5118007-02 Computer Architecture

Ch. 4 The Processor

6 May 2024

Shin Hong

Basic MIPS Implementation

- Let's discuss with a core subset of MIPS instructions
  - load word (lw), store word (sw)
  - add, sub, AND, OR and slt
  - branch equal (beq)

- Executing an instruction
  - send PC to the memory to fetch next instruction
  - read one or two registers based on the instruction
  - (optional) perform instruction-specific operations
    - e.g., use ALU to perform arithmetic operations on different registers

Instruction Formats

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

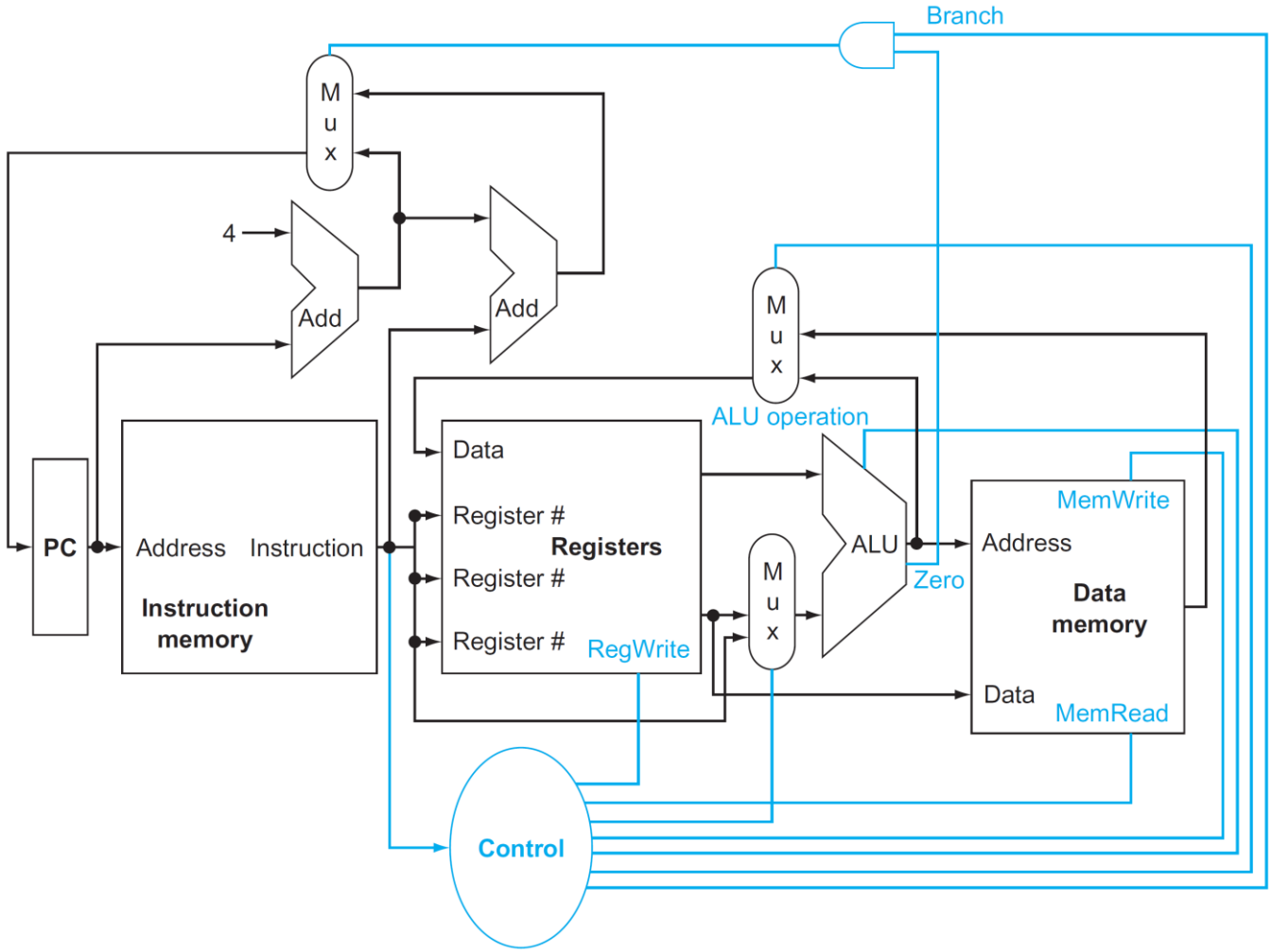
Field	35 or 43	rs	rt	address
Bit positions	31:26	25:21	20:16	15:0

