NWEN 242

7. Processor: single cycle datapath control

Agenda

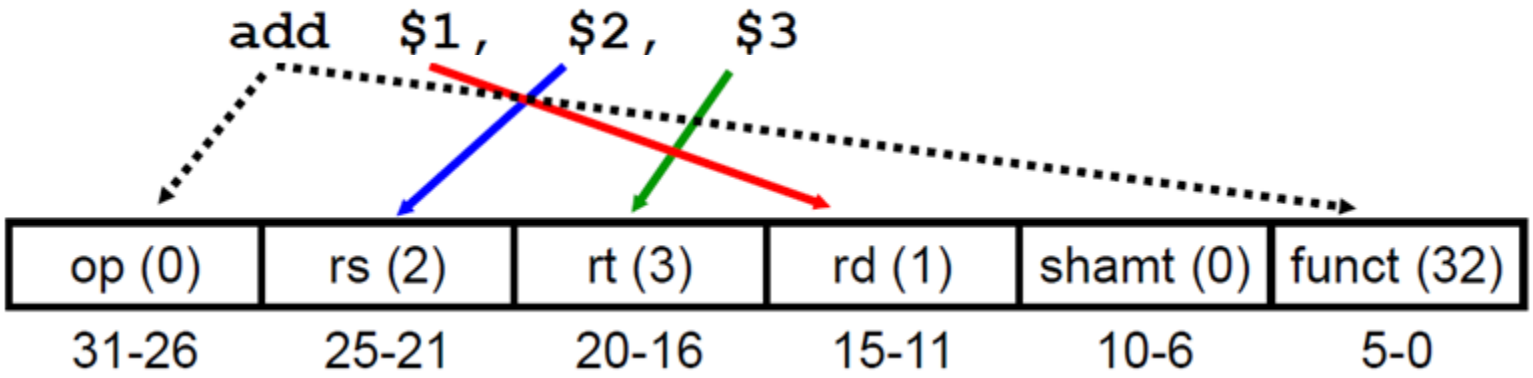
- Control for the datapath
  - What control signals are needed?
- Designing a controller for the datapath
  - Inputs
    - The instructions fields
    - The ALU zero
  - Outputs
    - To multiplexors
    - To ALU control inputs
    - To memory block and register file

Control for the datapath

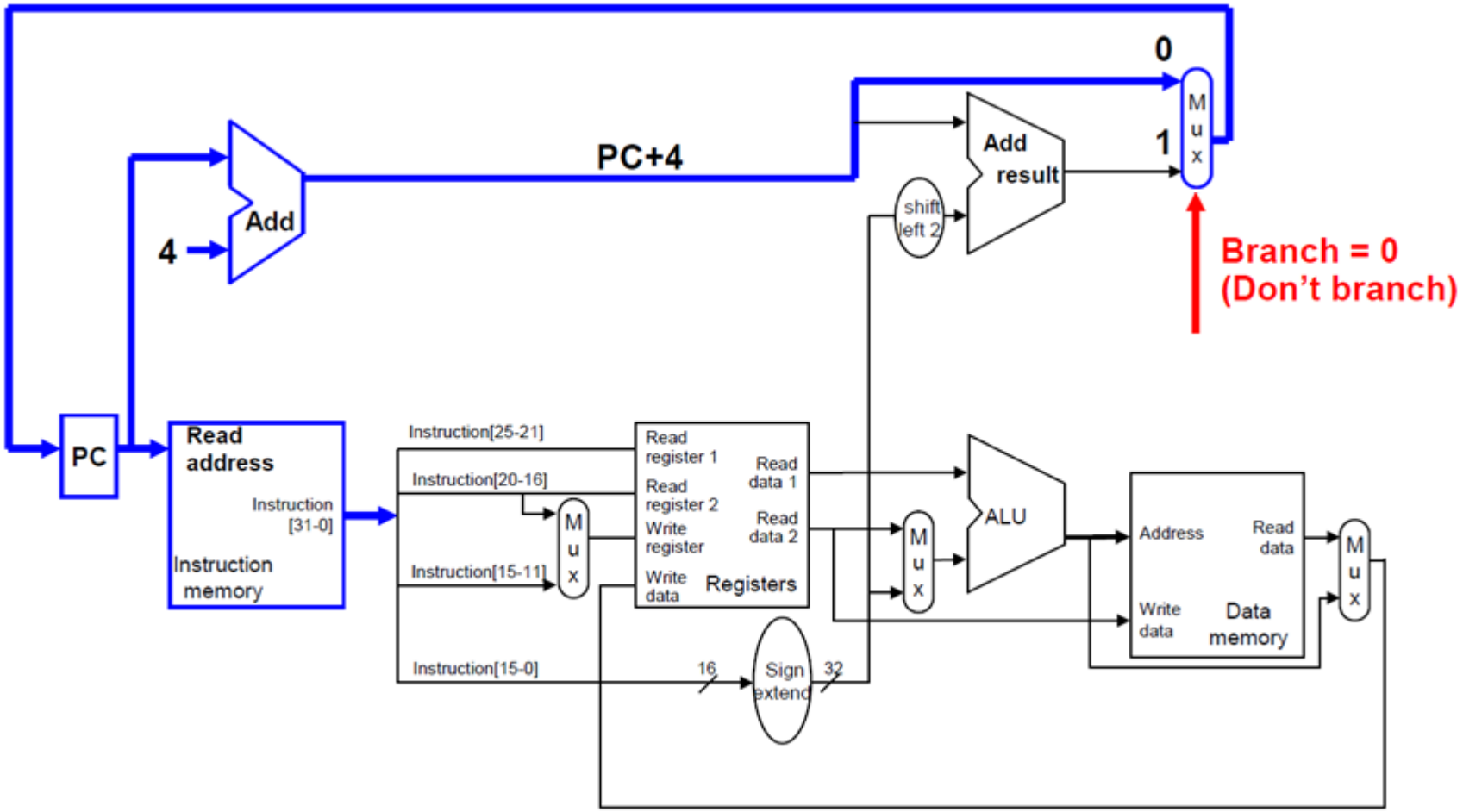
- Our datapath needs control signals
  - Multiplexors need a select input
  - ALUs need an operation input
  - Memories need read/write input
- Control signals are derived from the instruction opcode
- We shall see an R-type instruction in the datapath and
  - Consider 3 areas of the datapath in turn
  - See (a subset of) the control signals that are needed in each area
- Memory, branch, and jump instructions need similar control

