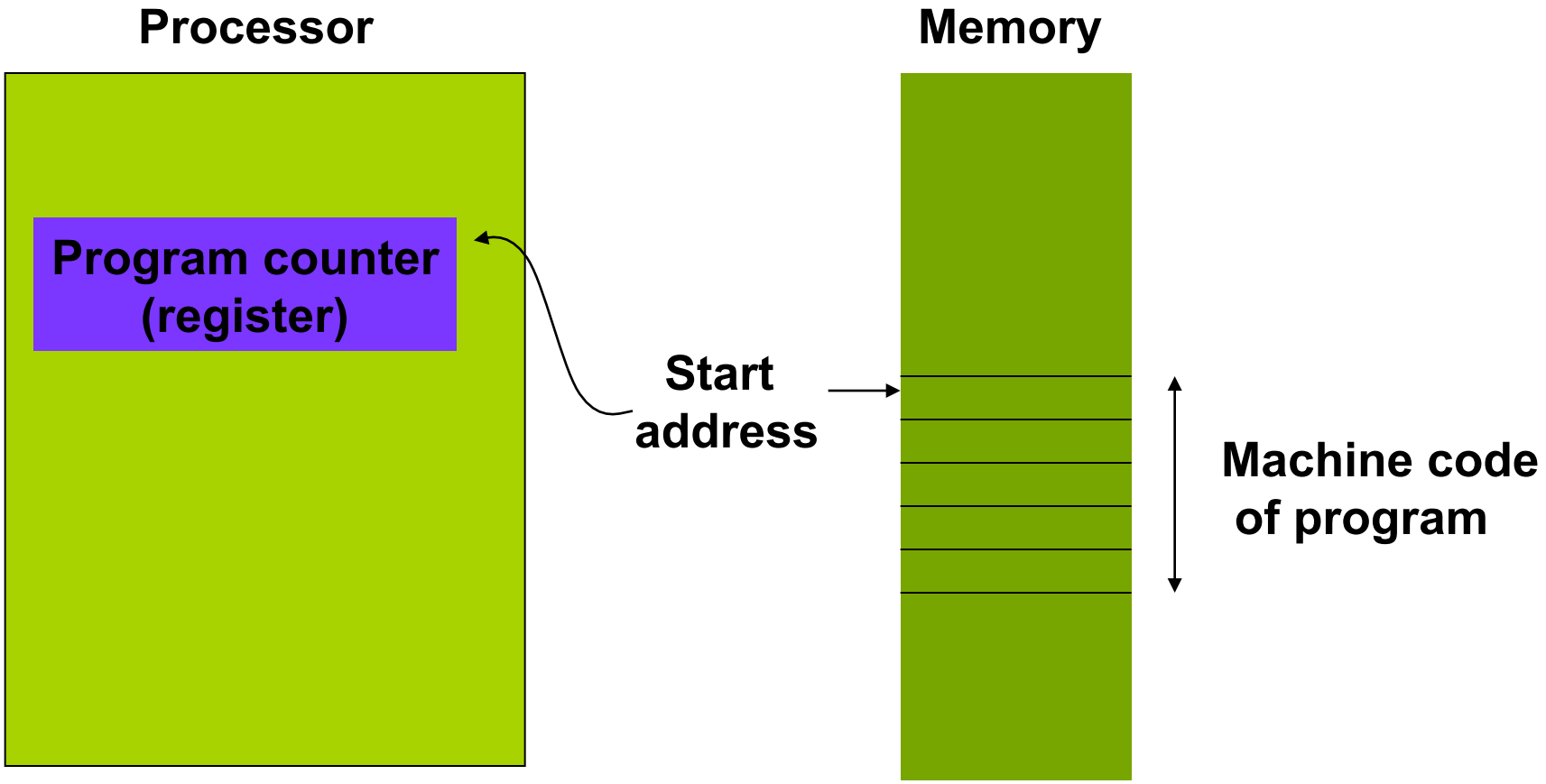


- Rely on compiler to achieve performance

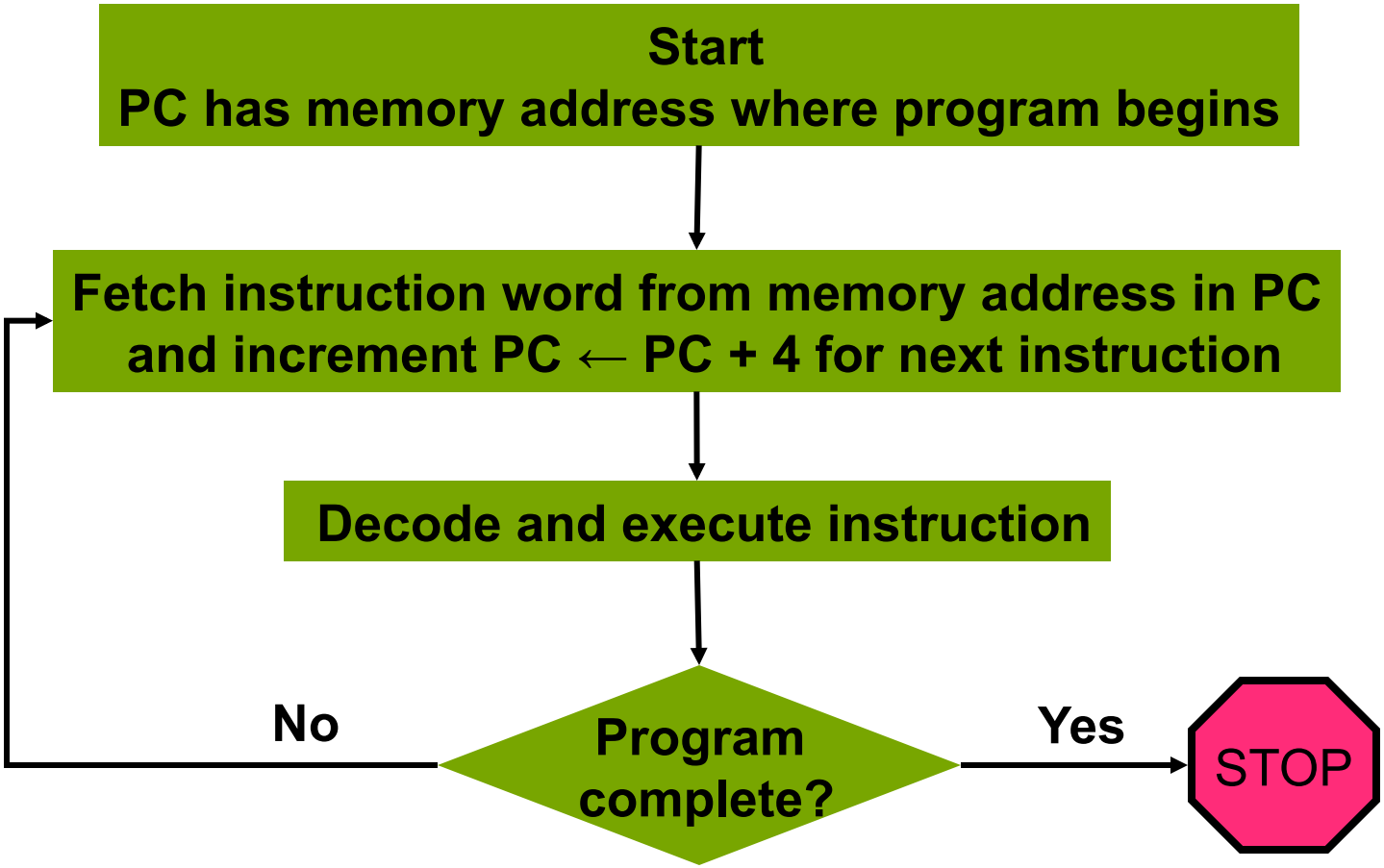
Where Does It All Begin?

