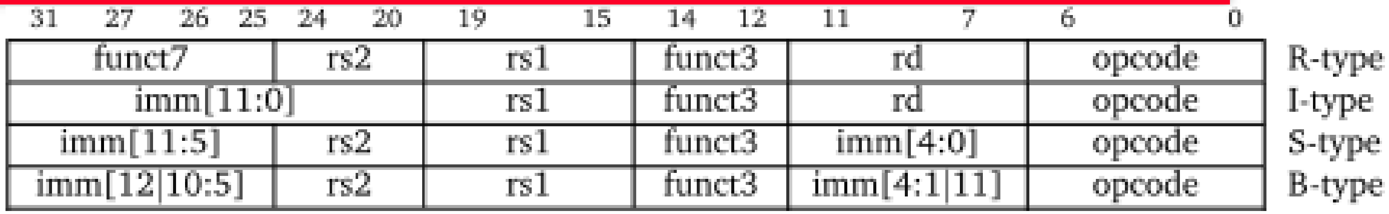
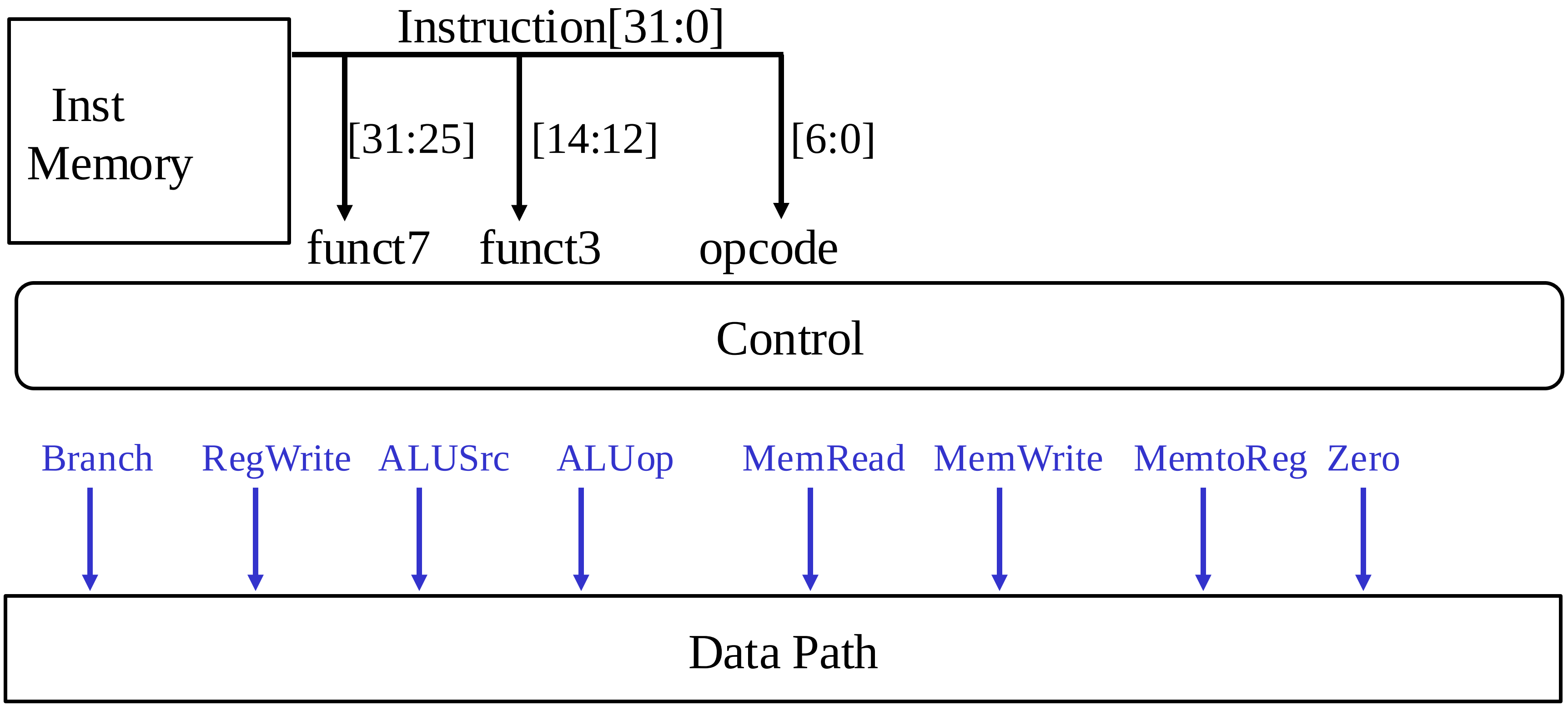
How to Design a Processor: Step by Step

1. Analyze instruction set => datapath requirements
   1. the meaning of each instruction is given by the *register transfers*
   2. datapath must include storage element for registers
   3. datapath must support each register transfer
2. Select the set of datapath components and establish clocking methodology
3. Assemble the datapath meeting the requirements
4. **Analyze the implementation of each instruction to determine the settings of the control points that affects the register transfer**
5. Assemble the control logic

Step 4: Given Datapath: RTL → Control



31 27 26 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]		rs1	funct3	rd	opcode	I-type
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode	B-type

