Computer Architecture
Chapter 4

The Processor – 1

Joonho Kong
School of EE, KNU

Introduction

A Basic RISC-V Processor Implementation

- We will examine an RTL (Register Transfer-Level) implementation of RISC-V processor that can execute instructions such as:

1. The memory-reference instruction
   : load doubleword (ld) and store doubleword(sd)

2. The arithmetic-logical instruction
   : add, sub, and, and or

3. The conditional branch instruction
   : beq

Introduction

Remind Key Design Principles

1) Simplicity favors regularity

2) Smaller is faster

3) Good design demands good compromises

4) Make common case fast

Overview of the Implementation

- What are the same steps for every instruction?
  1. Send the program counter (PC) to the memory and fetch the instruction from that memory
  2. Read one or two registers
     → For example, the add instruction need to read two registers as operands and the ld instruction need to read only one register

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

Overview of the Implementation

Arithmetic-Logical Unit (ALU)

- ALU: performs arithmetic and bitwise operations on integer binary operands
- The ALU is used in every instruction after reading the registers

1. The memory-reference instruction
   : for an address calculation

2. The arithmetic-logical instruction
   : for the operation execution

3. The conditional branch instruction
   : for the equality test

Overview of the Implementation

The Action after ALU

1. The memory-reference instruction
 - will access the memory to read or write data

2. The arithmetic-logical or ld instruction
 - write the data from the ALU or memory back into a register

3. The conditional branch instruction
 - change the next instruction address based on comparison between two values

Overview of the Implementation

An Abstract View of RISC-V Implementation

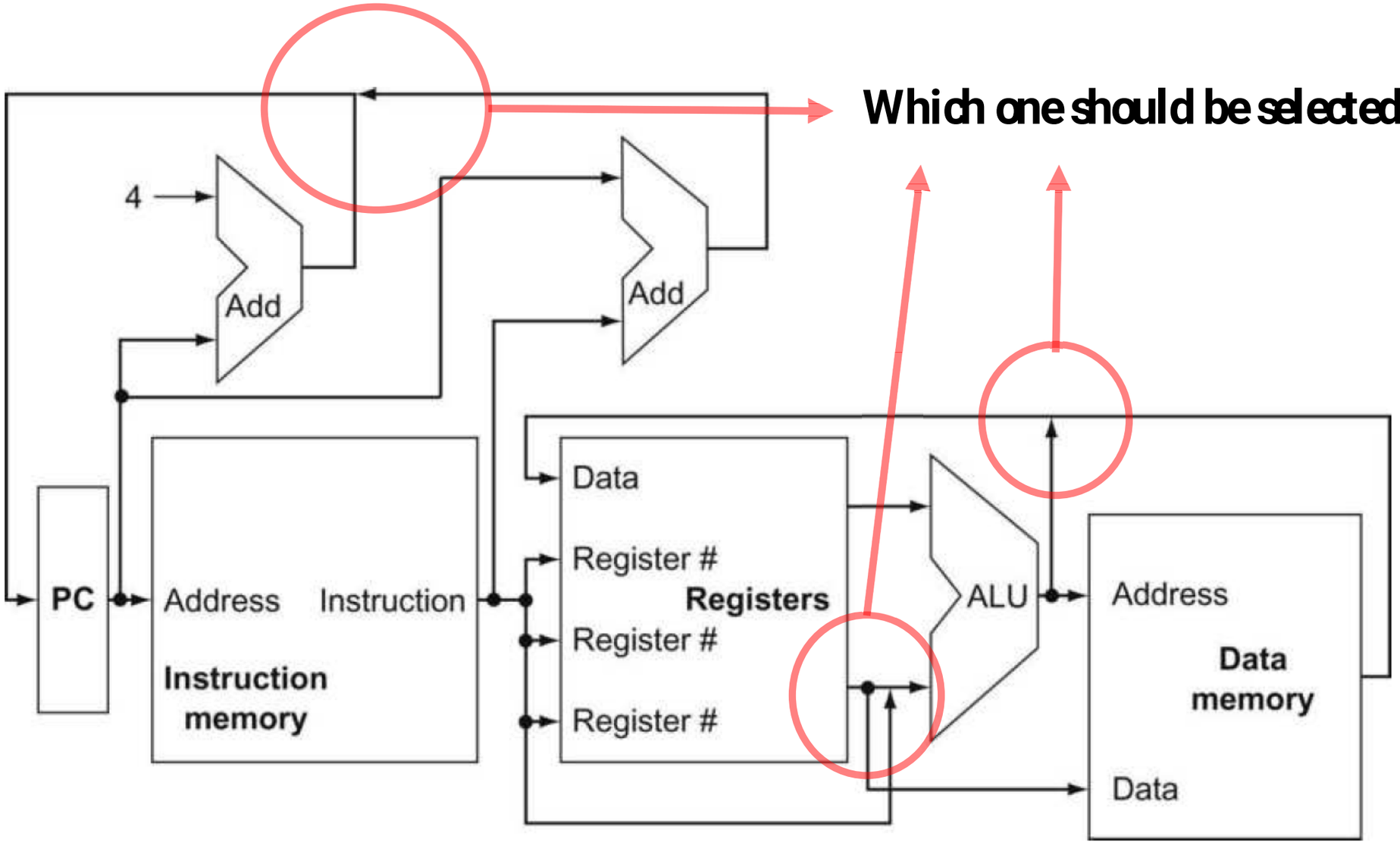


Figure 4.1 is from “Computer Organization and Design”

Overview of the Implementation

An Abstract View of RISC-V Implementation

- What is the problem of the design shown in the previous slide?

