

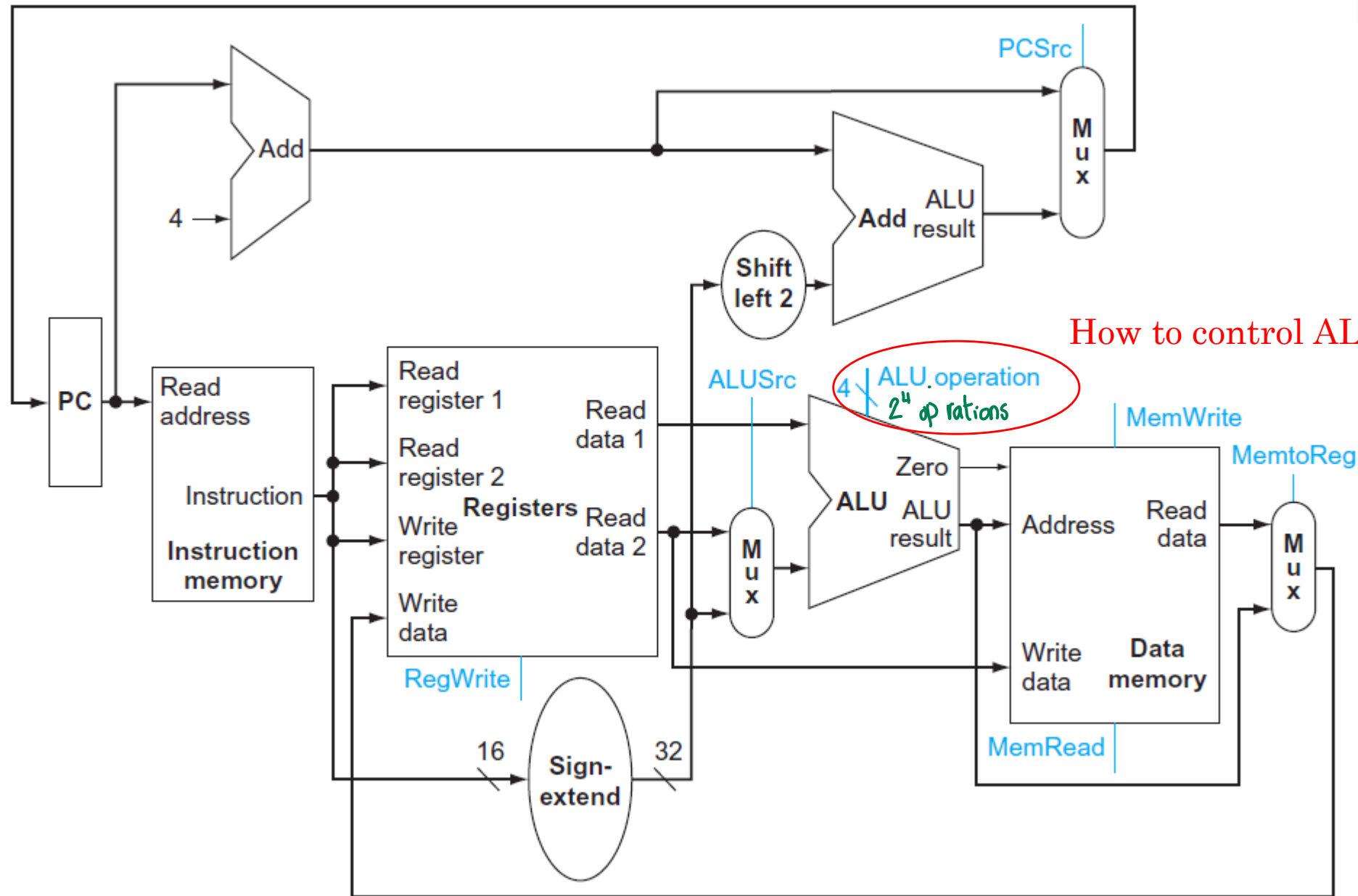
Computer System Architecture - DataPath

H. SOUBRA (c)2023

Disclaimer

This course contains copyrighted material the use of which has not always been specifically authorized by the copyright owner. They are used strictly for educational purposes. The principle of fair use applies.