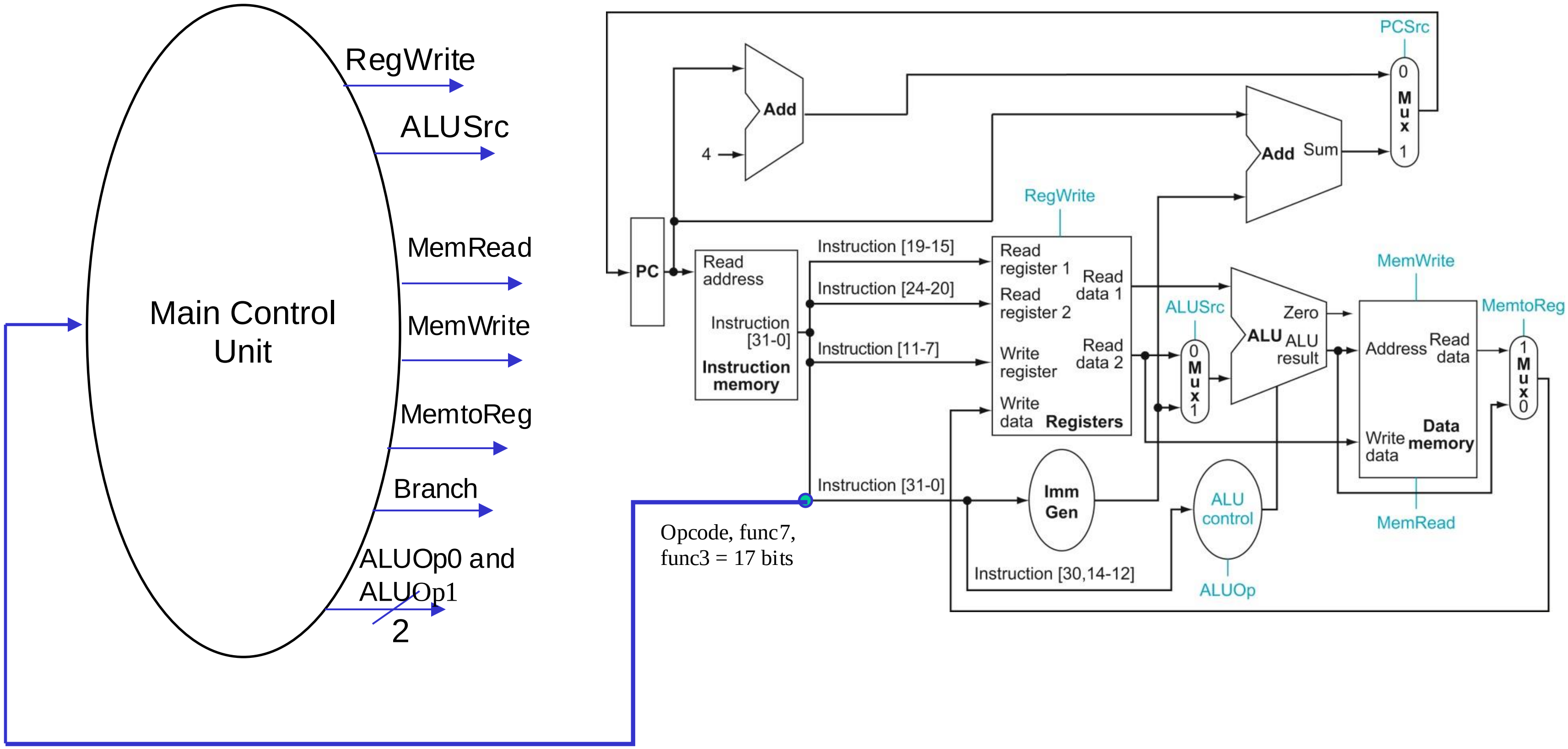


Main Control Unit

- Generate control signals for datapath elements & 2 bits for ALUop
- Control signals derived from instruction

R-type	funct7	rs2	rs1	funct3	rd	opcode
	7 bits	5 bits	5 bits	3 bits	5 bits	7 bits
Store	immediate[11:5] 7 bits	rs2 5 bits	rs1 5 bits	funct3 3 bits	immediate[4:0] 5 bits	opcode 7 bits
Load	immediate		rs1	funct3	rd	opcode
Branch	imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode
	7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