1. Being wired together like a part of red circles is impossible
   → We must use multiplexors to choice only one value

2. Control of several units depending on the type of instructions
   → We can implement control paths

Overview of the Implementation

The Basic Implementation of RISC-V

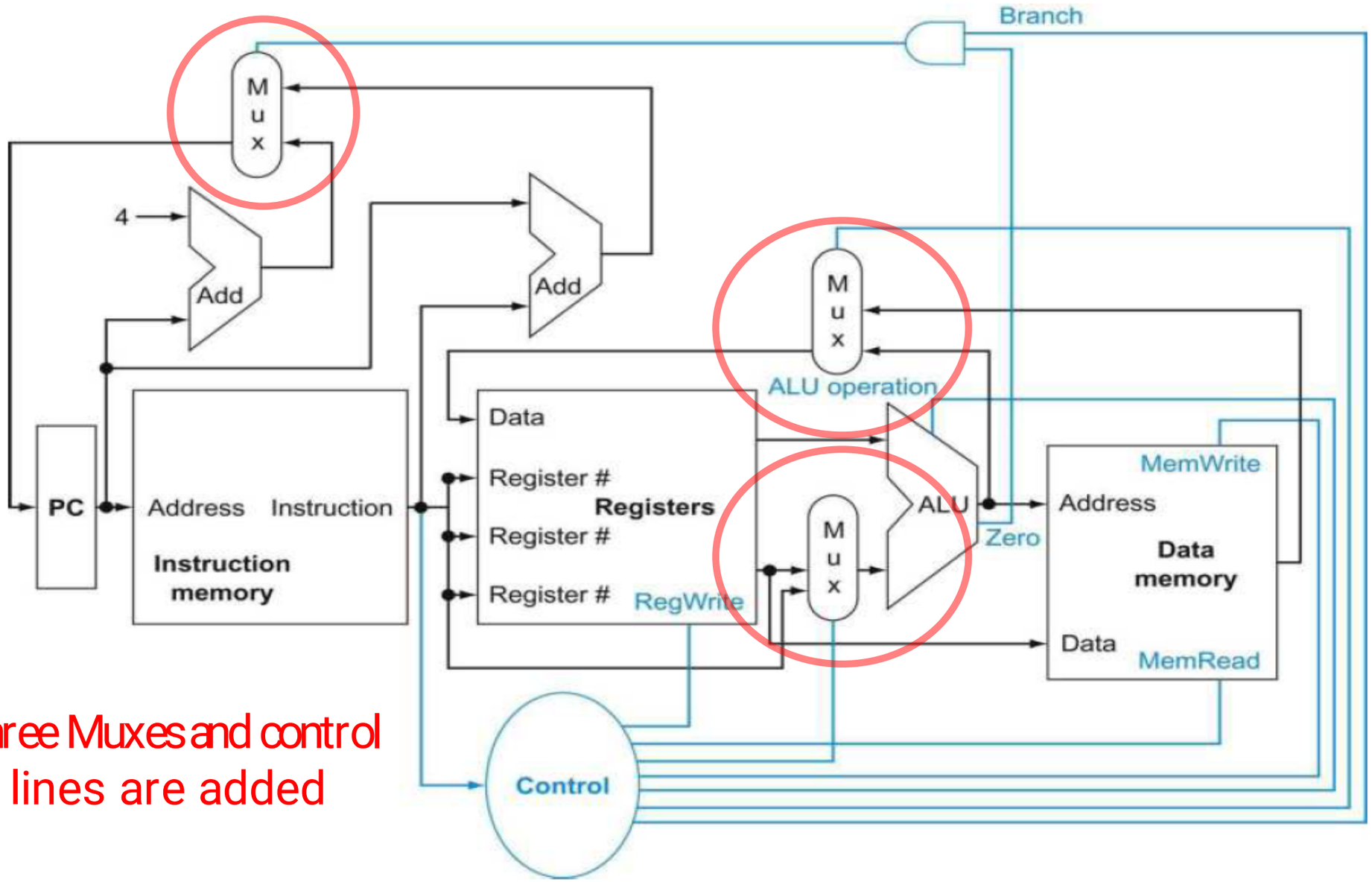


Figure 4.2 is from “Computer Organization and Design”

Logic Design Conventions

A Few Key Idea in Digital Logic

· Combinational elements
  - Its outputs depend only on the current inputs
  - The ALU is an example of combinational element

· State (sequential) elements
  - It has some internal storages
  - Output depends on the internal states and inputs
  - The instruction and data memories and the registers are the example of state element

Logic Design Conventions

How State Elements Work?

- They need at least two inputs and provide one output
  input: data value and clock
  output: data value

- D flip-flop is the simplest example of state element

- They are also called “sequential”

D Flip-flop

Symbol

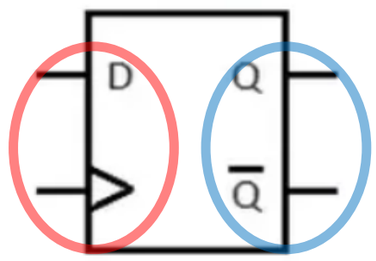


Table of truth:

clk	D	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

edge triggered – state is changed at clock edge

Logic Design Conventions

Clocking Methodology

- Edge-triggered clocking methodology
  : Any values are updated only on a clock edge for the state elements

- Combinational logic, state elements, and the clock are closely related

- All signals must propagate in one clock cycle

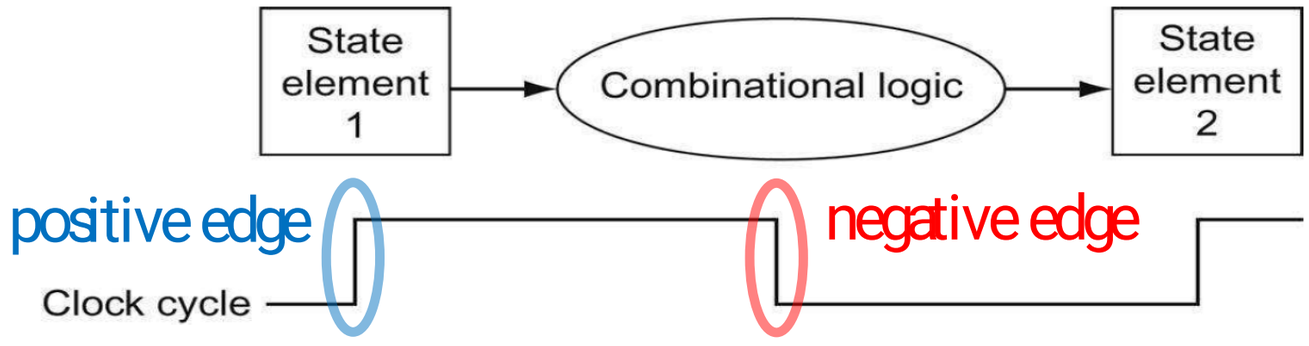


Figure 4.3 is from “Computer Organization and Design”

Logic Design Conventions

Clocking Methodology

- Whether we use rising clock edge (positive edge) or falling clock edge(negative edge) is not important

- Like Fig 4.4, there is no feedback within a single clock cycle
  It means the combinational logic shown below is only activated once in a single clock cycle

