CSC 3210 Computer organization and programming

Chapter 4

Chunlan Gao

Georgia State University

Instruction Execution (5 steps)

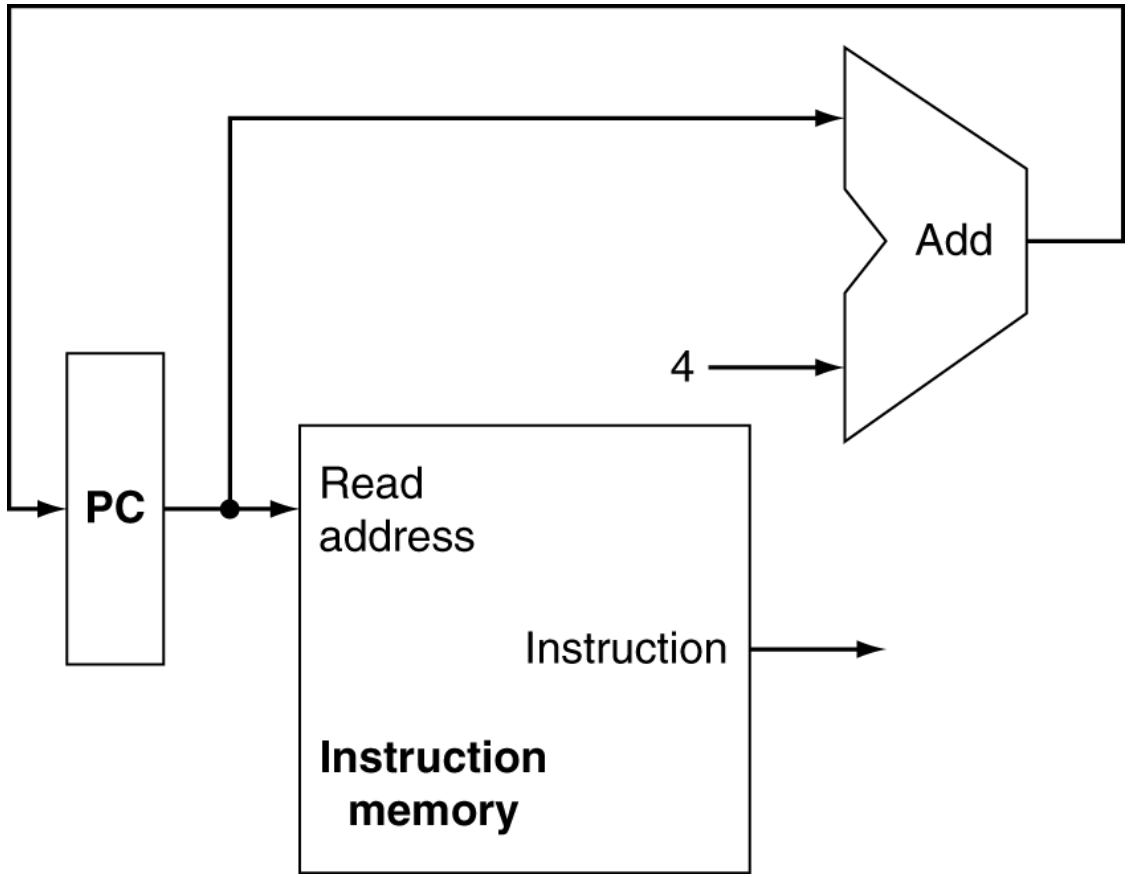
In a typical RISC-V 5-stage pipeline:

1. **IF (Instruction Fetch)** – IM (Instruction Memory)
2. **ID (Instruction Decode/Register Fetch)** – Reg
3. **EX (Execute)** – ALU
4. **MEM (Memory Access)** – DM (Data Memory)
5. **WB (Write Back)** – Reg

Building a Datapath

- Datapath
  - Elements that process data and addresses in the CPU
    - Registers, ALUs, mux’s, memories, …

Step 1 **Instruction Fetch**



Suppose we have these instructions in memory:

Address	Instruction
`0x1000`	`LOAD R1, 0x50`
`0x1004`	`ADD R1, R2`
`0x1008`	`JMP 0x2000`

- Initially, `PC = 0x1000` → fetches the first instruction.
- After execution, `PC` becomes `0x1004`.