Main Control Unit



Main Control Unit

RegWrite

ALUSrc

MemRead

MemWrite

MemtoReg

Branch

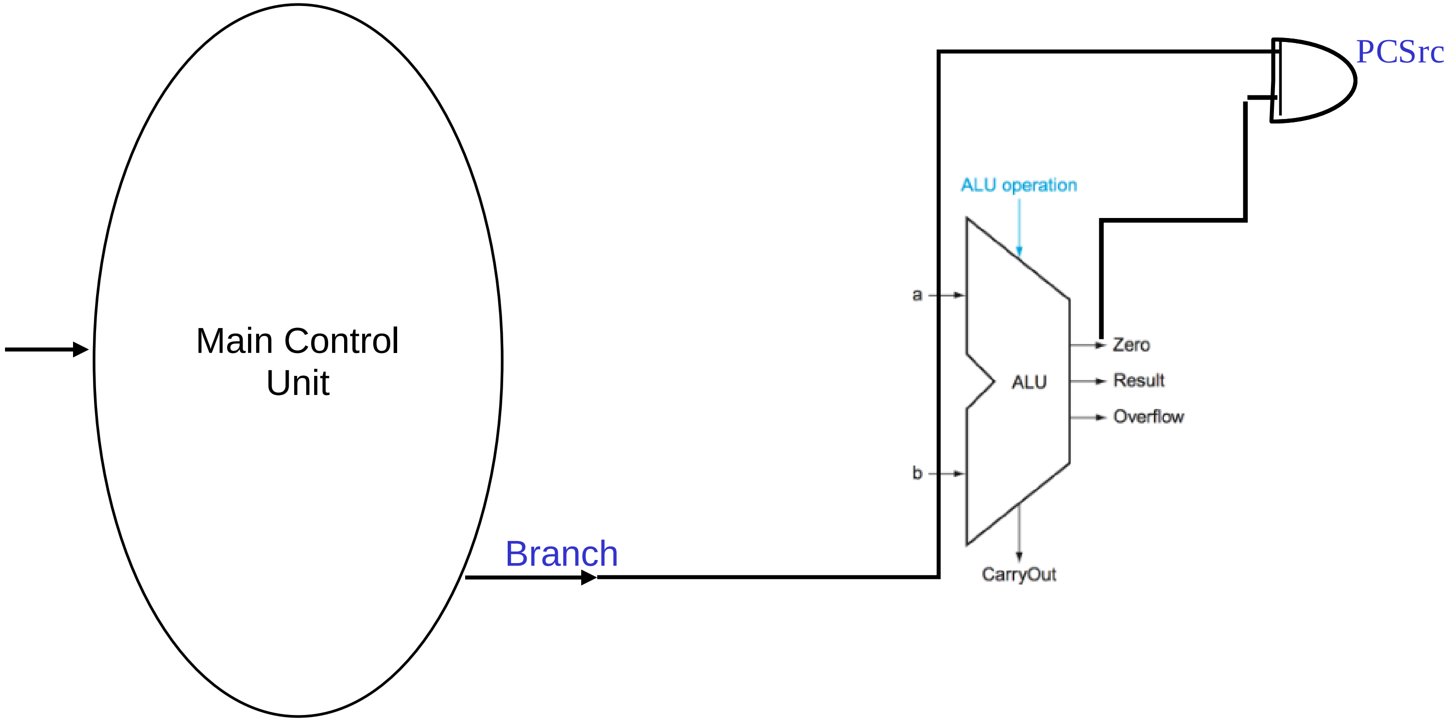
ALUOp0 and ALUOp1

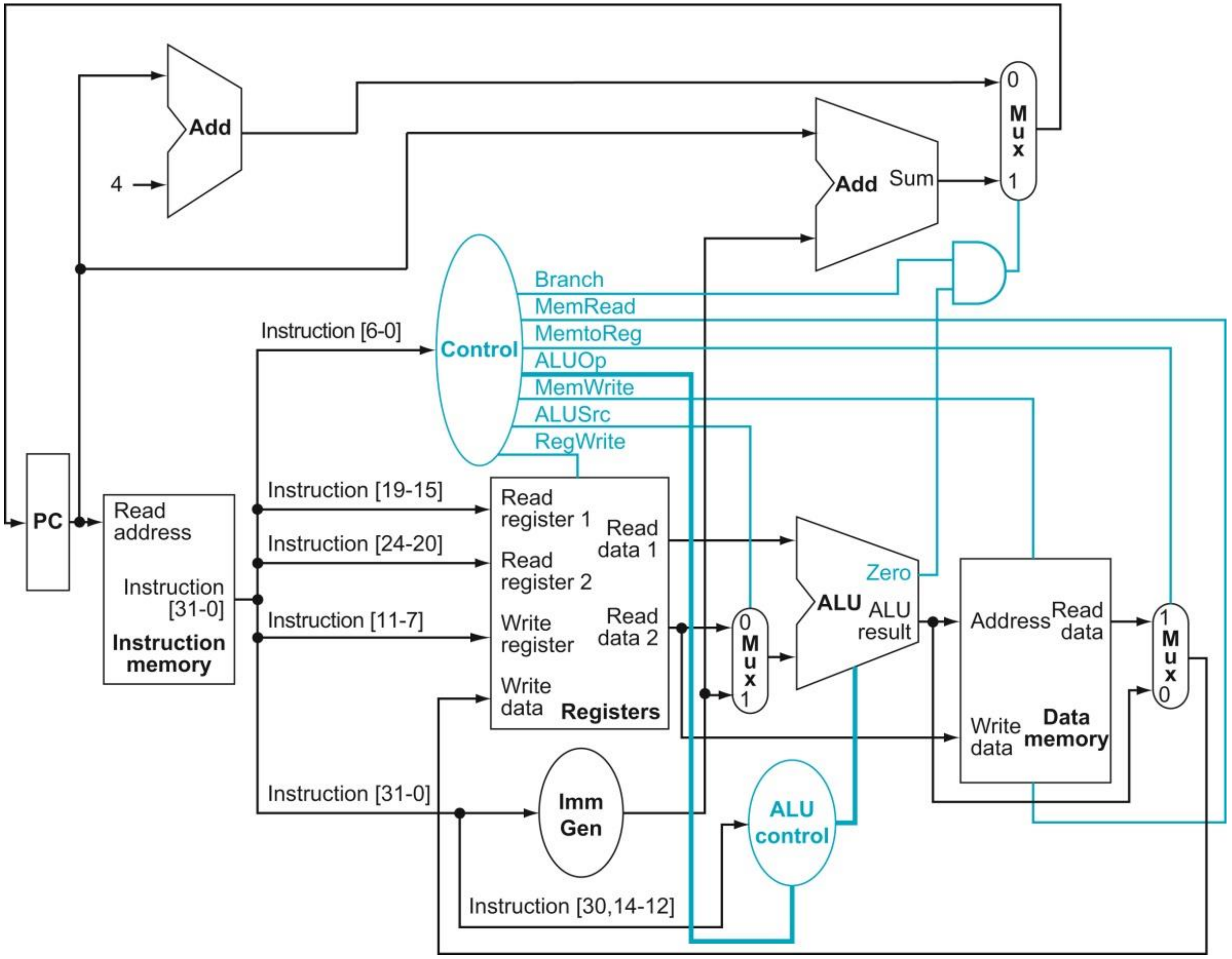
2

Opcode, func7, func3 = 17 bits

Branch is Special

- control alone is not enough




Add
4
Add
Sum
0
M
u
x
1
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [6-0]
PC
Read address
Instruction [31-0]
Instruction memory
Instruction [19-15]
Instruction [24-20]
Instruction [11-7]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
0
M
u
x
1
ALU
Zero
ALU result
Address
Read data
Write data
Data memory
1
M
u
x
0
Instruction [31-0]
Imm Gen
ALU control
Instruction [30,14-12]

Control

- **Controlling the flow of data (multiplexor inputs)**
  - Design the Main Control Unit
- **Selecting the operations to perform (ALU, read/write, etc.)**
  - Design the ALU Control Unit

- **Information comes from the 32 bits of the instruction**
- **Example: add t0, s1, s2**

Instruction Format

funct7	rs2	rs1	funct3	rd	opcode
0000000	01010	01001	000	00101	0110011

- **ALU's operation based on instruction type and function code**

Design of the Main Control Unit

- **Seven control signals**
  - RegWrite
  - ALUSrc
  - PCSrc
  - MemRead
  - MemWrite
  - MemtoReg

  - ALUOp
    - 00: loads and stores
    - 01: beq
    - 10: R-type, determined by func7 and func3

