Computer Architecture

Datapath and Control (Chapter 5)

Ferdowsi University of Mashhad

Fall 2011

Hamid Noori

hnoori@um.ac.ir

http://ece.ut.ac.ir/faculty/noori/

Outline

- Introduction
- Building a Datapath
- Designing the Control Unit
- A Single Cycle Implementation
- A Multicycle Implementation
- Designing the Control Unit for the Multicycle Implementation
- Exceptions

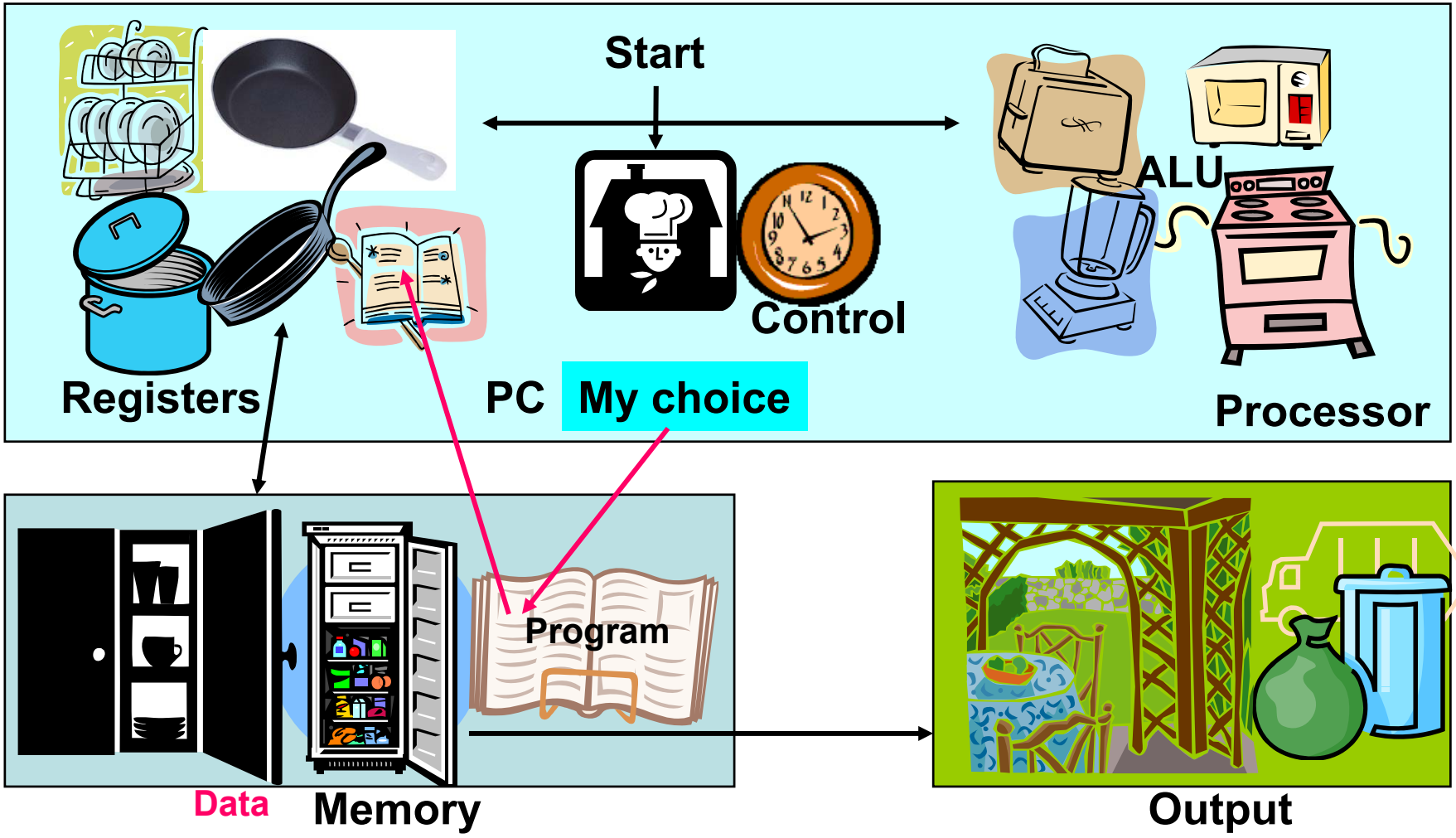
Outline

- Introduction
- Building a Datapath
- Designing the Control Unit
- A Single Cycle Implementation
- A Multicycle Implementation
- Designing the Control Unit for the Multicycle Implementation
- Exceptions

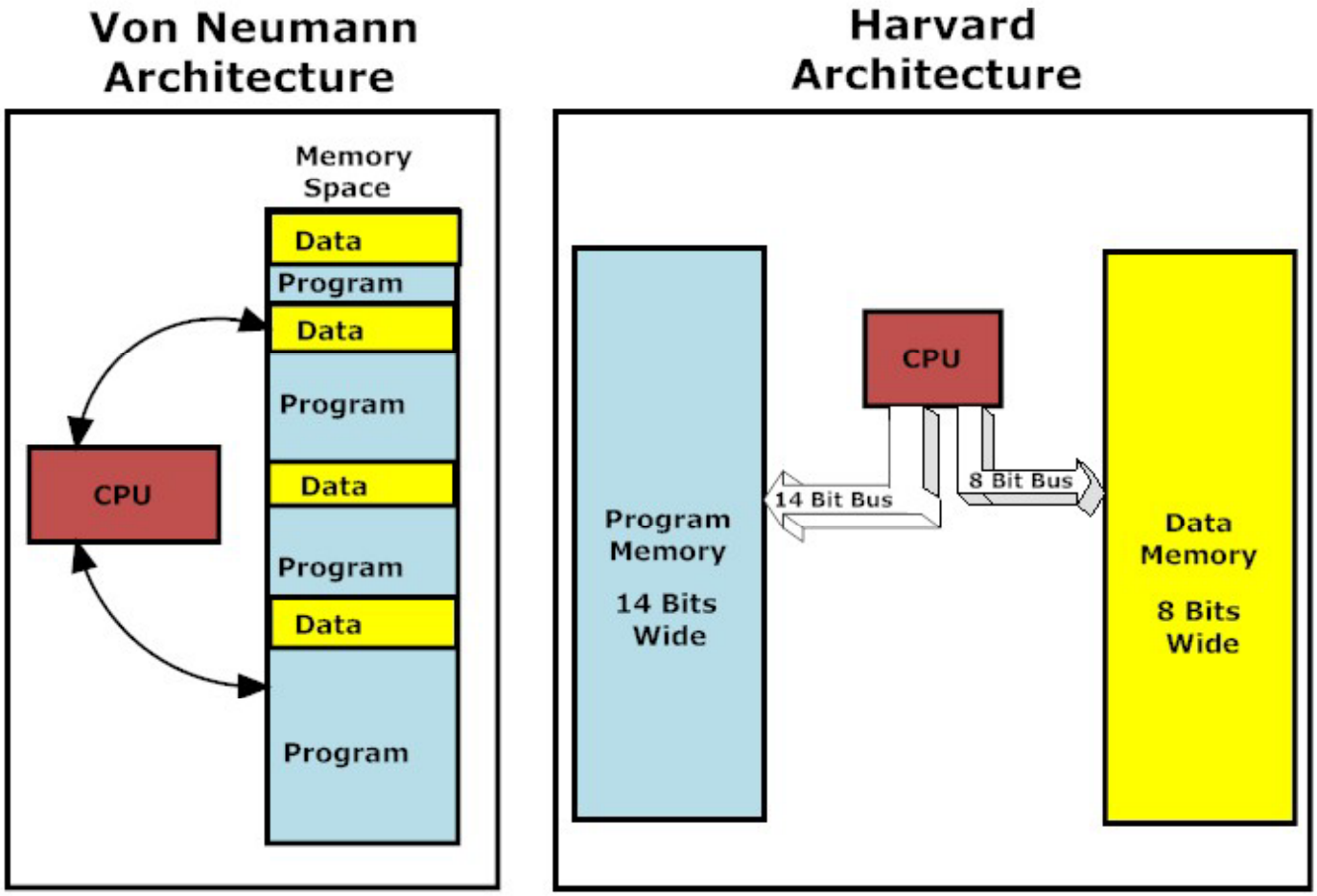
A Basic MIPS Implementation

- A subset of MIPS
  - Memory-reference instructions (lw, sw)
  - Arithmetic-logic instructions (add, sub, and, or, slt)
  - Control-instructions (beq, j)
  - Other integer instructions (shift, multiply, divide) are not supported
  - Floating point instructions are not supported