- *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.

Example Instruction Set: Mini-DLX (DLX Subset)

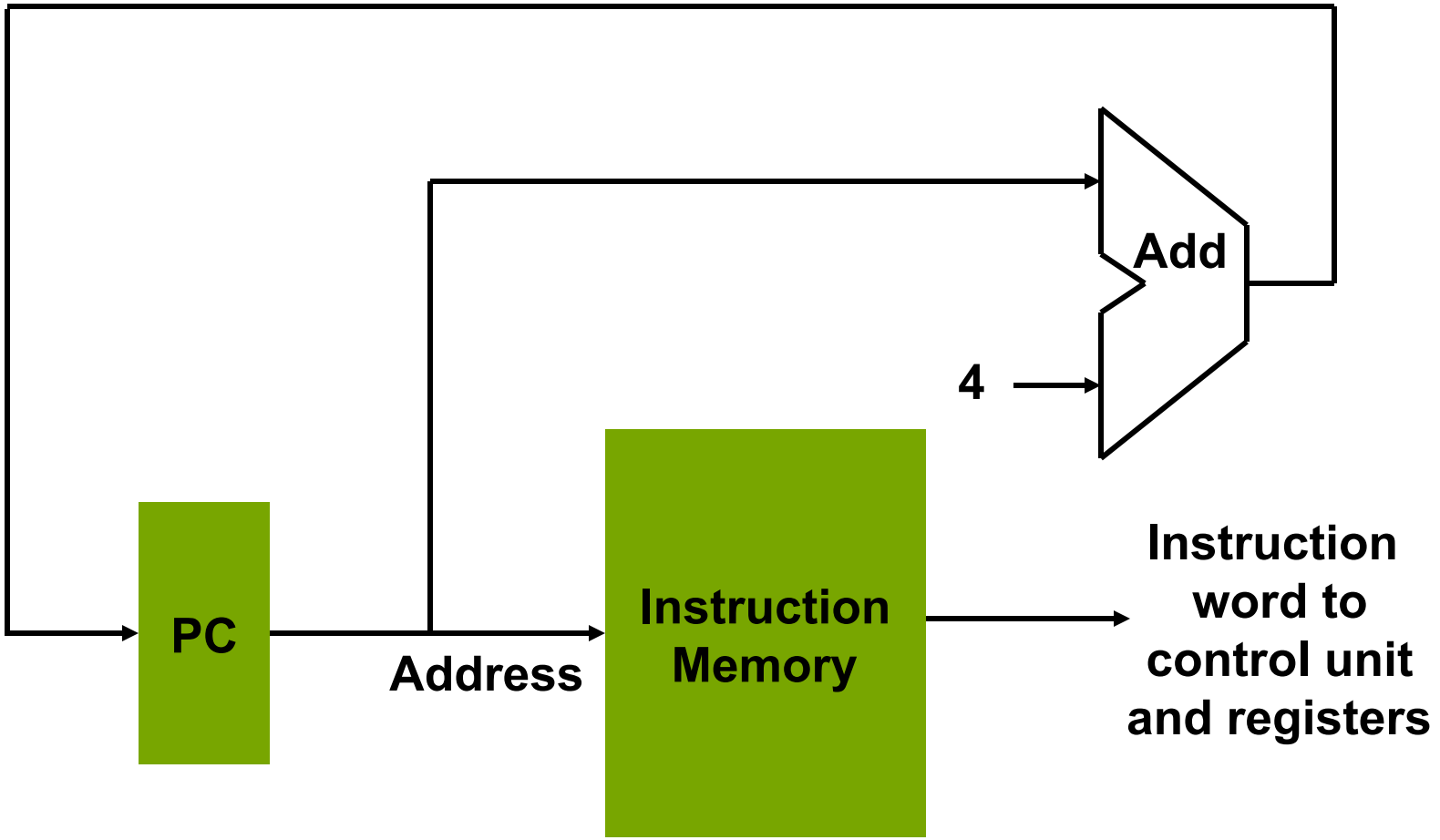
MIPS Instruction – Subset

- Arithmetic and Logical Instructions
  - add, sub, xor, and, sll
- Memory reference Instructions
  - lw, sw
- Branch
  - beq, j