R-type

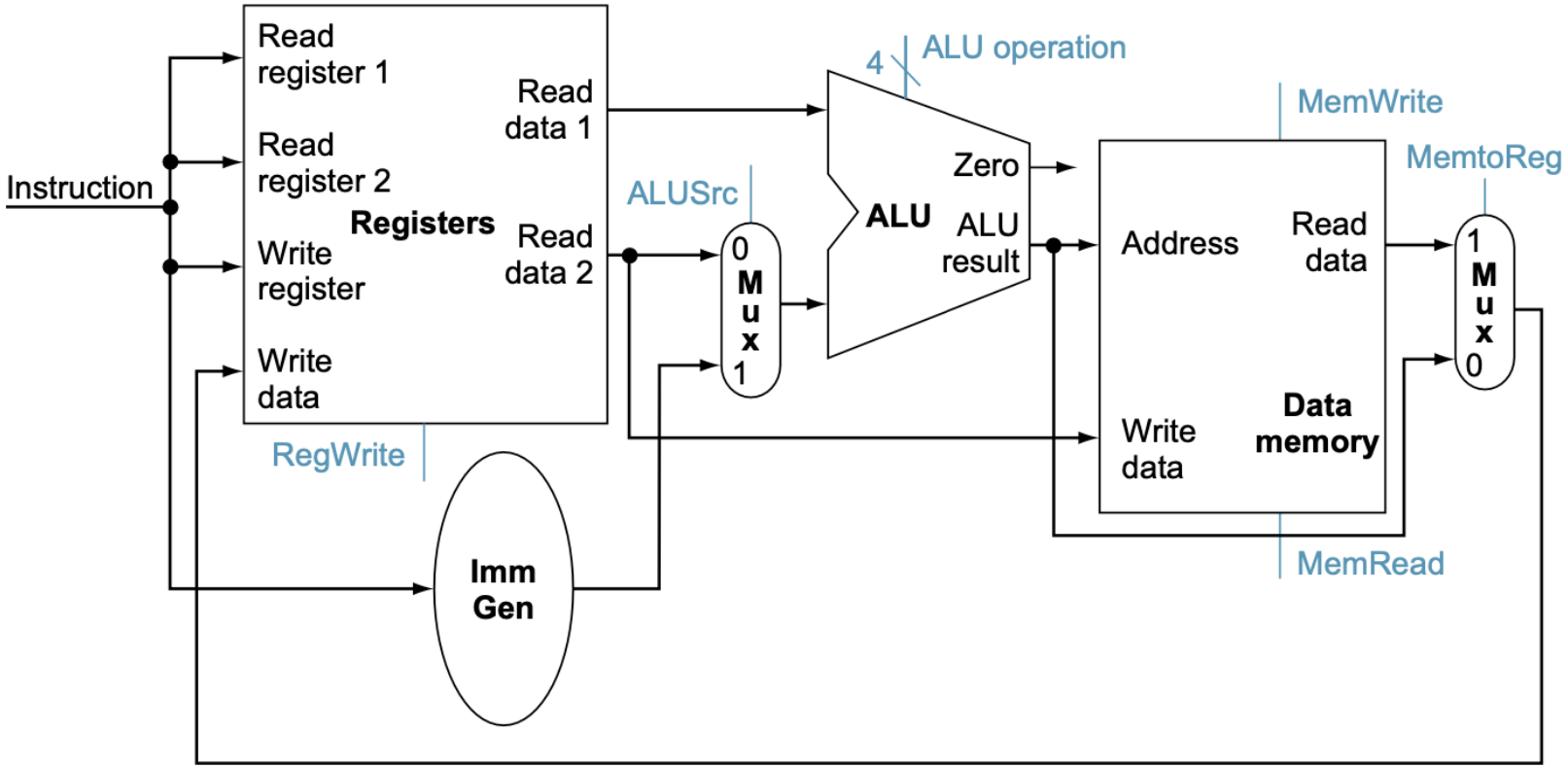


FIGURE 4.10 The datapath for the memory instructions and the R-type instructions. This example shows how a single datapath can be assembled from the pieces in Figures 4.7 and 4.8 by adding multiplexors. Two multiplexors are needed, as described in the example.

Copyright © 2021 Elsevier Inc. All rights reserved.