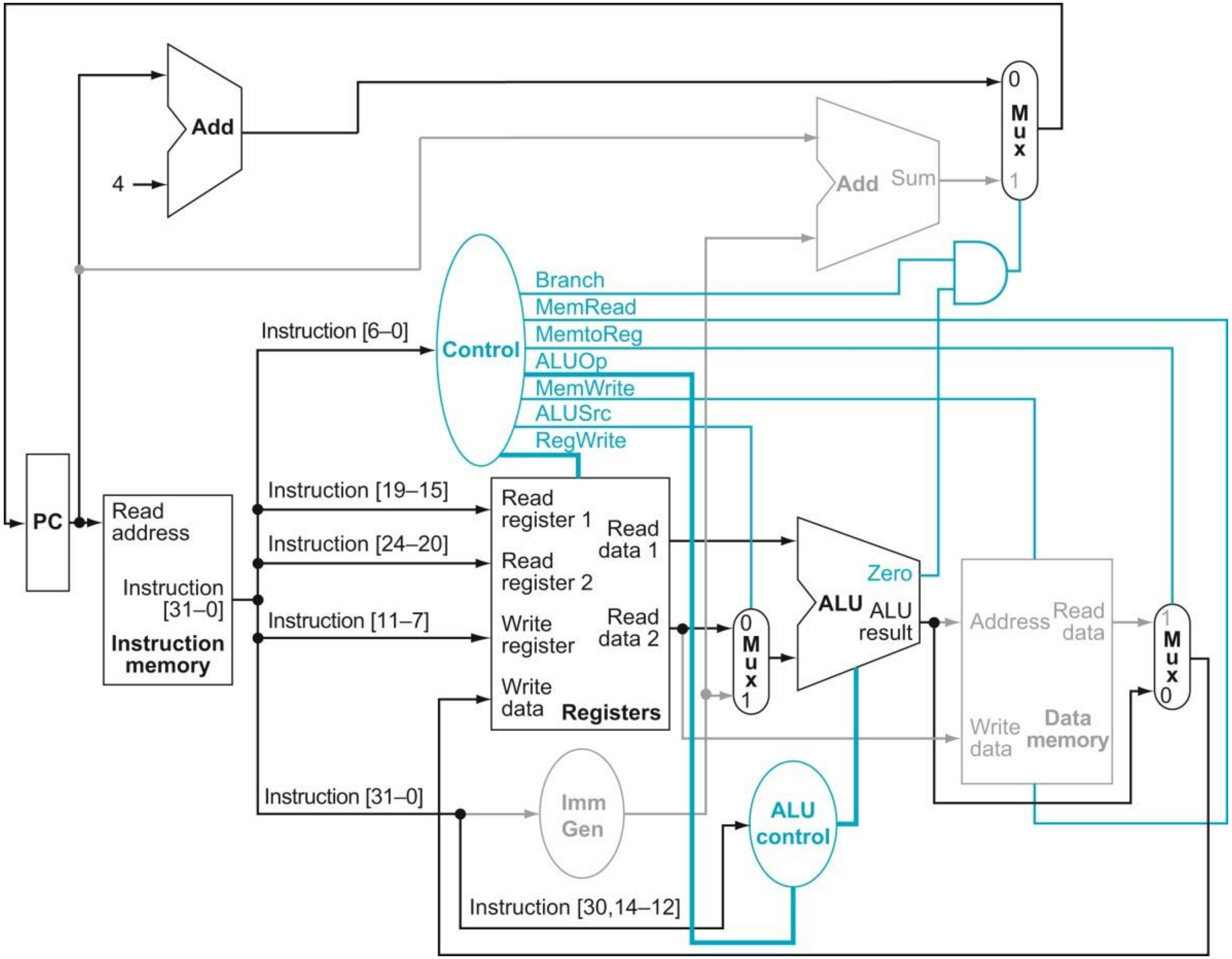
How to Design a Processor: Step by Step

1. Analyze instruction set => datapath requirements
    1. the meaning of each instruction is given by the *register transfers*
    2. datapath must include storage element for registers
    3. datapath must support each register transfer
2. Select the set of datapath components and establish clocking methodology
3. Assemble the datapath meeting the requirements
4. Analyze the implementation of each instruction to determine the settings of the control points that affects the register transfer
5. **Assemble the control logic**

Control Signals

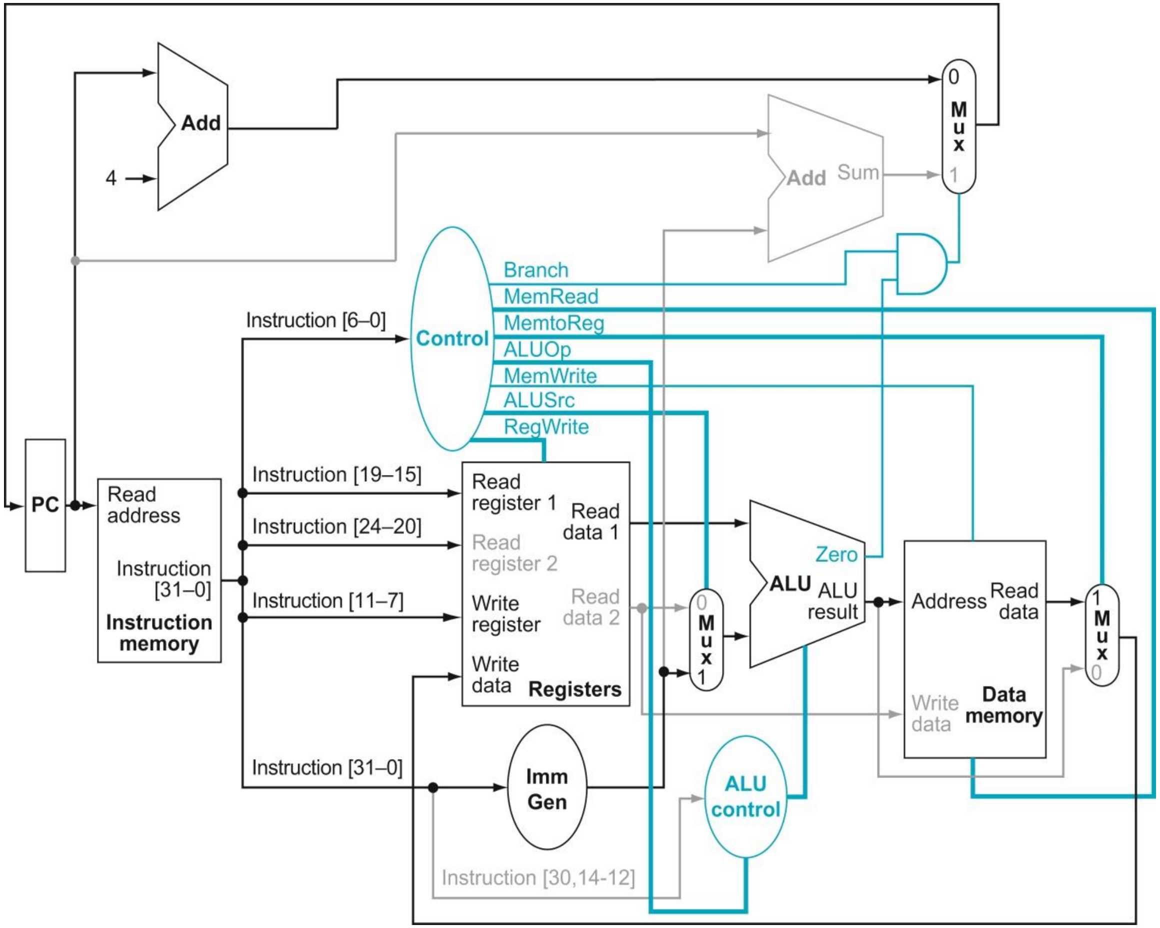
1	RegWrite = 1	The register on the Write register input is written with the data on the Write data input (at the next clock edge)
2	ALUSrc = 0	The second ALU operand comes from Read data 2
	ALUSrc = 1	The second ALU operand comes from the sign-extension unit
3	PCSrc = 0	The PC is replaced with PC+4
	PCSrc = 1	The PC is replaced with the branch target address
4	MemtoReg = 0	The value input to the regular write data input comes from the ALU
	MemtoReg = 1	The value input to the regular write data input comes from the data memory
5	MemRead = 1	Read data memory
6	MemWrite = 1	Write data memory