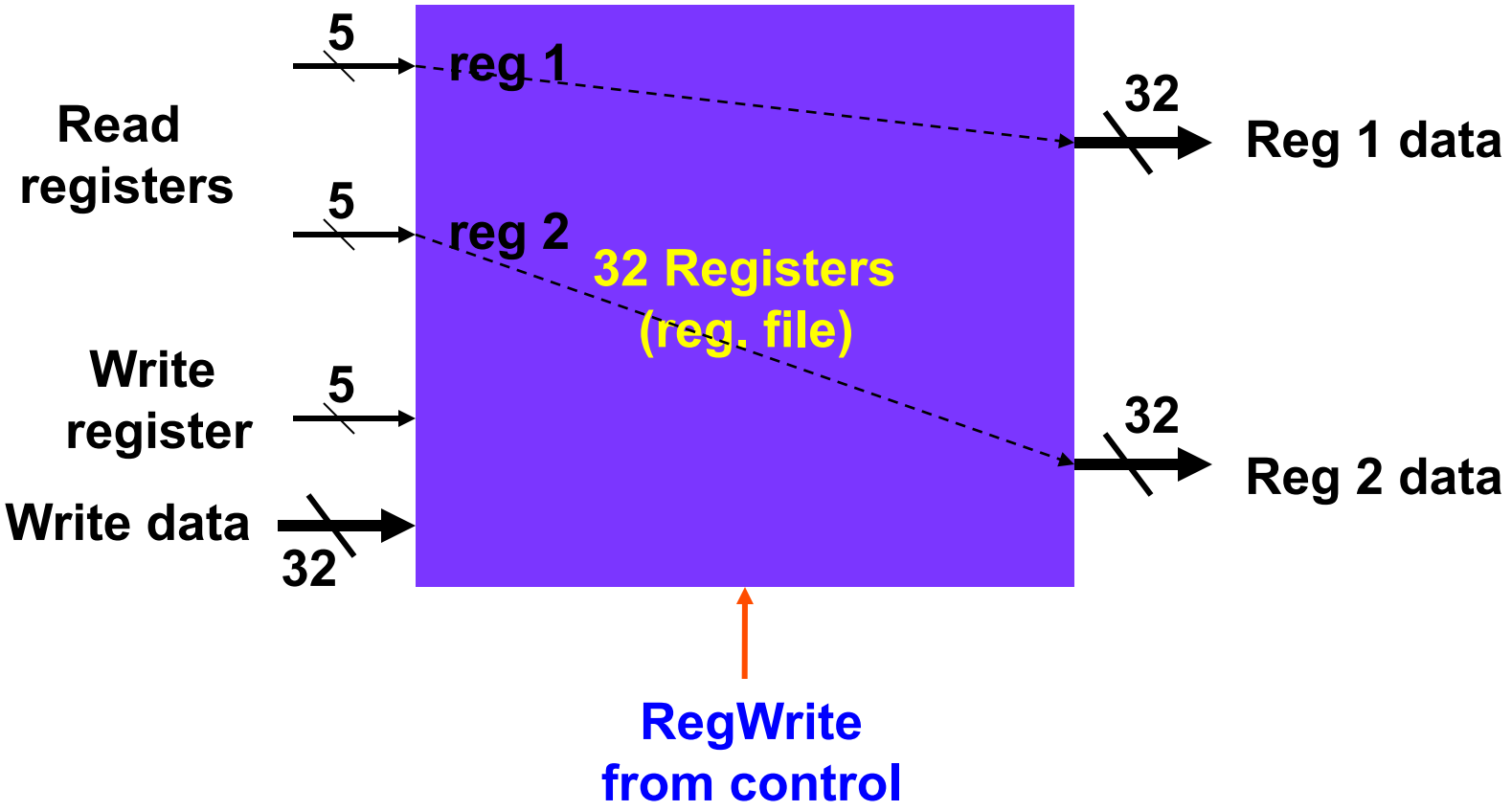
Mini-DLX Instruction Set

ADD Rd, Rs1, Rs2	Rd ← Rs1 + Rs2 (overflow – exception)	R	000_000/ 000_100
SUB Rd, Rs1, Rs2	Rd ← Rs1 - Rs2 (overflow – exception)	R	000_000/ 000_110
AND Rd, Rs1, Rs2	Rd ← Rs1 and Rs2	R	000_000/ 001_000
XOR Rd, Rs1, Rs2	Rd ← Rs1 xor Rs2	R	000_000/ 001_010
SLL Rd, Rs1, Rs2	Rd ← Rs1 << Rs2 (logical) (5 lsb of Rs2 are significant)	R	000_000/ 001_100
LW Rd, Imm (Rs1)	Rd ← M(Rs1 + Imm) (Imm: sign extended 16 bit)	I	000_100
SW Imm (Rs1), Rd	M(Rs1 + Imm) ← Rd	I	001_000
BEQ Rs1, Rs2, Label	Rs1 = Rs2: PC ← PC+4+Label Rs1 /= Rs2: PC ← PC+4	I	010_000
J Label	PC ← PC + 4 + sign_extd(imm26)	J	001_100

Datapath for Instruction Fetch

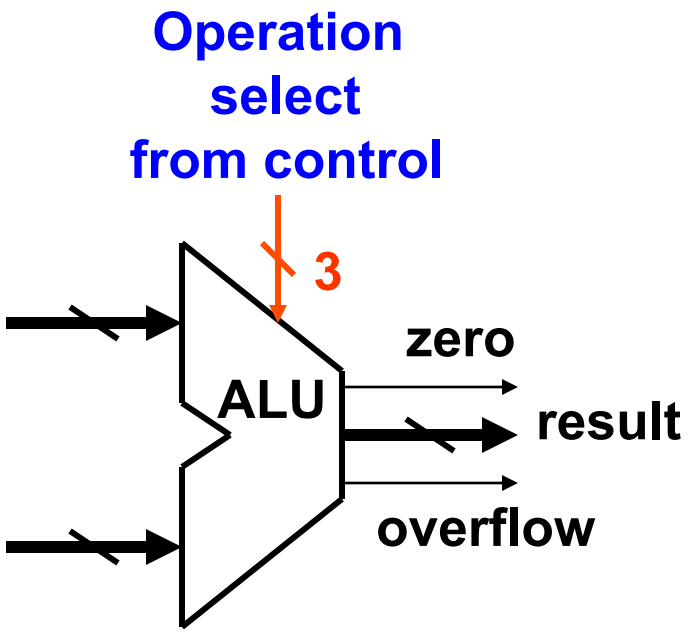


Register File: A Datapath Component



Multi-Operation ALU

Operation select	ALU function
010	Add
011	Subtract
100	AND
101	XOR
110	Shift left