Practice(Full data path)

Question: Question_datapath_four

Solution: Question_datapath

R-type Instructions	funct7	rs2	rs1	funct3	rd	opcode	Example
add (add)	0000000	00011	00010	000	00001	0110011	add x1, x2, x3
sub (sub)	0100000	00011	00010	000	00001	0110011	sub x1, x2, x3
I-type Instructions	**immediate**		**rs1**	**funct3**	**rd**	**opcode**	**Example**
addi (add immediate)	001111101000		00010	000	00001	0010011	addi x1, x2, 1000
lw (load word)	001111101000		00010	010	00001	0000011	lw x1, 1000 (x2)
S-type Instructions	**immed -iate**	**rs2**	**rs1**	**funct3**	**immed -iate**	**opcode**	**Example**
sw (store word)	0011111	00001	00010	010	01000	0100011	sw x1, 1000(x2)

FIGURE 2.6 RISC-V architecture revealed through Section 2.5. The three RISC-V instruction formats so far are R, I, and S. The R-type format has two source register operand and one destination register operand. The I-type format replaces one source register operand with a 12-bit *immediate* field. The S-type format has two source operands and a 12-bit immediate field, but no destination register operand. The S-type immediate field is split into two parts, with bits 11–5 in the leftmost field and bits 4–0 in the second-rightmost field.

Copyright © 2021 Elsevier Inc. All rights reserved.

Review of Format-Branch Instructions

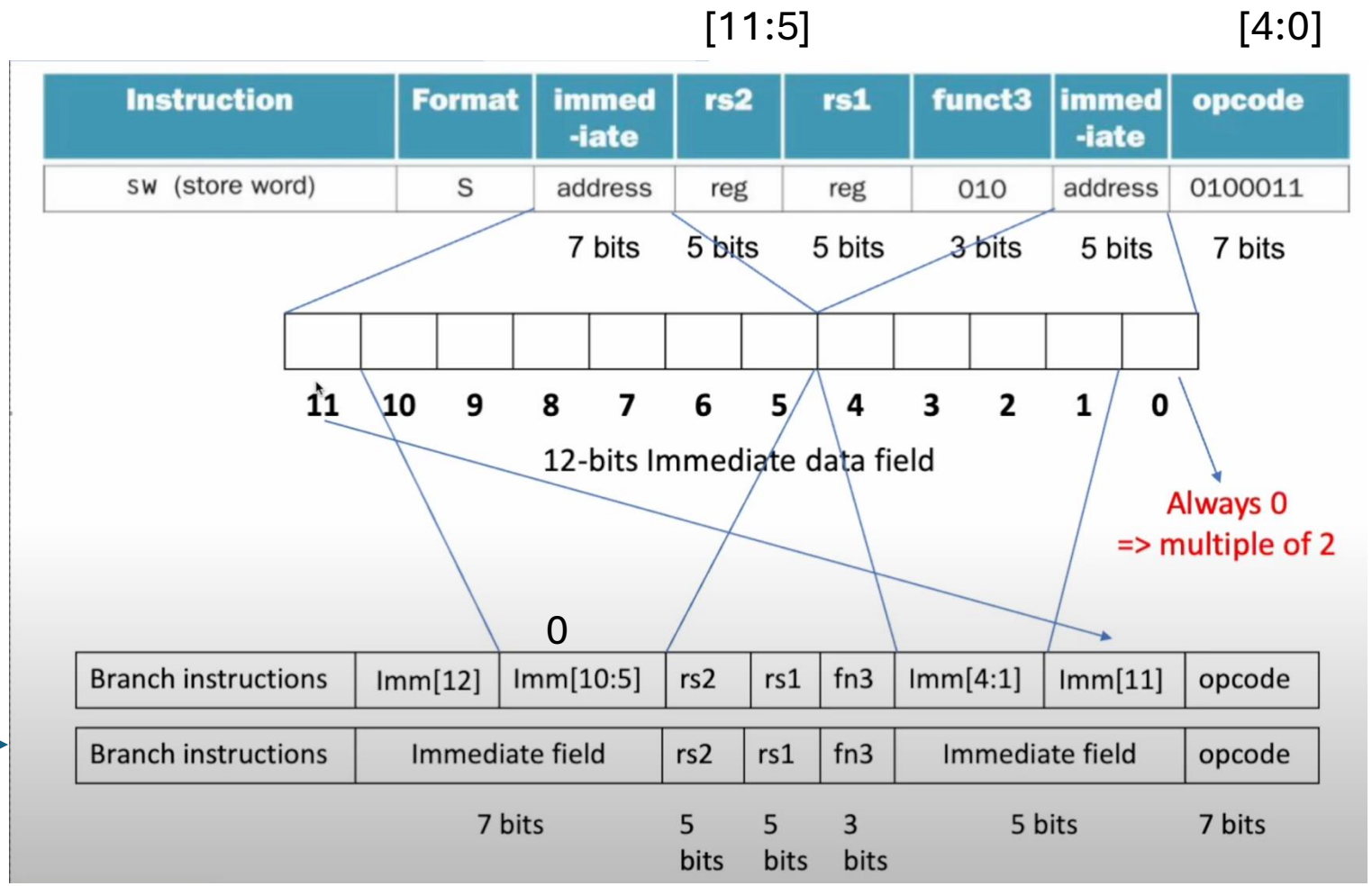
beq rs1, rs2, offset

If they are **equal**, the program counter (PC) jumps to the address at PC + offset.