Von Neumann Kitchen



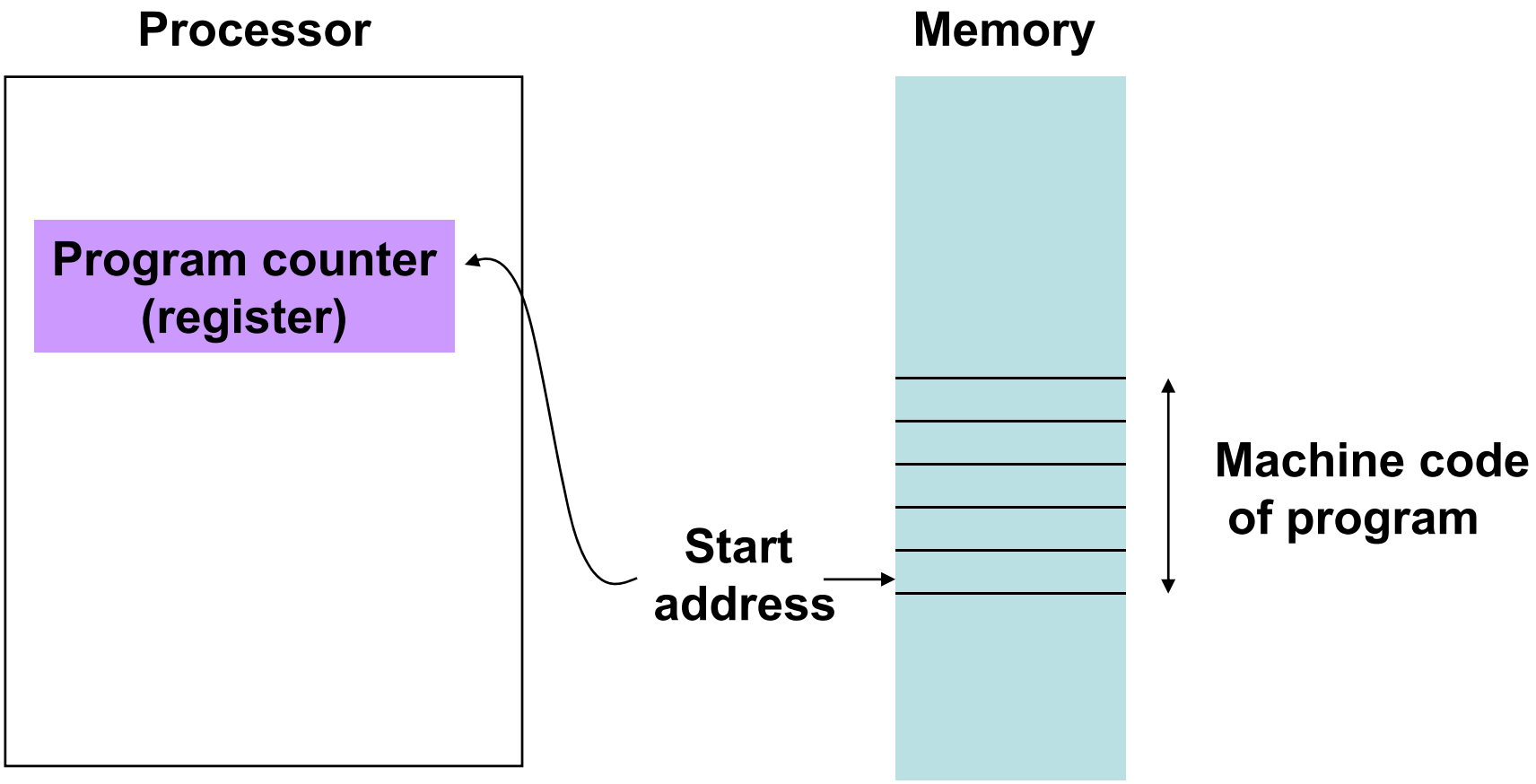
Von Neumann vs. Harvard Architecture



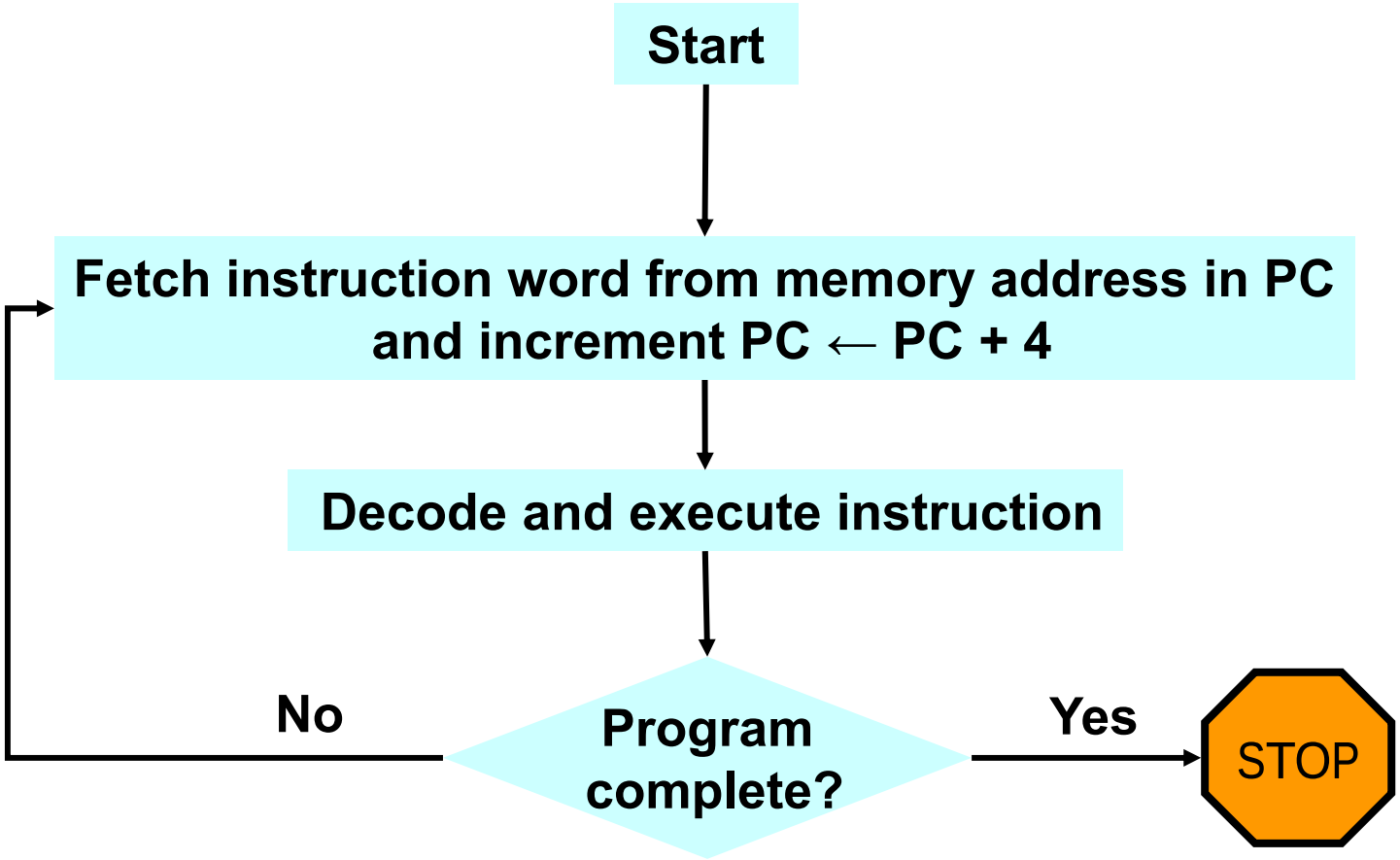
Where Does It All Begin?

- 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.

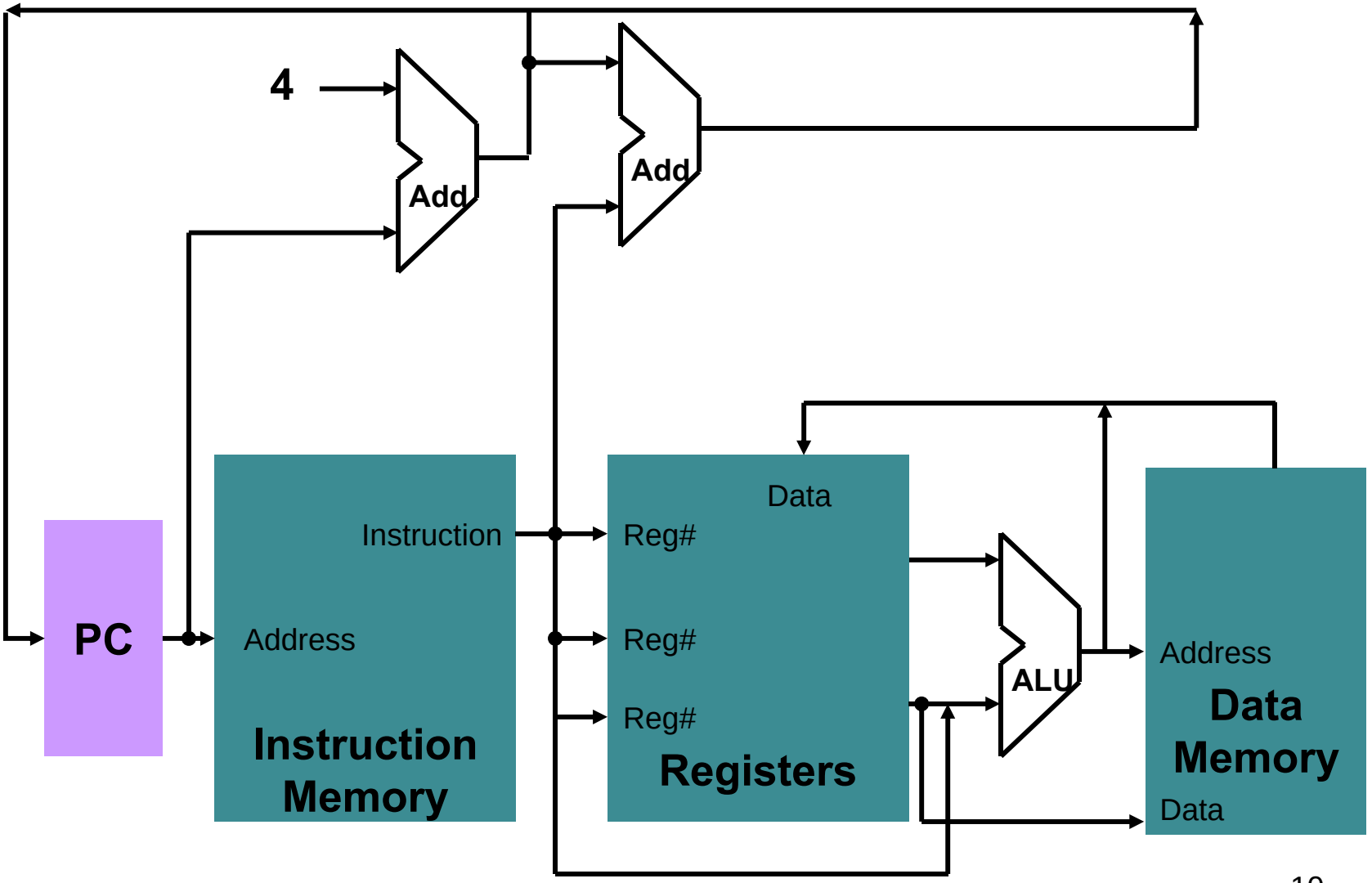
Where is the Program?



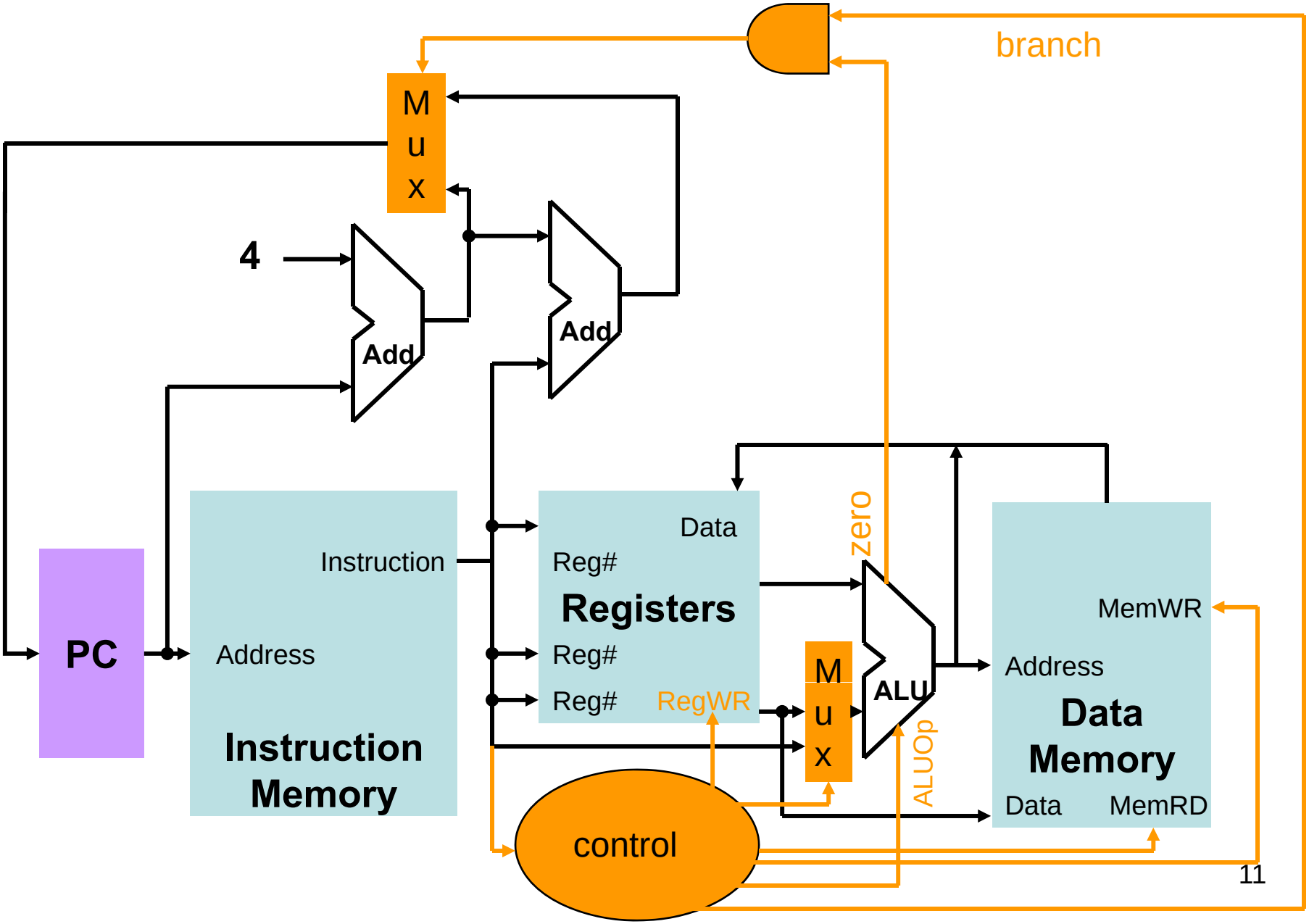
How Does It Run?



An Abstract View of MIPS Implementation



A Basic Implementation of MIPS including Control Signals



Datapath and Control

- **Datapath:** Memory, registers, adders, ALU, and communication buses. Each step (fetch, decode, execute) requires a communication (data transfer) path between memory, registers and ALU.
- Transfer, Execute, Store

- **Control:** Datapath for each step is set up by control signals that set up dataflow directions on communication bus and select ALU function. Control signals are generated by a control unit consisting of one or more finite-state machines.

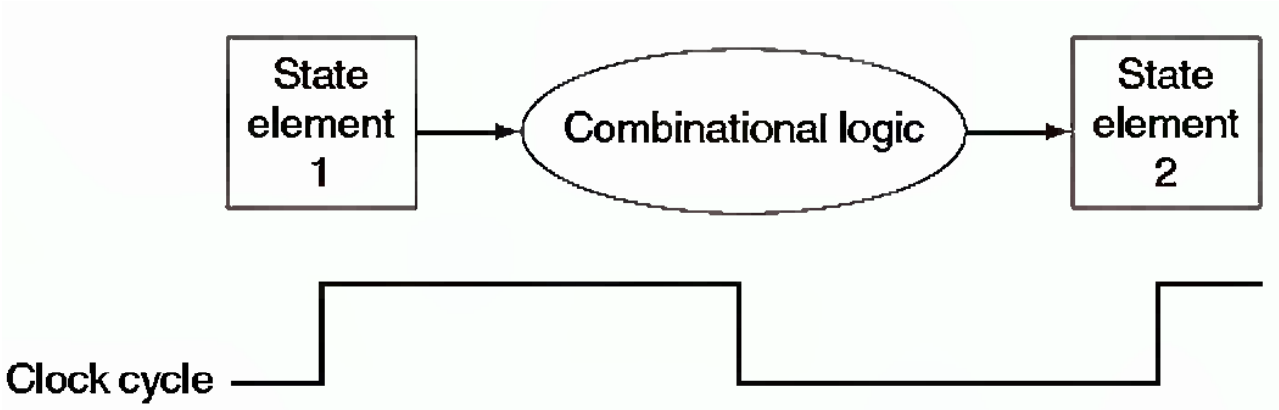
Outline

- Introduction
- Building a Datapath
- Designing the Control Unit
- A Single Cycle Implementation
- A Multicycle Implementation
- Designing the Control Unit for the Multicycle Implementation
- Exceptions

Introduction

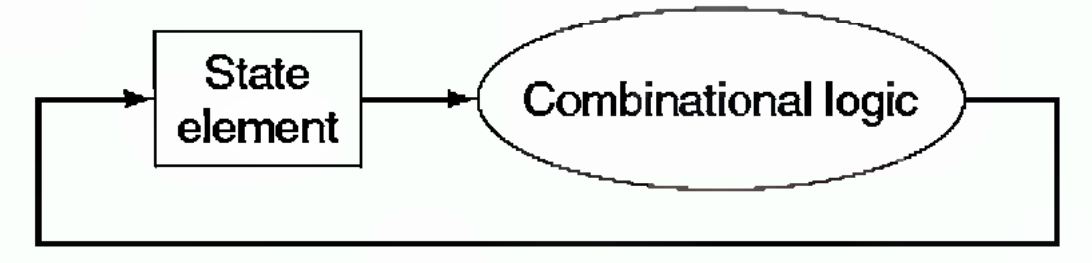
- Functional units of MIPS consist of two types of logic elements:
  - Combinational elements
  - State elements (sequential elements)