zero = 1, when all bits of result are 0

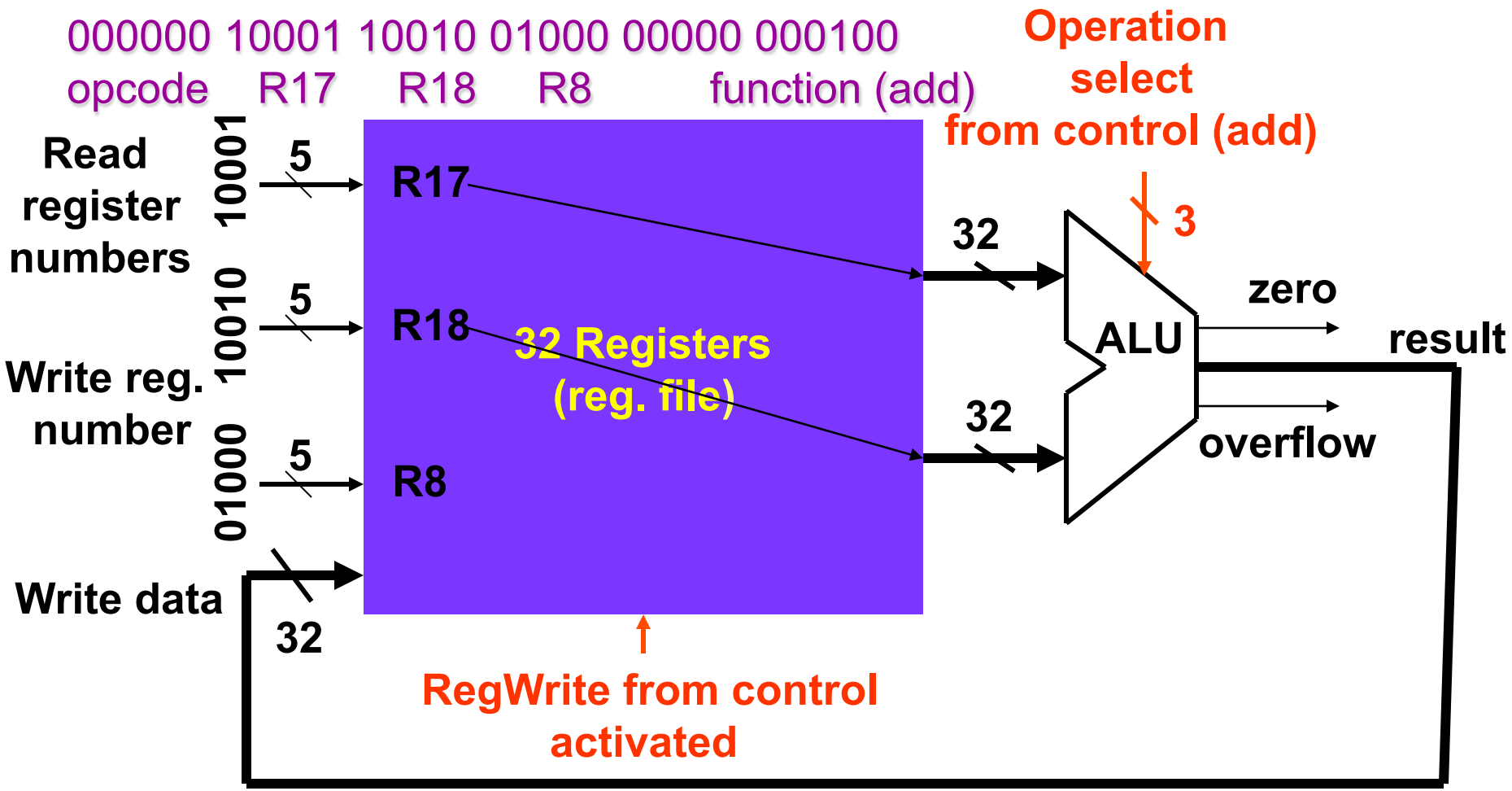
R-Type Instructions

- Also known as arithmetic-logical instructions
- add, sub, and, xor, sll
- Example: add R17, R18, R8
  - Machine instruction word

    000000 10001 10010 01000 00000 000100

    opcode R17 R18 R8 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 R8, 1200 (R9) # incr. in bytes

