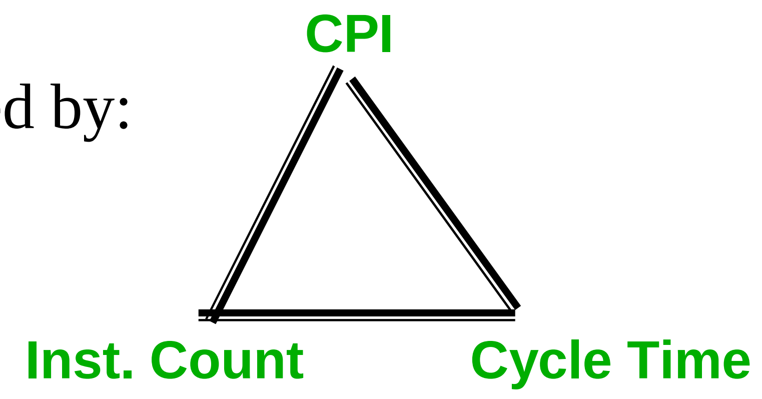
Chapter 4 The processor: Datapath & Control

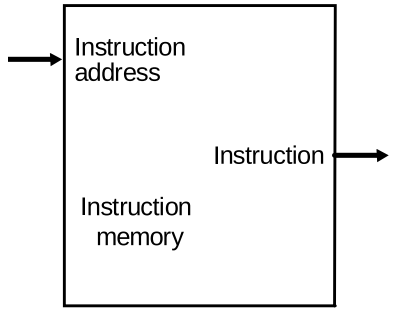
- Two implementations of MIPS instruction architecture
  - Single cycle implementation
    - Datapath & Control
  - Pipeline implementation
    - Datapath & Control
    - Data dependence/Hazard
- Consider only the core of MIPS inst. Set
  - memory-reference inst. lw, sw
  - ALU inst. add, sub, and or, slt -- R-type
  - branch inst. beq, j

The Performance Perspective

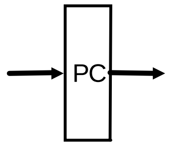


- Performance of a machine is determined by:
  - Instruction count
  - Clock cycle time
  - Clock cycles per instruction
- **Processor design (datapath and control) will affect:**
  - **Clock cycle time**
  - **Clock cycles per instruction**
- Single cycle processor:
  - A simple start to understand more complicated pipeline implementation

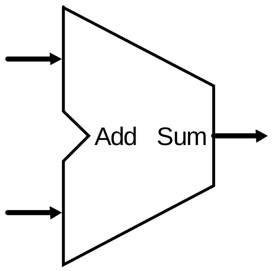
Functional Units for Implementing instructions



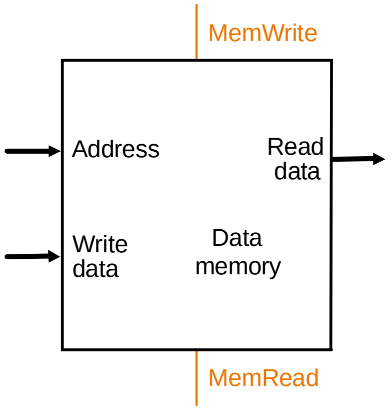
a. Instruction memory



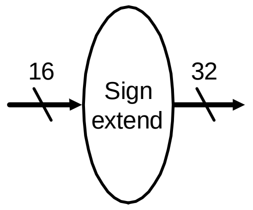
b. Program counter



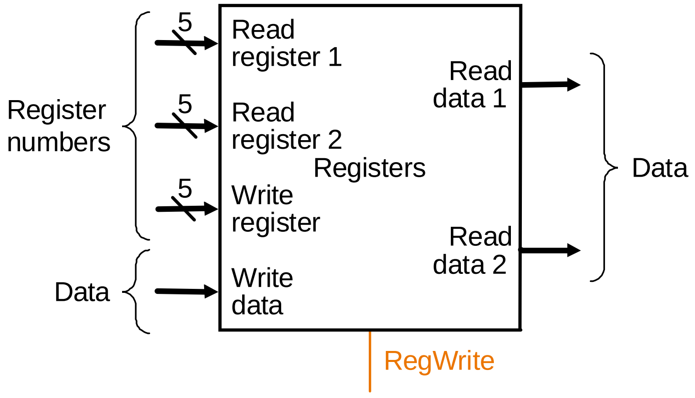
c. Adder



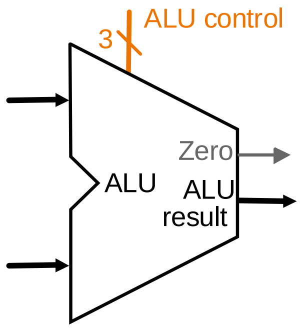
a. Data memory unit



b. Sign-extension unit



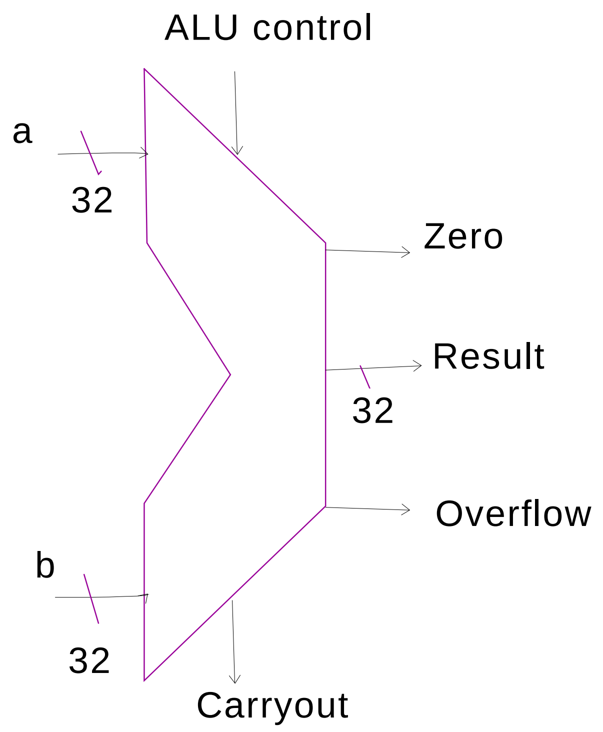
a. Registers



b. ALU

MIPS ALU Functions

Control	Funct	Result
0000	AND	aANDb
0001	OR	aORb
0010	add	a+b
0110	sub	a-b
0111	slt	1 if a<b



Zero flag:

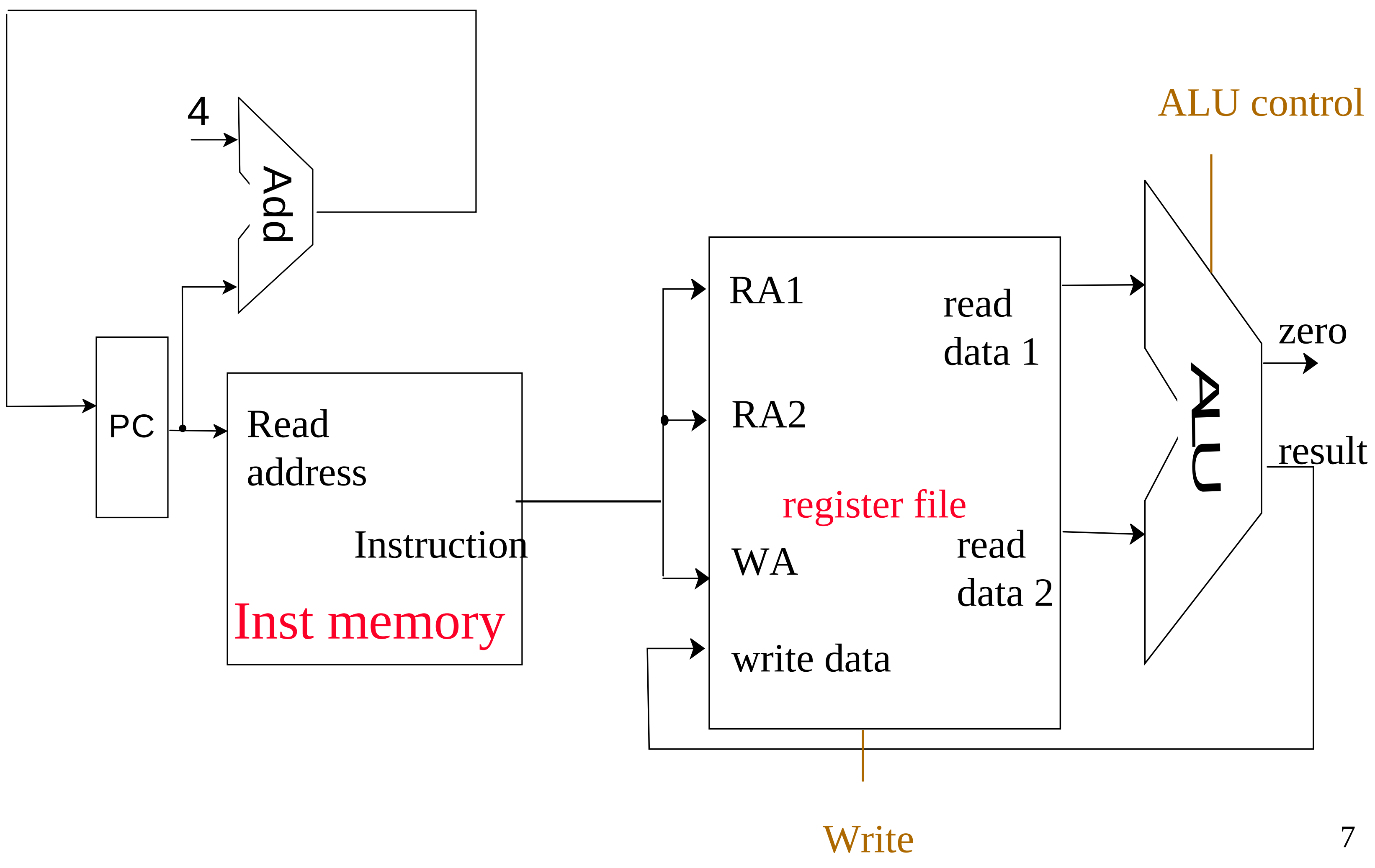
Basic Idea of Datapath Design

- Design a datapath for each type of instruction
  - R
  - lw
  - sw
    - Combine lw and sw
  - beq
- Combine all

Datapath for R-type

- Example: add $3, $4, $5
- Basic steps
  - fetch instruction
  - select registers (rs, rt),
  - ALU operation on two data, need ALU
  - write back registers

Single-Cycle: Datapath for R-type



Datapath for Load Word

- Basic steps: example: `lw $3, 300($4)`
  - fetch instruction
  - select a register (rs)
  - calculate address, need ALU
  - access memory (read memory)
  - write register file

Datapath for Store Word

- Basic steps: example: `sw $3, 300($4)`
  - fetch instruction
  - select a register (rs)
  - calculate address, need ALU
  - access memory (write memory)

Combining datapaths of `lw & sw`

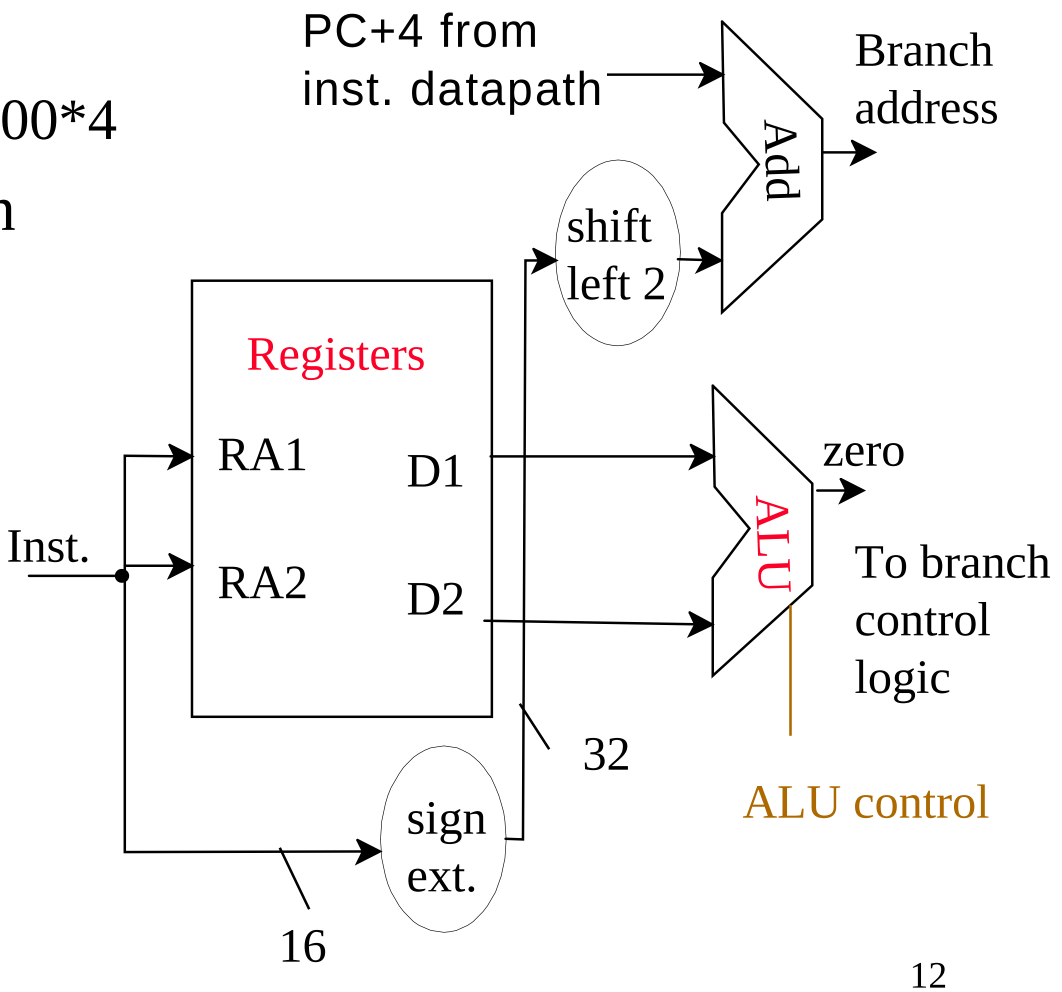
- `rt`: write address for `lw`, read address for `sw`.
- Register write control signal.