Name (Field size)	Field 7 bits	5 bits	5 bits	3 bits	5 bits	7 bits	Comments
R-type	funct7	rs2	rs1	funct3	rd	opcode	Arithmetic instruction format
I-type	immediate[11:0]		rs1	funct3	rd	opcode	Loads & immediate arithmetic
S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode	Stores
SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode	Conditional branch format
UJ-type	immediate[20,10:1,11,19:12]				rd	opcode	Unconditional jump format
U-type	immediate[31:12]				rd	opcode	Upper immediate format

Copyright © 2021 Elsevier Inc. All rights reserved.

Review of Format- Branch Instructions



SB format

PC

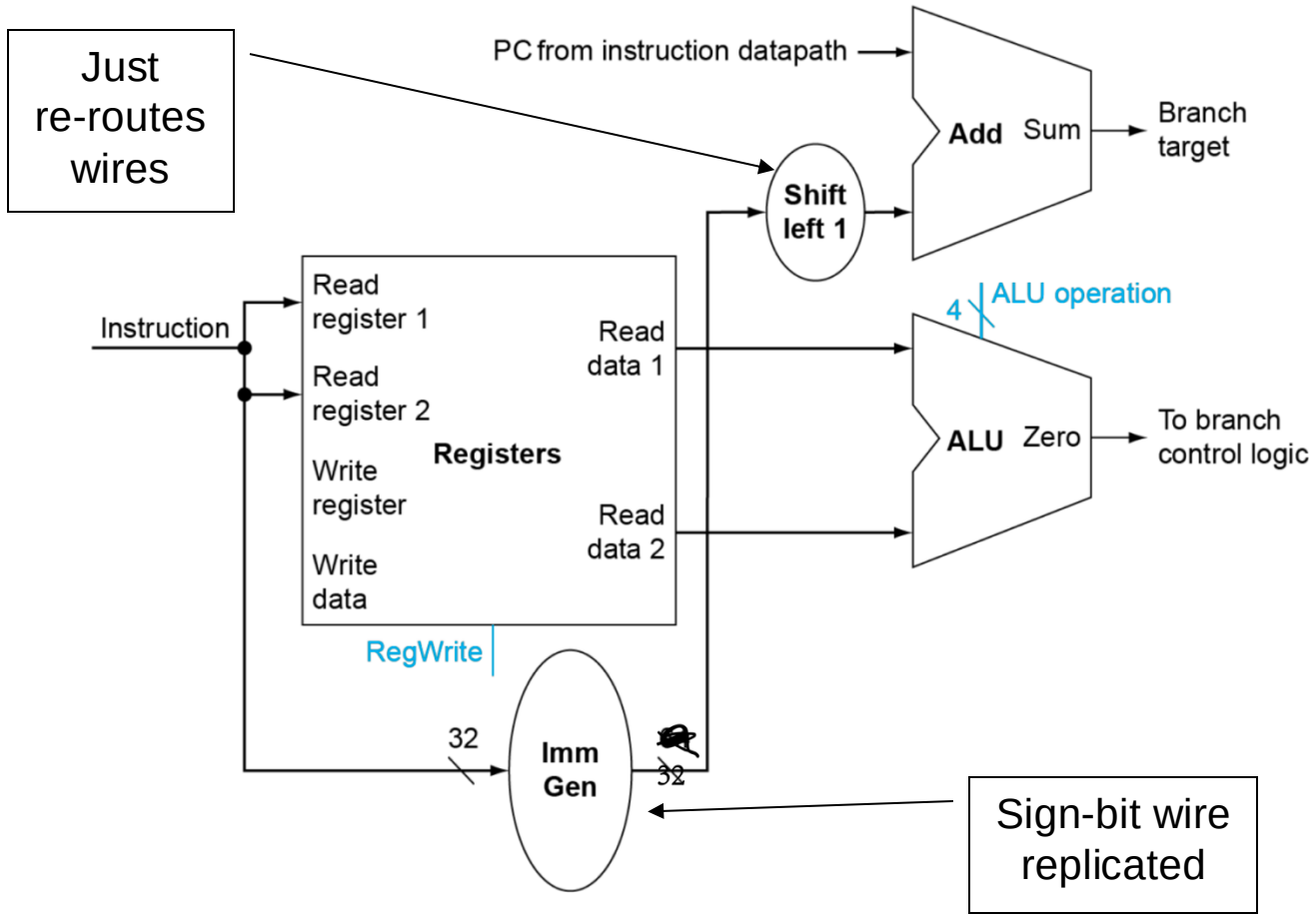
address	value
0X0012	xxxx
0X0008	xxxx
0X0004	xxxx
0X0000	xxxx

offset always power of 2，we can use 13 immediate number (last bit always 0) -4096~+4094