Review of R-type instruction

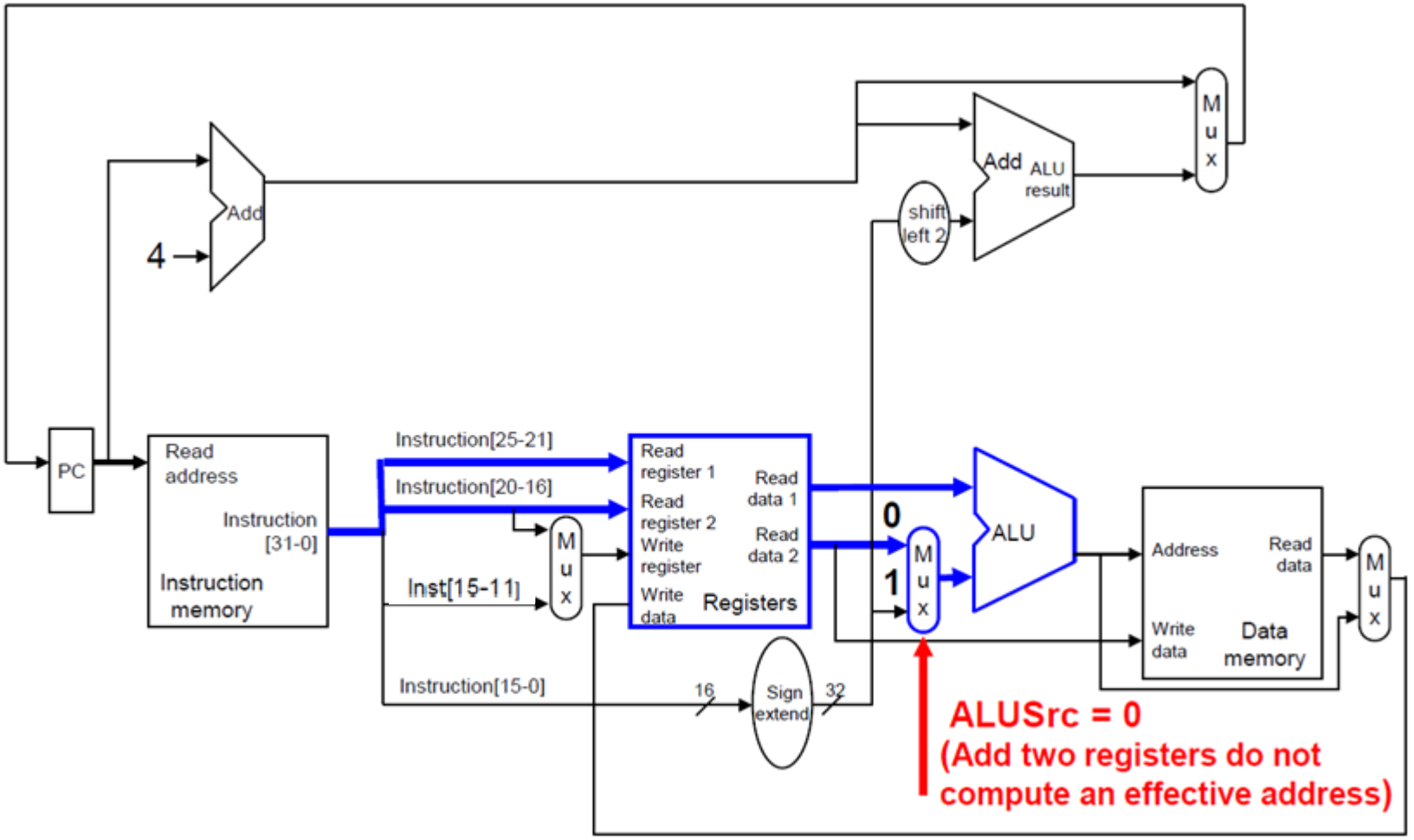
- Typical instruction