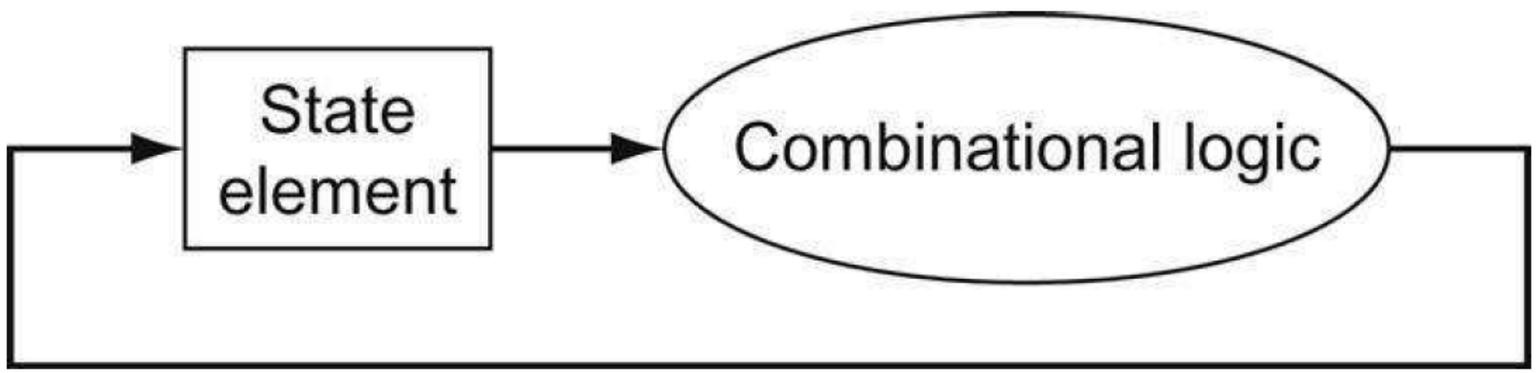


Figure 4.4 is from “Computer Organization and Design”

Building a Datapath

The Basic Elements for Datapath

Program counter (PC)
- Holds the address of the current instruction

Instruction memory
- Stores the instructions of a program
- Supplies an instruction at PC to processor datapath

Adder
- Combinational logic
- It is basically from the ALU, but works only an addition operation
- Increase the PC for the address of the next instruction

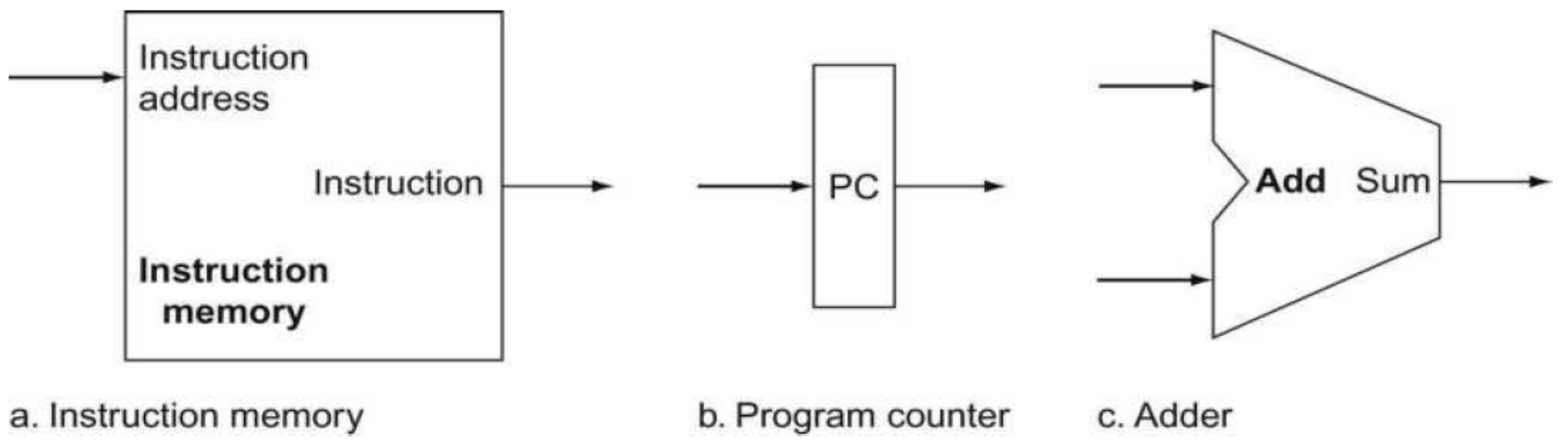


Figure 4.5 is from “Computer Organization and Design”

Building a Datapath

The Basic Elements for Datapath (Instruction fetch unit)

- Fetch the instruction from memory

- Prepare next instruction by increasing the PC by 4 (PC+4)

- These two stages can be worked at the same time because they are independent

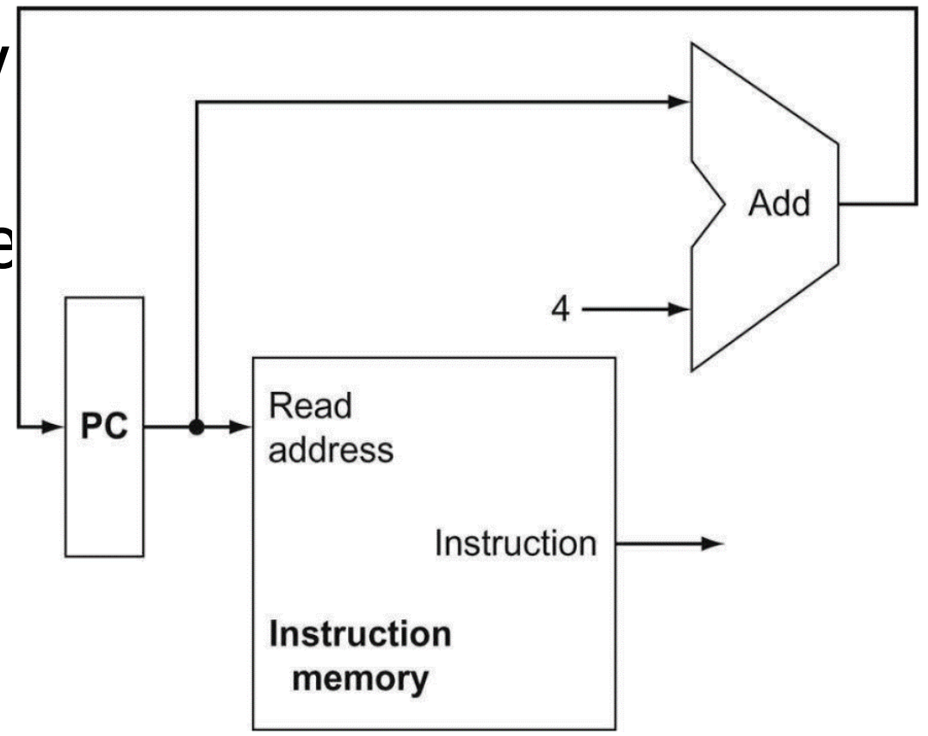


Figure 4.6 is from “Computer Organization and Design”

Building a Datapath

The Basic Elements for Datapath (Register file and ALU)

- Register file: 32 general-purpose registers (x0~x31) of the processor
- After fetch the (R-format) instruction:
  - R-format instruction needs to read two data from register file and write one data into register file
  - Input for the register file is register number and output is corresponding data
  - RegWrite signal controls the write operation for the registers
- Then, the ALU takes two 64-bit inputs and produces a 64-bit one output (4-bit ALU operation signal and 1-bit Zero signal will be covered soon)