Branch Instructions

- Read register operands
- Compare operands
  - Use ALU, subtract and check Zero output
- Calculate target address
  - Sign-extend displacement
  - Shift left 1 place (halfword displacement)
  - Add to PC value

Branch Instructions



The simple data path for different instruction

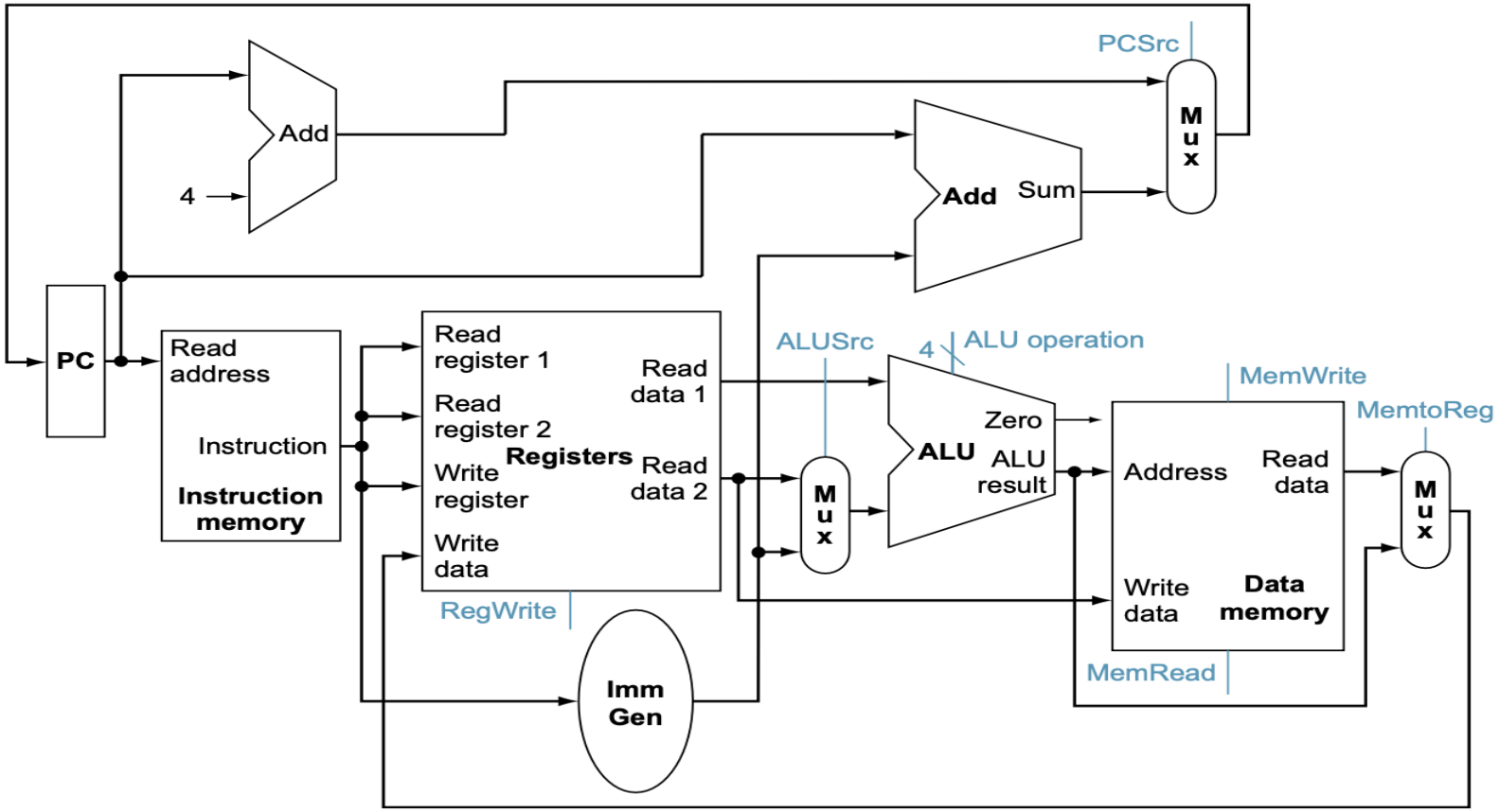


FIGURE 4.11 The simple datapath for the core RISC-V architecture combines the elements required by different instruction classes. The components come from Figures 4.6, 4.9, and 4.10. This datapath can execute the basic instructions (load-store register, ALU operations, and branches) in a single clock cycle. Just one additional multiplexor is needed to integrate branches.