b. Load or store instruction

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

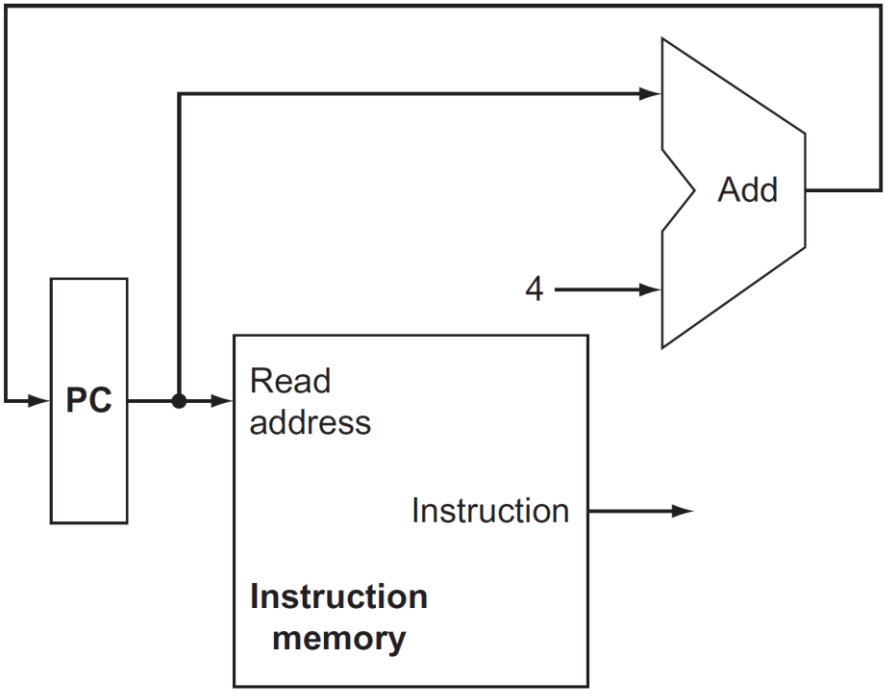
c. Branch instruction

Abstract View of a Single Cycle MIPS Processor



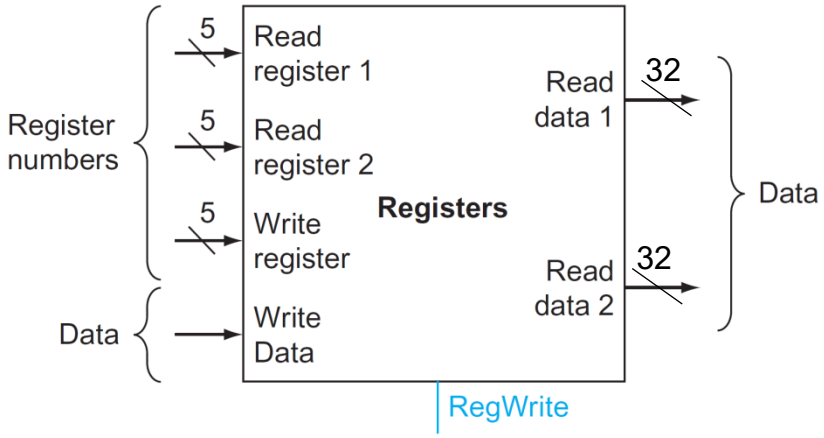
- `add $t1, $t2, $t3`
- `lw, $t1, 4($t2)`
- `beq, $s1, $s2, 32`
- `j 14428`

Fetching Instructions



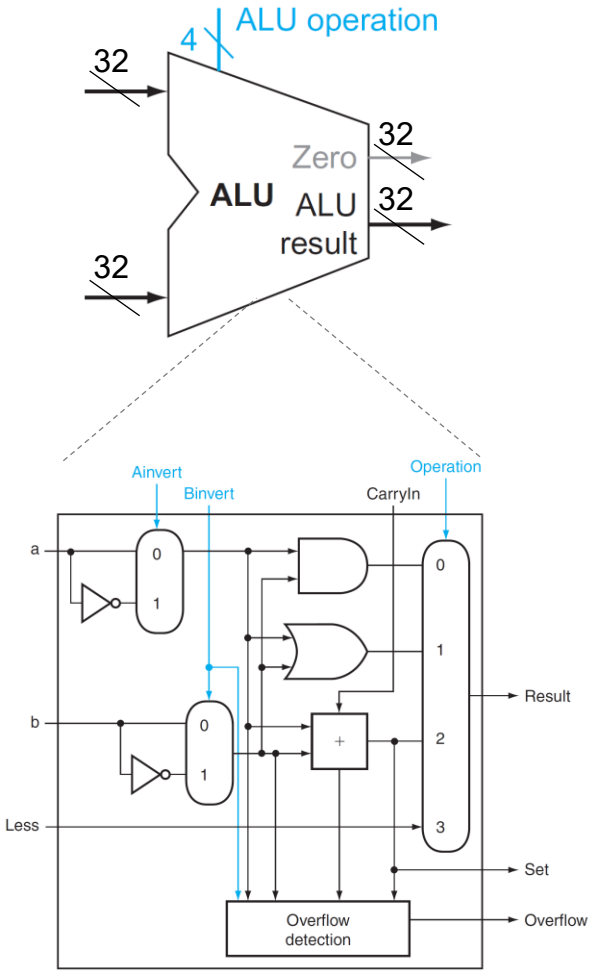
- To execute any instruction, the instruction must be fetched from memory
- To prepare for executing next instruction, the PC must be incremented