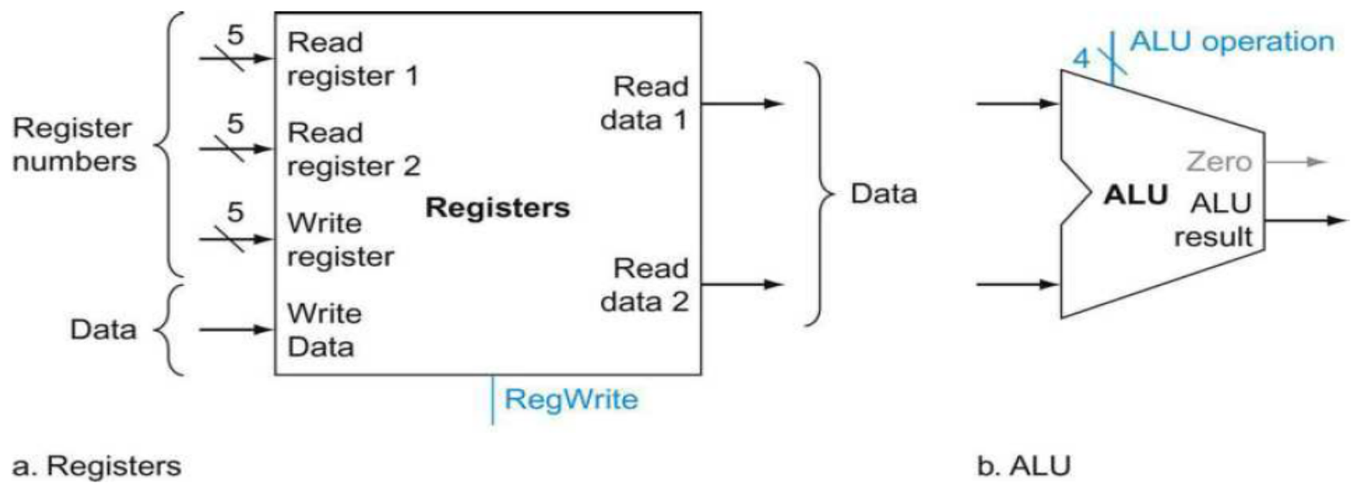
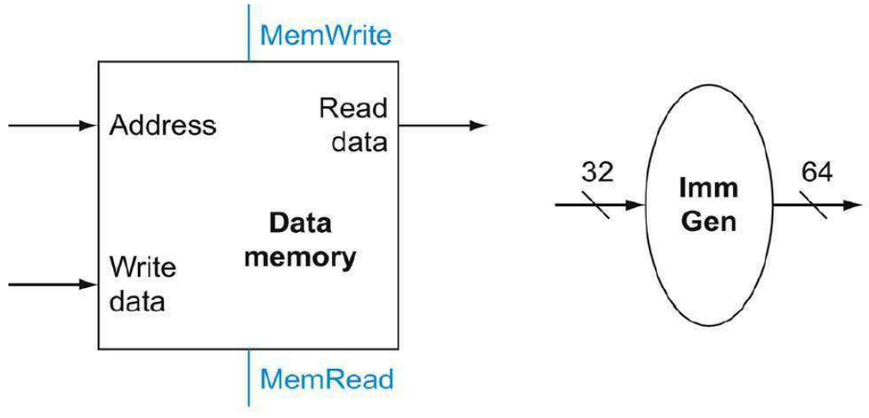


Figure 4.7 is from “Computer Organization and Design”

Building a Datapath

The Basic Elements for Datapath (Data memory)

- To access memory, load and store instructions add the base address to the 12-bit offset
- Data memory is controlled by the control signals such as MemWrite and MemRead
- For the 12-bit offset:
  - Instruction [31:0] is fed into the immediate generation unit (IGU)
  - IGU will combine the immediate bit fields and extend it to 64-bit signed value



a. Data memory unit b. Immediate generation unit

Figure 4.8 is from “Computer Organization and Design”

Building a Datapath

The Basic Elements for Datapath (Branch Instruction)