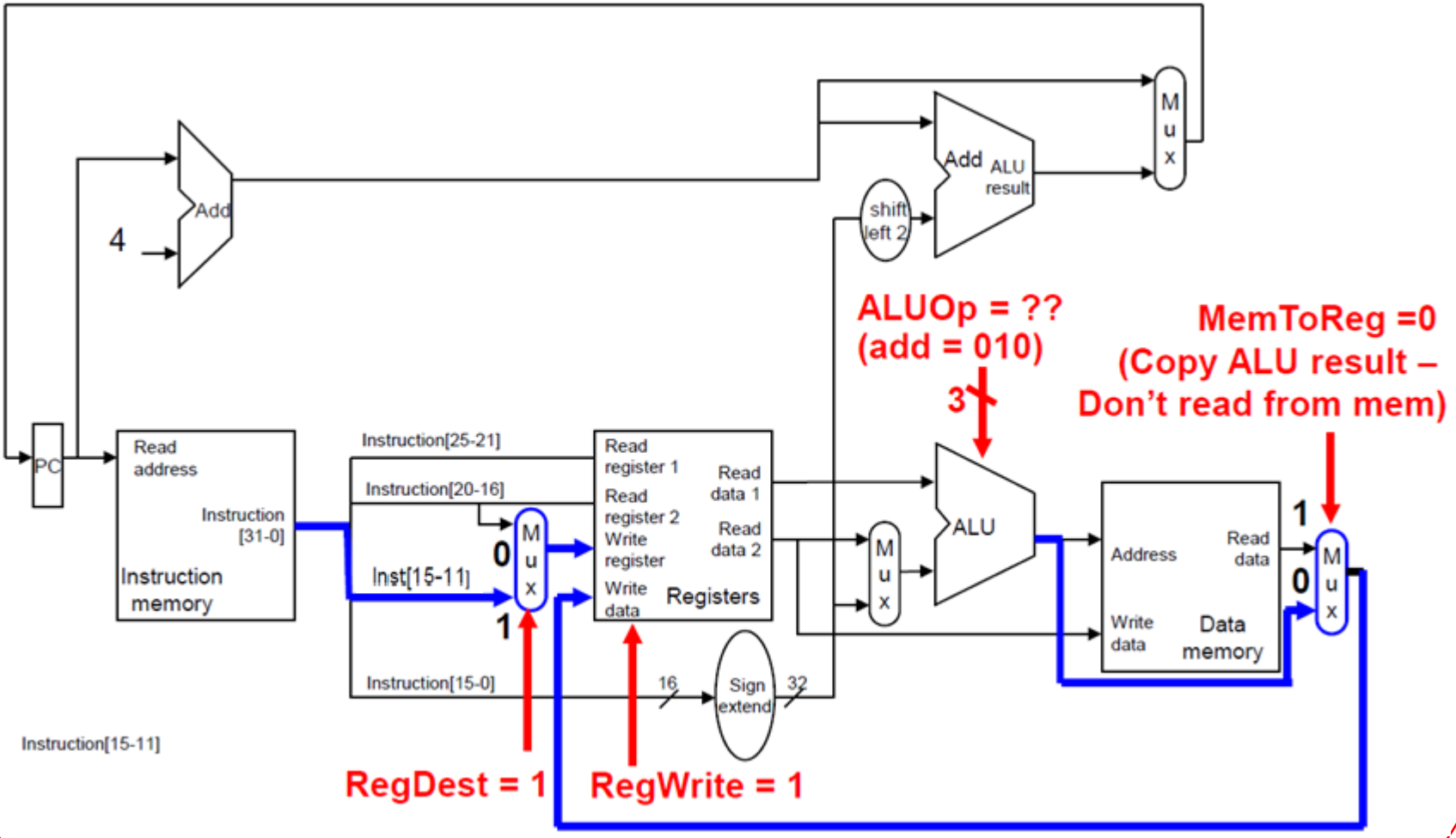
R-type (area 1 – instruction fetch)