Clock Cycle

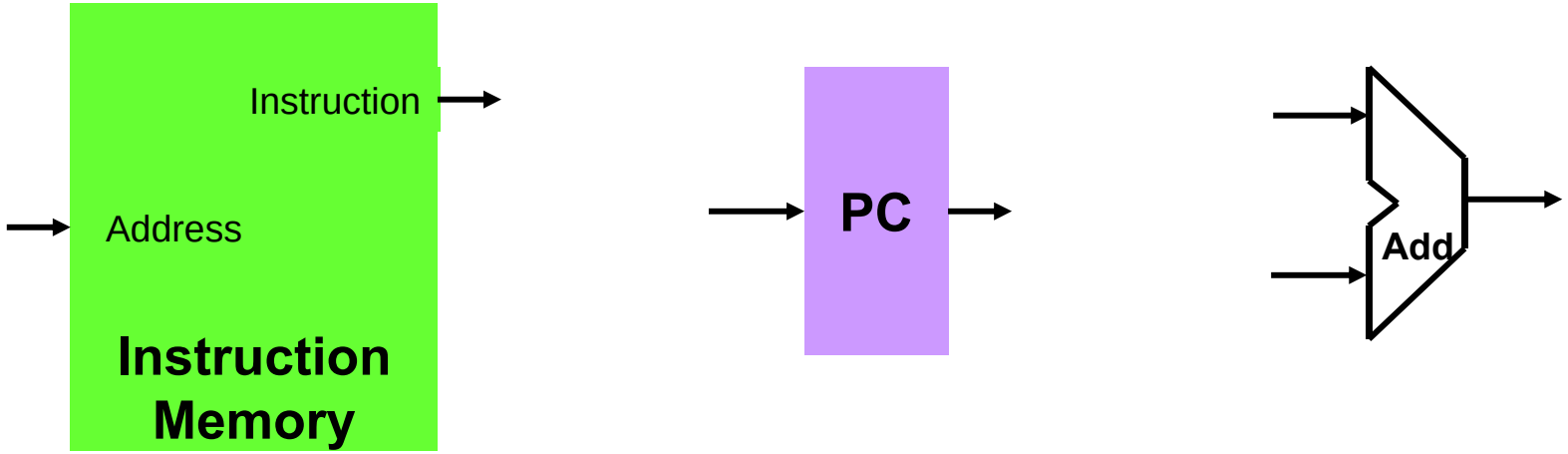

State element 1
Combinational logic
State element 2
Clock cycle

Edge Triggered Clocking

- An edge-triggered clocking methodology means that any values stored in a sequential logic element are updated only on a clock edge.
- An edge-triggered clocking methodology allows to read the contents of a register, send the value through some combinational logic, and write that register in the same clock cycle.