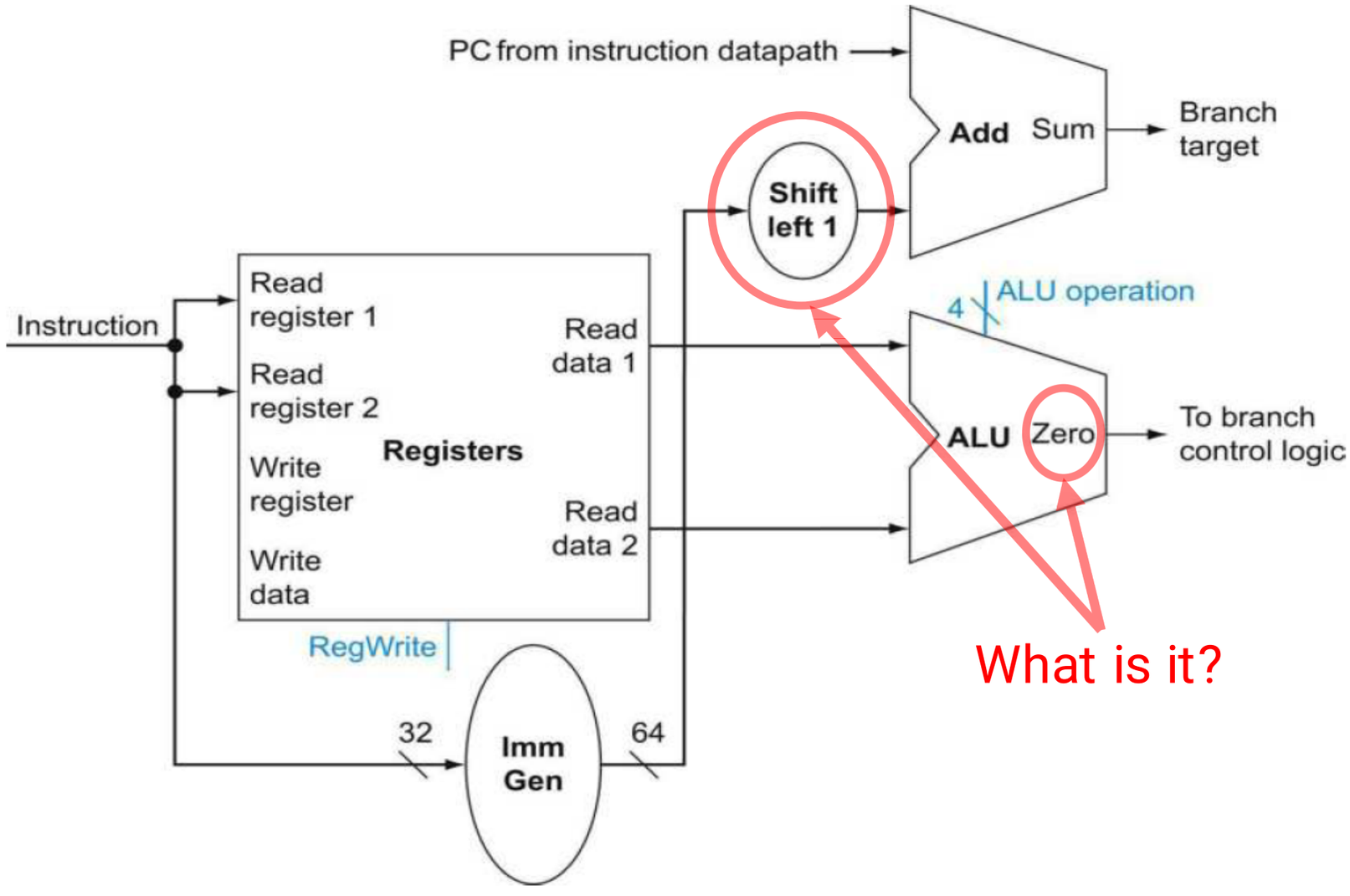


Figure 4.9 is from “Computer Organization and Design”

Building a Datapath

The Basic Elements for Datapath (The Conditional Branch Instruction)

What is the 1-bit Zero signal?

- ALU receives two data from registers and compare them.
- If they are equal, 1-bit Zero signal will be asserted (logic high (1) value)
- How to test those two values are equal or not?
  → By subtracting two values

Why 64-bit sign-extended value is shifted left 1?

- For the effective range of the offset field
- In other words, branch can be taken to farther address from the PC

Building a Datapath

Creating a Single Datapath

The simple datapath can be created by adding some basic datapath for:

1. Instruction fetch and preparing next PC
2. Datapath for register file and ALU
3. Datapath for memory and immediate generation unit
4. Datapath for branches

Building a Datapath

Creating a Single Datapath

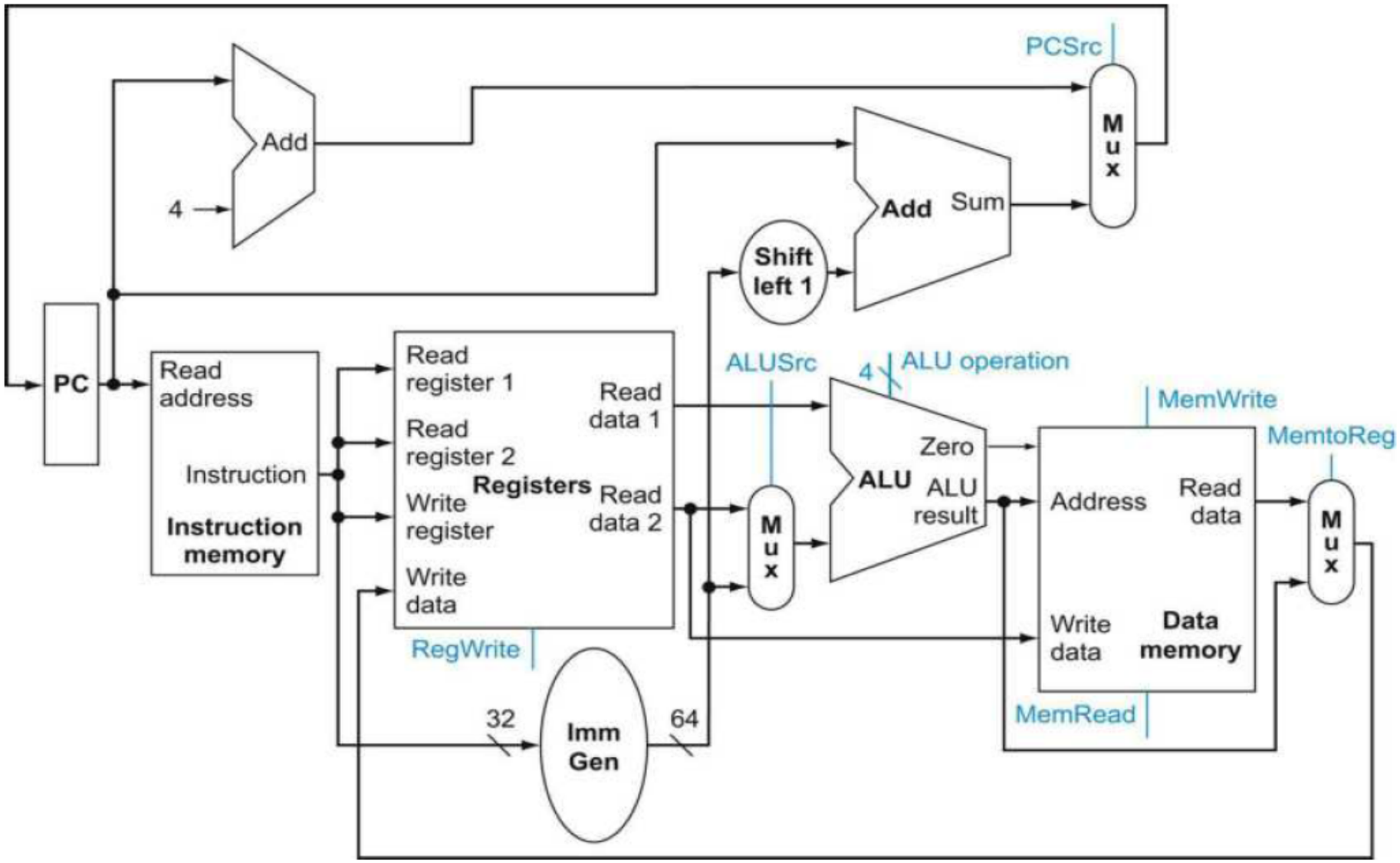


Figure 4.11 is from “Computer Organization and Design”

Building a Datapath

The ALU Control

- Use of ALU:
  1. The memory-reference instruction
     : for an address calculation

  2. The arithmetic-logical instruction
     : for the operation execution

  3. The conditional branch instruction
     : for the equality test

Building a Datapath

Using ALU Control Unit

- The "ALU control unit" controls the ALU
- 2-bit ALUOp input determines the operation of the ALU

- ALUOp (00 and 01) doesn't care about funct7 and funct3 fields
- ALUOp (10) determines add, sub, AND, and OR operation using two funct fields
- "x" means "don't care"
- Please don't memorize the tables shown below

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract

Instruction opcode	ALUOp	Operation	Funct7 field	Funct3 field	Desired ALU action	ALU control input
ld	00	load doubleword	XXXXXXX	XXX	add	0010
sd	00	store doubleword	XXXXXXX	XXX	add	0010
beq	01	branch if equal	XXXXXXX	XXX	subtract	0110
R-type	10	add	0000000	000	add	0010
R-type	10	sub	0100000	000	subtract	0110
R-type	10	and	0000000	111	AND	0000
R-type	10	or	0000000	110	OR	0001