Copyright © 2021 Elsevier Inc. All rights reserved.

Control

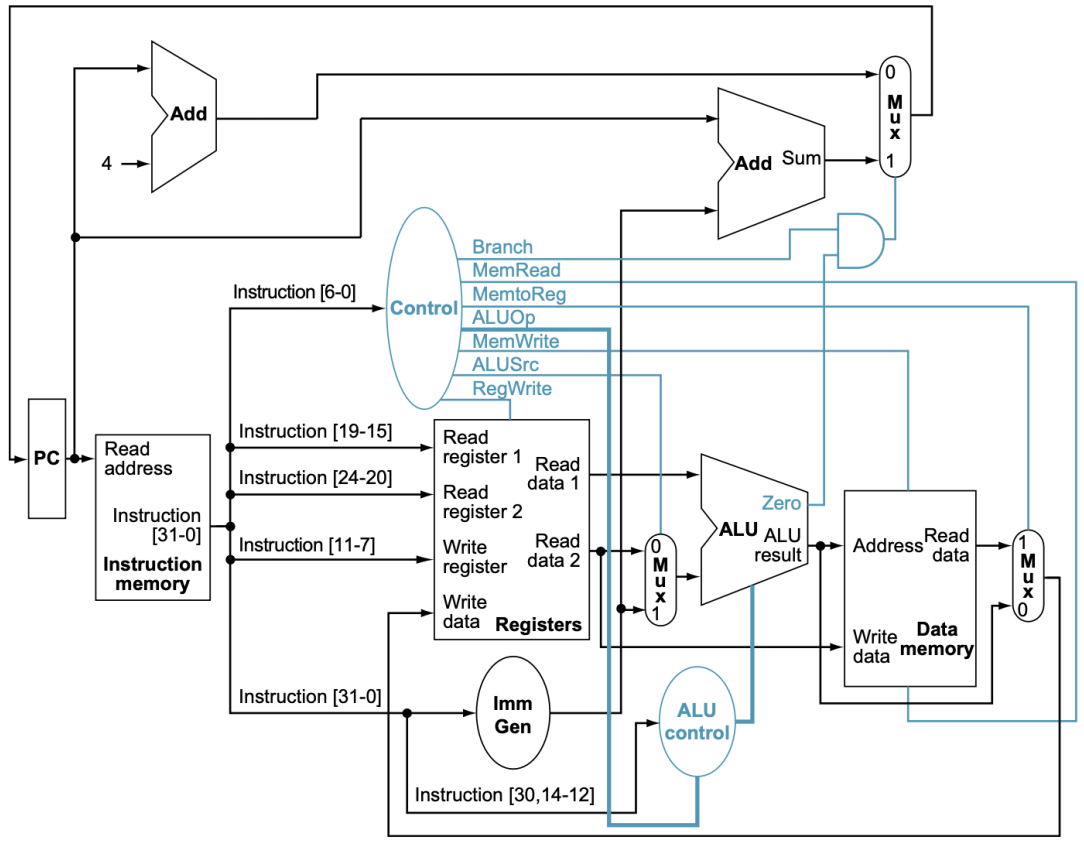


FIGURE 4.21 The simple datapath with the control unit. The input to the control unit is the 7-bit opcode field from the instruction. The outputs of the control unit consist of two 1-bit signals that are used to control multiplexors (ALUSrc and MemtoReg), three signals for controlling reads and writes in the register file and data memory (RegWrite, MemRead, and MemWrite), a 1-bit signal used in determining whether to possibly branch (Branch), and a 2-bit control signal for the ALU (ALUOp). An AND gate is used to combine the branch control signal and the Zero output from the ALU; the AND gate output controls the selection of the next PC. Notice that PCSrc is now a derived signal, rather than one coming directly from the control unit. Thus, we drop the signal name in subsequent figures.

Copyright © 2021 Elsevier Inc. All rights reserved.