Control for R-Format Instructions



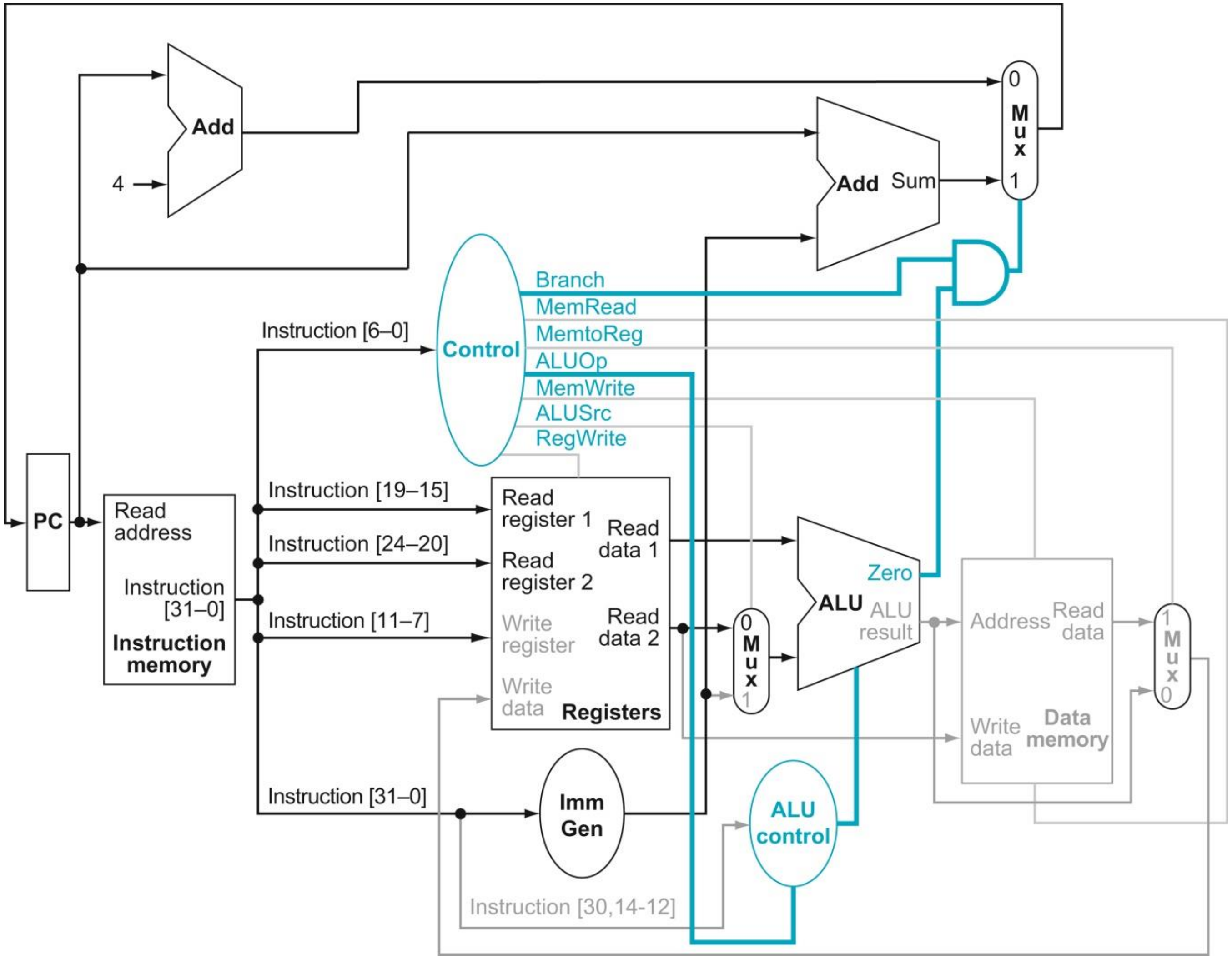
RegWrite	1
ALUSrc	0
Branch	0
MemtoReg	0
MemRead	0
MemWrite	0
ALUOp1	1
ALUOp0	0

Control for Load/Store Instructions



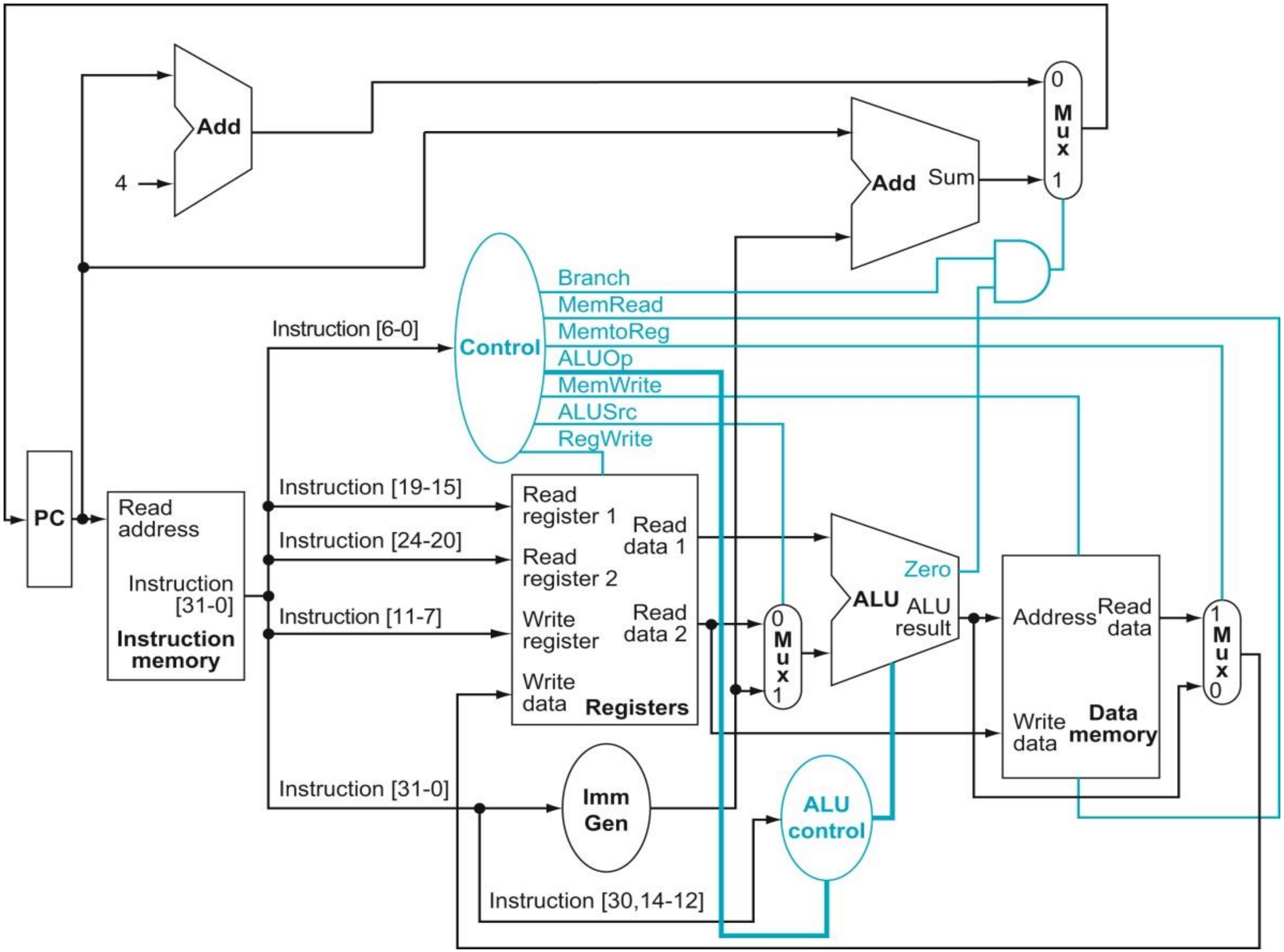
	lw	sw
RegWrite	1	0
ALUSrc	1	1
Branch	0	0
MemtoReg	1	X
MemRead	1	0
MemWrite	0	1
ALUOp1	0	0
ALUOp0	0	0