  000100 01001 01000 0000 0100 1011 0000

  opcode R9 R8 1200

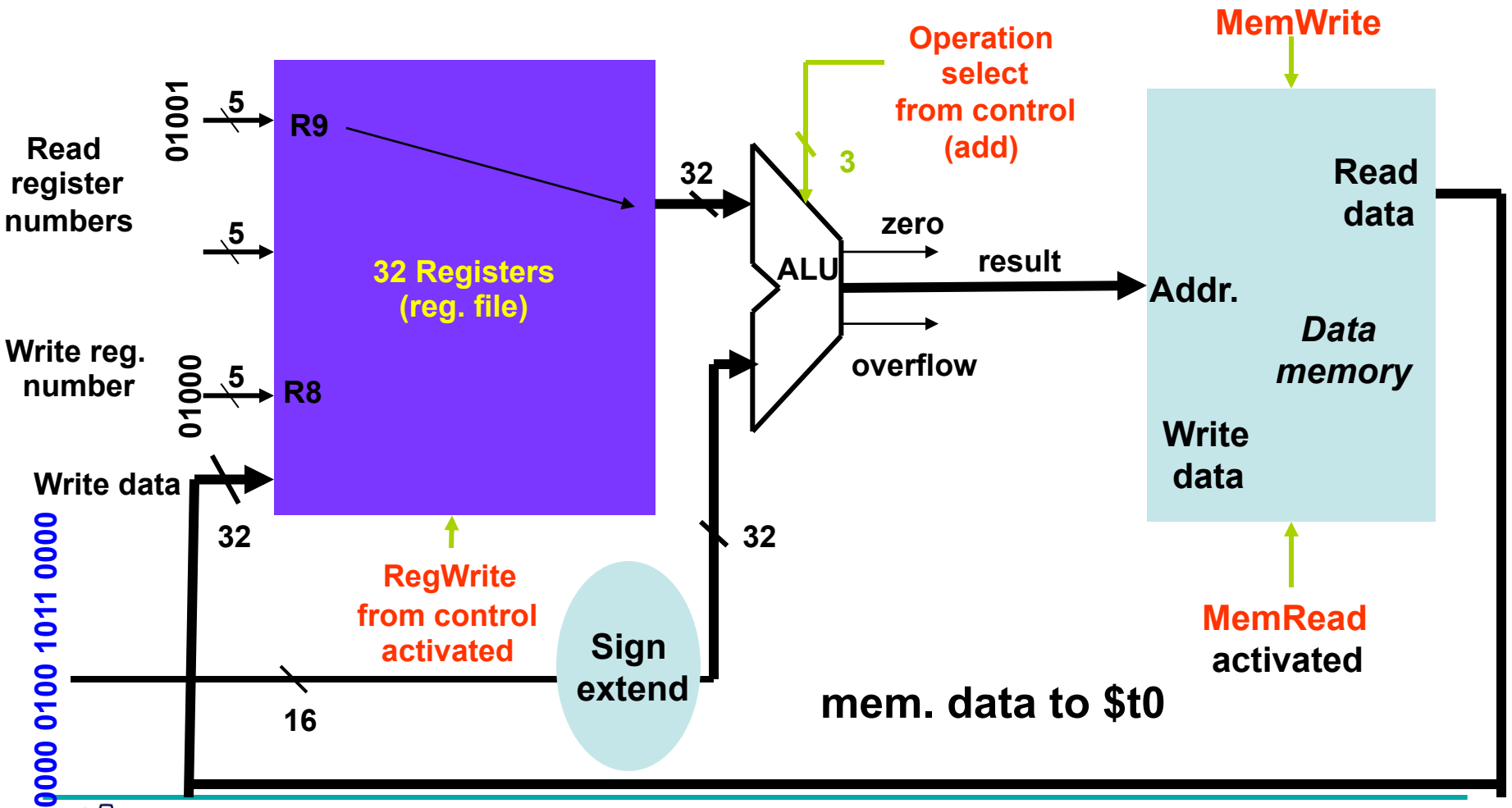
- sw R8, 1200 (R9) # incr. in bytes

  001000 01001 01000 0000 0100 1011 0000

  opcode R9 R8 1200

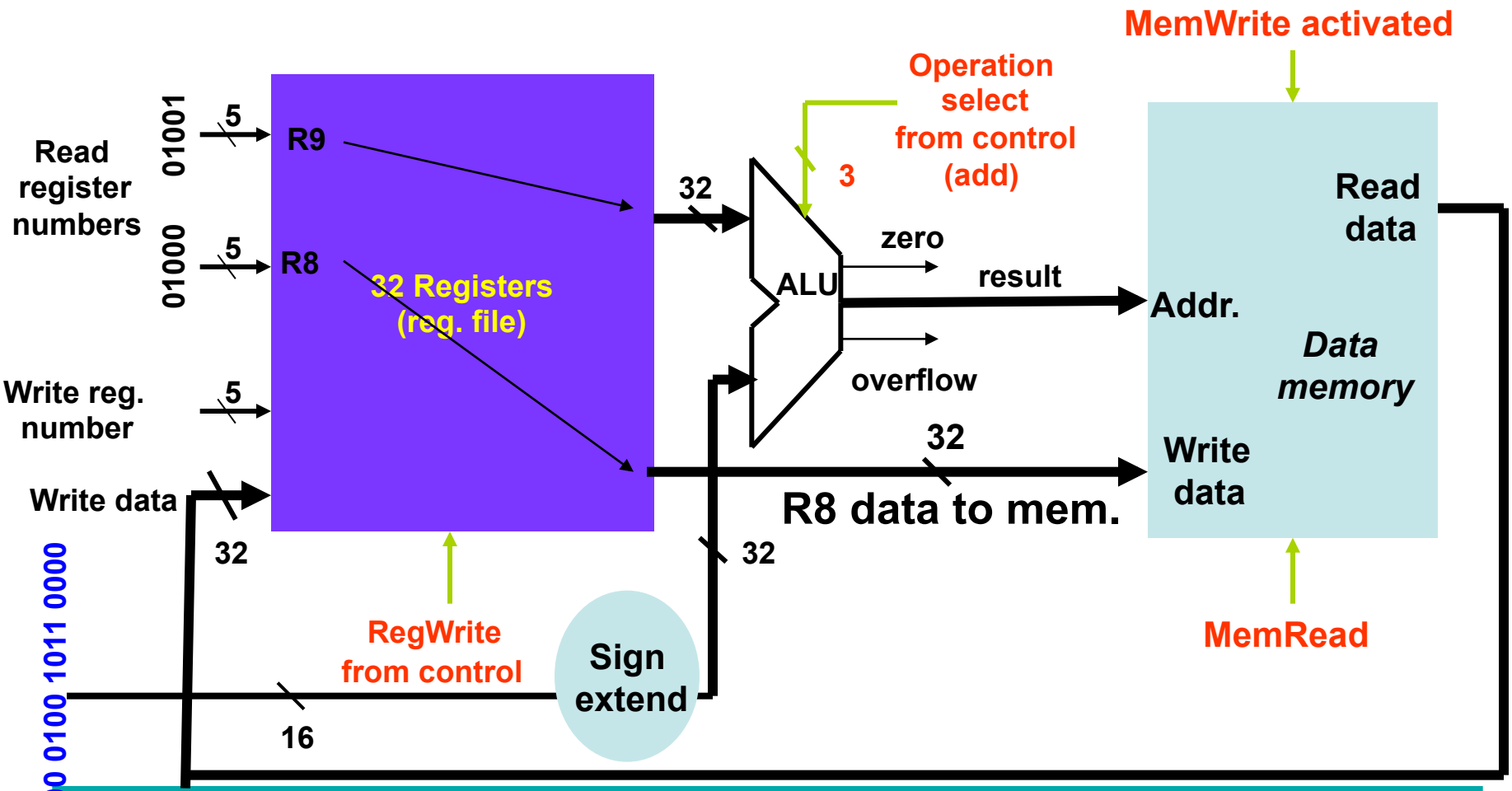
Datapath for lw Instruction

000100 01001 01000 0000 0100 1011 0000
opcode R9 R8 1200



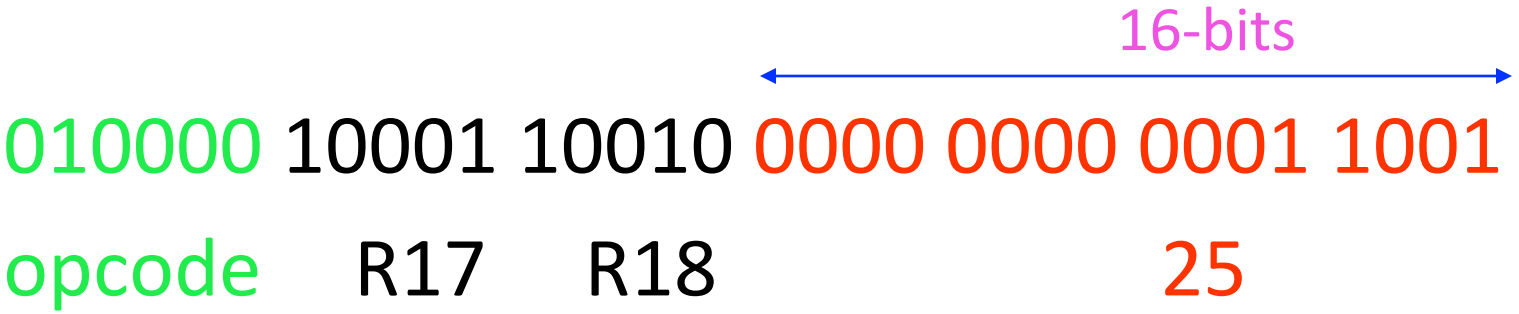