Datapath for `beq`

- Basic steps
  - fetch the instruction
  - select registers
  - test condition, calculate branch address (need additional ALU)

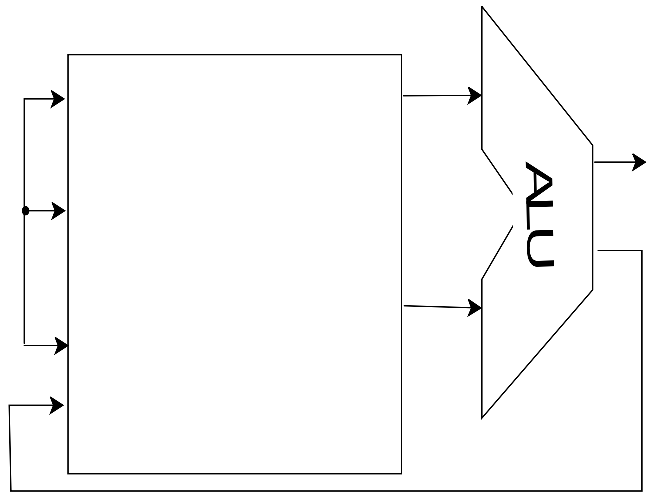
Datapath for Branch Inst.

- beq \$1, \$2, 300
  - if (\$1=\$2) goto PC+4+300*4
- ALU for branch condition
- Adder for branch address
- Shift left 2: X by 4
- Zero: control logic to decide if branch.

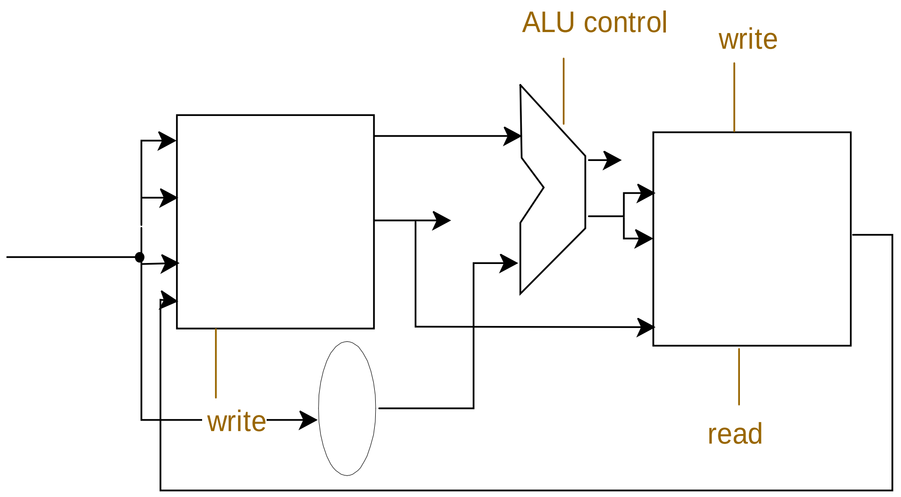


Creating a Single Datapath for All Inst.

- Combine datapaths for R-type, memory inst. and branch inst.
  - using multiplexers
  - without duplicating common functional units.
- As an example, combine R-type and memory type datapaths
- Final goal: combine all types of datapaths



R-type data path



Memory type datapath

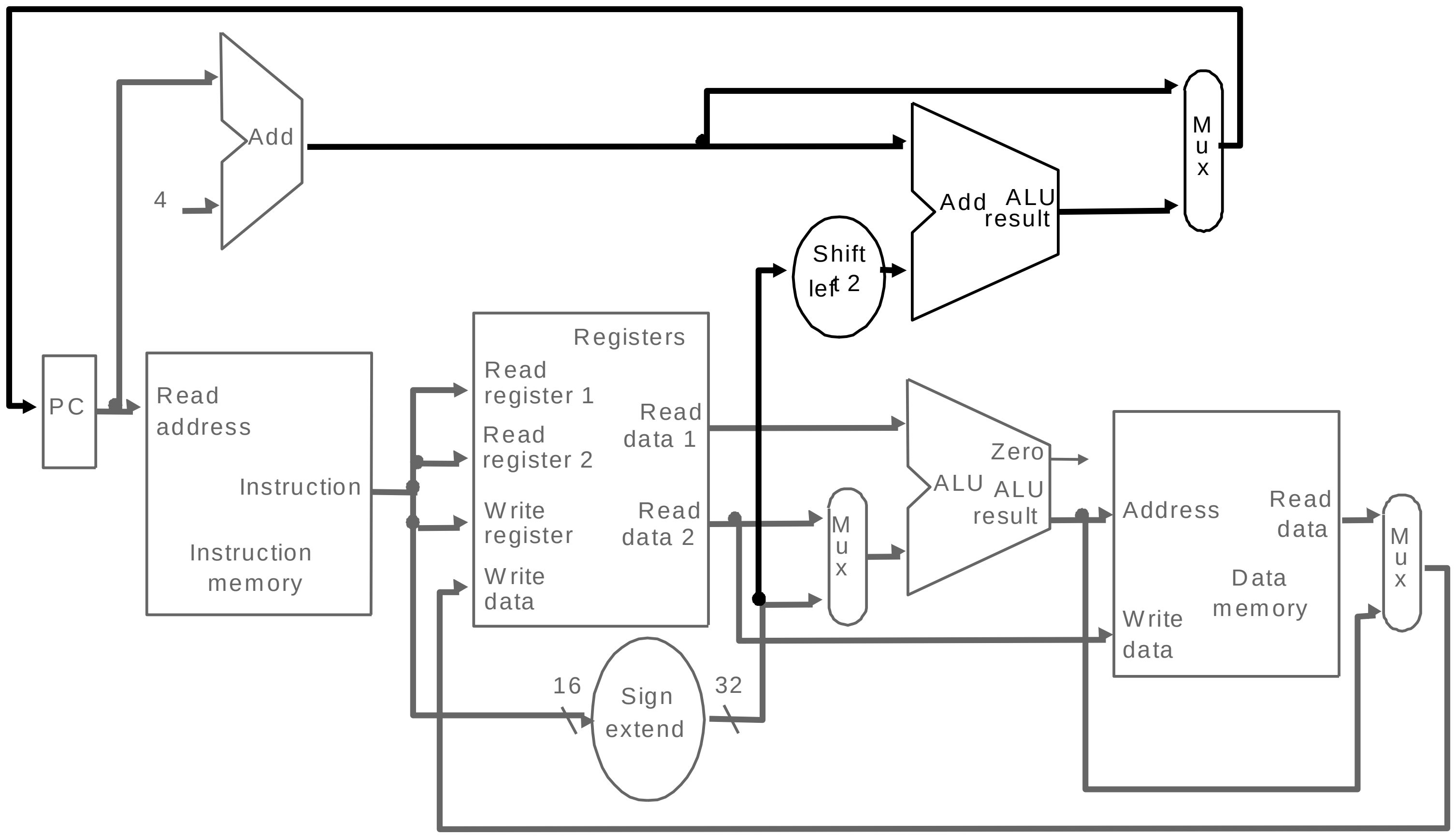
Key Differences:

2nd input to ALU

Value written into register file

Register write address

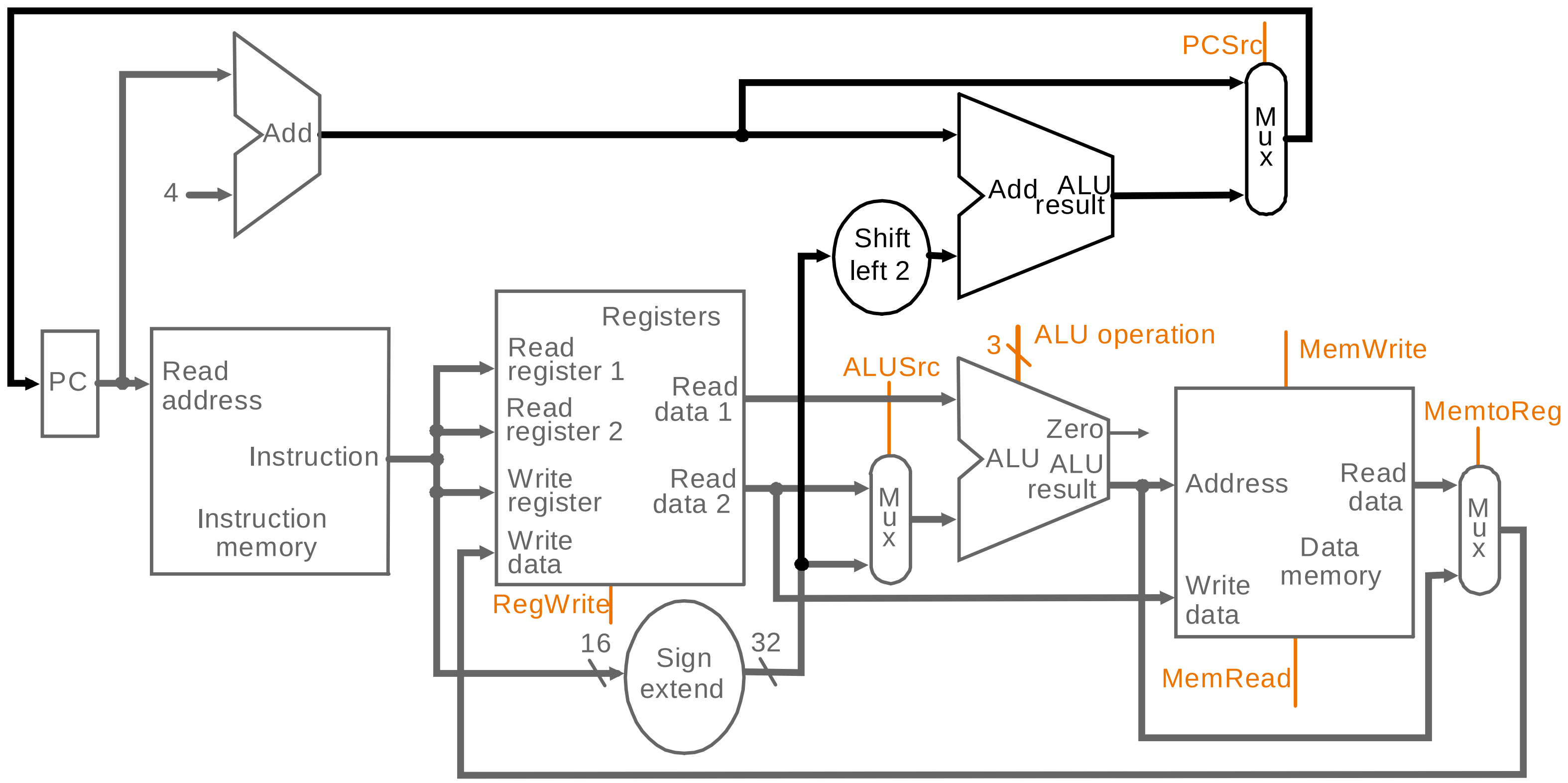
Simple Datapath for Single Cycle



Can execute basic instructions in a single clock cycle

No resource can be used more than once during a single cycle

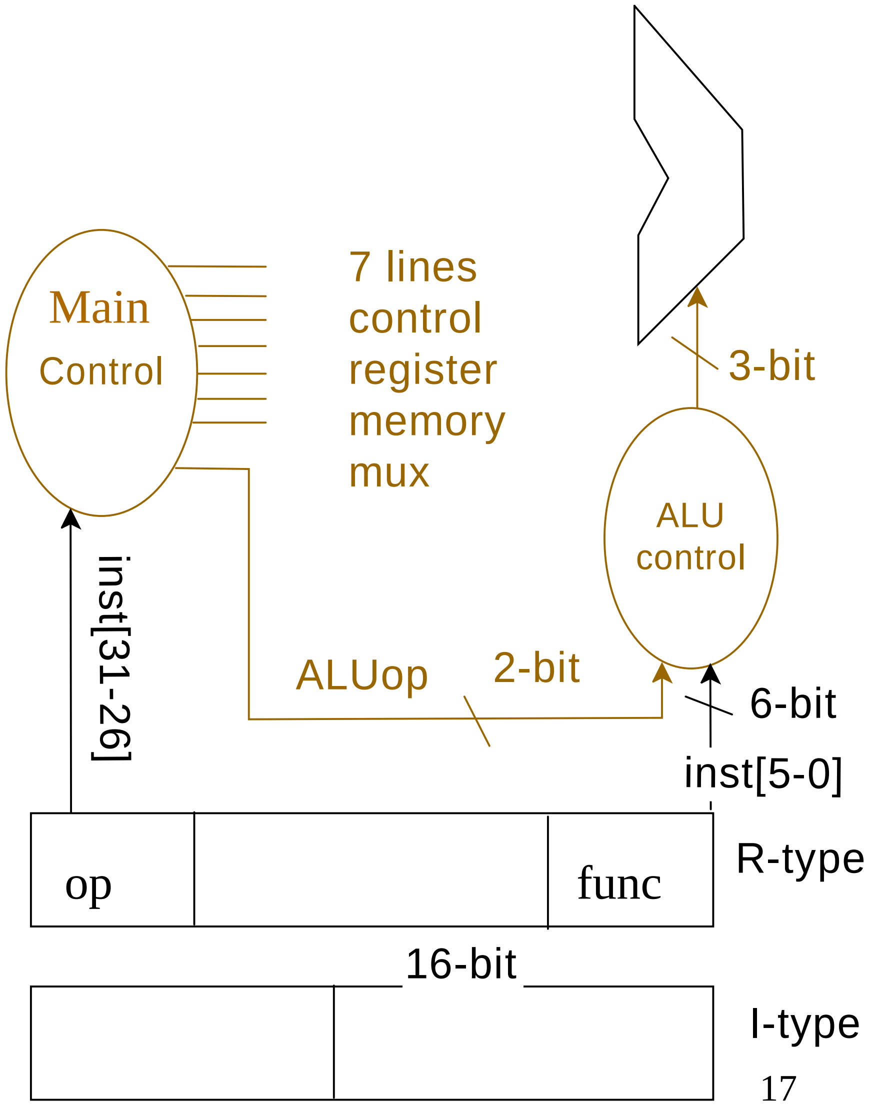
Control?