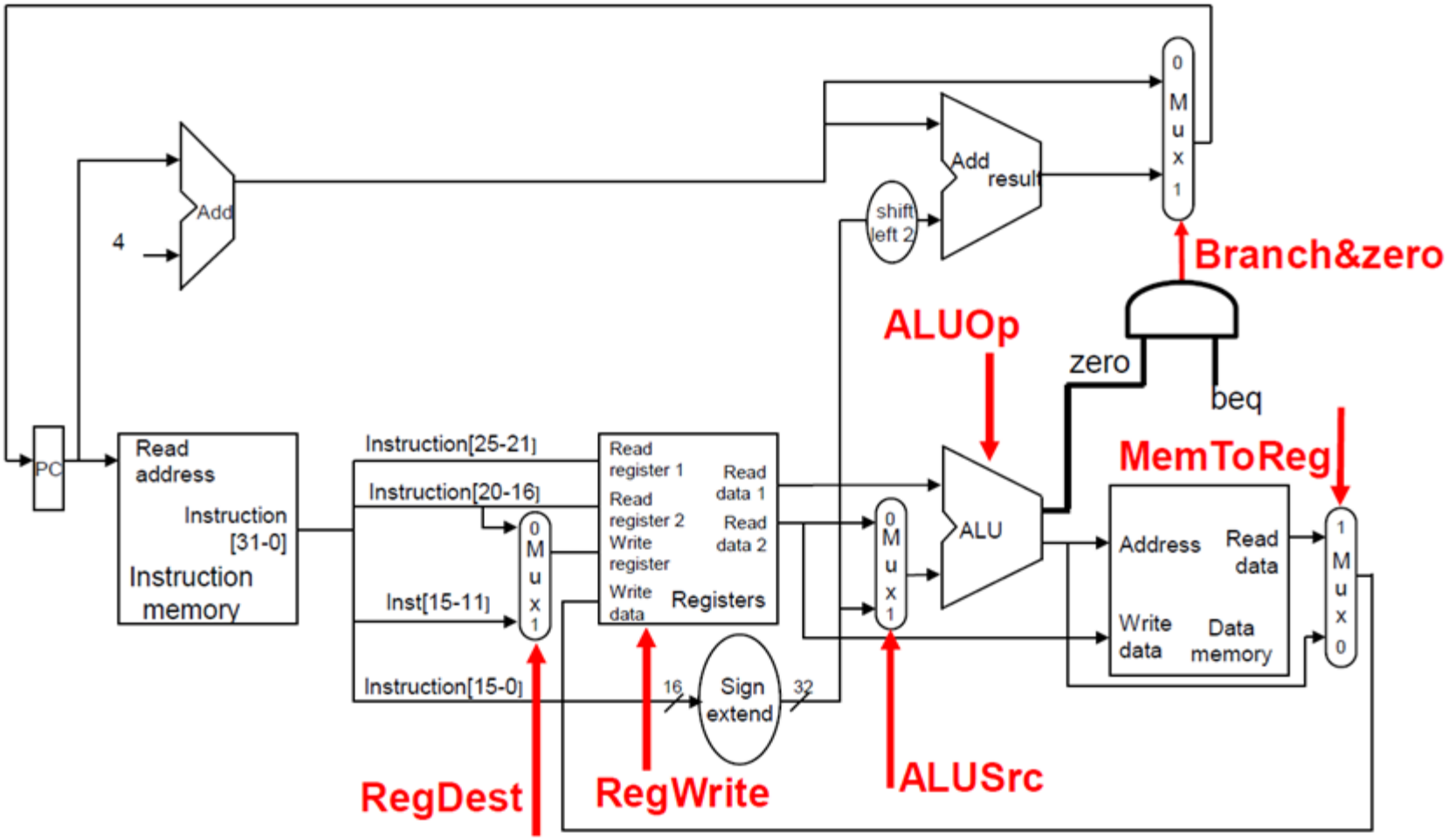
R-type (area 2 – ALU input)



R-type (area 3 – compute & store result)



Summary: the six control signals



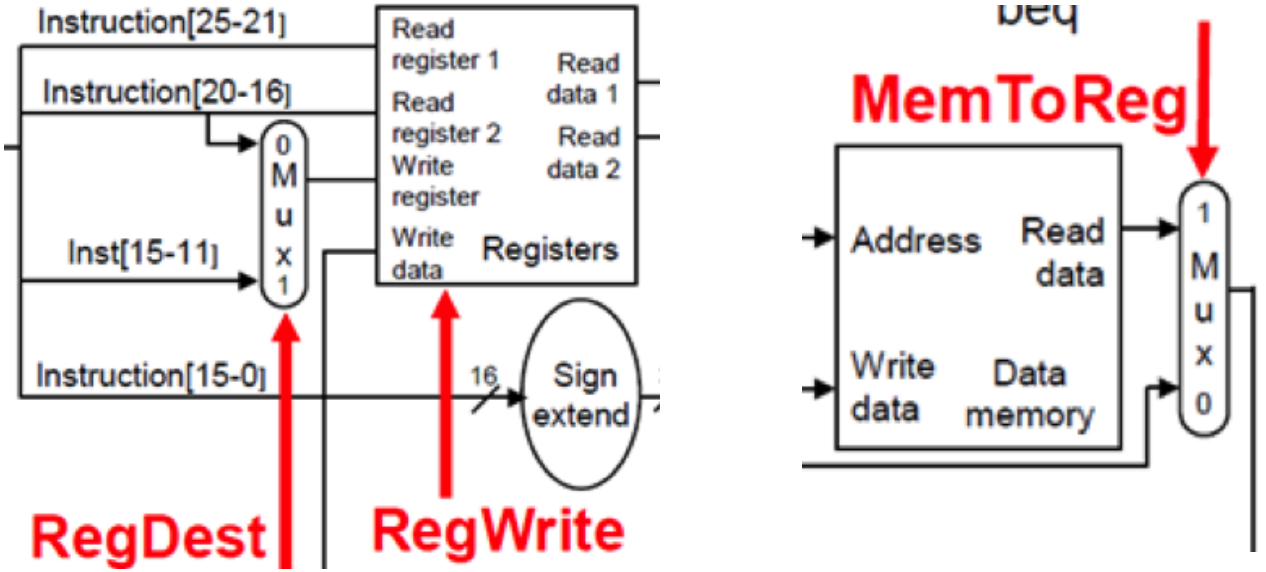
Truth table for control signals

OpCode	Instruction	Reg Dst	ALU Op	ALU Src	BEQ	MemTo Reg	Reg Write
000000	R-type	1	funct	0	0	0	1
100011	lw	0	add	1	0	1	1
101011	sw	x	add	1	0	x	0
000100	beq	x	sub	0	1	x	0

- X's are don't cares
- Can be either 0 or 1 but will not influence our design of the datapath for the instruction considered