Register Files



- Register file is a collection of registers where a register is read/written by specifying its number
  - a R-type instruction gives two read register and one write register
  - RegWrite is edge-triggered

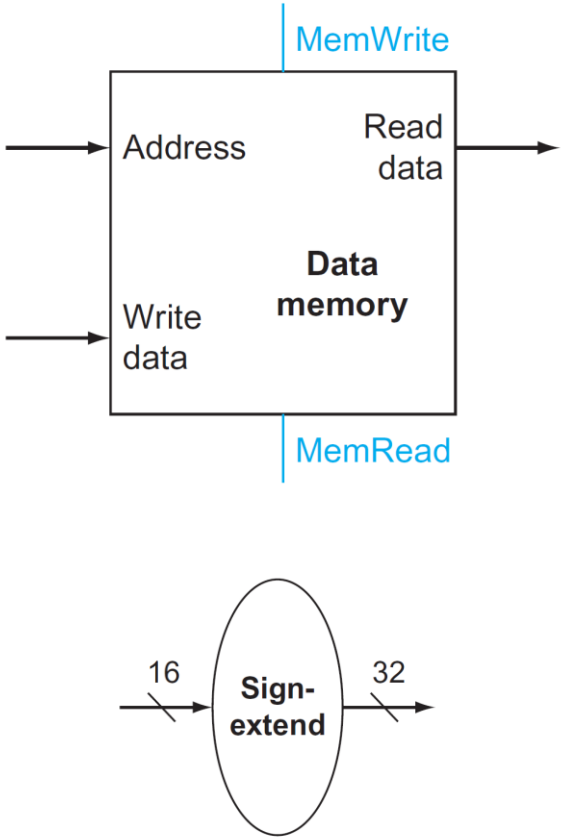
Arithmetic Logic Unit (ALU)



- ALU reads two words from register files, and produces a words and two 1-bit signals (Zero and Overflow)
- ALU is configured by four-bits control input
  - A-invert (1 bit)
  - B-invert (1 bit)
  - Operation (2-bits)

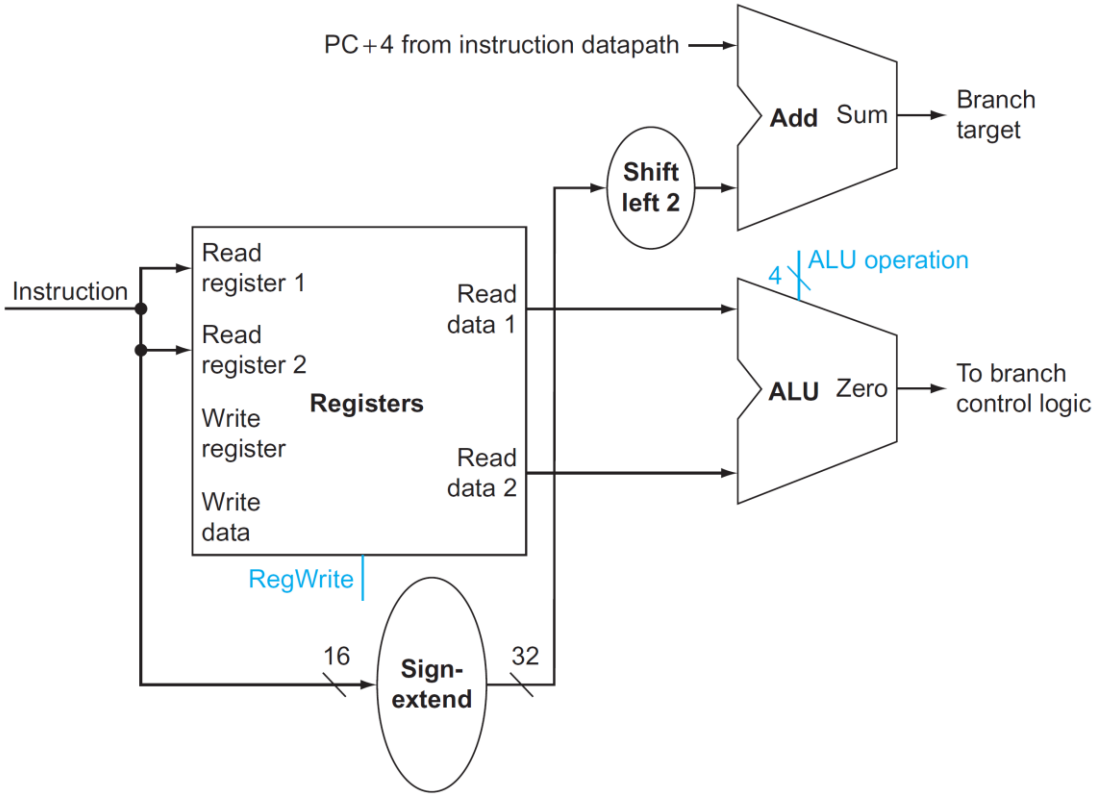
ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