ALU Control

The simple implementation of RISC-V ALU used for the following instructions:

F: Function

- Load/Store: F = add
- Branch(beq): F = subtract
- R-type: F depends on opcode function field

ALU control	Function
0000	AND
0001	OR
0010	add
0110	subtract

Checking Opcode

- ALU operation is generated in two steps:

The main control checks opcode and generates a 2-bit signal ALUOp that indicates the instruction type.

(Check the opcode only is faster)

ALU control uses 2-bit ALUOp, instead of opcode directly

Instructions	ALUOp
lw	00
sw	00
beq	01
R-type	10

ALU Control

ALUOp
00
01
10

no 11

Instruction opcode	ALUOp	Operation	funct7	funct3	ALU function	ALU operation
lw	00	load word	xxx xxxx	xxx	add	0010
sw	00	store word	xxx xxxx	xxx	add	0010
beq	01	branch if equal	xxx xxxx	xxx	subtract	0110
R-type	10	add	000 0000	000	add	0010
		subtract	010 0000	000	subtract	0110
		and	000 0000	111	and	0000
		or	000 0000	110	or	0001

ALUOp		Funct7 field							Funct3 field			
ALUOp1	ALUOp0	I[31]	I[30]	I[29]	I[28]	I[27]	I[26]	I[25]	I[14]	I[13]	I[12]	Operation
0	0	X	X	X	X	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	X	X	X	X	0110
1	X	0	0	0	0	0	0	0	0	0	0	0010
1	X	0	1	0	0	0	0	0	0	0	0	0110
1	X	0	0	0	0	0	0	0	1	1	1	0000
1	X	0	0	0	0	0	0	0	1	1	0	0001