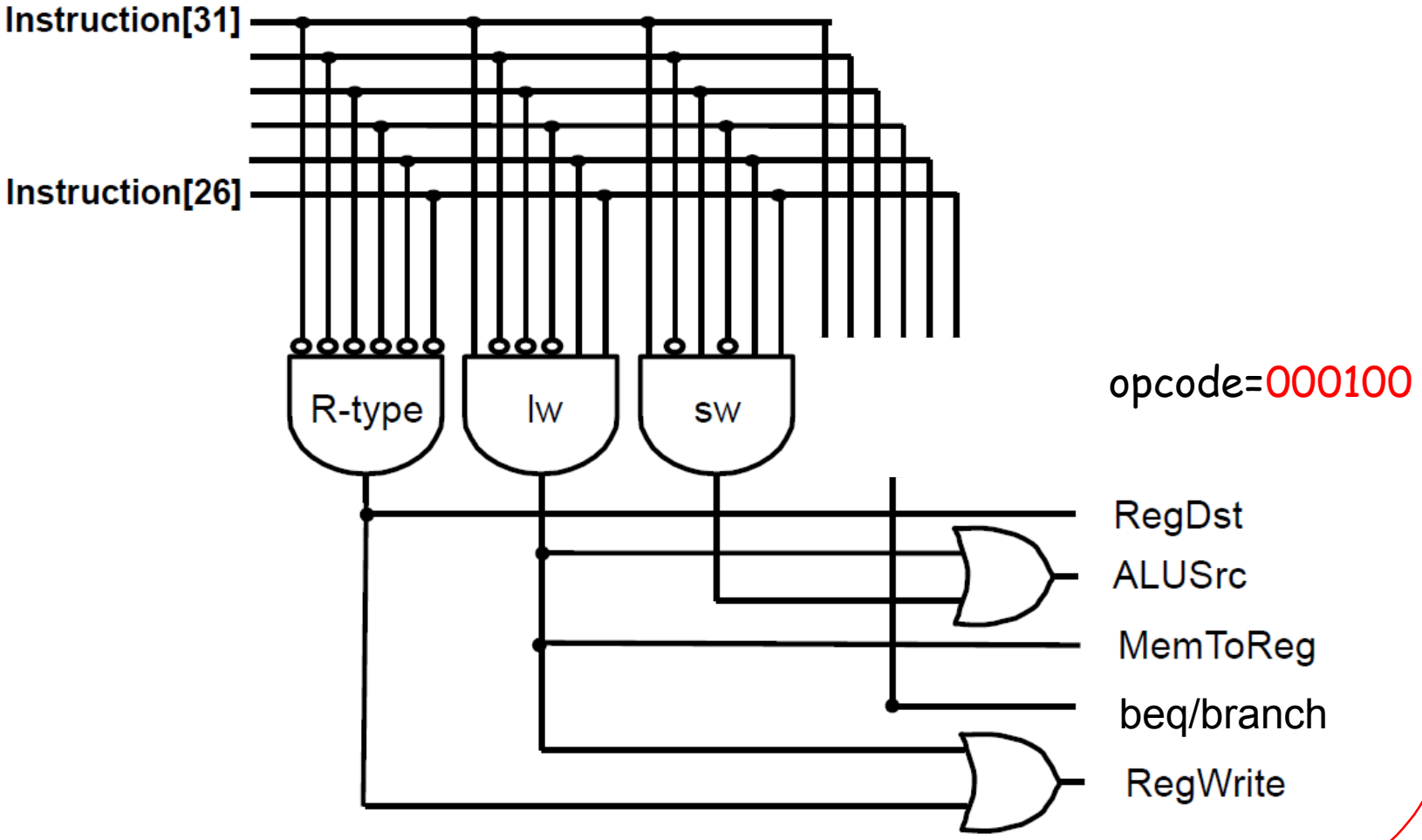
Quick exercise

- For the sw instruction, why RegDst=x and MemToReg=x?