Loading/Storing Memory



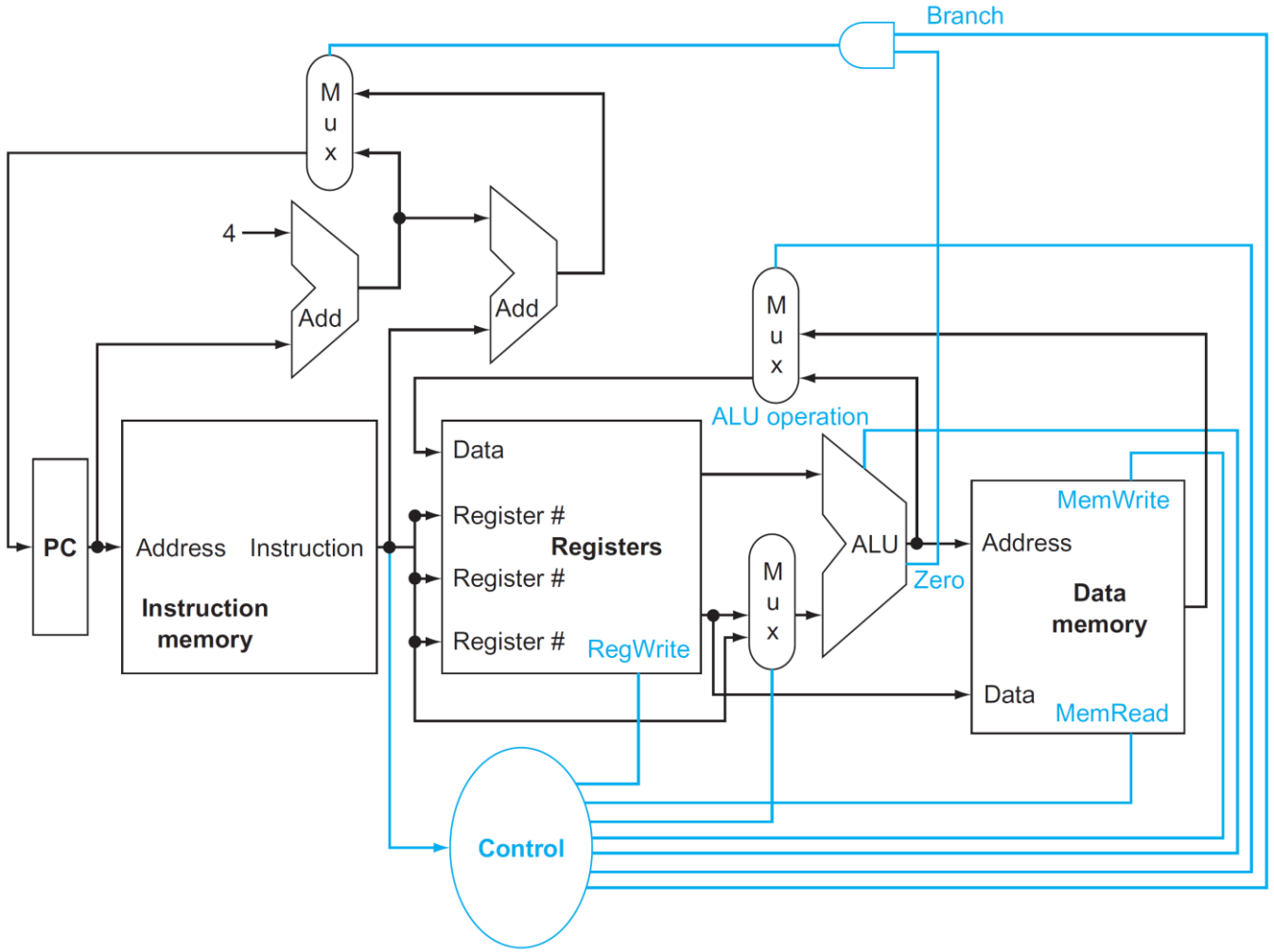
- A load/store instruction obtains the target memory address by adding a base register with a 16-bit constant
  - lw $t1, 4($t2)