Datapath Elements for Instruction Fetch

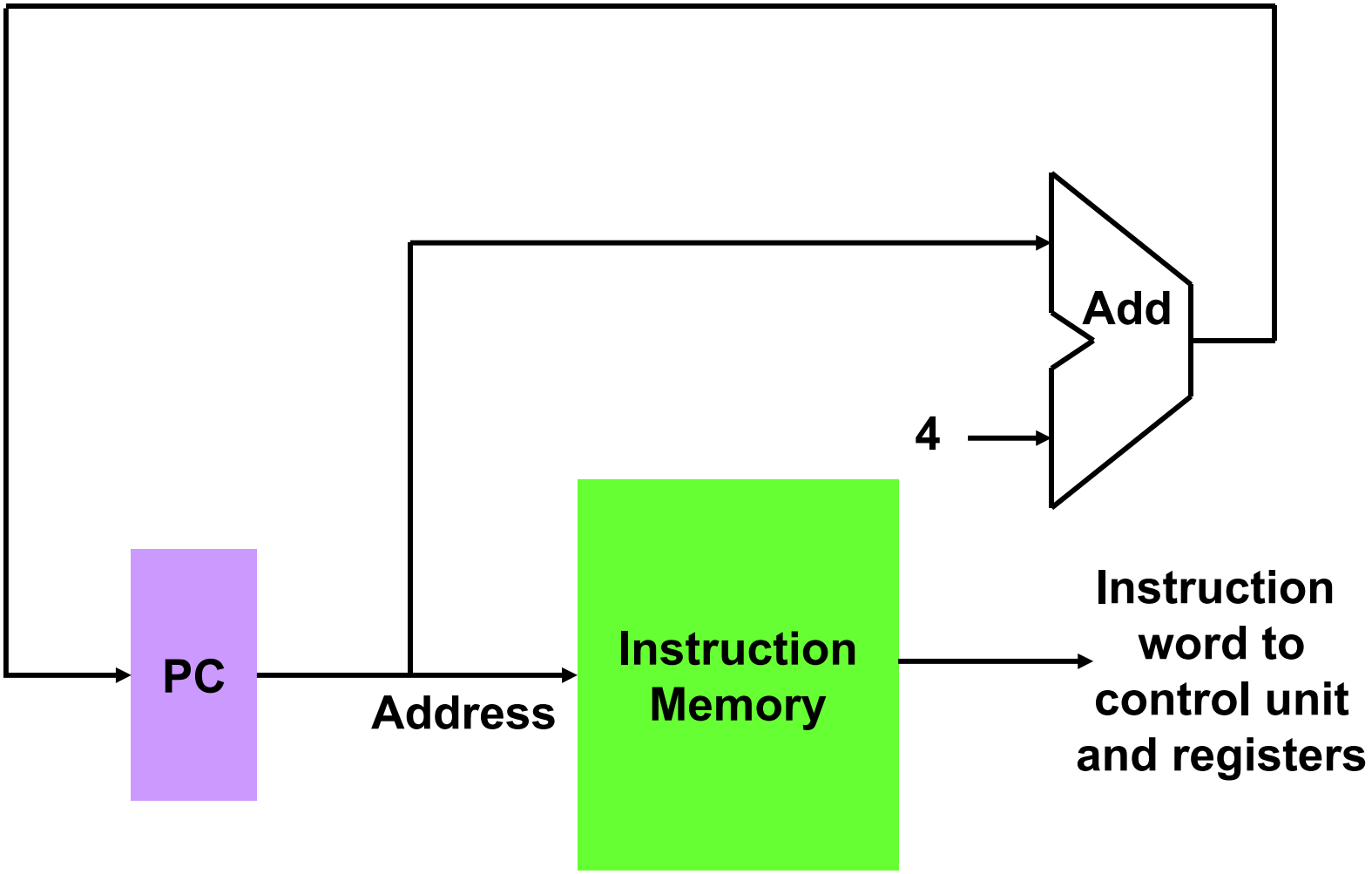


a. Instruction Memory

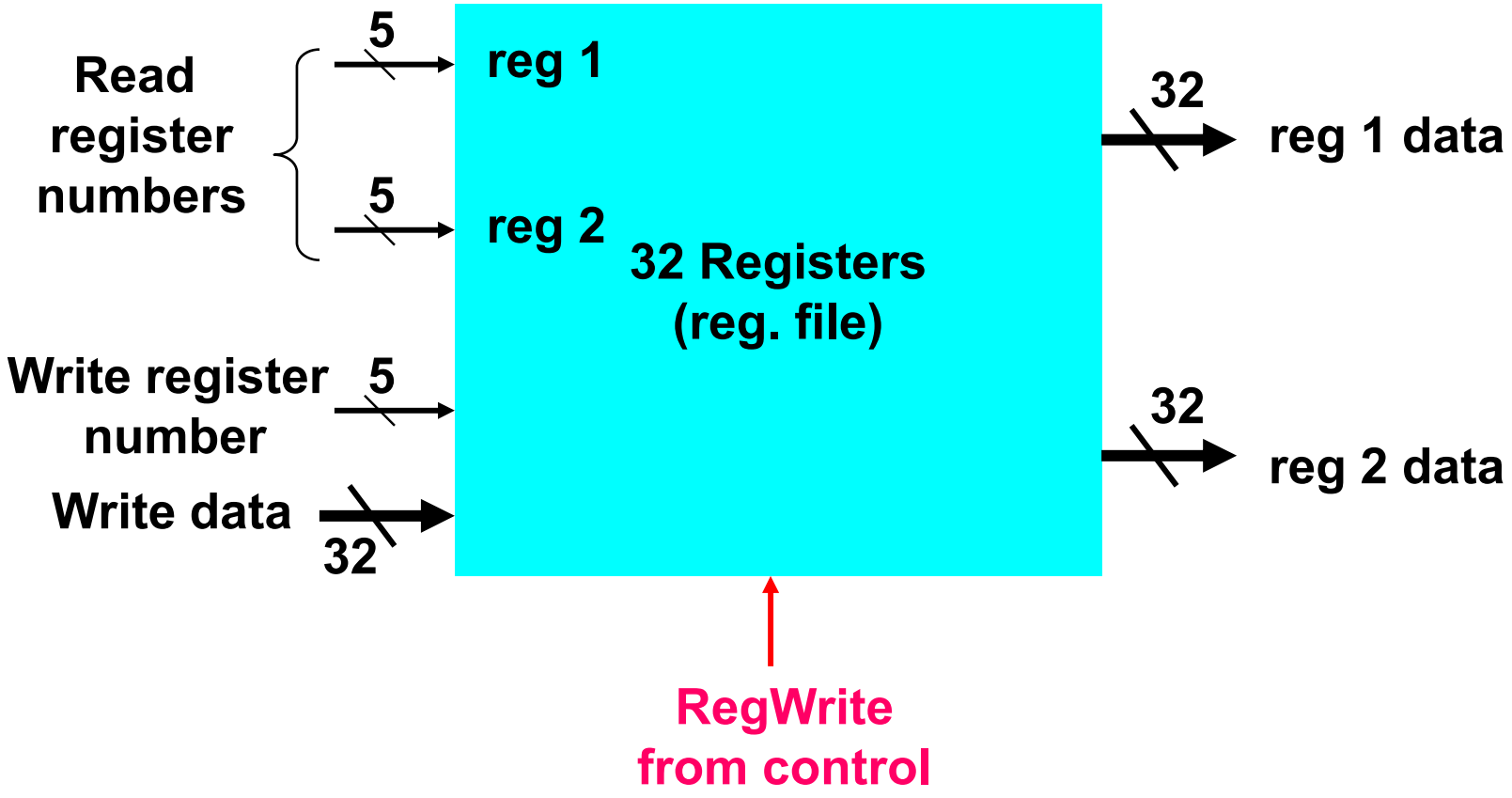
b. Program Counter

c. Adder

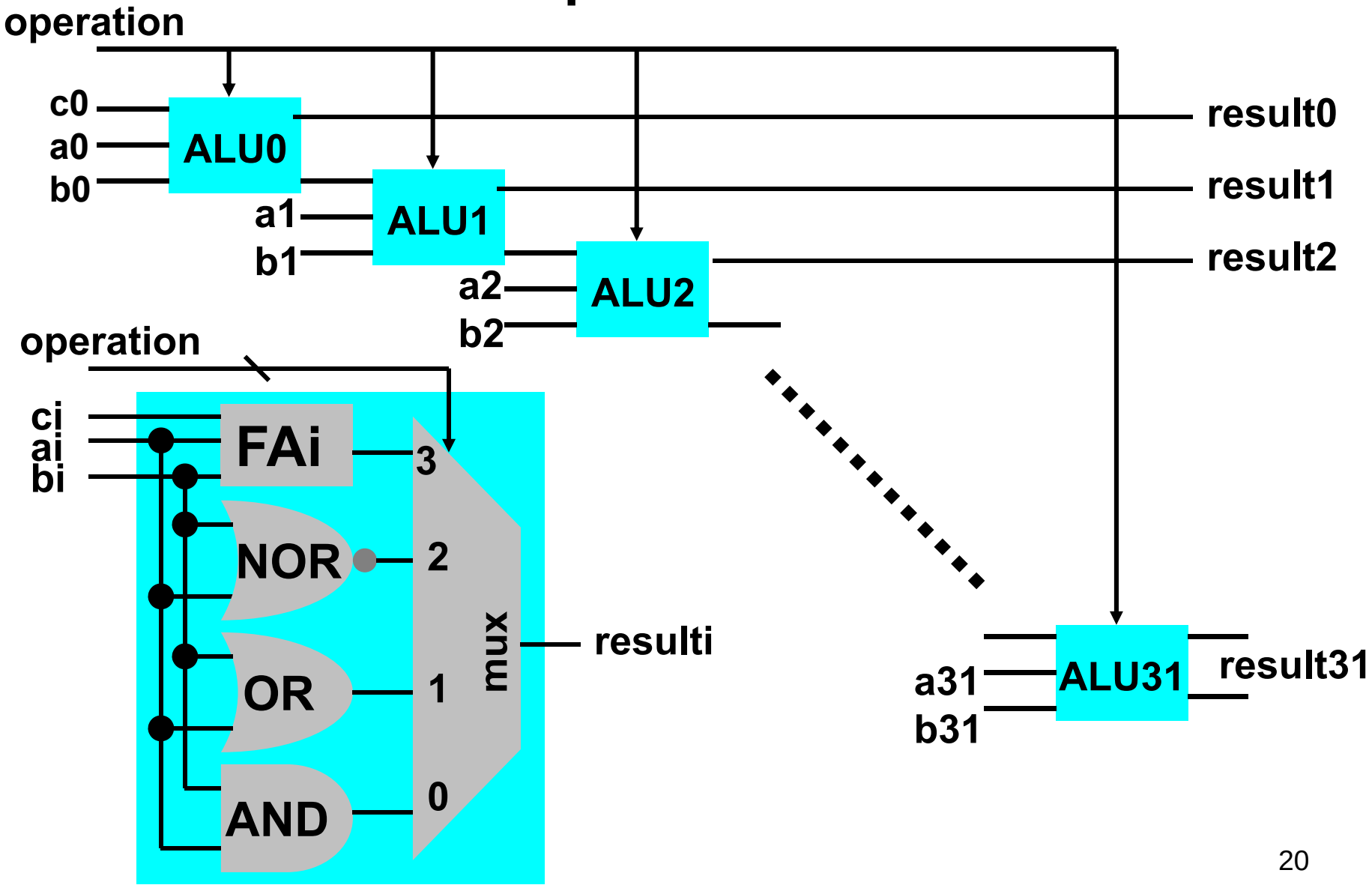
Datapath for Instruction Fetch



Register File

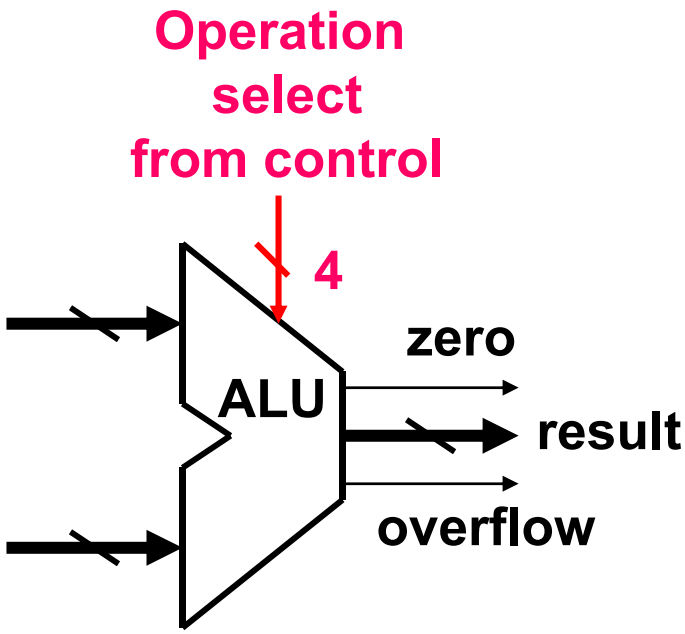


Multi-Operation ALU



Multi-Operation ALU

Operation select	ALU function
0000	AND
0001	OR
0010	Add
0110	Subtract
0111	Set on less than
1100	NOR



zero = 1, when all bits of result are 0

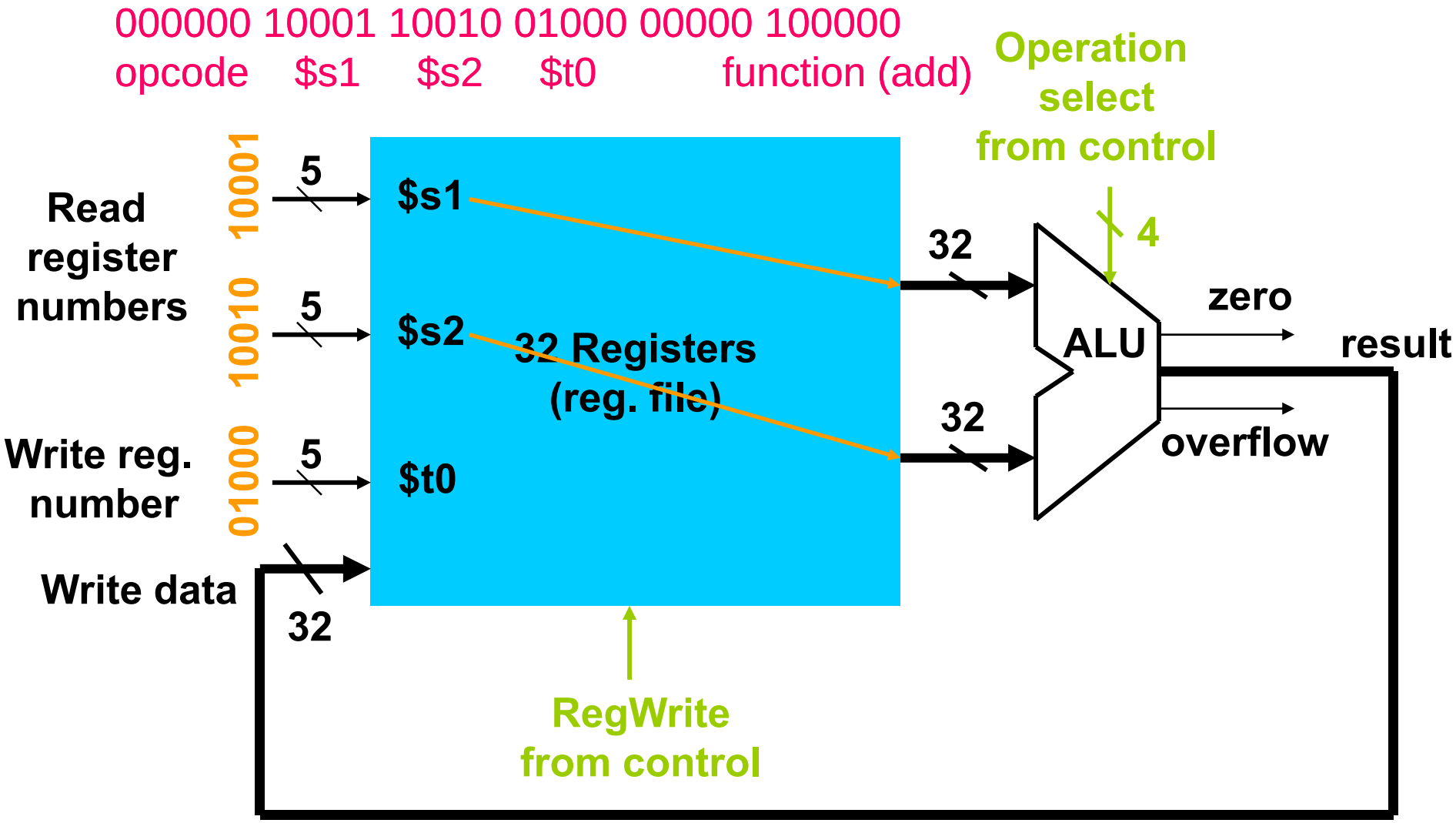
R-Type Instructions

- Also known as arithmetic-logical instructions
  - add, sub, and, or, 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)

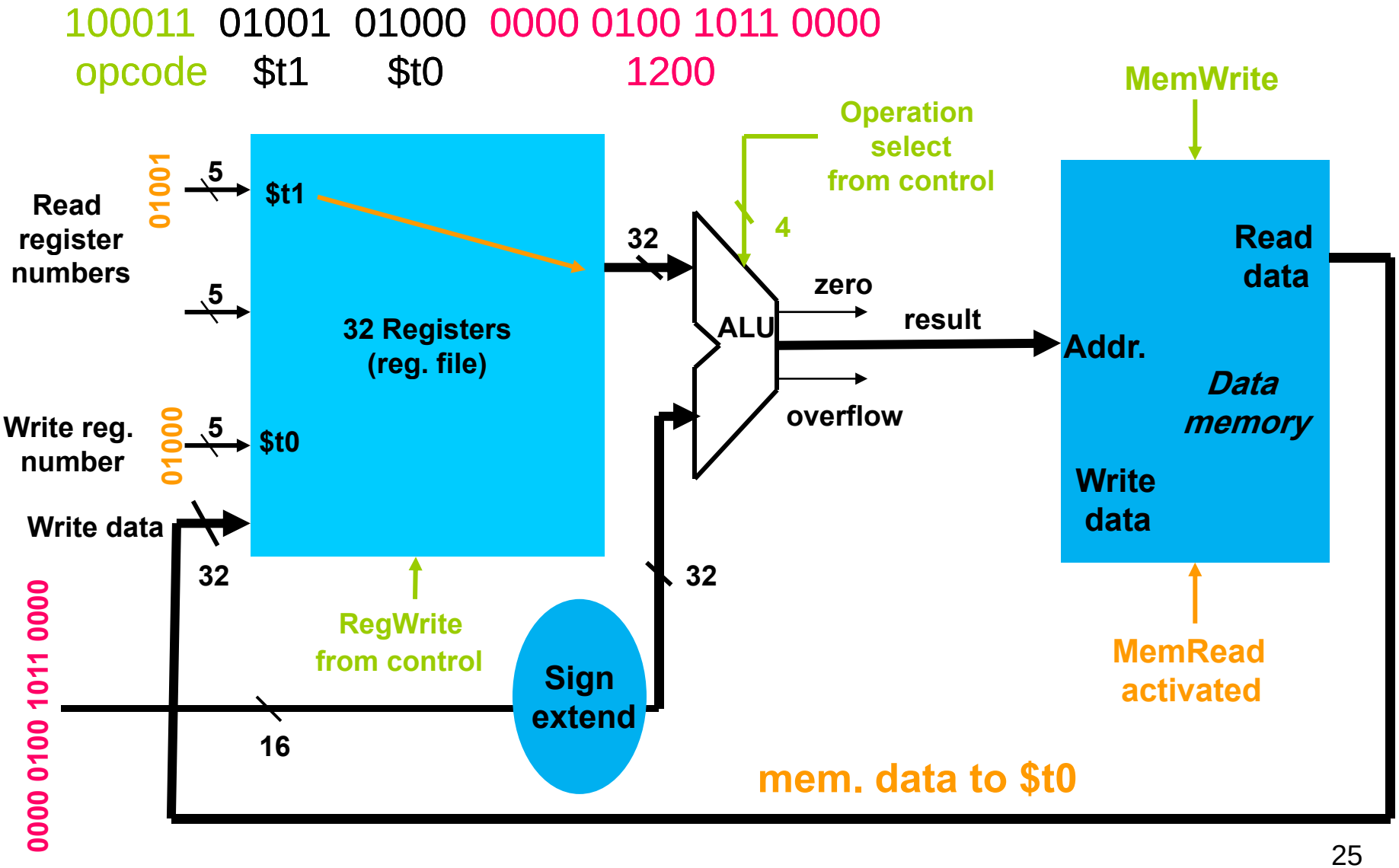
  100011 01001 01000 0000 0100 1011 0000

  opcode $t1 $t0 1200
- sw $t0, 1200 ($t1)

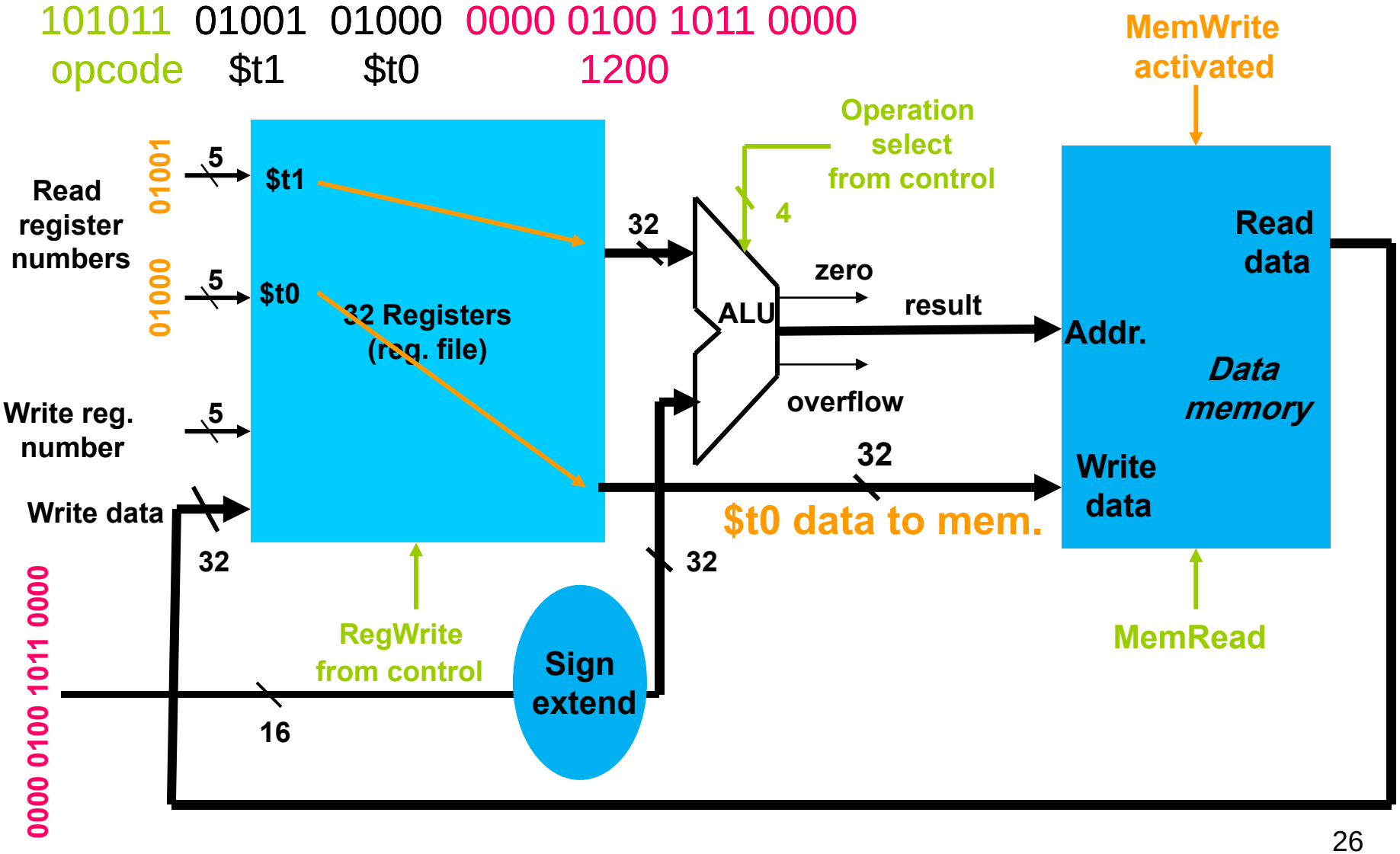
  101011 01001 01000 0000 0100 1011 0000