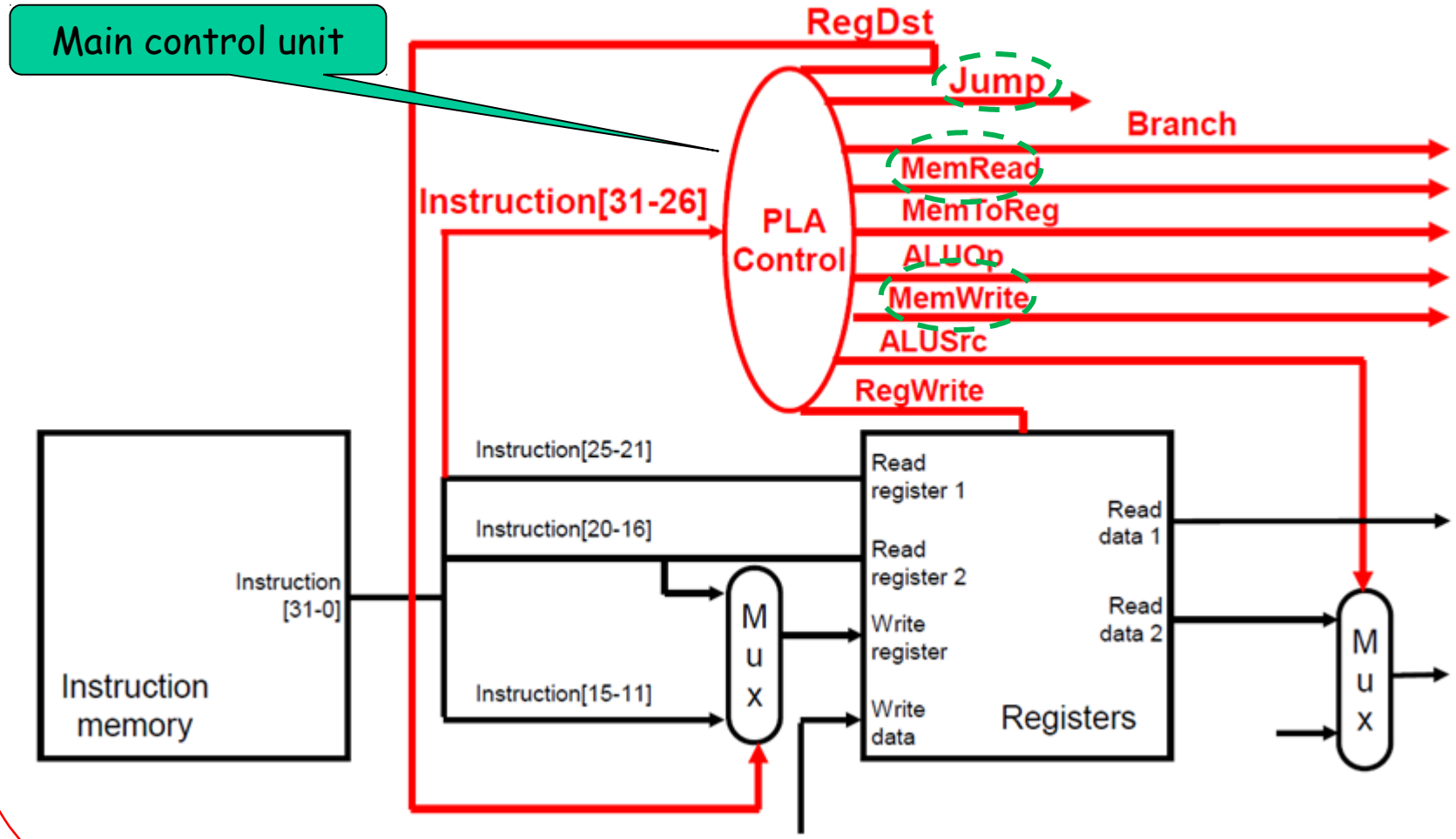


- A. Because no inputs will be provided to the respective multiplexors.
- B. Because the outputs from the respective multiplexors will make no change to the datapath

An implementation of control



Where the control logic fits in



Short summary

- Control logic
  - Inputs: the instruction opcode and funct fields
  - Outputs: control signals to multiplexors, ALU, register file, and memory
  - Can be defined using combinational logic

- Control signals are fed to
  - Multiplexors to decide which input will be passed to which output and vice versa
  - To devices to tell them what to do
    - ALU to add or sub
    - Register file to read or write
    - Memory to read or write

The ALU control

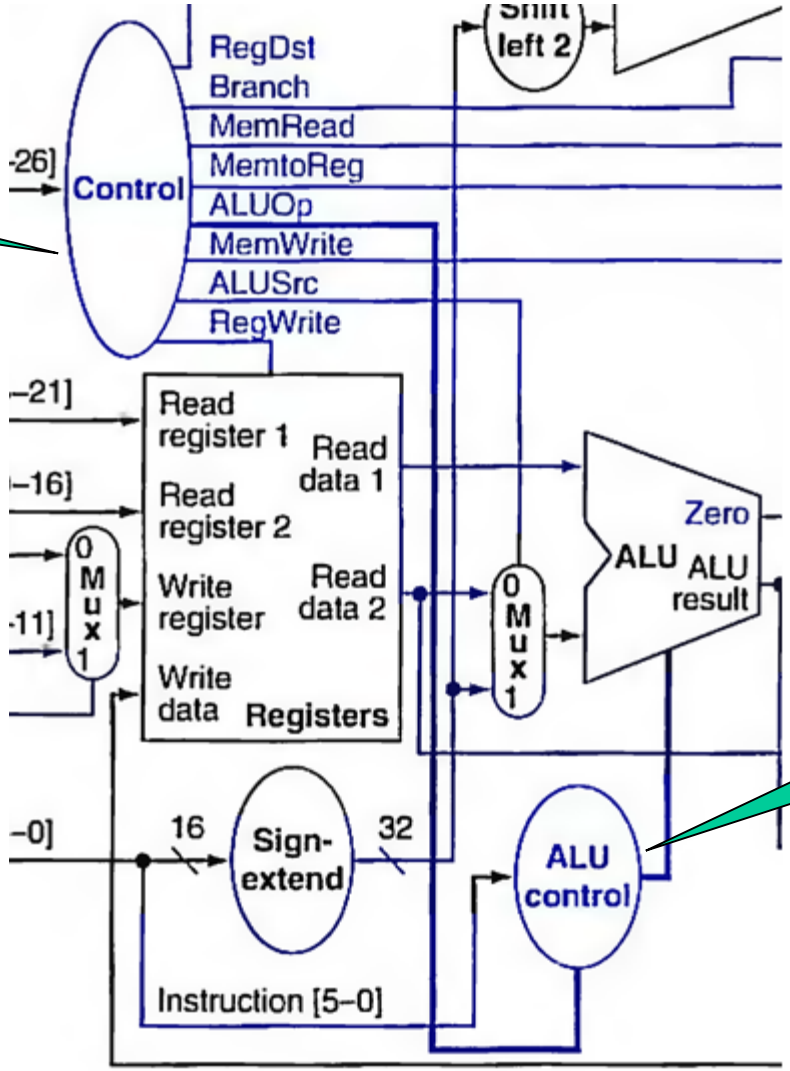
- ALUOp
  - A 2-bit control code that indicates which ALU functions to use
    - **00**: for load and stores
    - **01**: for beq
    - **10**: for R-format, use the function field of the instruction