Control for Branch on Equal



RegWrite	0
ALUSrc	0
Branch	1
MemtoReg	X
MemRead	0
MemWrite	0
ALUOp1	0
ALUOp0	1

Single-Cycle Data Path with Control



Design of the ALU Control Unit

- **ALU used for R-type**
  - Function depends on funct field

ALU control	Function
0000	AND
0001	OR
0010	add
0110	subtract

ALU Control for R-Type

- **2-bit ALUOp derived from opcode**
  - Combinational logic derives ALU control
  - Think of the table as IF statements
    - If opcode says it’s R-type, then set ALUOp=10
    - If ALUOp=10, then tell ALU Control to use funct3 and funct7 to set ALU Control

opcode	**ALUOp**	**Operation**	**funct7**	**funct3**	**ALU function**	**ALU control**
R-type	10	add	0000000	000	add	0010
		subtract	0100000	000	subtract	0110
		AND	0000000	111	AND	0000
		OR	0000000	110	OR	0001

ALU Control for lw/sw

- Assume 2-bit ALUOp derived from opcode
  - Combinational logic derives ALU control
- ALU used for Load/Store: F = add

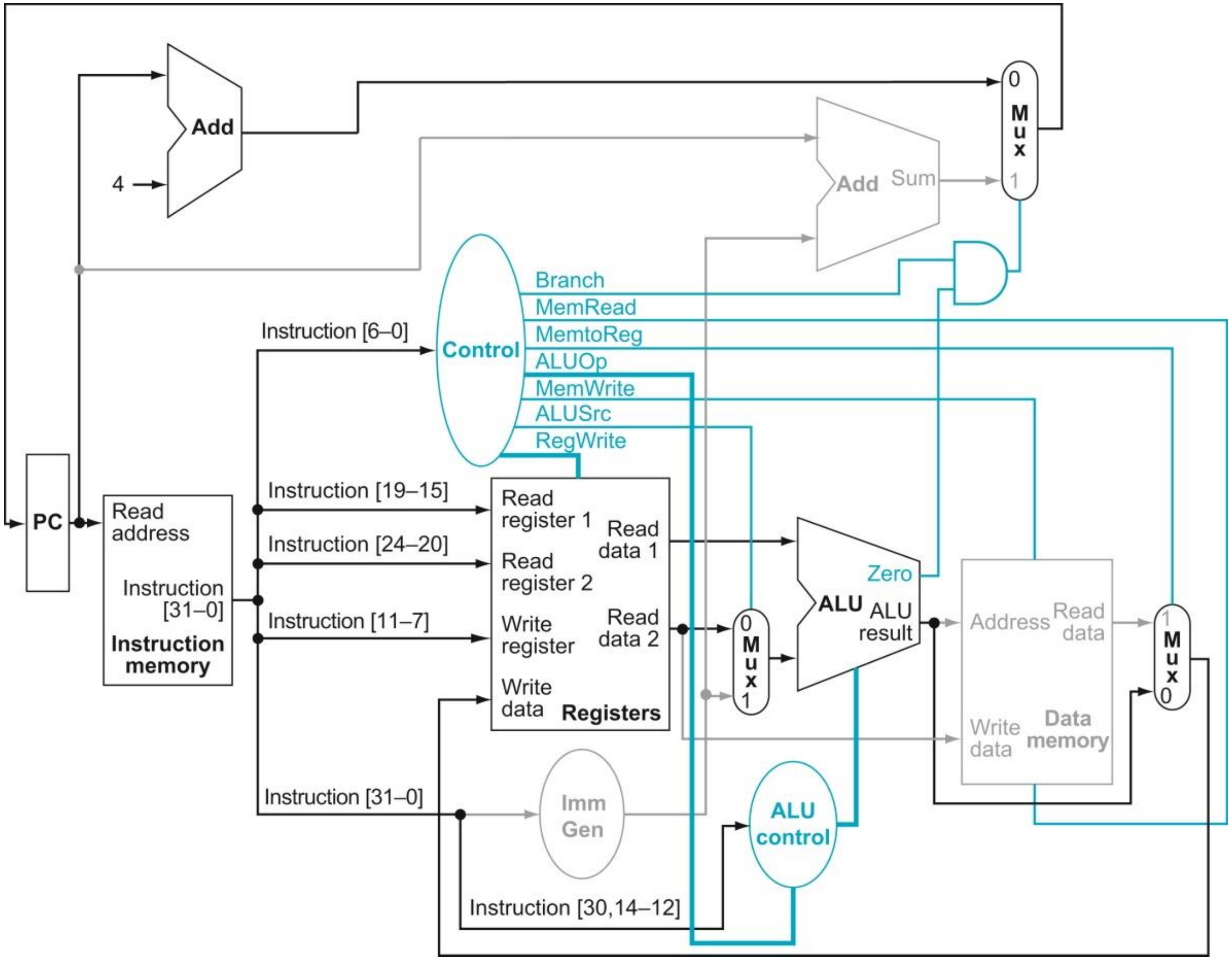
opcode	ALUOp	Operation	funct7	funct3	ALU function	ALU control
lw	00	load word	XXXXXXX	XXX	add	0010
sw	00	store word	XXXXXXX	XXX	add	0010
R-type	10	add	0000000	000	add	0010
		subtract	0100000	000	subtract	0110
		AND	0000000	111	AND	0000
		OR	0000000	110	OR	0001