  opcode $t1 $t0 1200

Datapath for “lw” Instruction

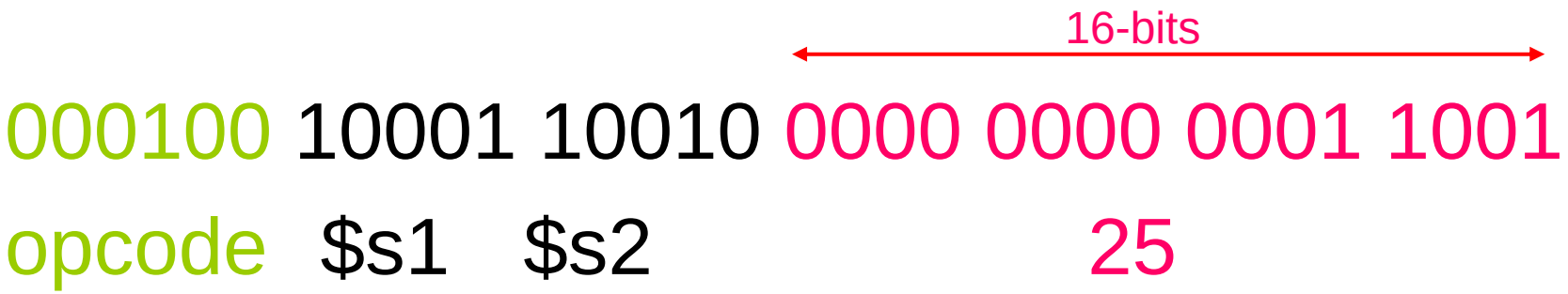


Datapath for “sw” Instruction



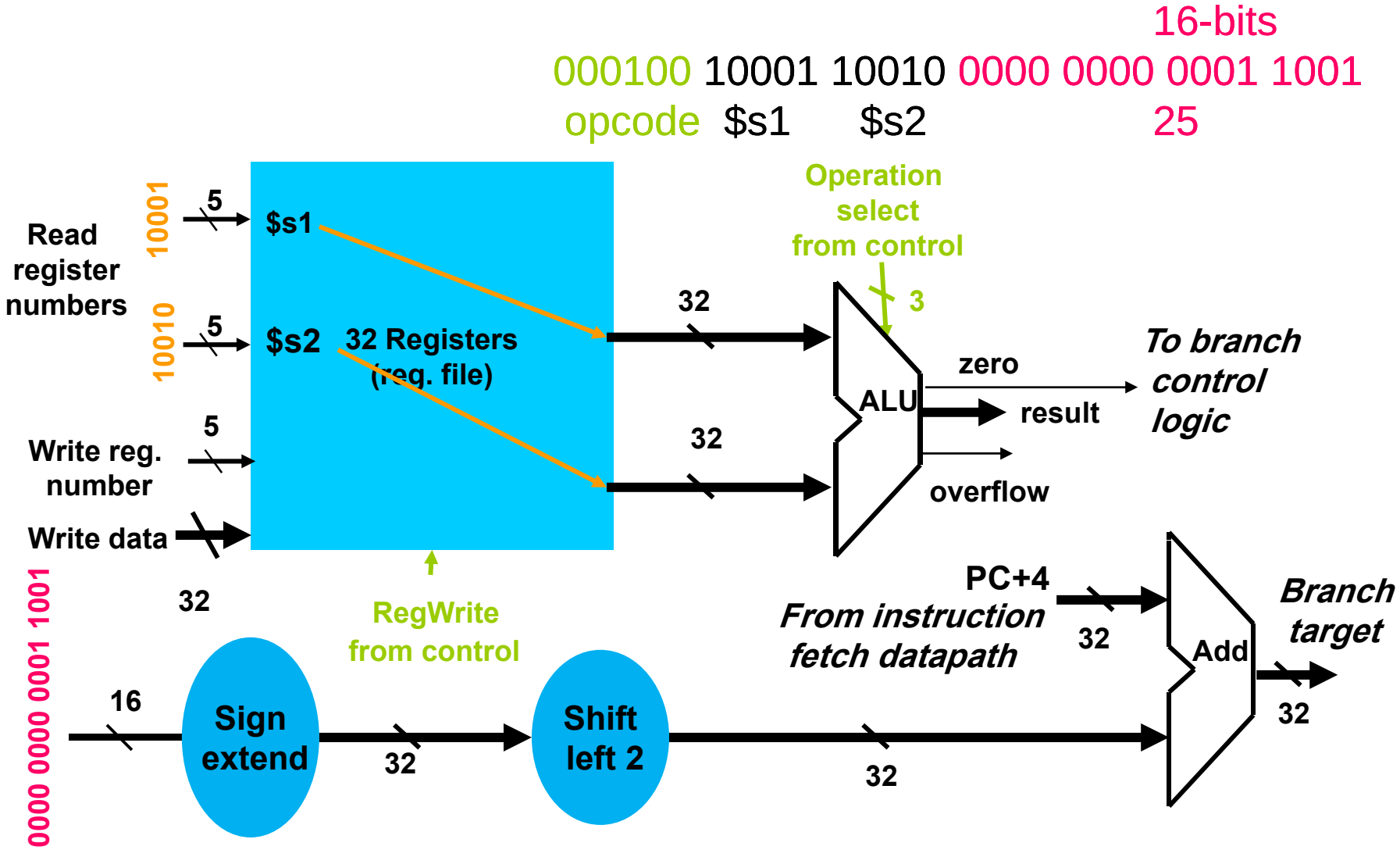
Branch Instruction (I-Type)

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



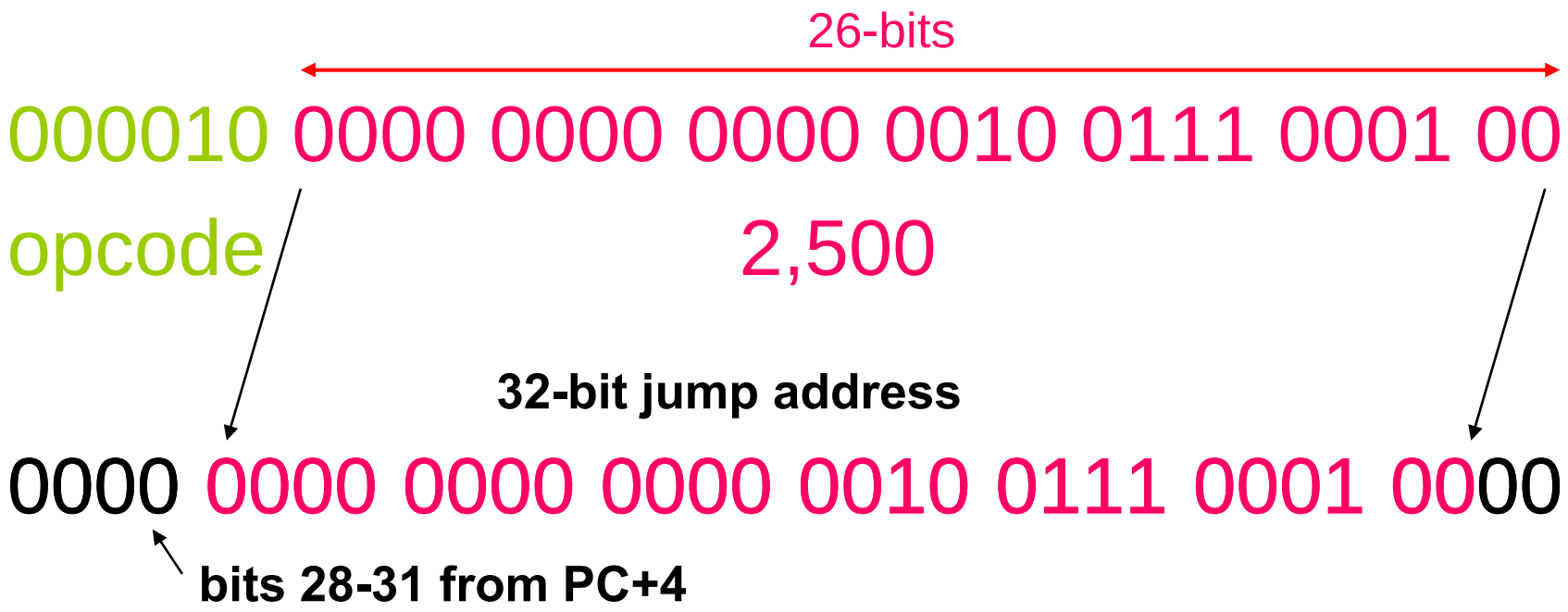
Note: Can branch within ± 2^{15} words from the current instruction address in PC.