Two-stage control signal generation

- Multiple levels of decoding
  - Main control unit generated the ALUOp bits
  - ALU control unit uses ALUOp and function field to generate the control inputs to the ALU

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

Implement the ALU control unit

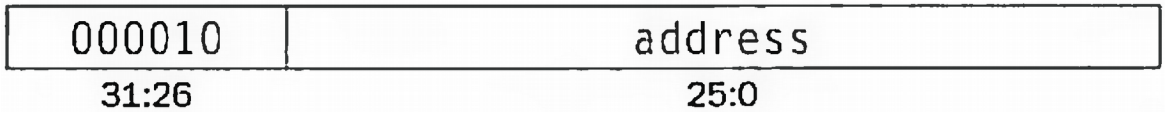


Quick exercise

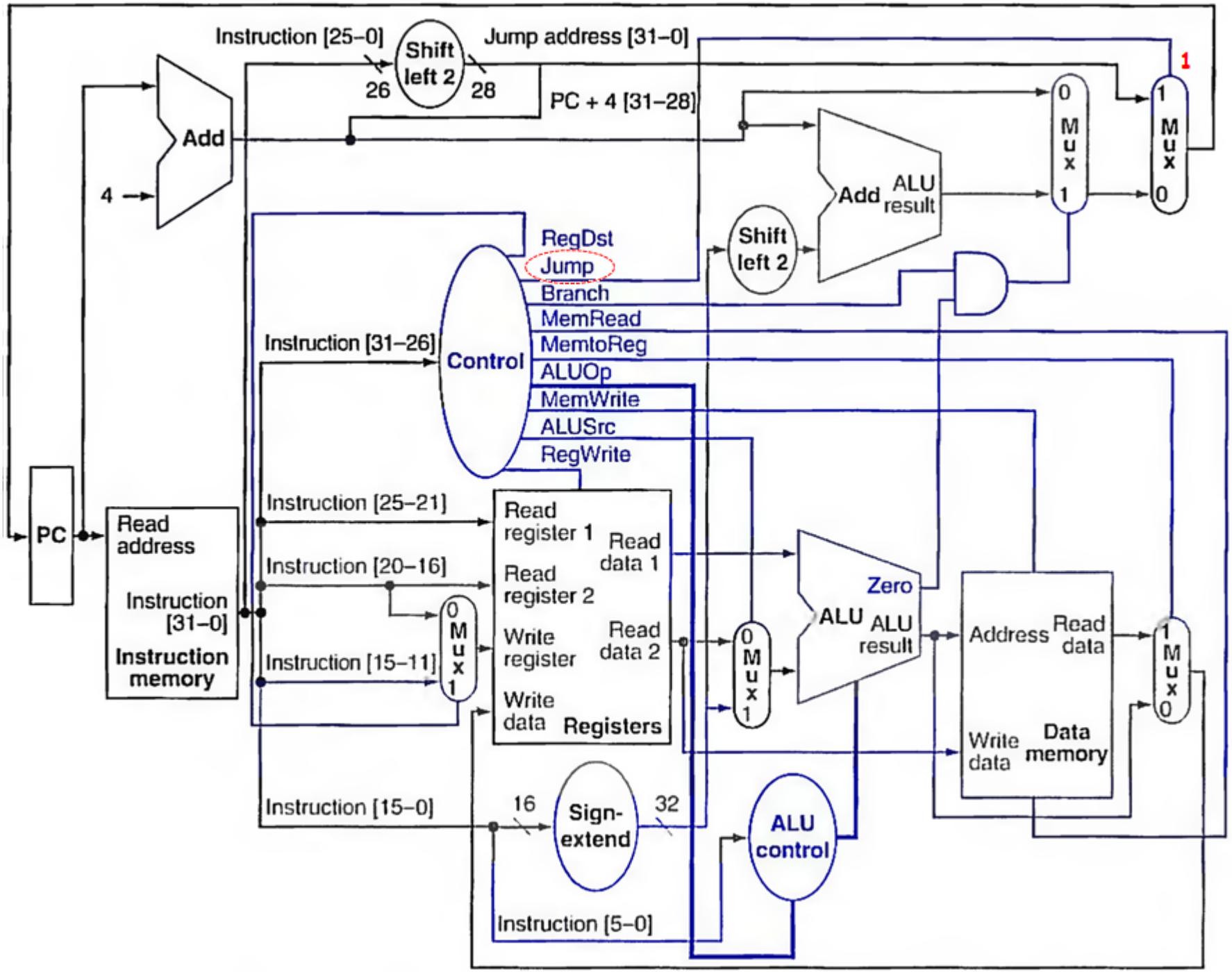
- What could be the benefits of a two-leveled decoding process for generating ALU control signals?
  - A. ALU control can be treated completely separately.
  - B. ALU control signals can be generated more quickly.
  - C. The size of the main control unit can be reduced.
  - D. The speed of the main control unit can be potentially increased.