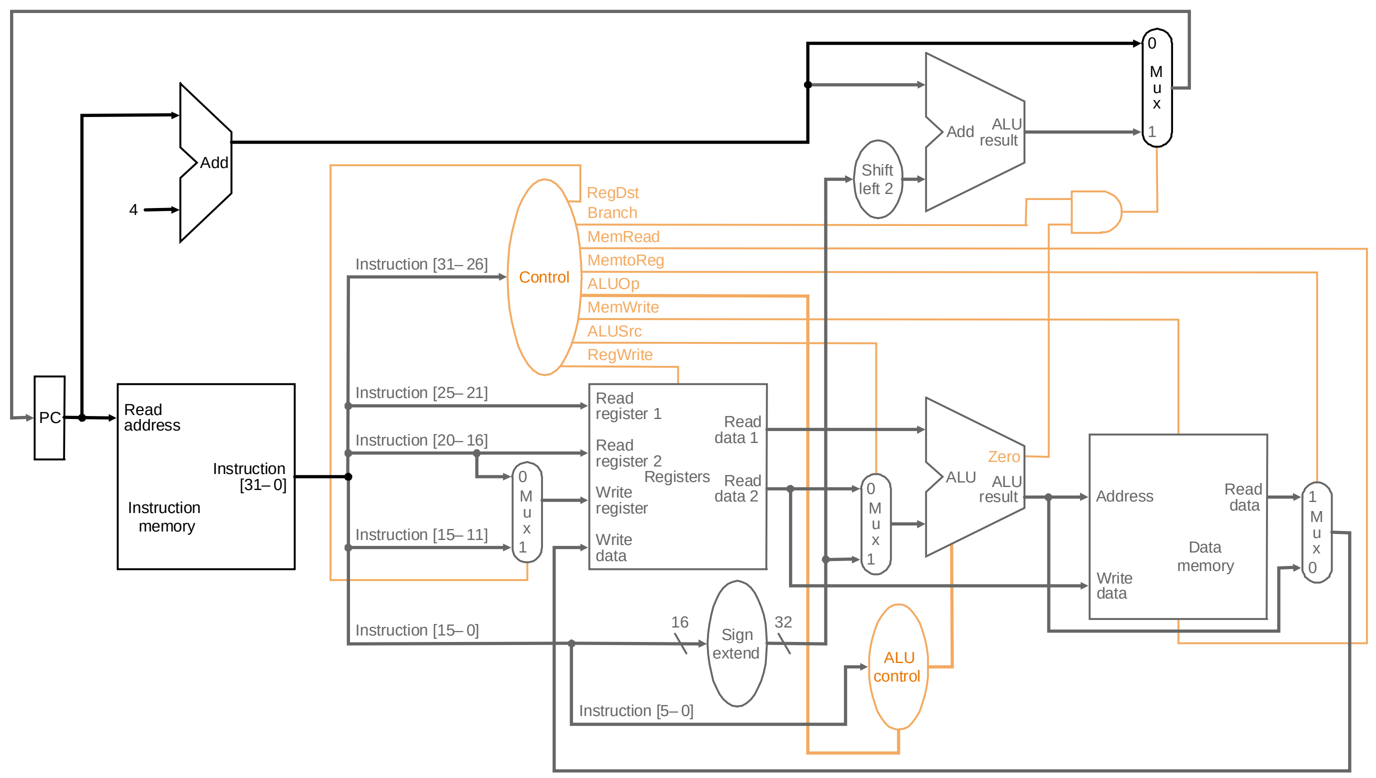
PCSrc
Add
4
M
u
x
Add
ALU
result
Shift
left 2
PC
Read
address
Instruction
Instruction
memory
Registers
Read
register 1
Read
register 2
Write
register
Write
data
Read
data 1
Read
data 2
RegWrite
16
Sign
extend
32
ALUSrc
M
u
x
3
ALU operation
Zero
ALU
ALU
result
MemWrite
Address
Read
data
Data
memory
Write
data
MemRead
MemtoReg
M
u
x

Single-Cycle: Control

- Main control:
  - input: 6-bit from op field
  - output: 9 control lines
- ALU control:
  - input: ALUop + 6-bit (function field)
  - output: 3 lines
  - for I, J type, ALU control depends on only ALUop



Control



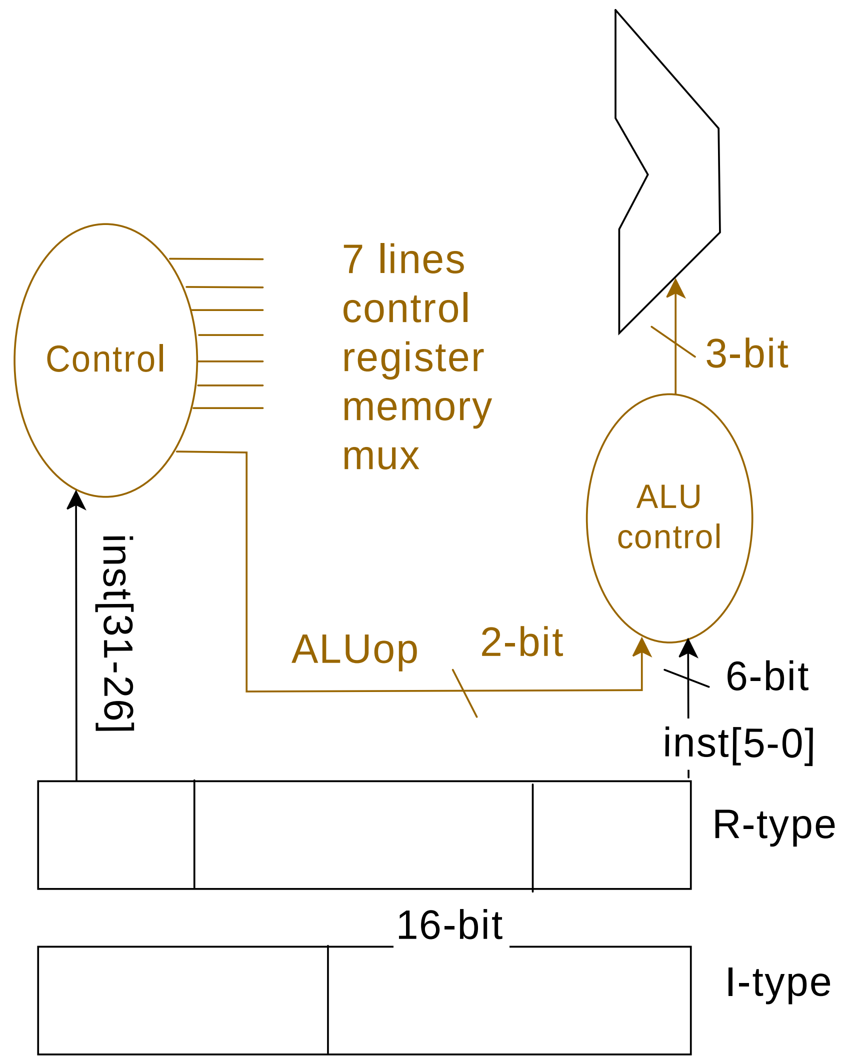
Instruction	RegDst	RegWrt	Memto-Reg	Brch	Mem Read	Mem Write	ALUSrc	ALUOp1	ALUp0
R-format									
`lw`									
`sw`									
beq									