Datapath for “beq” Instruction

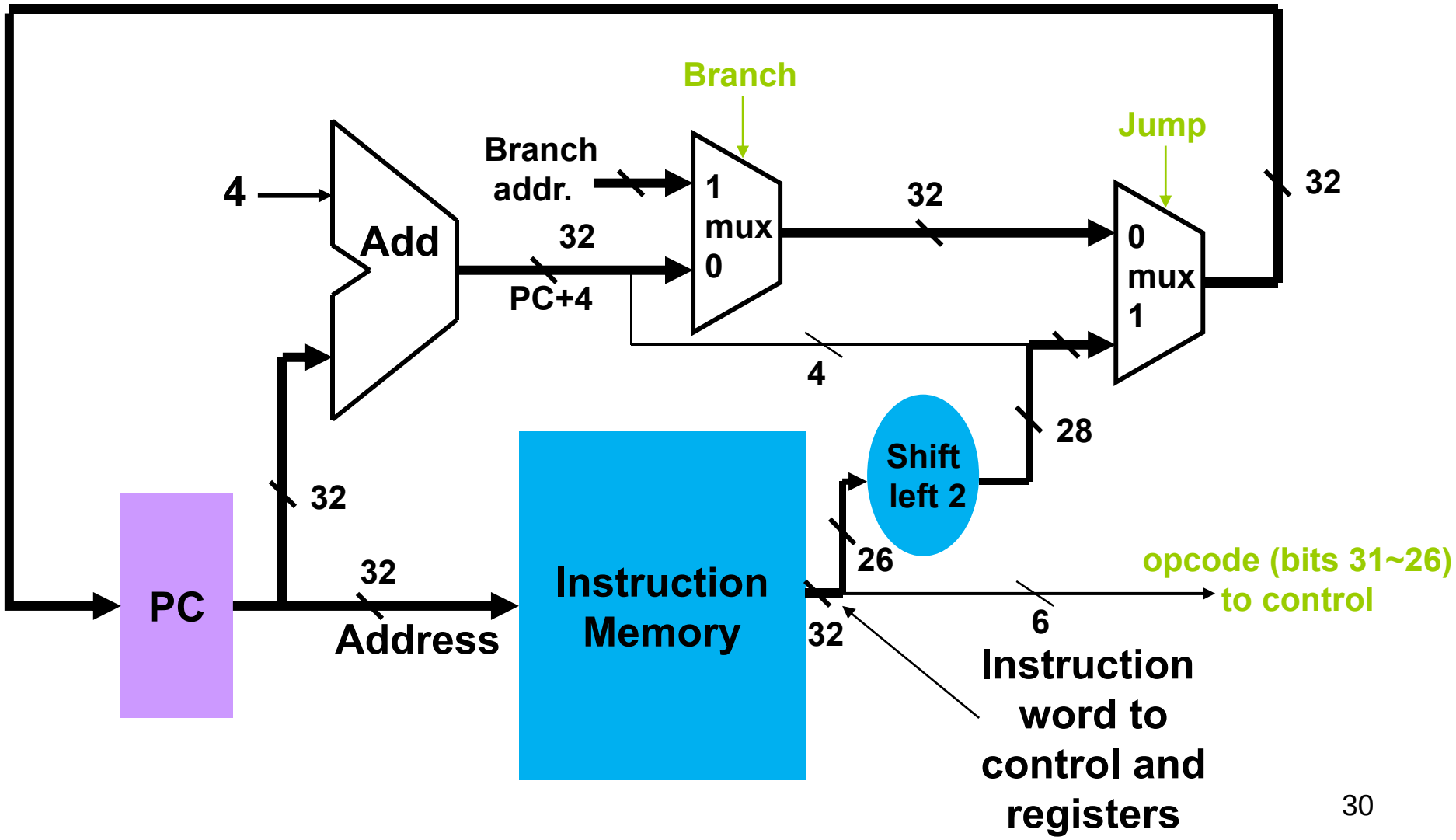


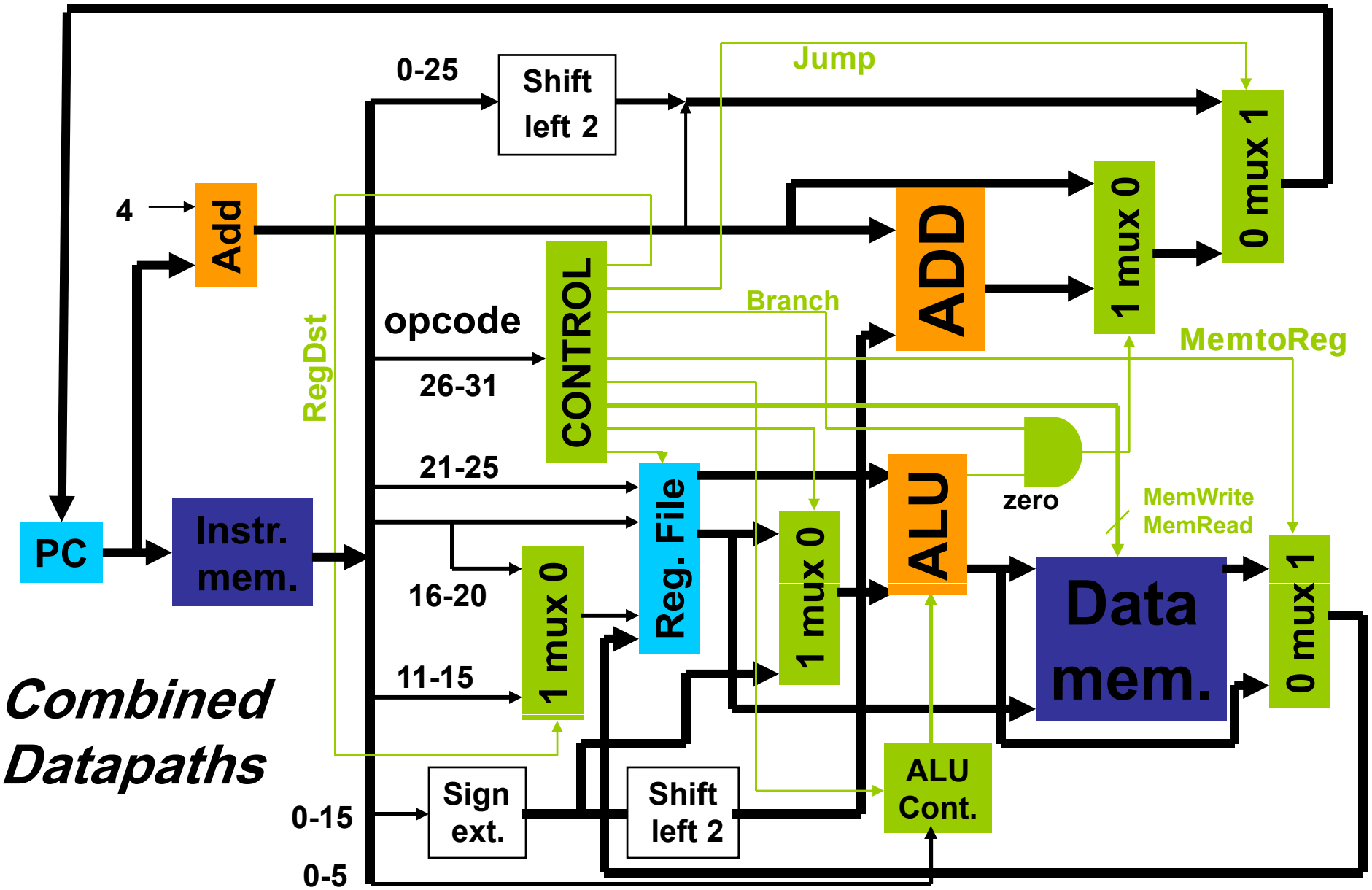
J-Type Instruction

- j 2500 # jump to instruction 2,500



Datapath for Jump Instruction





Combined Datapaths

Outline

- Introduction
- Building a Datapath
- Designing the Control Unit
- A Single Cycle Implementation
- A Multicycle Implementation
- Designing the Control Unit for the Multicycle Implementation
- Exceptions

The ALU Control

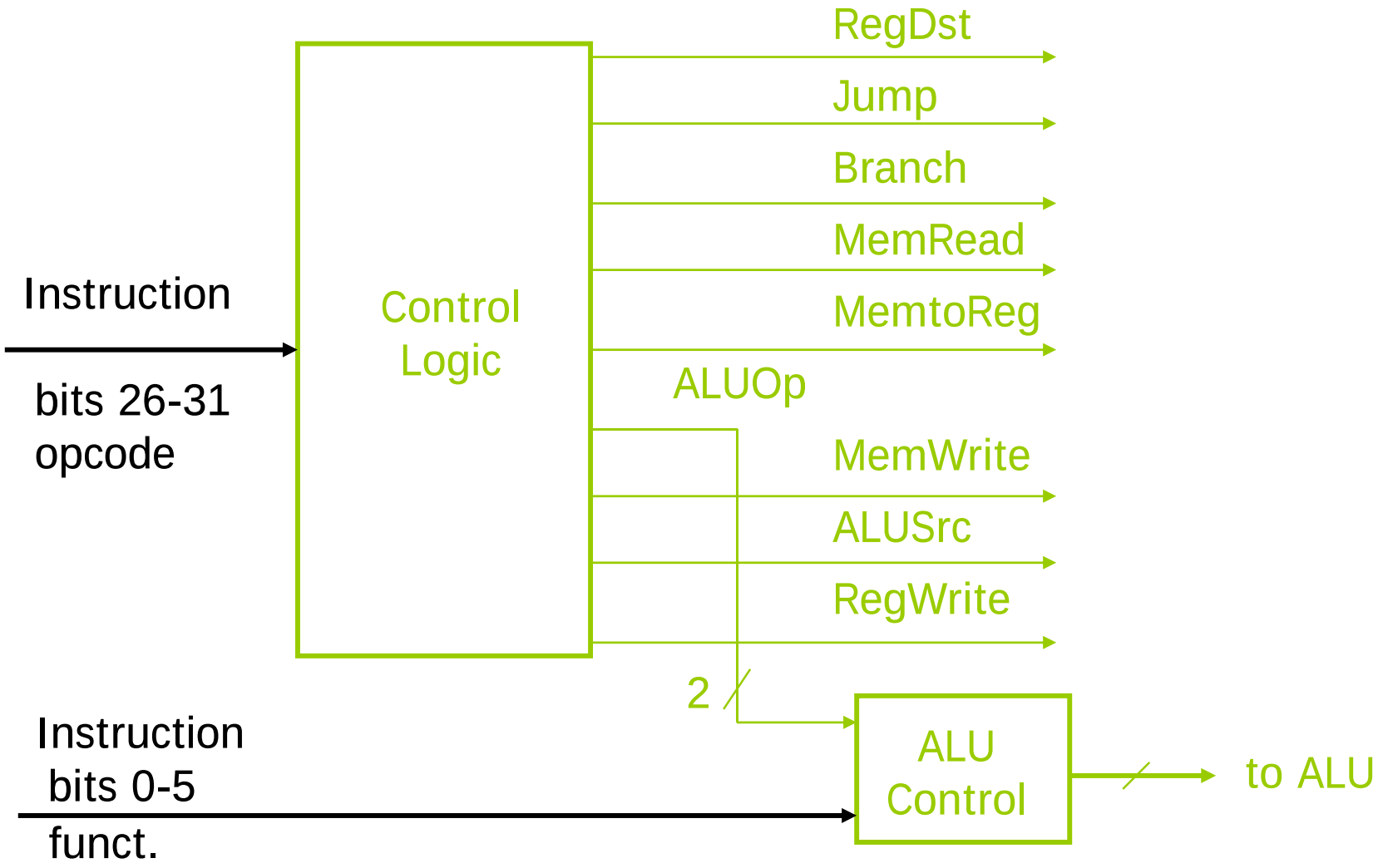
ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	and	0000
R-type	10	OR	100101	or	0001
R-type	10	set on less than	101010	set on less than	0111

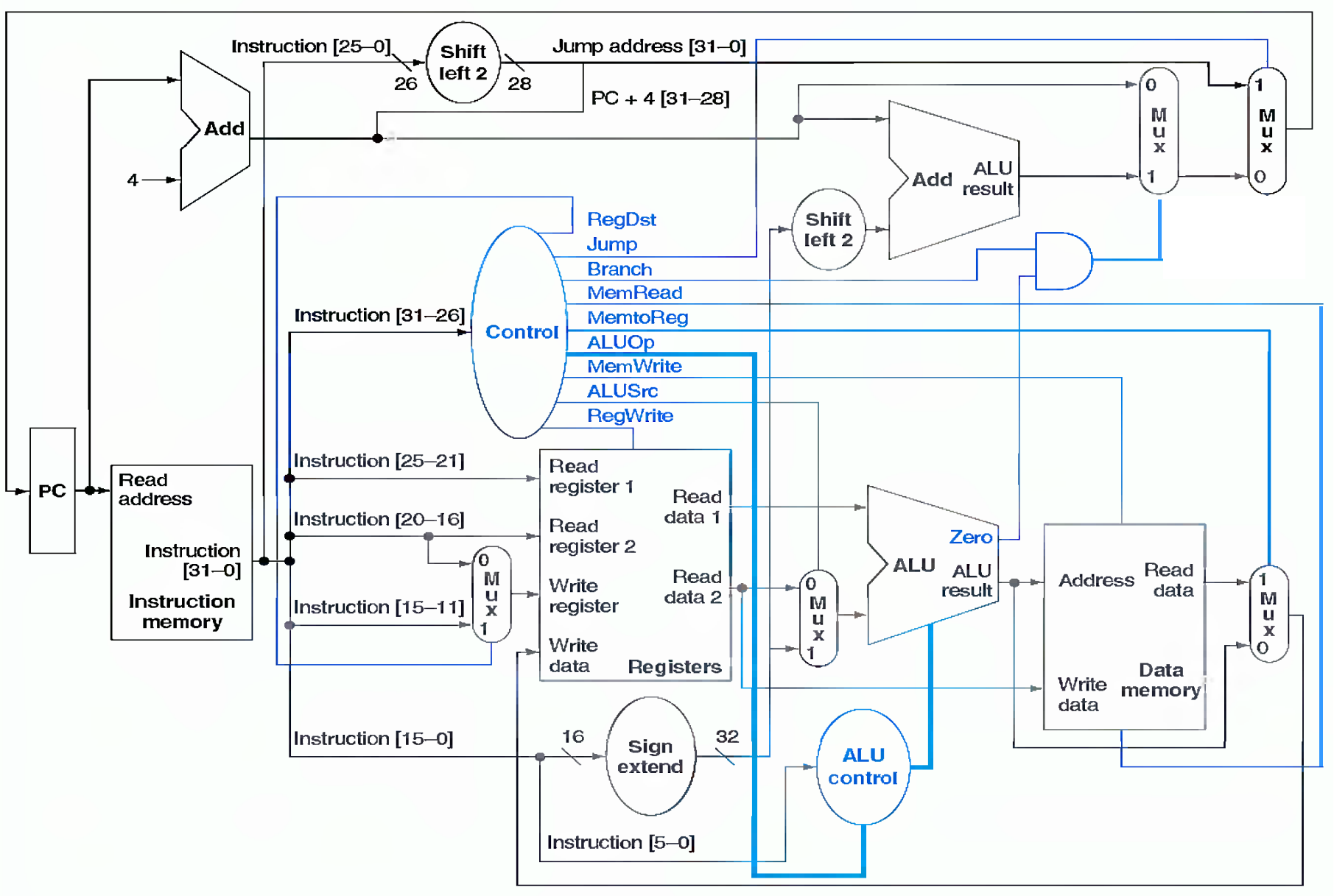
The ALU Control

ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

Control


RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction
bits 26-31
opcode
Control
Logic
2
Instruction
bits 0-5
funct.
ALU
Control
to ALU

The Simple Datapath with Control Unit



The Three Instruction Classes

Field	0	rs	rt	rd	shamt	funct
Bit positions	31:26	25:21	20:16	15:11	10:6	5:0

a. R-type instruction

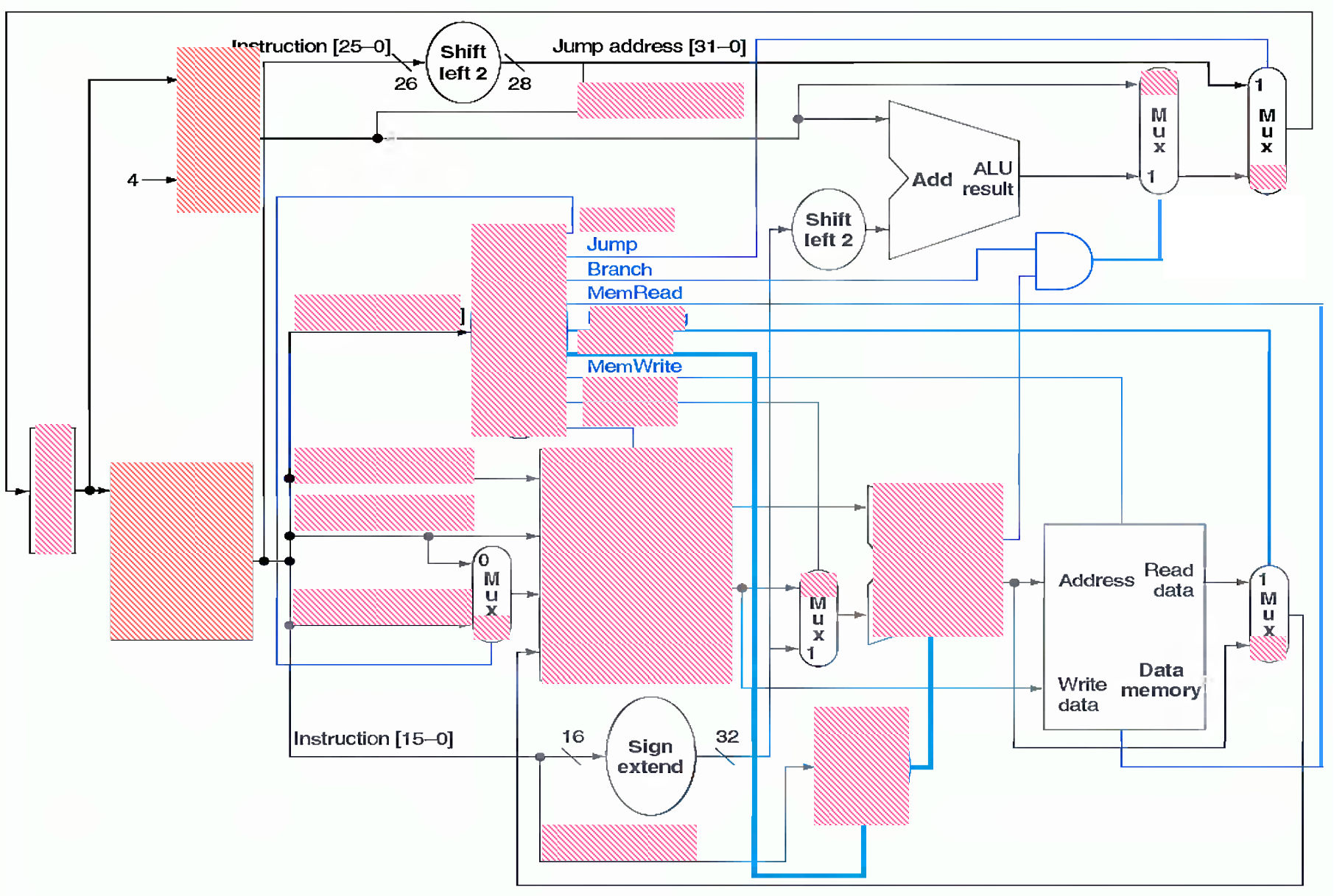
Field	35 or 43	rs	rt	address
Bit positions	31:26	25:21	20:16	15:0