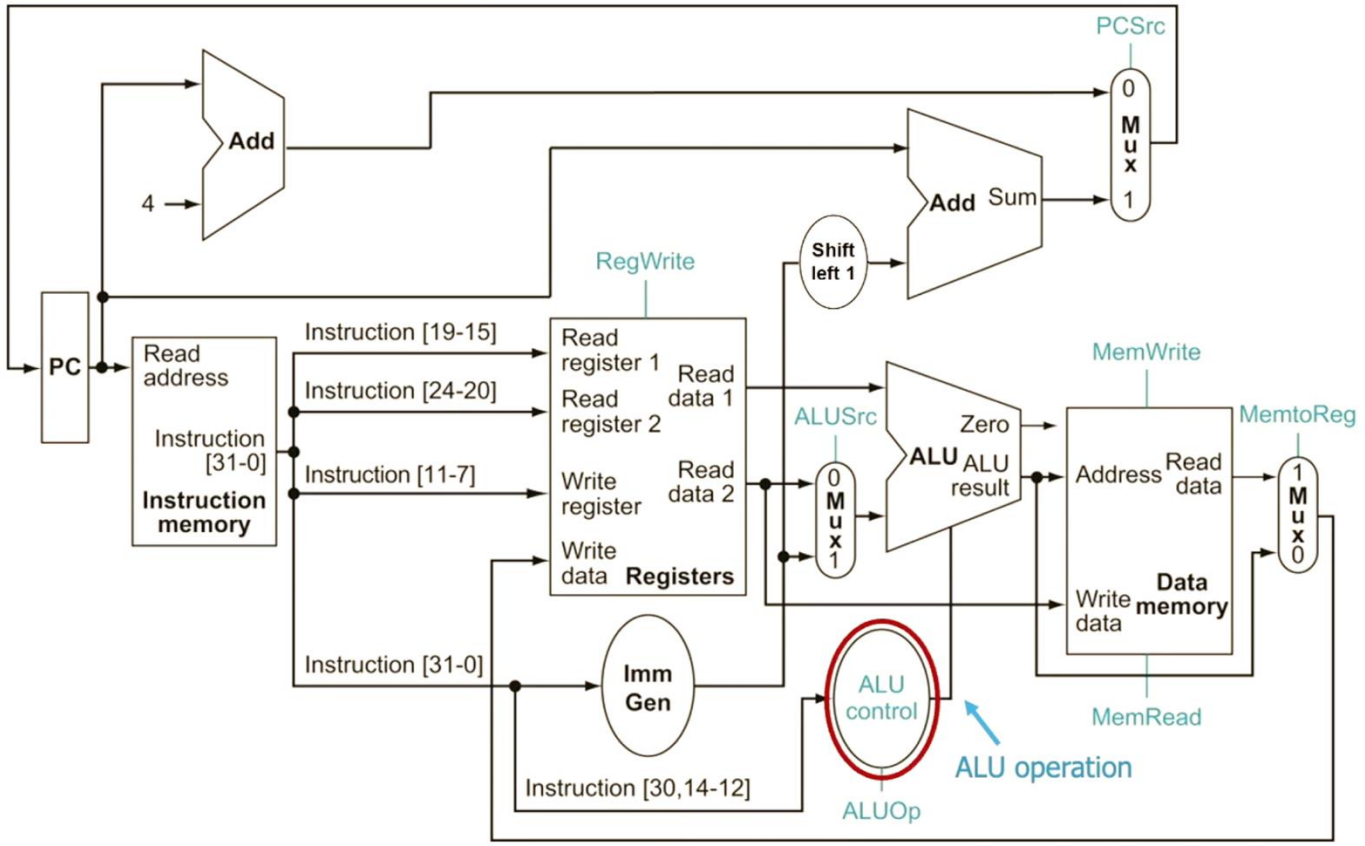
ALU control

Instruction opcode	ALUOp	Operation	funct7	funct3	ALU function	ALU operation
lw	00	load word	xxx xxxx	xxx	add	0010
sw	00	store word	xxx xxxx	xxx	add	0010
beq	01	branch if equal	xxx xxxx	xxx	subtract	0110
R-type	10	add	000 0000	000	add	0010
		subtract	010 0000	000	subtract	0110
		and	000 0000	111	and	0000
		or	000 0000	110	or	0001

ALUOp		Funct7 field							Funct3 field			
ALUOp1	ALUOp0	I[31]	I[30]	I[29]	I[28]	I[27]	I[26]	I[25]	I[14]	I[13]	I[12]	Operation
0	0	X	X	X	X	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	X	X	X	X	0110
1	X	0	0	0	0	0	0	0	0	0	0	0010
1	X	0	1	0	0	0	0	0	0	0	0	0110
1	X	0	0	0	0	0	0	0	1	1	1	0000
1	X	0	0	0	0	0	0	0	1	1	0	0001

need 1 bit in function 7 and 3 bits in function 3

ALU control



Copyright © 2021 Elsevier Inc. All rights reserved.

Questions

1. Which of the following is correct for a load instruction about MemtoReg?

- It should be set to cause the data from memory to be sent to the register file
- It should be set to cause the correct register destination to be send to the register file
- We don not care about the setting of MemotoReg for loads.

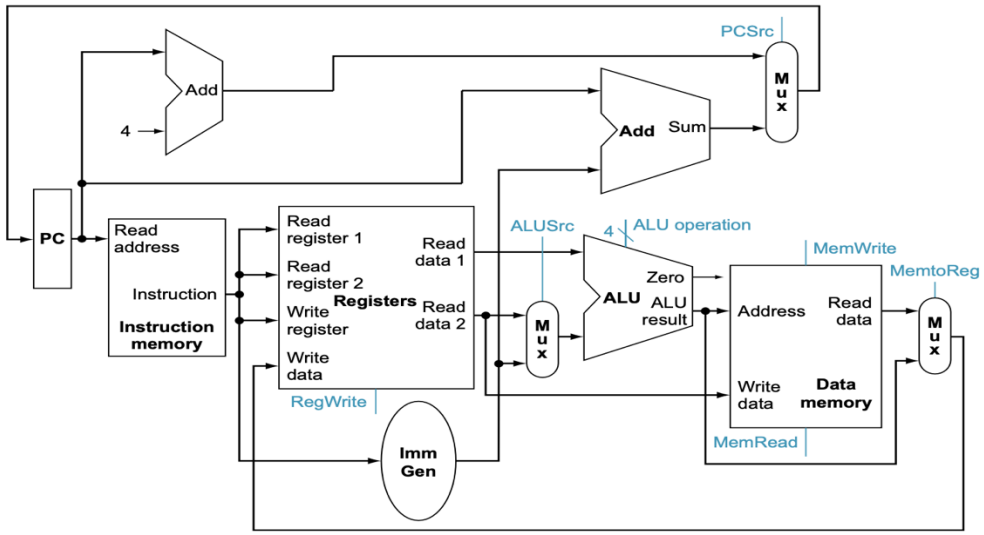


FIGURE 4.11 The simple datapath for the core RISC-V architecture combines the elements required by different instruction classes. The components come from Figures 4.6, 4.9, and 4.10. This datapath can execute the basic instructions (load-store register, ALU operations, and branches) in a single clock cycle. Just one additional multiplexor is needed to integrate branches.

Copyright © 2021 Elsevier Inc. All rights reserved.