Figure 4.12 is from "Computer Organization and Design"

Building a Datapath

Datapath with ALU Control Block

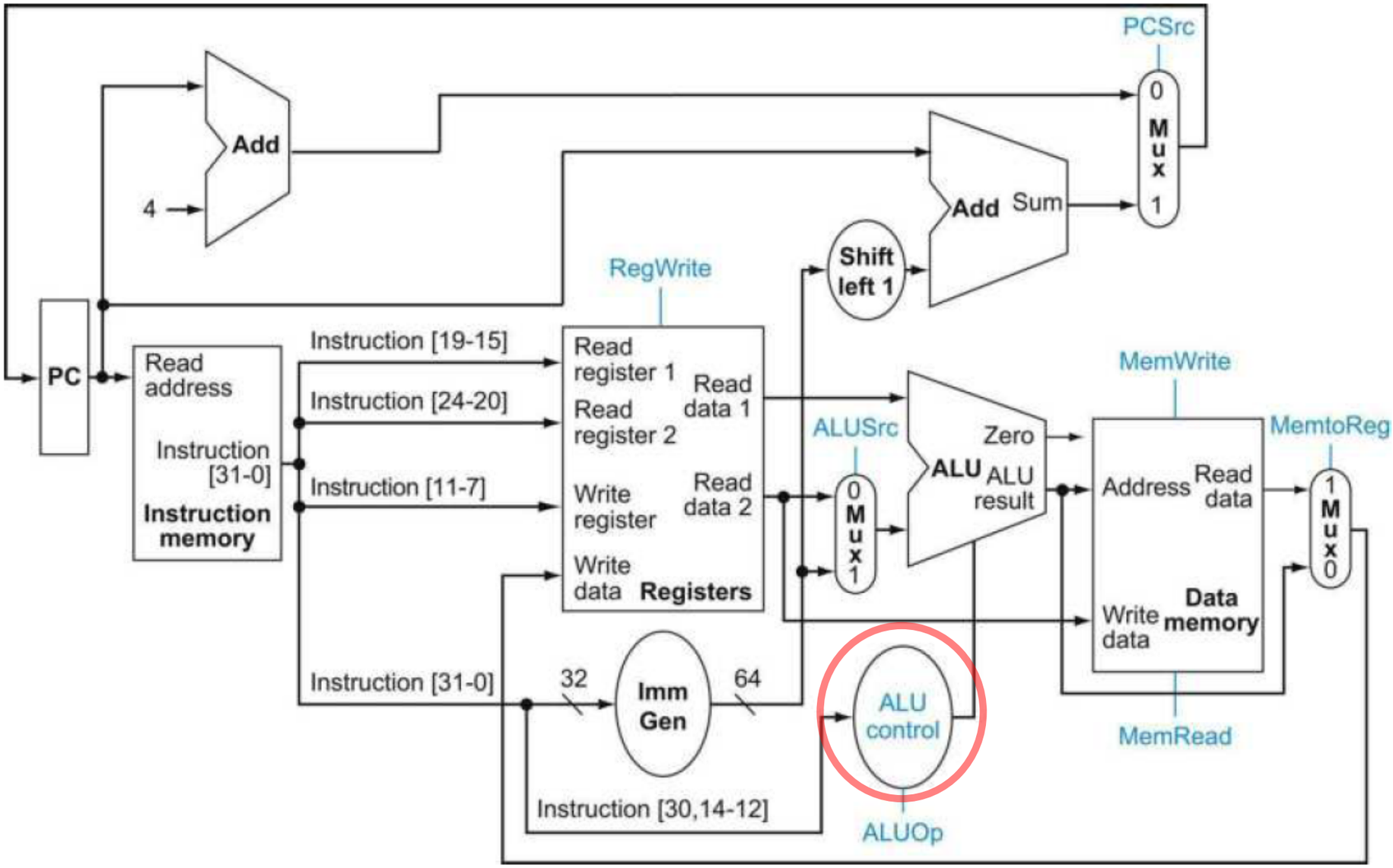


Figure 4.15 is from “Computer Organization and Design”