b. Load or store instruction

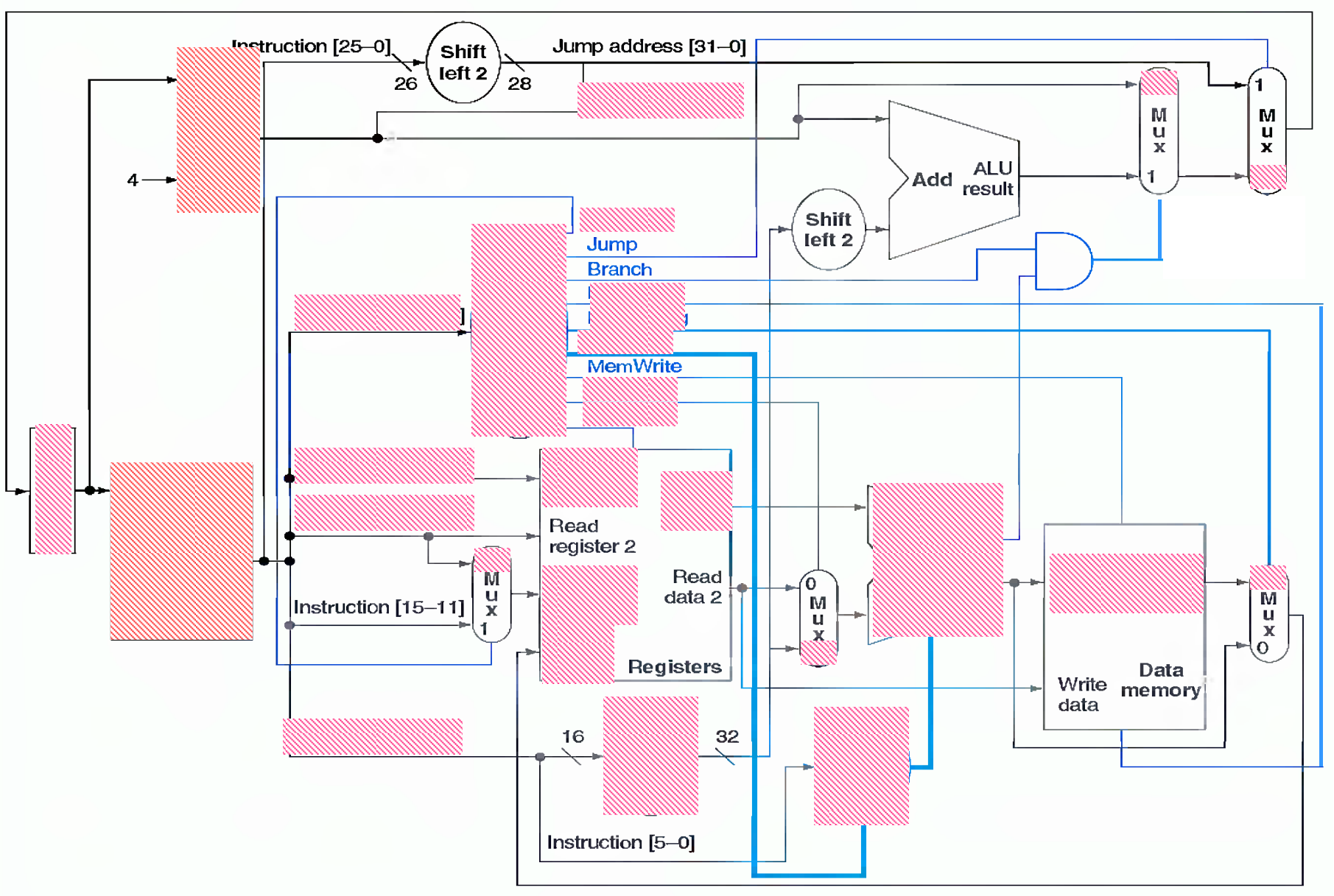
Field	4	rs	rt	address
Bit positions	31:26	25:21	20:16	15:0

c. Branch instruction

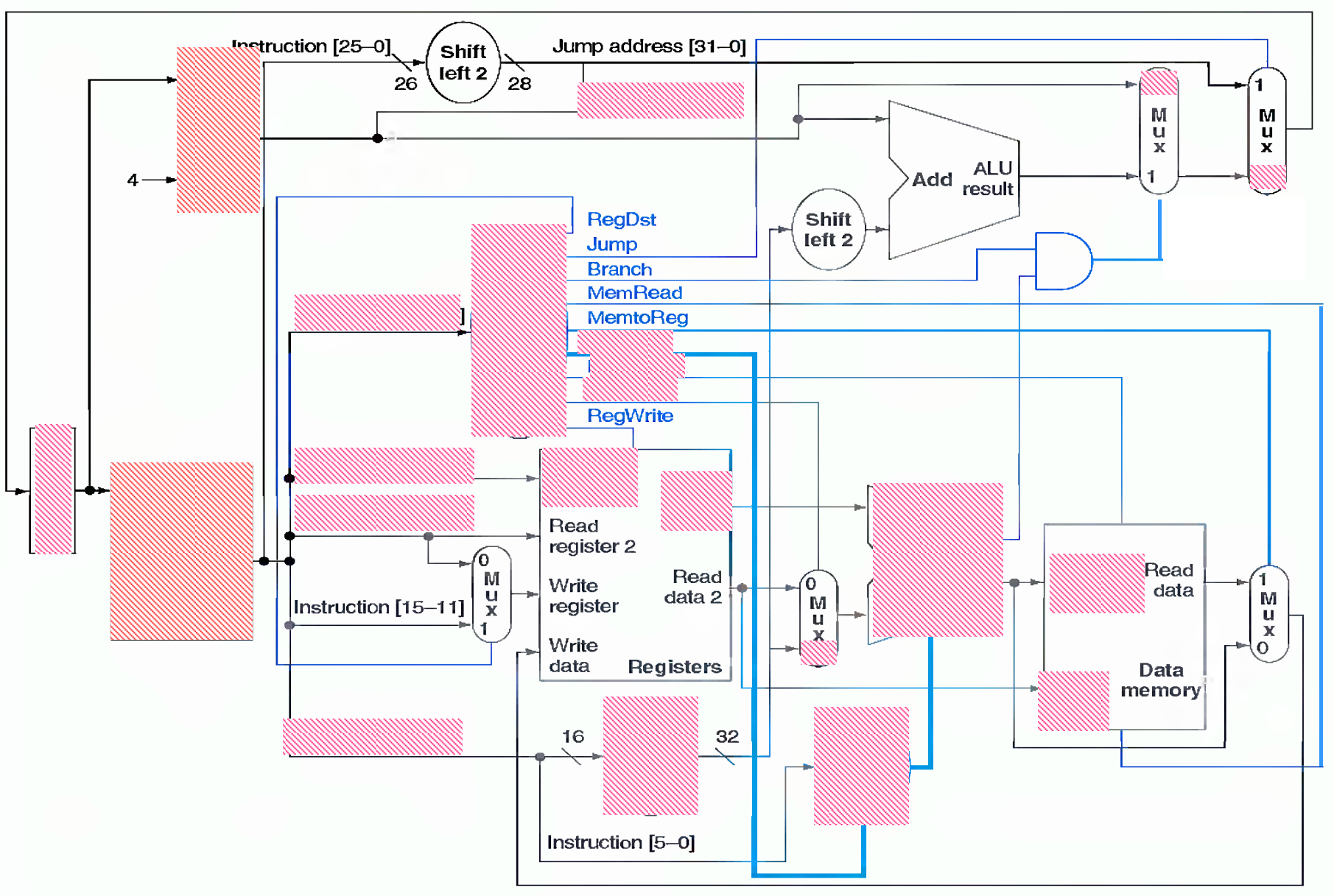
The Datapath in Operation for an R-Type Instruction



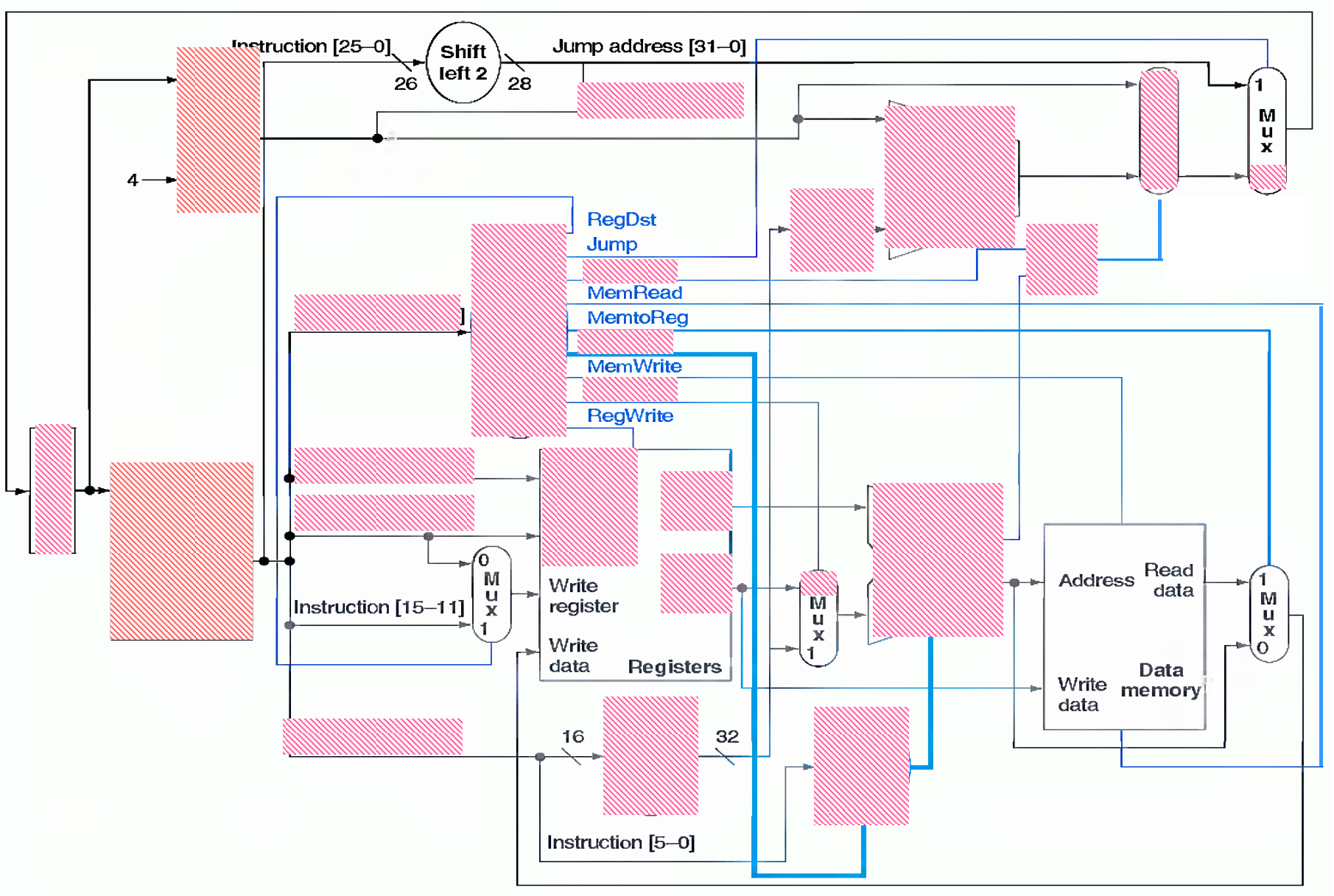
The Datapath in Operation for LOAD Instruction