Main Control Unit -- Definitions of Control Signals

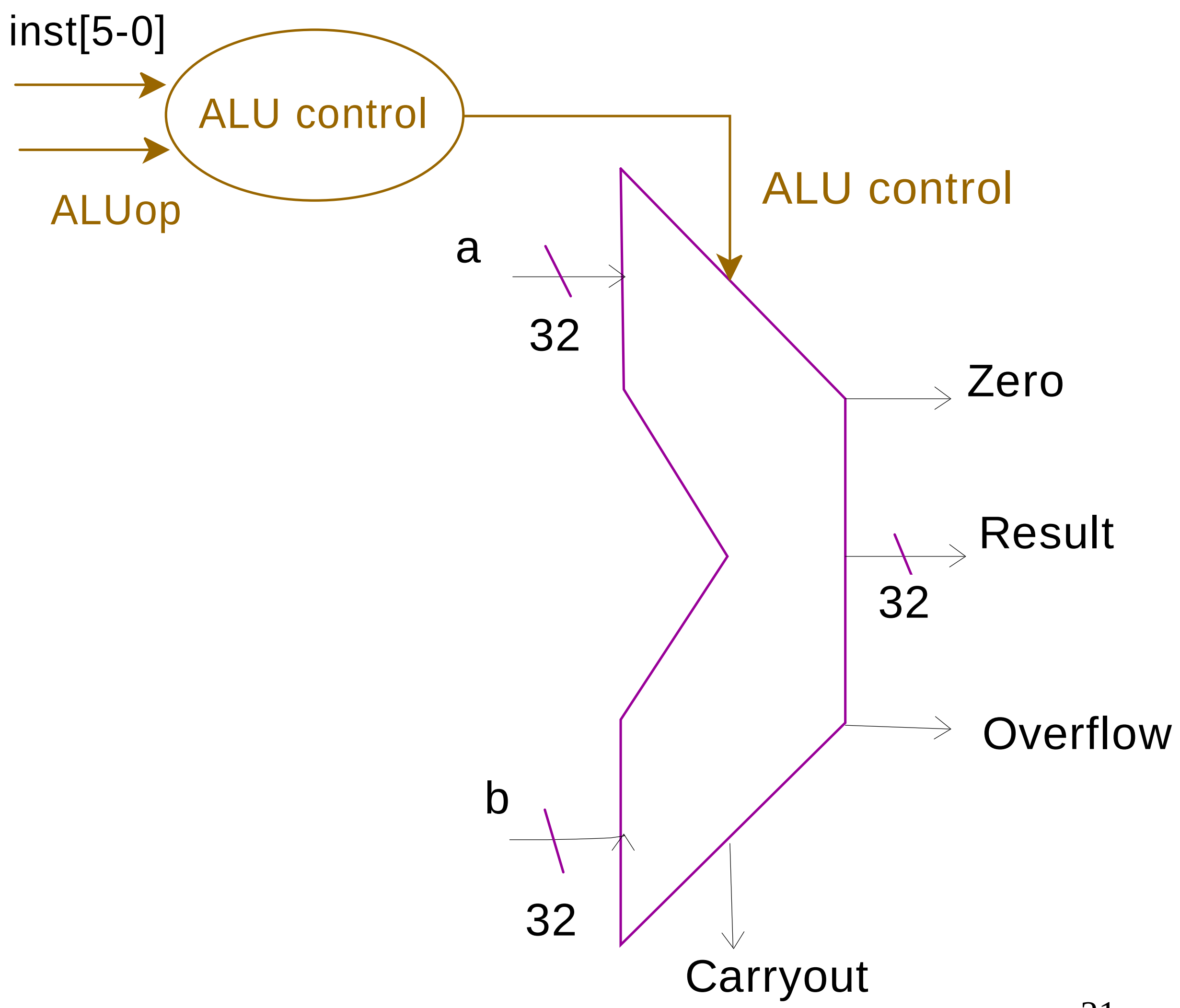
Signal Name	Effect when deasserted	Effect when asserted
MemRead	None	Data put on read dataoutput
MemWrite	None	Write data into memory
RegWrite	None	Write register
ALUSrc	2nd ALUinput from register file	2nd ALUinput from inst[15-0]
PCSrc	PC = PC+4	PC = branch address
MemtoReg	result of ALU is sent	data in memory is sent
RegDst	inst[20-16] (rt) provides register write address	inst[15-11]] (rd) provides register write address