Conditional Jump



- A beq instruction has two registers to be compared, and a 16-bit offset to compute the branch target address
  - e.g., beq $t1, $t2, 32
- Use PC+4 for computing the branch target

Abstract View of a Single Cycle MIPS Processor



- `add $t1, $t2, $t3`
- `lw, $t1, 4($t2)`
- `beq, $s1, $s2, 32`

ALU Control (1/2)

- Depending on the 6-bit funct value, ALU is controlled to perform different operations for a R-type instruction
  - the main control unit generates the ALUOp bits
  - ALU control unit generates ALU operation bits depending on ALUOp

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

ALU Control (2/2)

- 4-bit ALU control is determined by the combination of the ALUOp and the funct field values

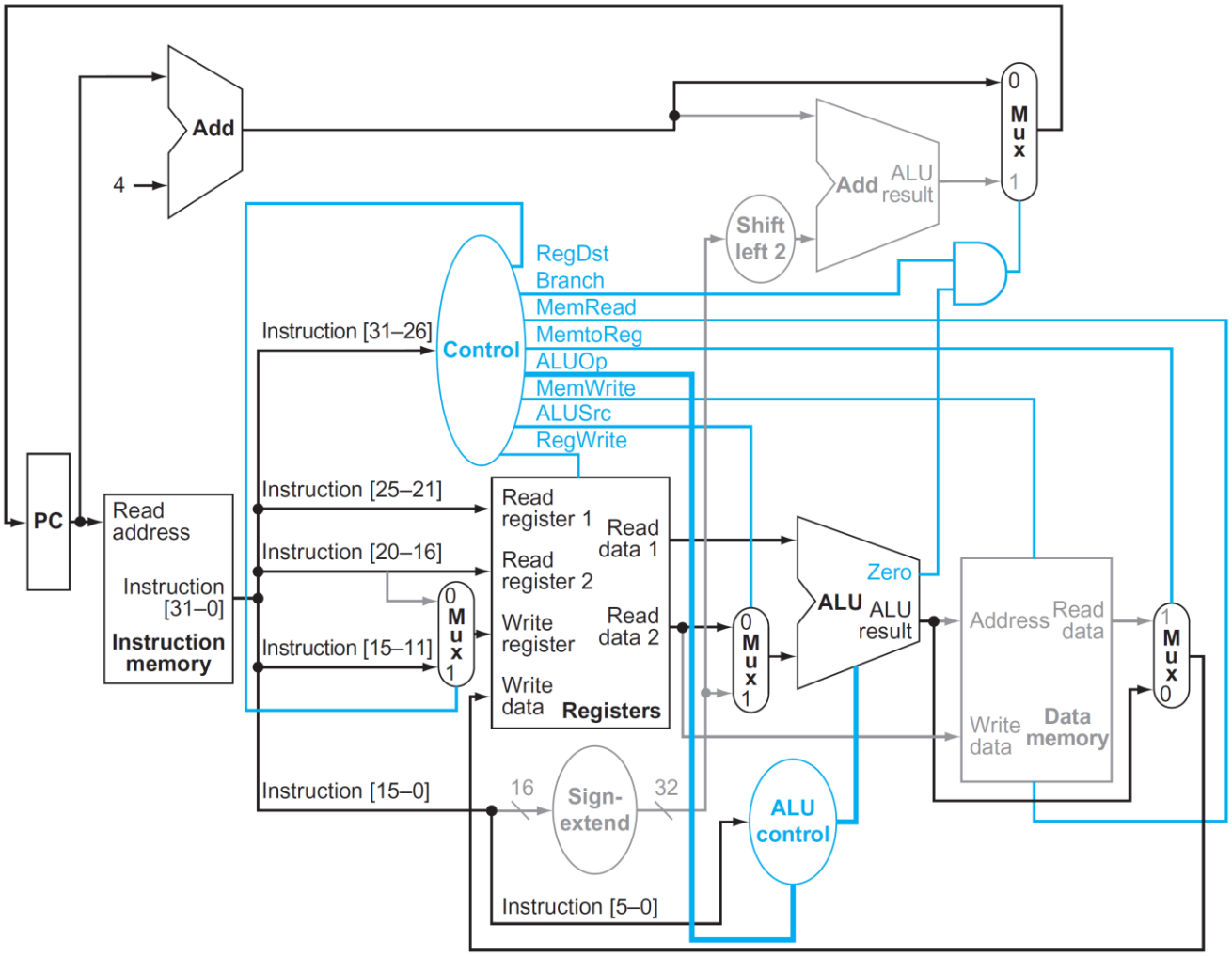
ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