ALU Control for beq

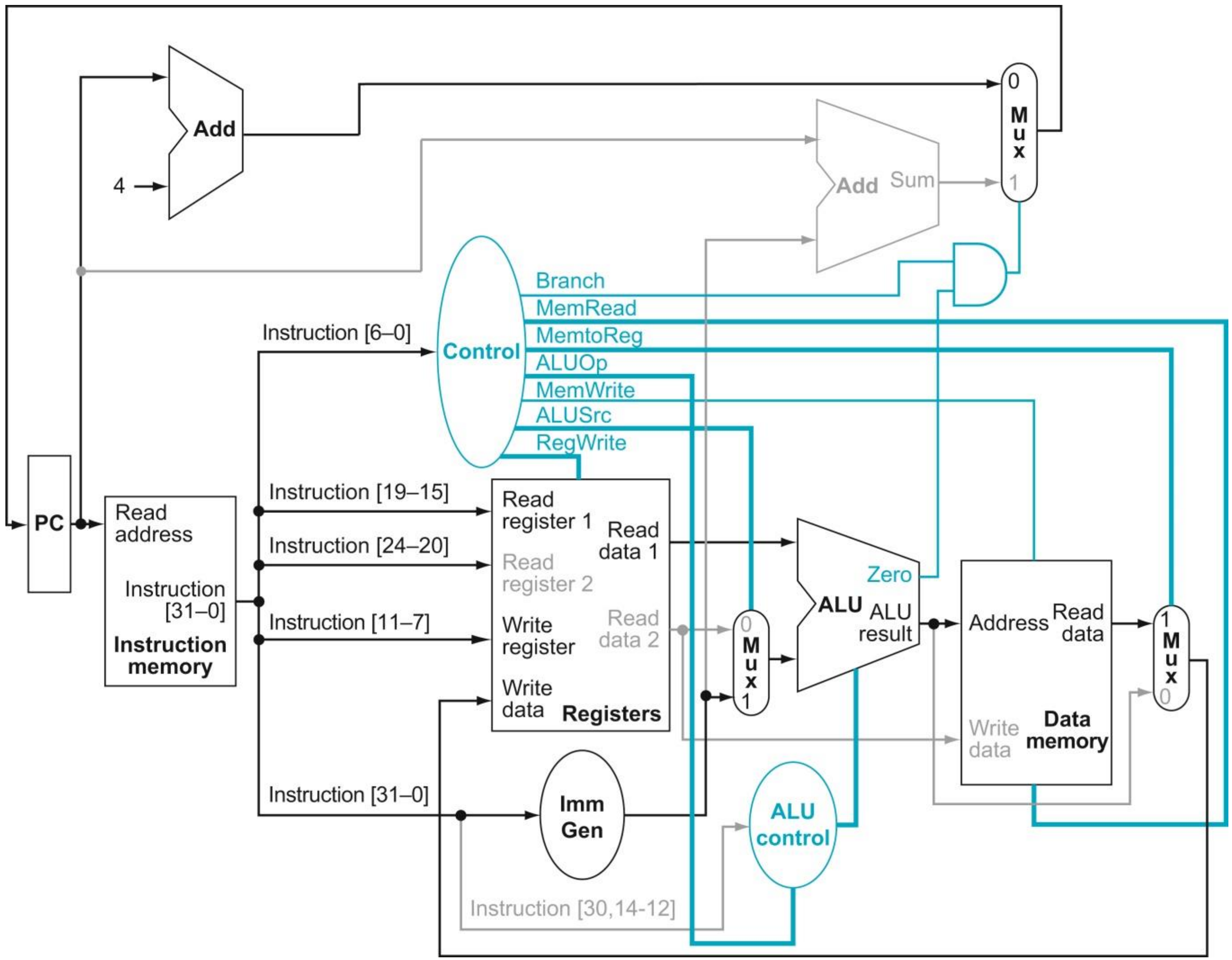
- beq F = sub
- lw/sw F = add
- R-type F depends on funct3/funct7

opcode	ALUOp	Operation	funct7	funct3	ALU function	ALU control
beq	01	branch on equal	XXXXXXX	XXX	sub	0110
lw	00	load word	XXXXXXX	XXX	add	0010
sw	00	store word	XXXXXXX	XXX	add	0010
R-type	10	add	0000000	000	add	0010
		subtract	0100000	000	subtract	0110
		AND	0000000	111	AND	0000
		OR	0000000	110	OR	0001