Datapath for sw Instruction

001000 01001 01000 0000 0100 1011 0000
opcode $t1 $t0 1200



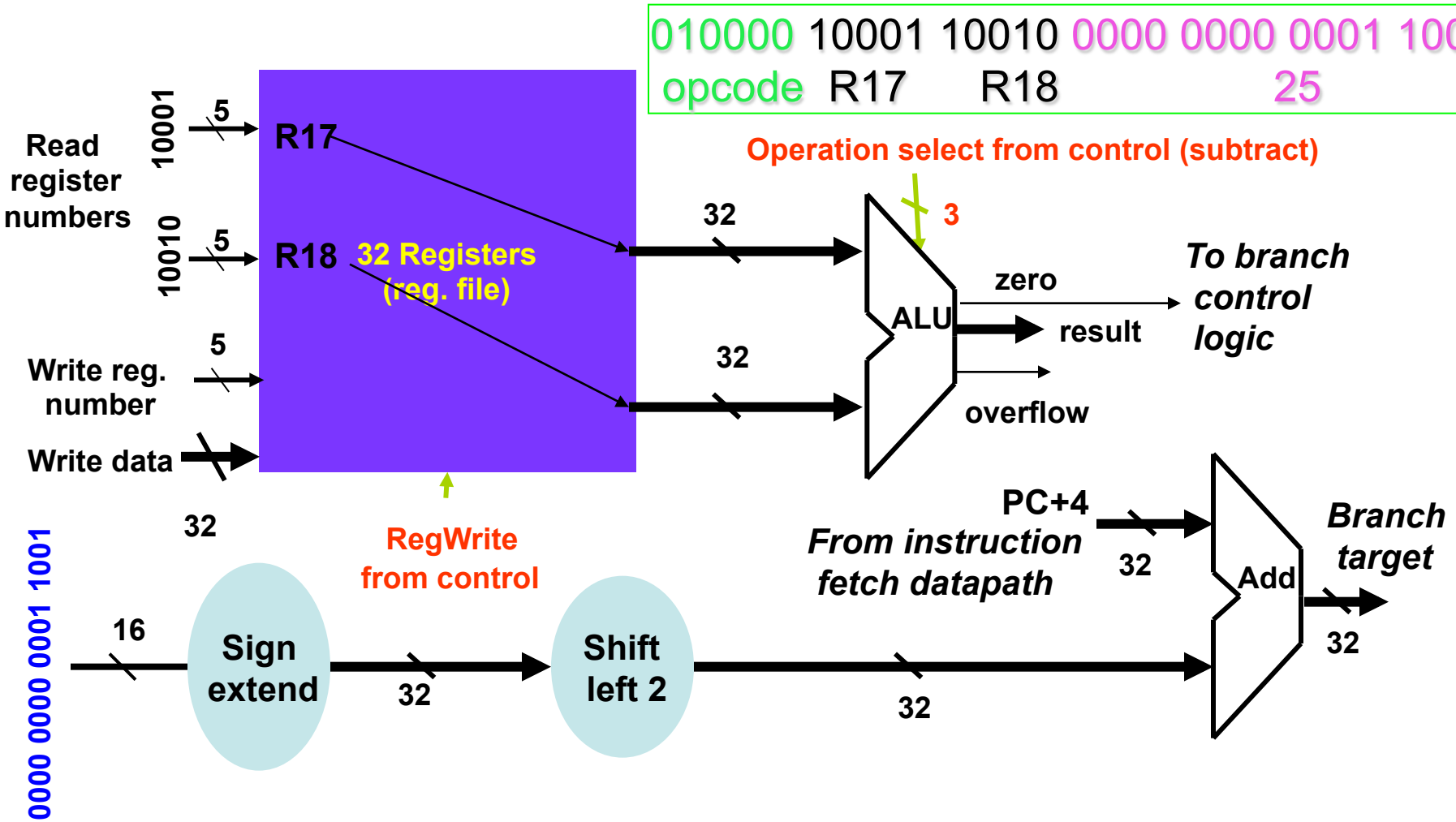
Branch Instruction (I-Type)

- beq R17, R18, 25 # if R17 = R18, advance PC through 25 instructions



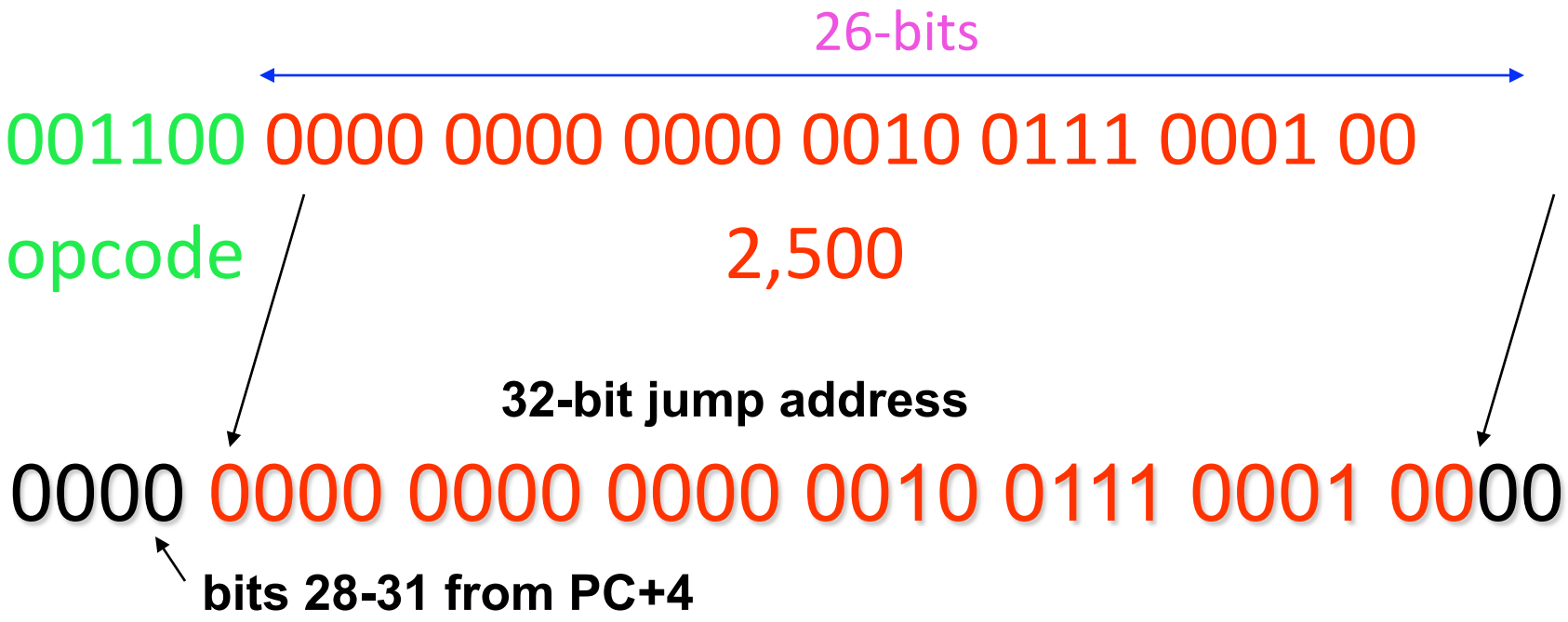
Note: Can branch within ± 2^{15} words from the current instruction address in PC.