Building a Datapath

The Simple Datapath with Main Control Unit

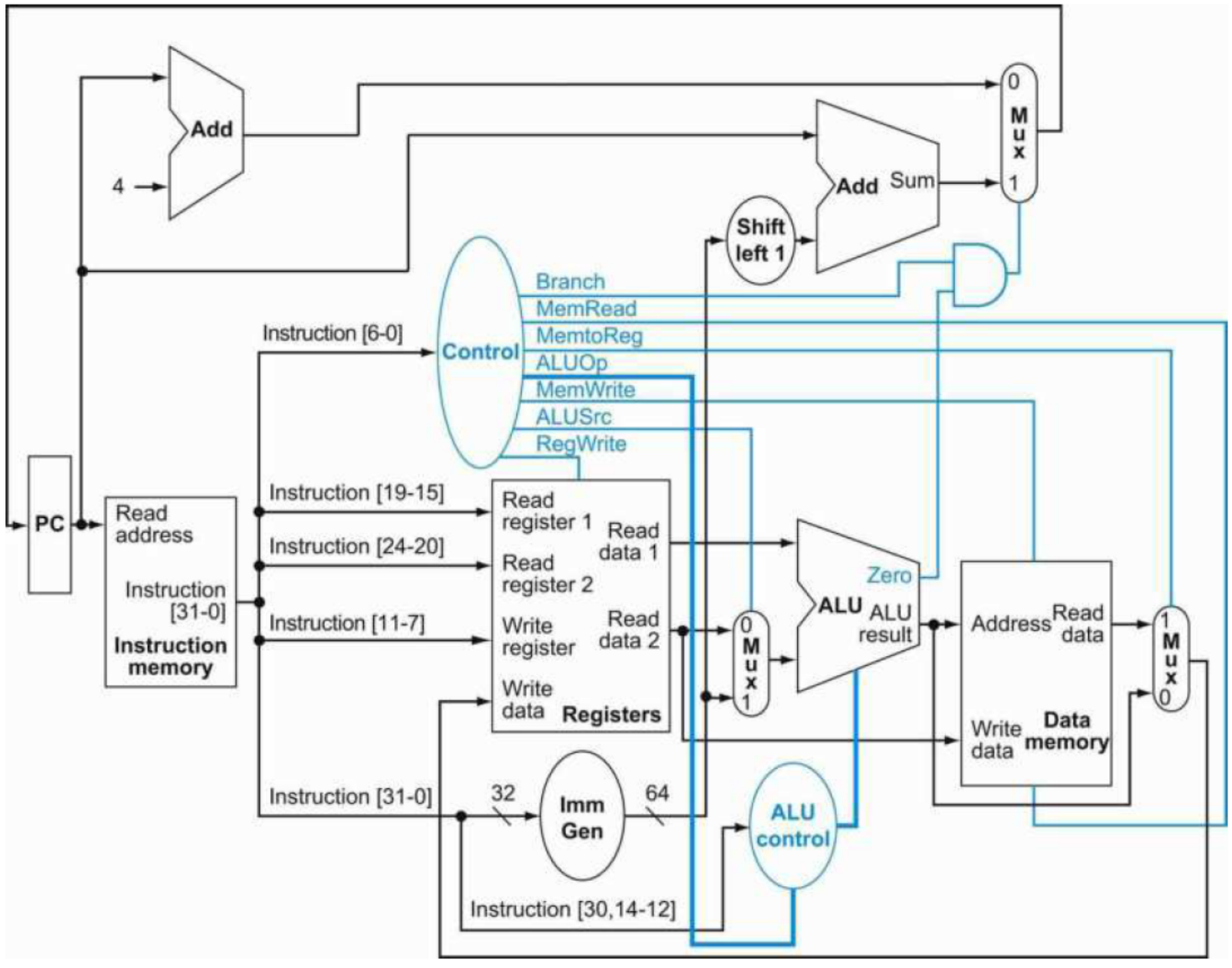


Figure 4.15 is from “Computer Organization and Design”

Building a Datapath

Summarize the Control Signal for each Instruction

Fig 4.18 shows which control signals are asserted or not for each instruction

Instruction	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	0	0	1	0	0	0	1	0
ld	1	1	1	1	0	0	0	0
sd	1	X	0	0	1	0	0	0
beq	0	X	0	0	0	1	0	1

Let’s see the example for each instruction using this information

Figure 4.18 is from “Computer Organization and Design”

Building a Datapath

Example of R-format Instruction

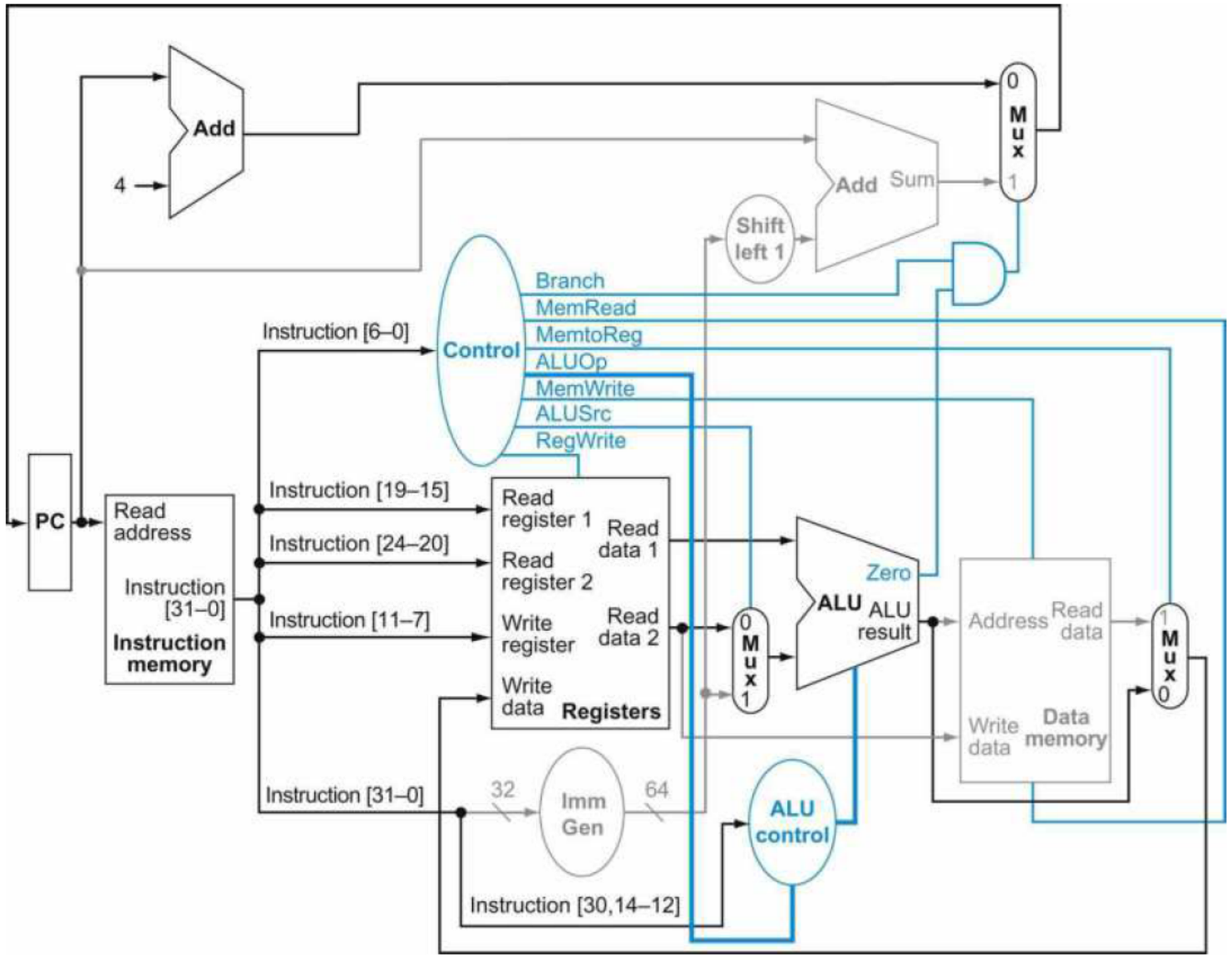


Figure 4.19 is from “Computer Organization and Design”