PCsrc = branch AND zero

Single cycle control



ALU Control



ALU Control	Funct
000	AND
001	OR
010	add
110	sub
111	slt

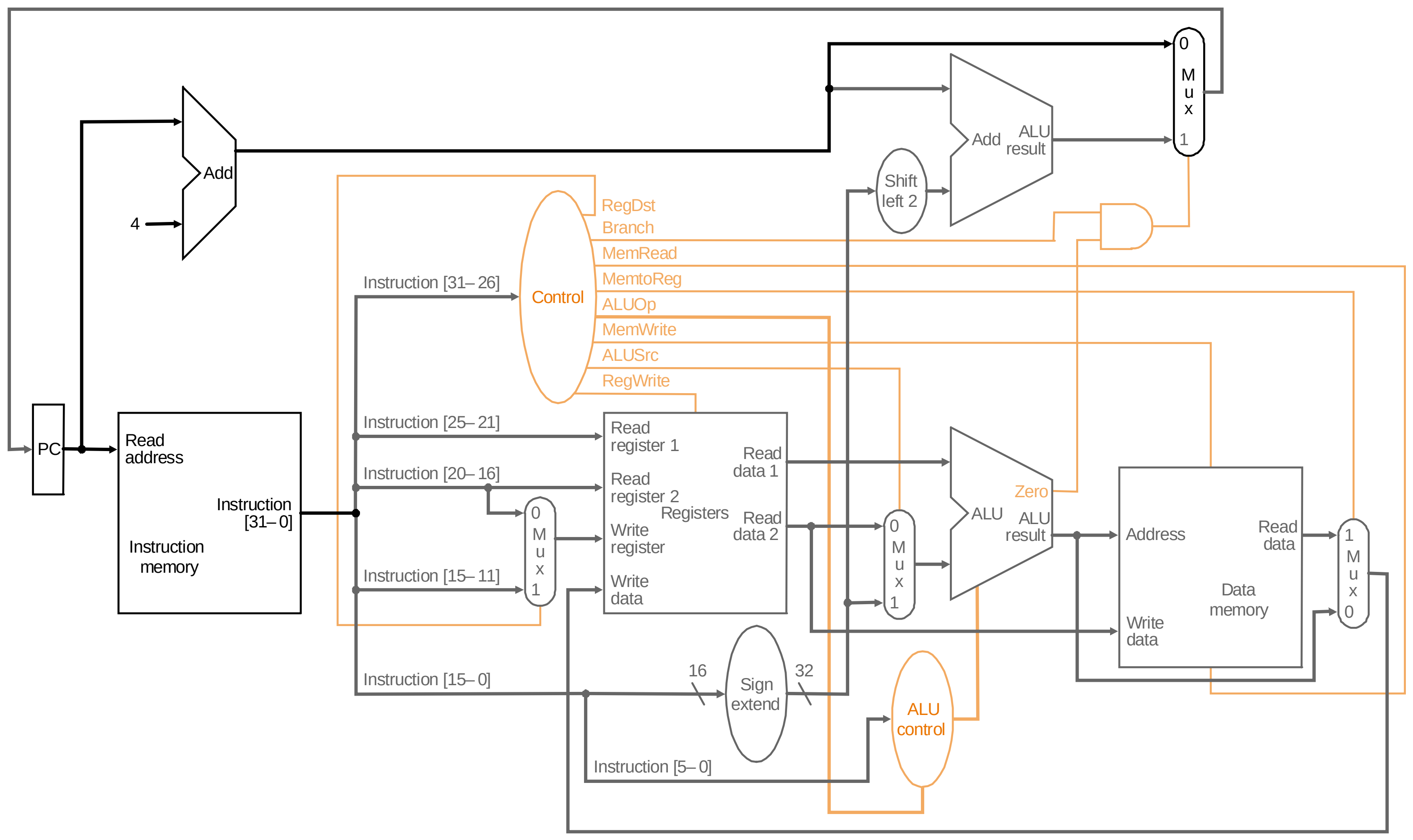
ALU Control, Truth Table

Inst opcode	Inst. operation	Desired ALU act.	ALUop	Function code	ALU control
lw	load word		00	xxxxxx	
sw	store word		00	xxxxxx	
beq	branch equal		01	xxxxxx	
R-type	add		10	10 0000	
R-type	sub		10	10 0010	
R-type	AND		10	10 0100	
R-type	OR		10	10 0101	
R-type	slt		10	10 1010	

*ALUop: output of main control
R-: ALUop=10,
lw/sw: ALUop=00

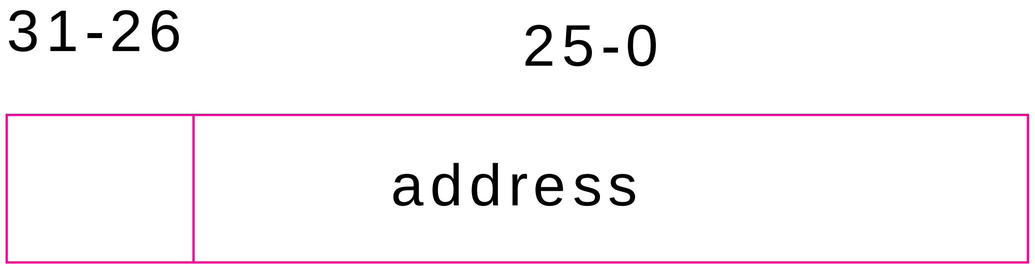
*ALU Control: combinational logic
8 inputs, 3 output.

Execution of *add $1, $1, $3* in roughly 4 steps



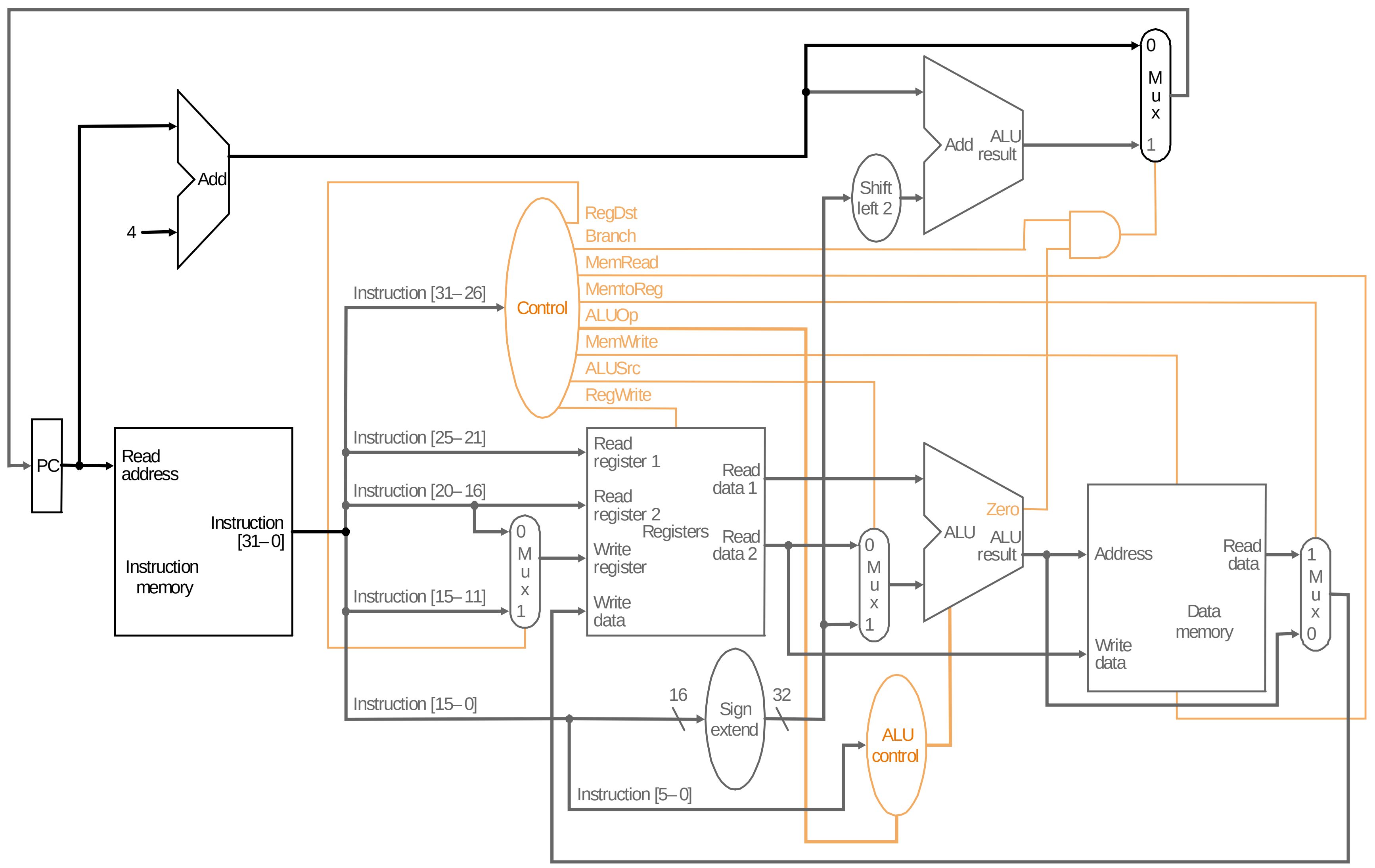
Single-Cycle: J-type

- So far, our datapath can handle R-type, lw/sw, beq
  - How about J-type?
  - J-type j L1