Main Control Unit (1/2)

- Input: instruction
  - opcode
  - input registers: rs, rt registers, base register
  - input constant: 16-bit offset, 26-bit offset
  - destination register
- Output
  - control bits to register file and data memory
  - select bit of three mux's
  - ALU control bits

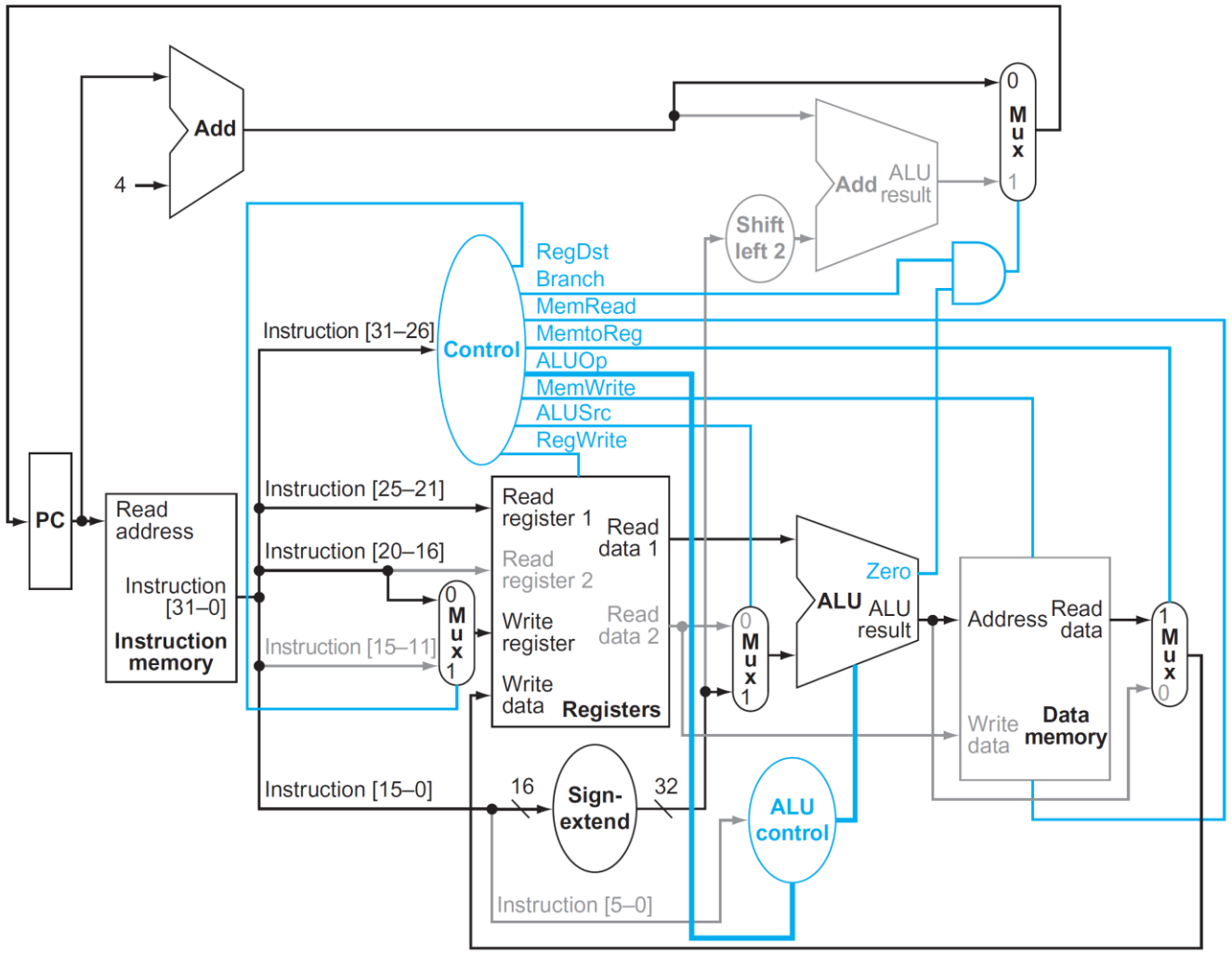
Instruction	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