Building a Datapath

Example of Load Instruction

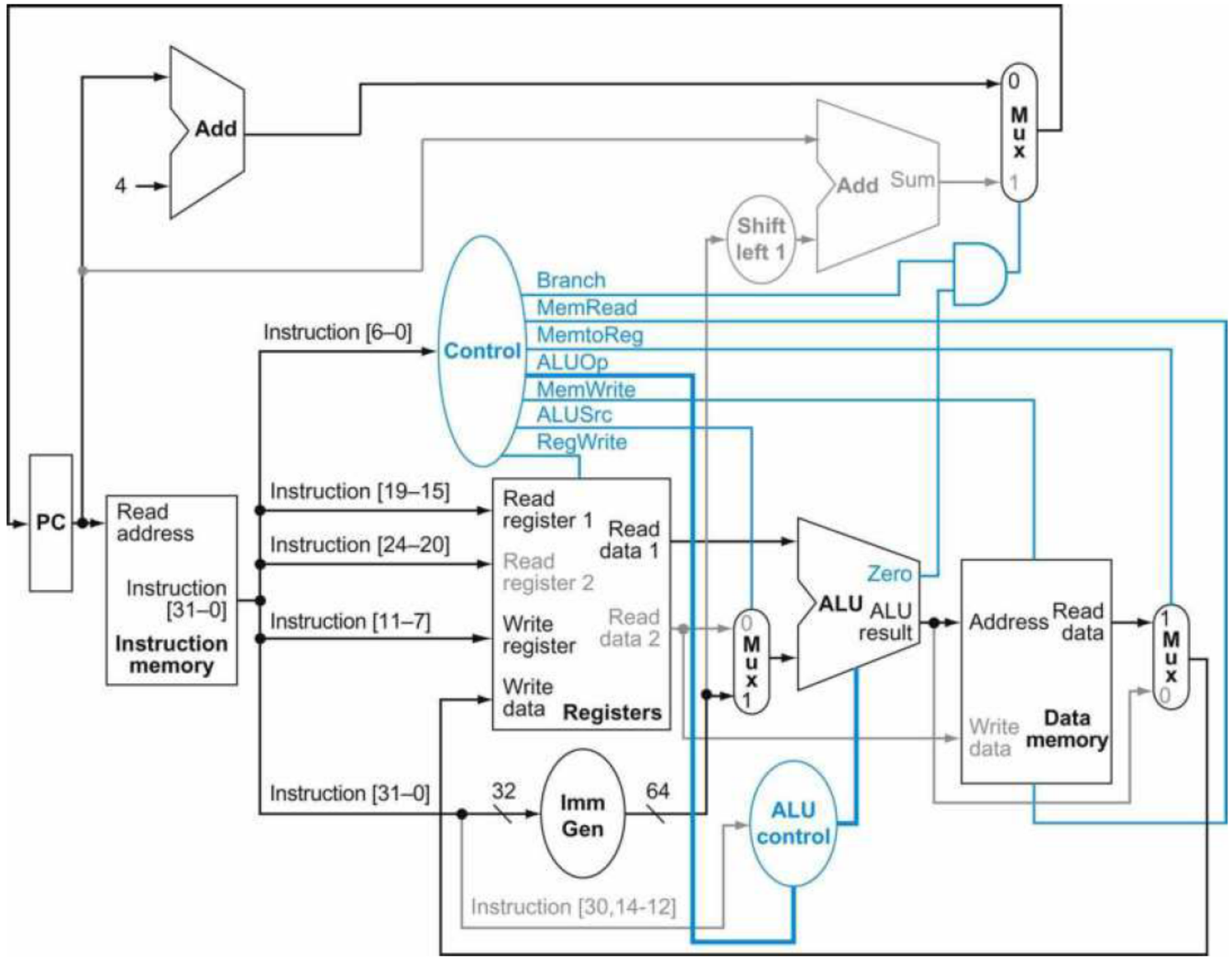


Figure 4.20 is from “Computer Organization and Design”

Building a Datapath

Example of beq Instruction

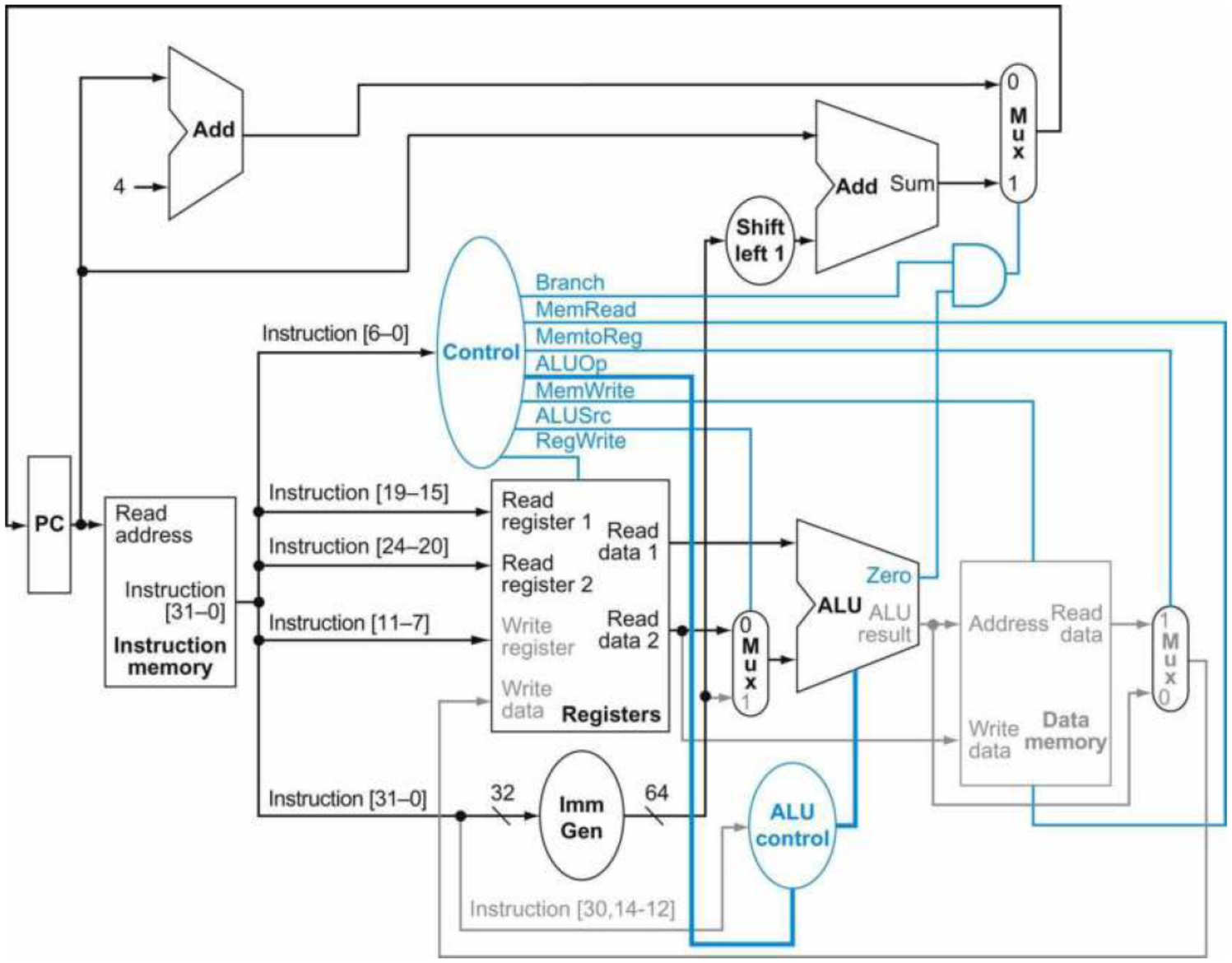


Figure 4.21 is from “Computer Organization and Design”

Building a Datapath

Single-Cycle Implementation

- Single-cycle design can work correctly

- But single-cycle implementation is inefficient. Why?
  - The longest path in the processor determines the clock cycle for every instruction
  - It is not suitable for high-performance processor

  - Violates **make common case fast**