Datapath for beq Instruction

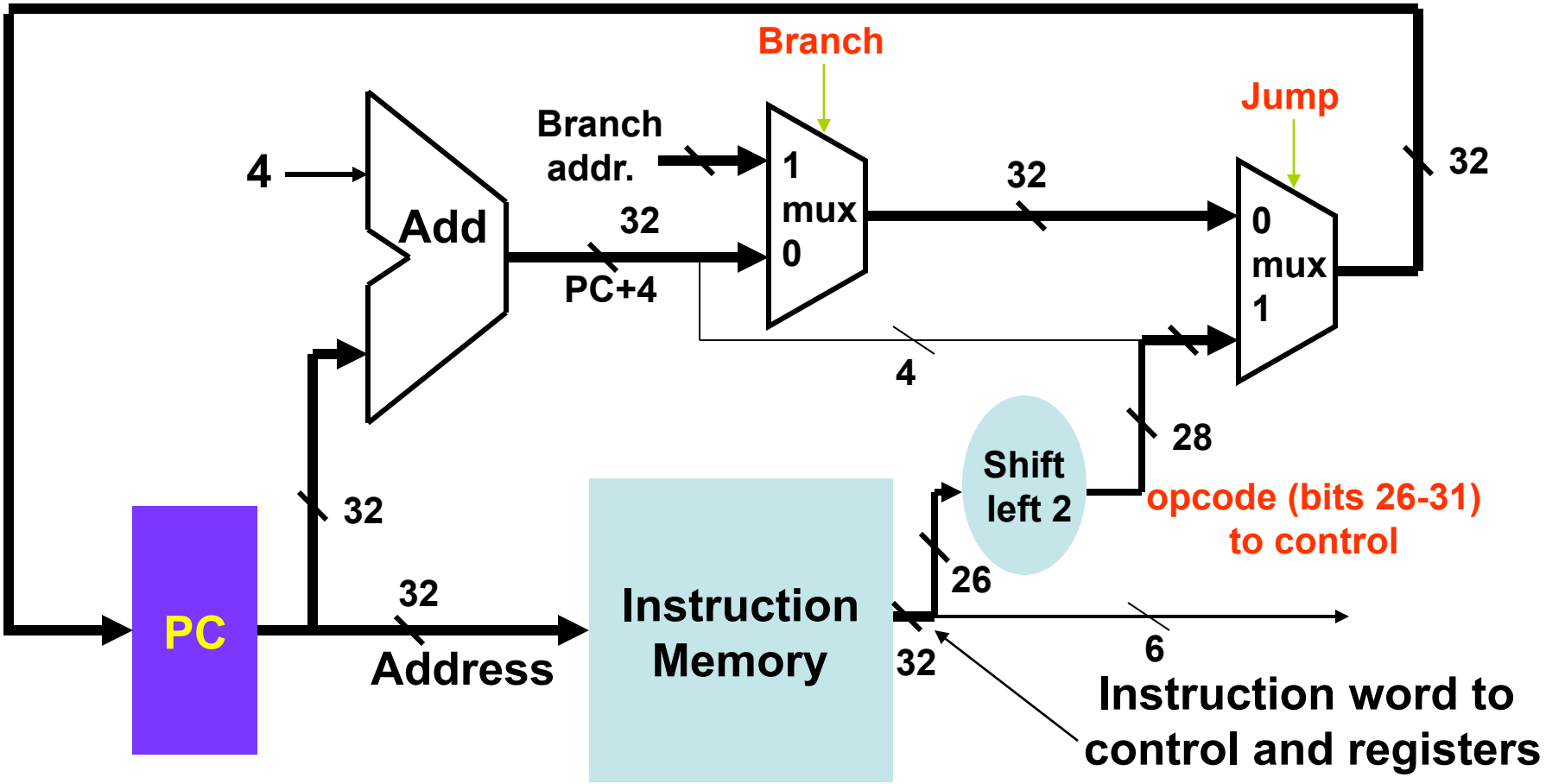


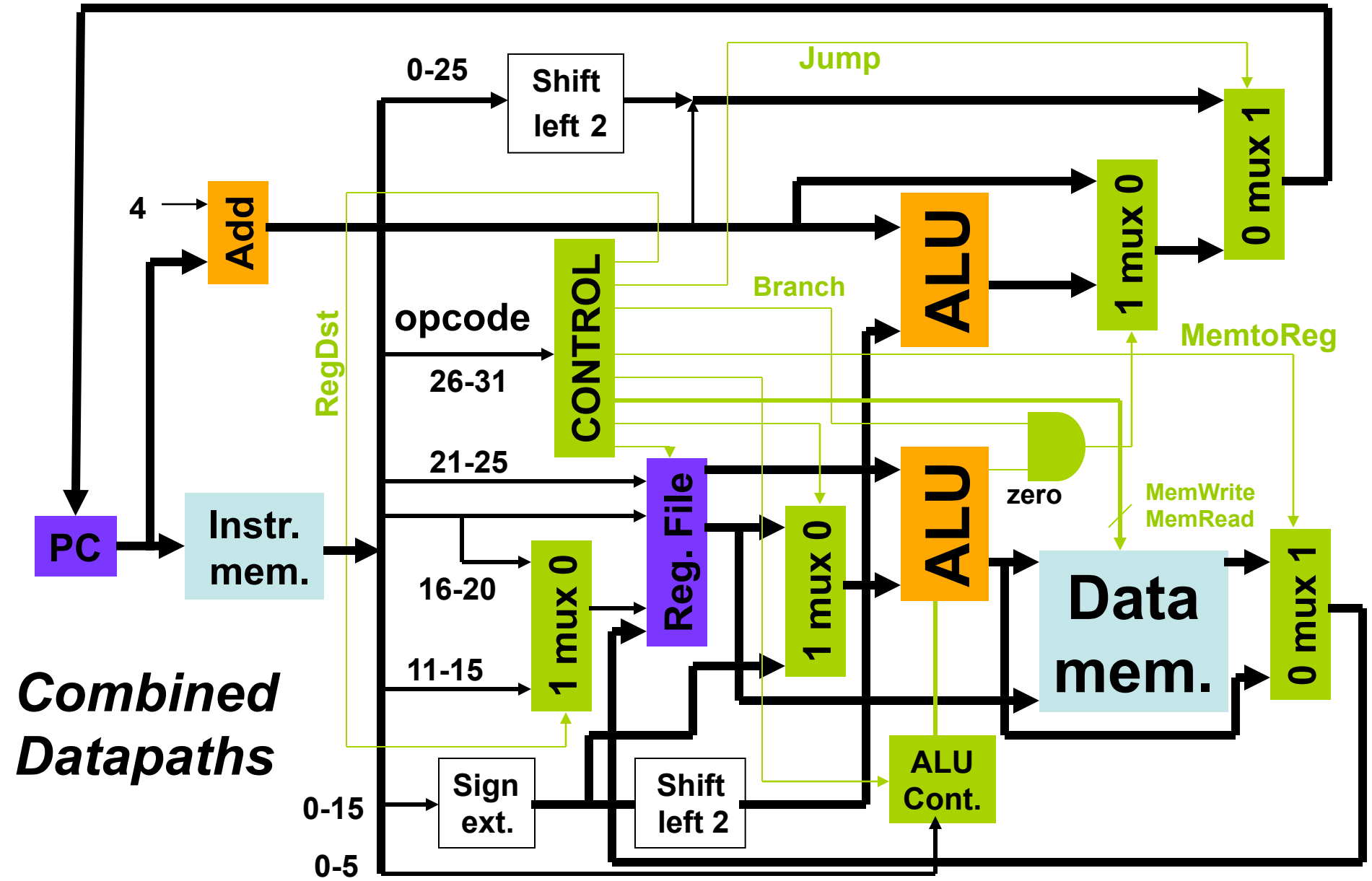
J-Type Instruction

- j 2500 # jump to instruction 2,500