Datapath for R-type Instruction



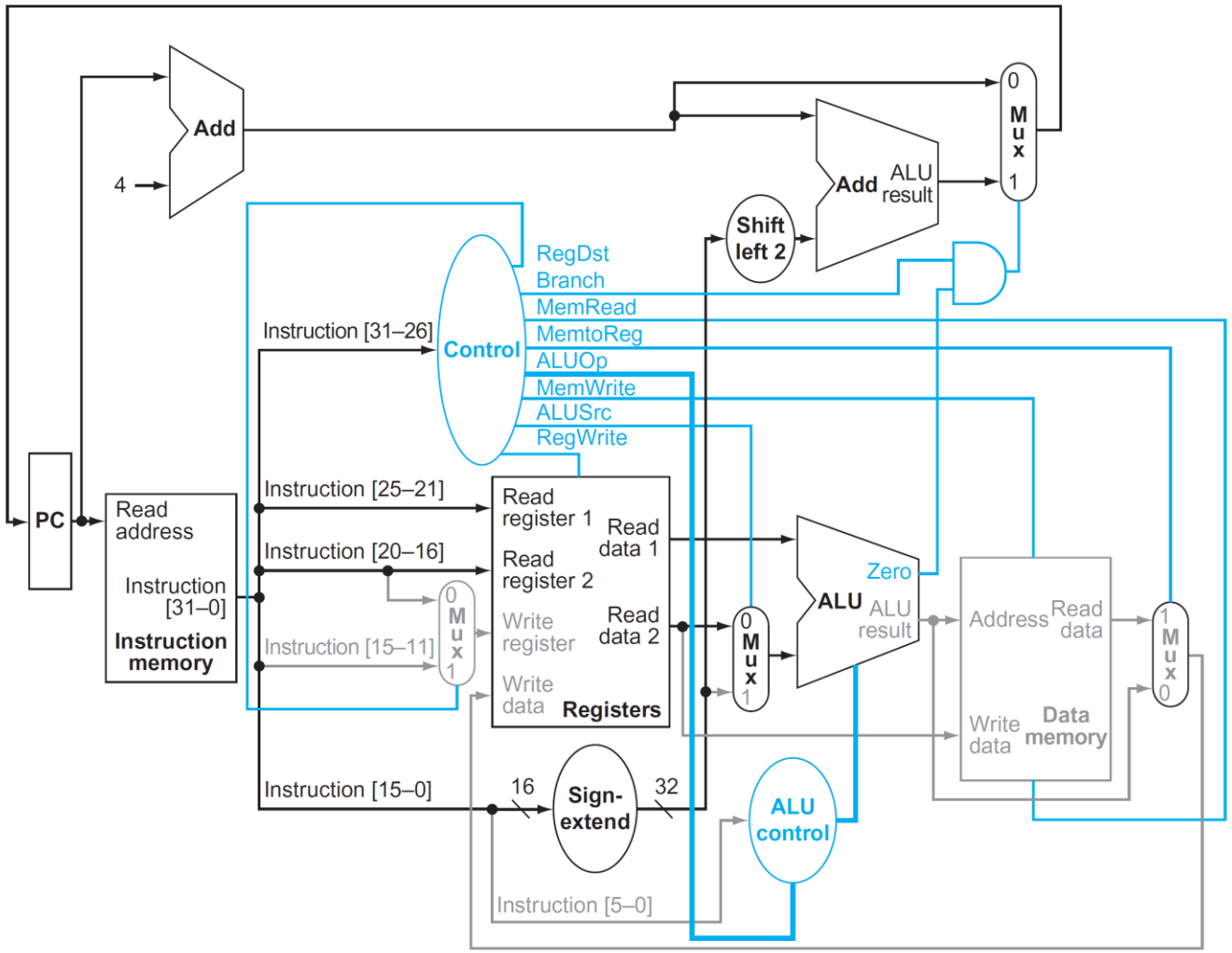
RegDst: 1
ALUSrc: 0
MemToReg: 0
RegWrite: 1
MemRead: 0
MemWrite: 0
Branch: 0
ALUOp: 10

Datapath for Load Instruction



RegDst: 0
ALUSrc: 1
MemToReg: 1
RegWrite: 1
MemRead: 1
MemWrite: 0
Branch: 0
ALUOp: 00

Datapath for Branch-On-Equal Inst.