    jal L1
  - 

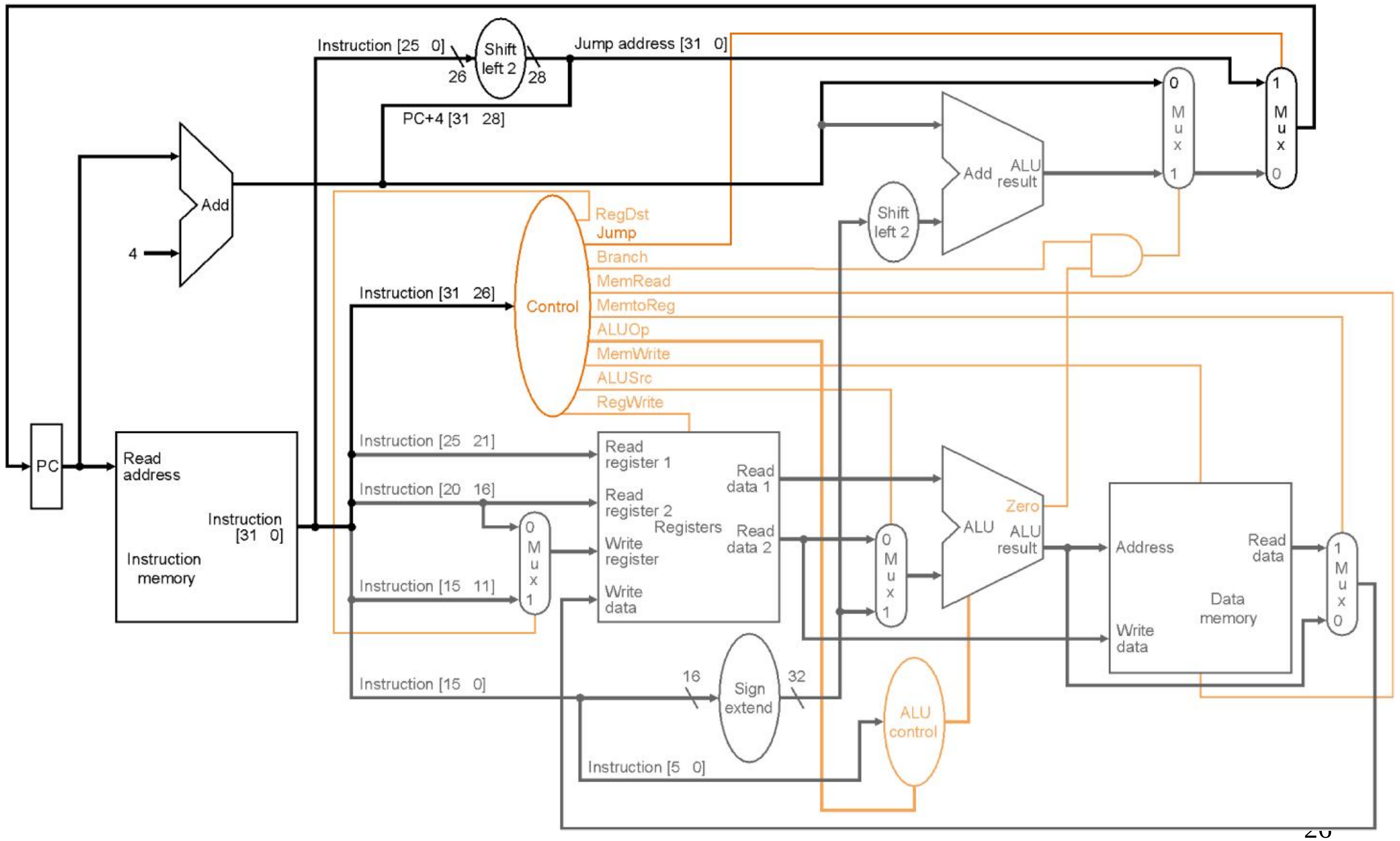
address= $a_{25} \ldots a_0$ current PC +4= $PC_{31}PC_{30} \ldots PC_0$

Actual address L1 = $pc_{31}\,pc_{30}\,pc_{29}pc_{28}\,a_{25} \ldots a_0\,00$



PC
Add
4
Read address
Instruction [31–0]
Instruction memory
Instruction [31–26]
Control
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
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
Registers
Read data 1
Read data 2
16
Sign extend
32
Shift left 2
Add
ALU result
0 M u x 1
ALU
Zero
ALU result
ALU control
Address
Read data
Write data
Data memory
1 M u x 0

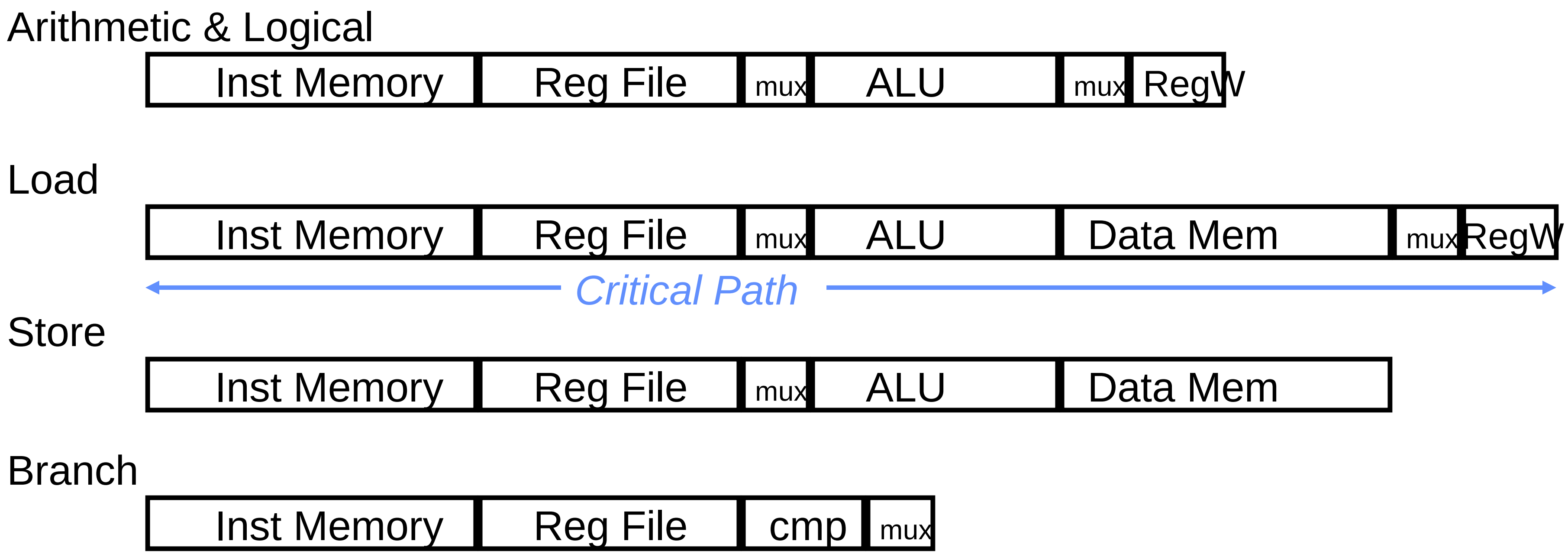
Single-Cycle: Datapath + Control including jump inst



Single-Cycle: Summary

- How to construct datapath for R-type, lw/sw, beq, j
- How to construct the control
  - ALU control
  - main control
- How the datapath + control to execute instruction

Execution (cycle) Time Analysis



- Cycle time >= the critical path length
- Long Cycle Time
  - All instructions take as much time as the slowest

Exercise: Can MemtoReg be eliminated and replaced by MemRead?

	RgDst	Jp	Brnch	MemRd	MemtoRg	ALUop	MemWrt	ALUSrc	RegWrt
lw	0	0	0	1	1	00	0	1	1
sw	x	0	0	0	x	00	1	1	0
R	1	0	0	0	0	10	0	0	1
beg	x	0	1	0	x	01	0	0	0
Jp	x	1	0	0	x	xx	0	x	0
			B						

Exercise: What would be the cycle time for the following cases?

- Assume
  - ALU delay=2ns,
  - Adder (PC+4) delay=x ns,
  - Adder (branch address) delay = y ns
- X=3, y=3
- x=5, y=5
- x=1, y=8