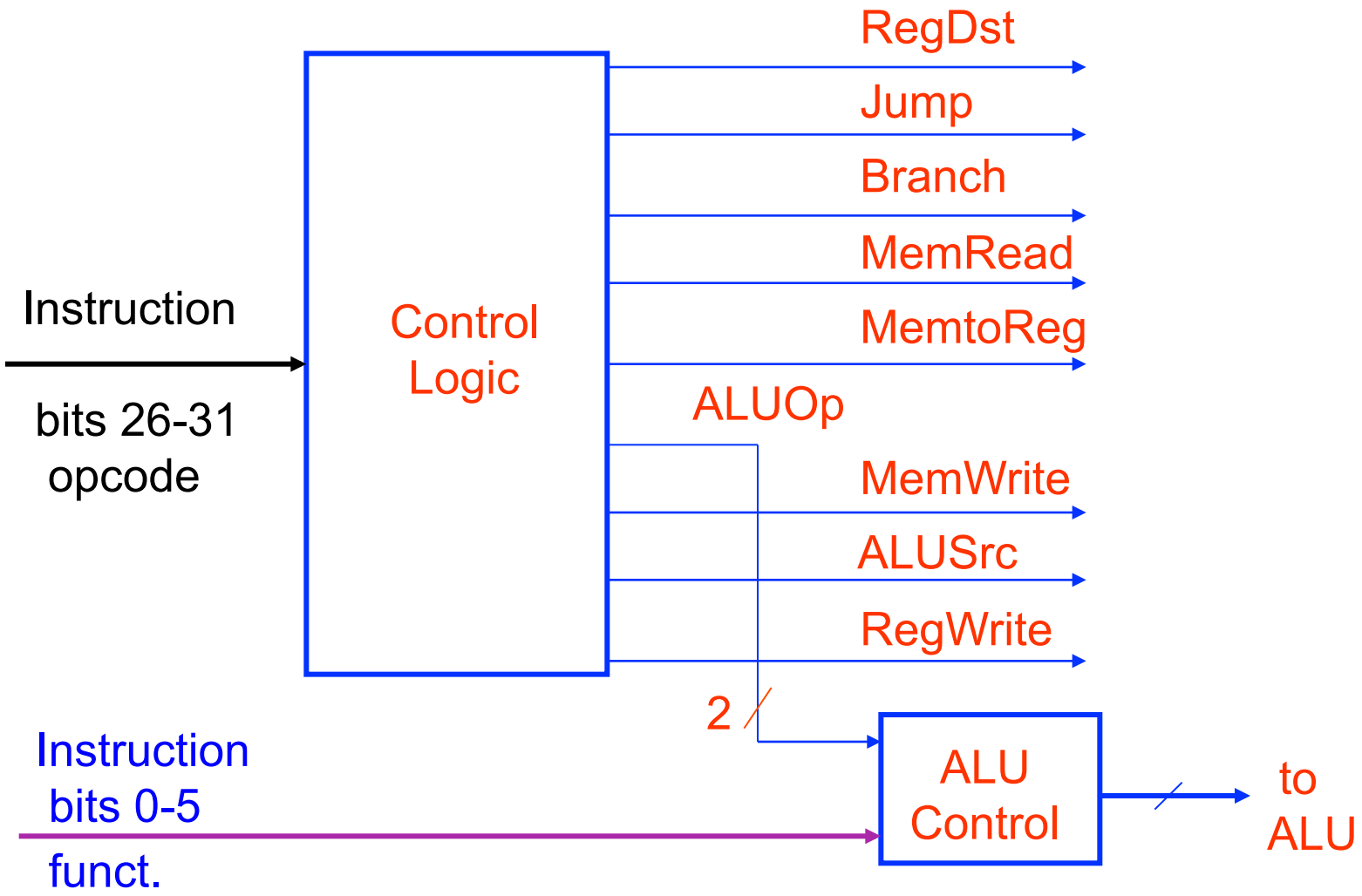


Datapath for Jump Instruction




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

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