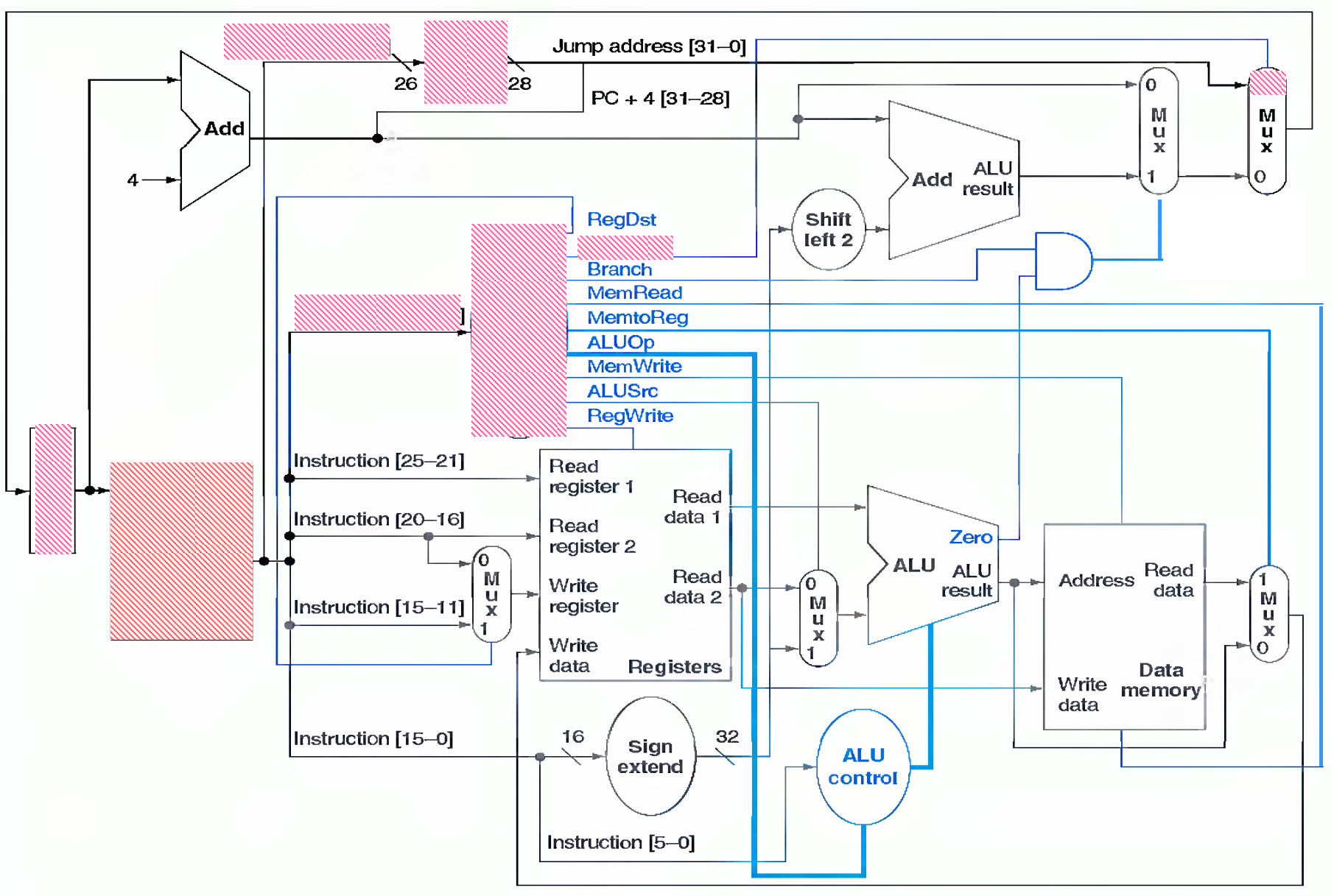
The Datapath in Operation for STORE Instruction



The Datapath in Operation for BEQ Instruction



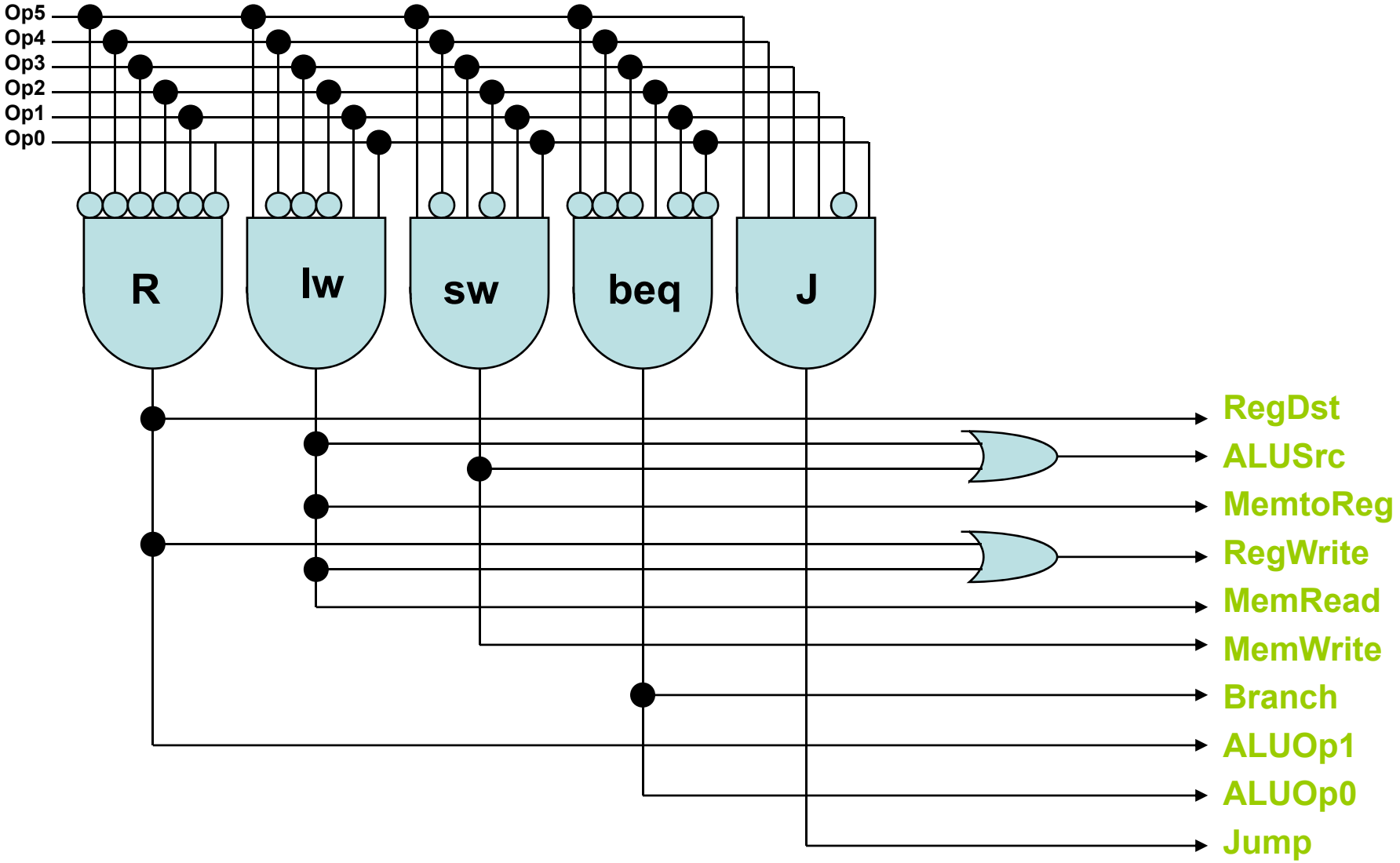
The Datapath in Operation for J Instruction



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	0	X	0	X	X	X

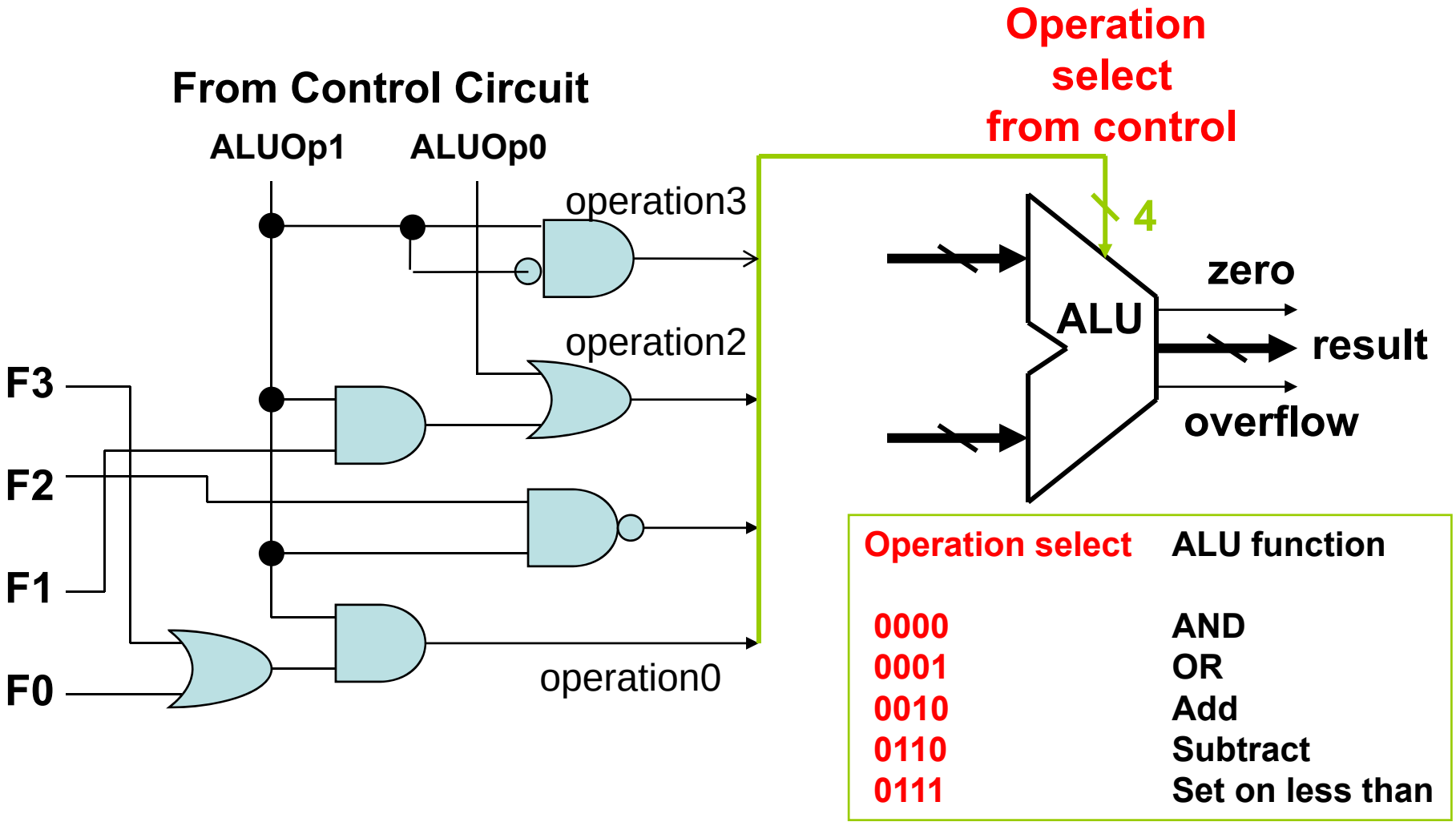
Single-Cycle Control Circuit



ALU Control Logic

Inputs								Outputs to ALU	
From CU		Funct. Code from IR (bits 0-5)						3-bit code	Opera-tion
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0		
0	0	X	X	X	X	X	X	0010	Add
X	1	X	X	X	X	X	X	0110	Subtract
1	X	X	X	0	0	0	0	0010	Add
1	X	X	X	0	0	1	0	0110	Subtract
1	X	X	X	0	1	0	0	0000	AND
1	X	X	X	0	1	0	1	0001	OR
1	X	X	X	1	0	1	0	0111	slt

ALU Control



Operation select	ALU function
0000	AND
0001	OR
0010	Add
0110	Subtract
0111	Set on less than

Outline

- Introduction
- Building a Datapath
- Designing the Control Unit
- A Single Cycle Implementation
- A Multicycle Implementation
- Designing the Control Unit for the Multicycle Implementation
- Exceptions

How Long Does It Take?

- Assume control logic is fast and does not affect the critical timing.
- Major time components are
  - ALU (2ns)
  - memory read/write (2ns)
  - register read/write (1ns)
- 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

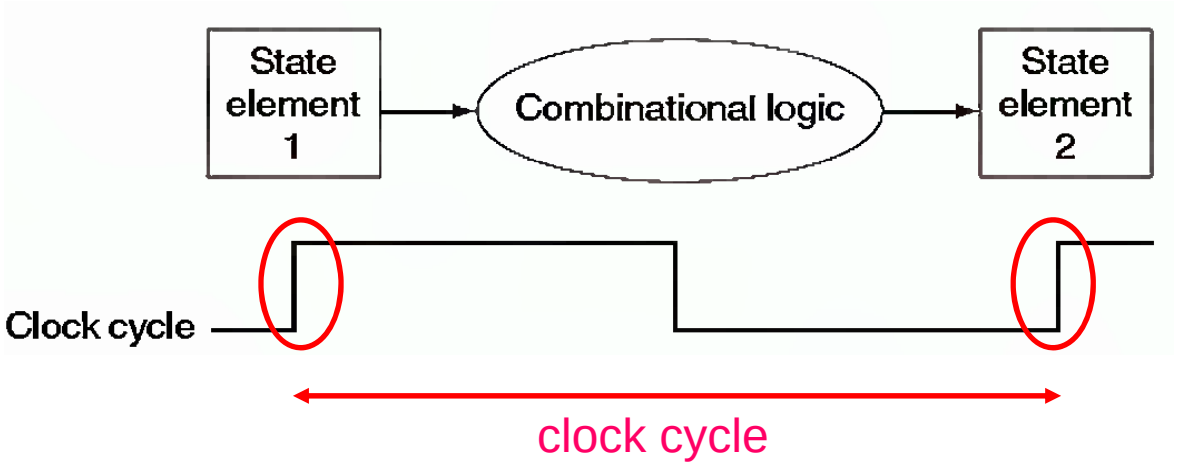
Time for beq (I-Type)

- 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?



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.
  - 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.