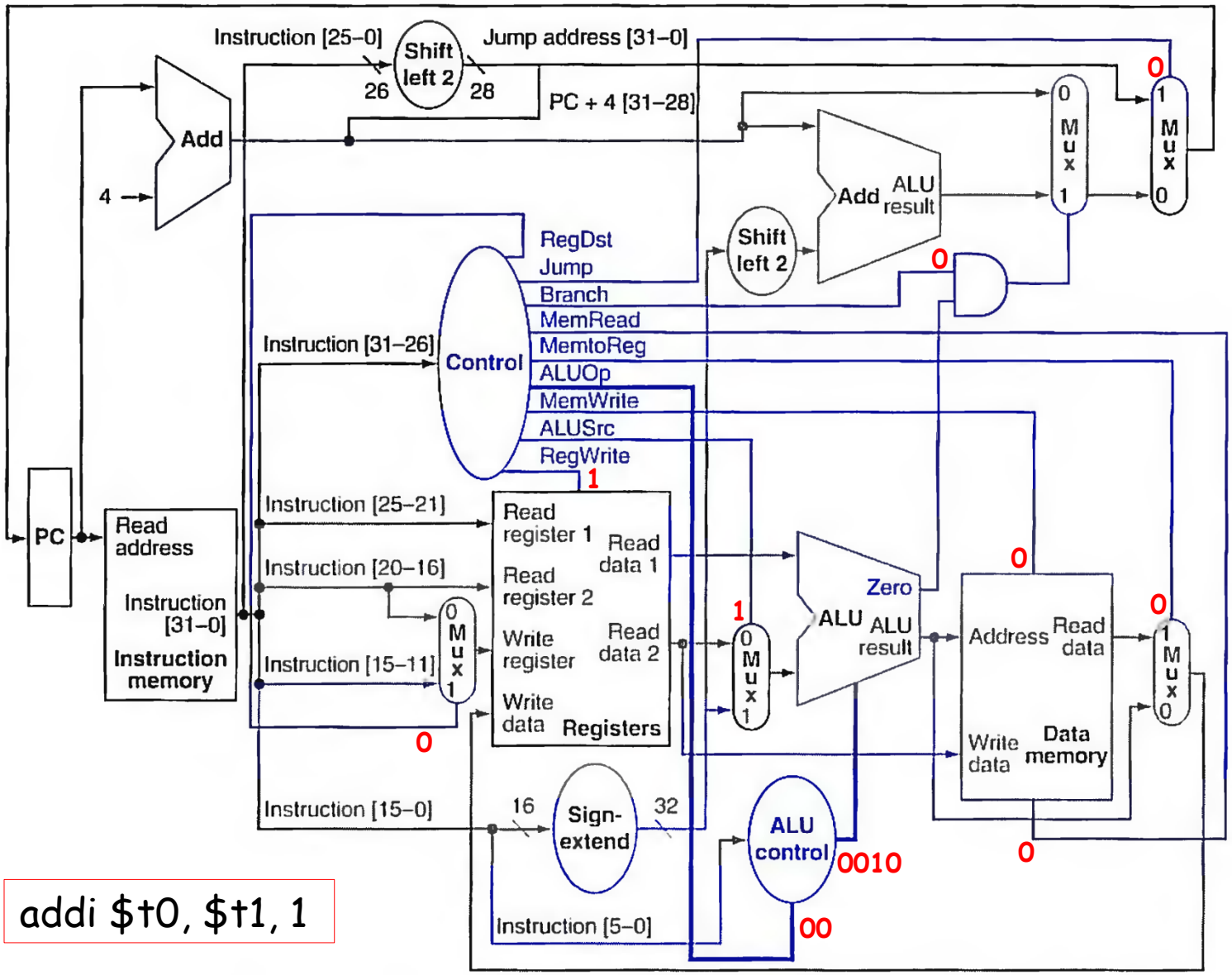
Implementing jump

- One class of instructions missing so far is that of the jump instruction.



- Formation of the jump address
  - The upper 4 bits of the current PC+4
  - The 26-bit immediate field of the jump instruction
  - The bits 00_{two}


Instruction [25–0]
Shift left 2
Jump address [31–0]
26
28
PC + 4 [31–28]
Add
4
1
0 M u x 1
1 M u x 0
Add
ALU result
Shift left 2
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
0 M u x 1
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign-extend
32
ALU control
Zero
ALU
ALU result
Address
Read data
Write data
Data memory


Instruction [25–0]
Shift left 2
Jump address [31–0]
26
28
PC + 4 [31–28]
Add
4
M u x
0
1
Add
ALU result
Shift left 2
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign-extend
32
Zero
ALU
ALU result
Address
Read data
Write data
Data memory
ALU control
0010
00
1
0

addi $t0, $t1, 1

Summary

- Single cycle datapath is easy to implement
- Control unit can be constructed by using combinational logic
  - Control multiplexors
  - Control ALUs
- Single cycle design is no longer popular because it is inefficient
  - The clock cycle is determined by the longest path in the processor