RegDst: X
ALUSrc: 0
MemToReg: 0
RegWrite: 0
MemRead: 0
MemWrite: 0
Branch: 1
ALUOp: 01

Main Control Unit (2/2)

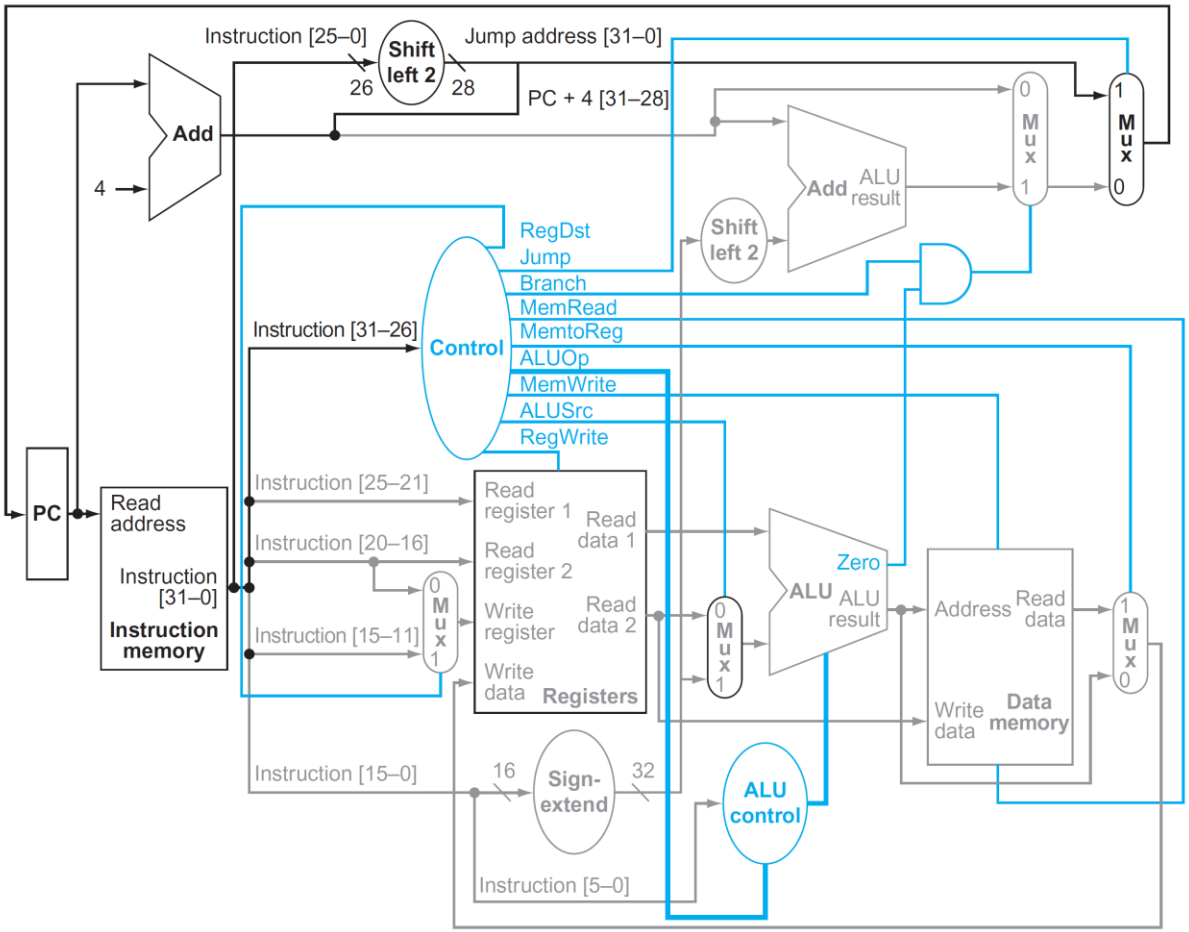
	Opcode						Control Lines								
	Op5	Op4	Op3	Op2	Op1	Op0	Reg Dst	ALU Src	Mem ToReg	Reg Write	Mem Read	Mem Write	Branch	ALU Op1	ALU Op2
R-inst.	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0
load	1	0	0	0	1	1	0	1	1	1	1	0	0	0	0
store	1	0	1	0	1	1	0	1	X	0	0	1	0	0	0
beq	0	0	0	1	0	0	0	0	X	0	0	0	1	0	1

Jump

- Instruction format

Field	000010	address
Bit positions	31:26	25:0

- Datapath



Limitation of Single-Cycle Design

- Modern architectures do not use single-cycle designs (with a fixed clock cycle) for its inefficiency
  - the clock cycle must have the same length for every instruction, and it follows the longest possible path (i.e., load)

- Instead, modern architectures use pipelined design
  - The execution of an instruction takes multiple clock cycle each of which processes only one operation of a unit
  - Depending on an instruction type, a different sequence of operations are performed