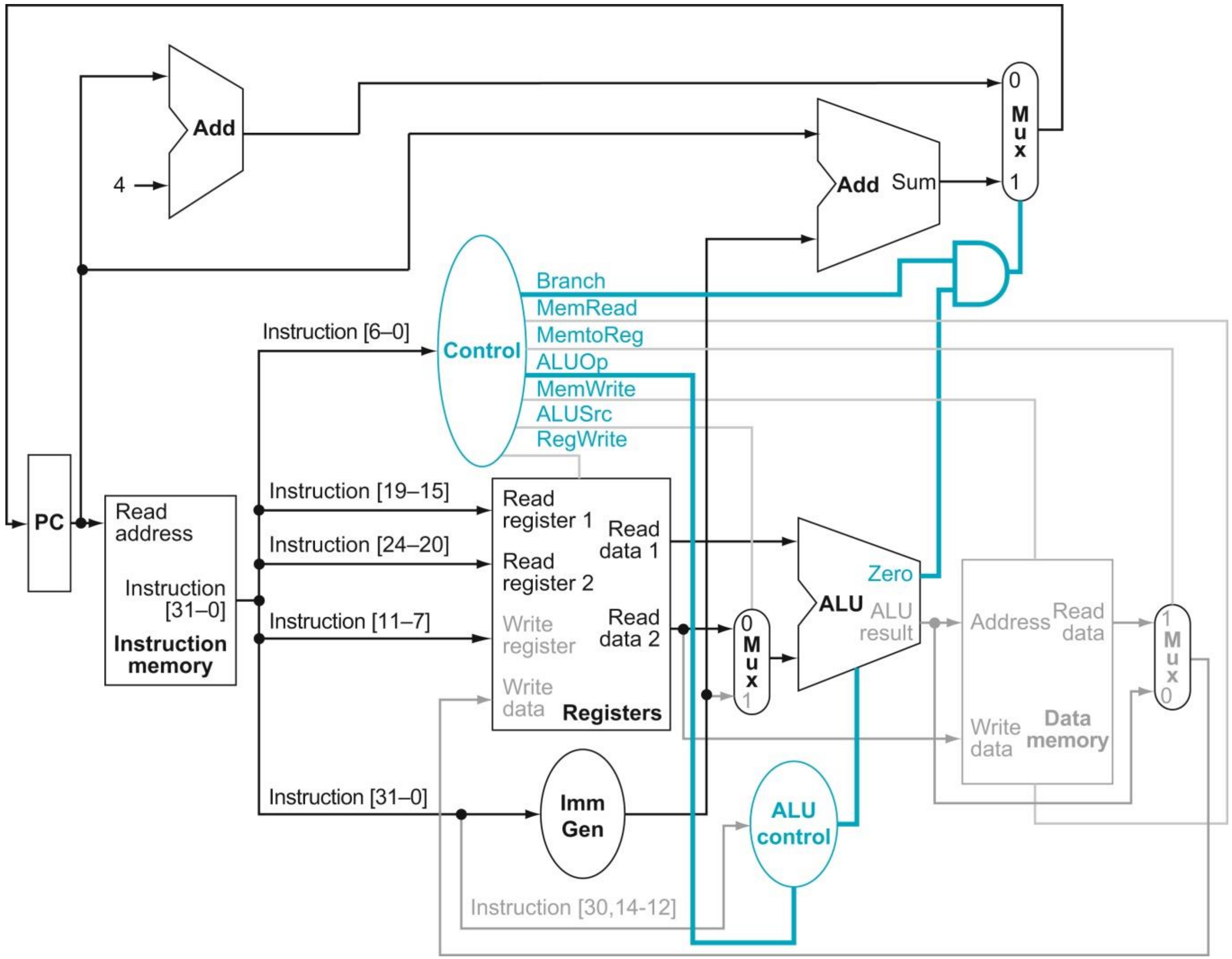
The Datapath in Operation for R-type Instructions (e.g. `add t1, t2, t3`)



The Datapath in Operation for `lw`



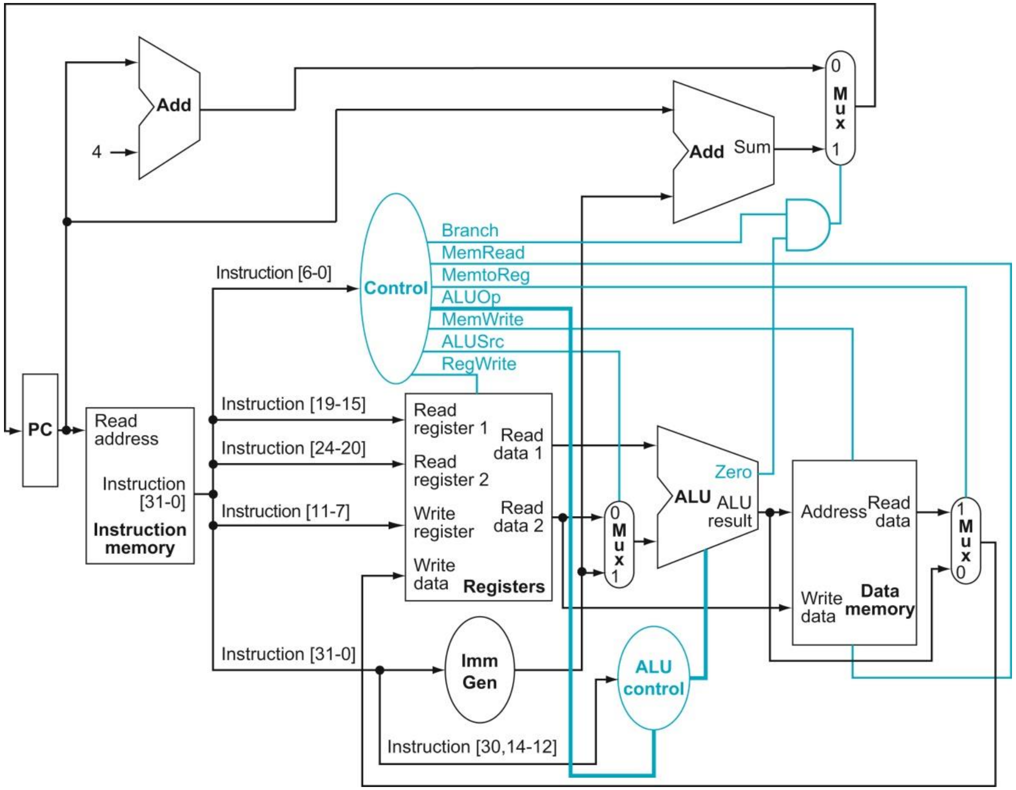
The Datapath in Operation for `beq`



Problem: `lw t1, 32(t2)`

31 27 26 25 24 20	19 15	14 12	11 7	6 0	
imm[11:0]	rs1	funct3	rd	opcode	I-type

- **Inactive blocks:**
- **Instruction [19-15]=**
- **Instruction [24-20] =**
- **Instruction [11-7]=**
- **Branch=**
- **MemRead=**
- **MemtoReg=**
- **ALUOp= 00**
- **MemWrite=**
- **ALUSrc=**
- **RegWrite=**



More Questions

- What is routed to ‘Write register’?
- What value comes out of ImmGen?
- Why does it not matter what value goes to Read register 2?
- Why does it not matter if the Zero flag is set or not coming out of the ALU?
- What other inputs are ignored due to control signals?

Summary of Single-Cycle Datapath

- 5 steps to design a processor
  - 1. Analyze instruction set => datapath <u>requirements</u>
  - 2. Select set of datapath components & establish clock methodology
  - 3. <u>Assemble</u> datapath meeting the requirements
  - 4. Analyze implementation of each instruction to determine setting of control points that affects the register transfer
  - 5. Assemble the control logic
- RISC-V makes it easier
  - Instructions same size
  - Source registers always in same place
  - Immediates minimize placement differences
  - Operations always on registers/immediates
- Single cycle datapath => CPI=1, CCT = long