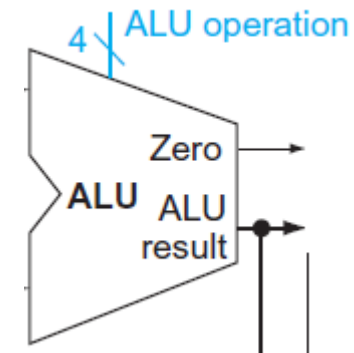
R-format + loads and stores + Branch



ALU operations

- What are the different operations that should be done in the ALU?
 1. AND
 2. OR
 3. Add
 4. subtract
 5. set on less than
 6. NOR

How many bits at least to select an operation?



ALU operations

- What are the different operations that should be done in the ALU?

1. AND
2. OR
3. Add
4. subtract
5. set on less than
6. NOR

Operation INPUT	ALU OPERATION
0000	AND
0001	OR
0010	Add
0110	subtract
0111	set on less than
1100	NOR

How to generate the operation input?



ALU in MIPS=combinational circuit that calculates a 32-bit output based on two 32-bit inputs and a 4-bit input specifying the ALU operation to perform. The ALU also computes three flag bits, **C (Carry)**, **V(Overflow)**, and Z(?)

ALU Control Unit

- What kind of **operation** is needed for **LW/SW**?

Add dd offset to the base

- What kind of **operation** is needed for **Branch equal**?

subtract (if result = 0 then they're equal)

- What kind of **operation** is needed for **R-type**?

AND, OR, Add, subtract, set on less than, NOR

Depending on the **funct** field

- What is the opcode for R-Type?

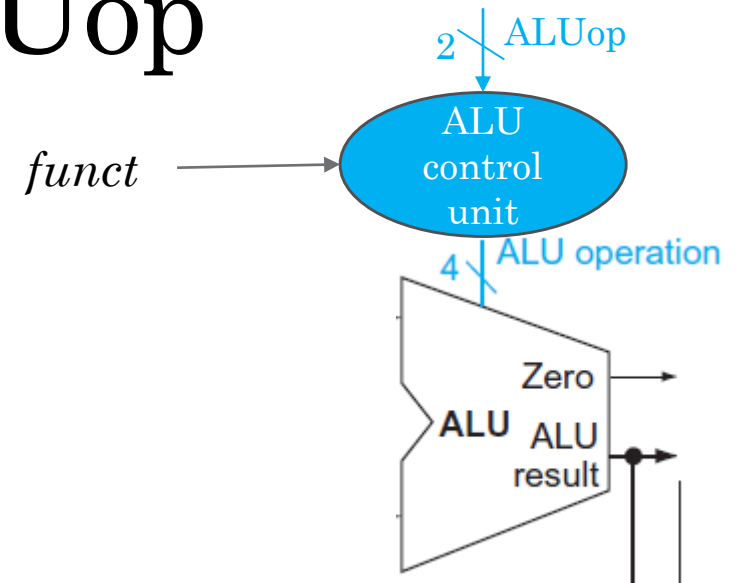
000000

2-bit control unit input

Instruction	Funct
add	rd, rs, rt
addu	rd, rs, rt
and	rd, rs, rt
break	
div	rs, rt
divu	rs, rt
jalr	rd, rs
jr	rs
mfhi	rd
mflo	rd
mthi	rs
mtlo	rs
mult	rs, rt
multu	rs, rt
nor	rd, rs, rt
or	rd, rs, rt
sll	rd, rt, sa
sllv	rd, rt, rs
slt	rd, rs, rt
sltu	rd, rs, rt
sra	rd, rt, sa
srav	rd, rt, rs
srl	rd, rt, sa
srlv	rd, rt, rs
sub	rd, rs, rt
subu	rd, rs, rt
syscall	
xor	rd, rs, rt

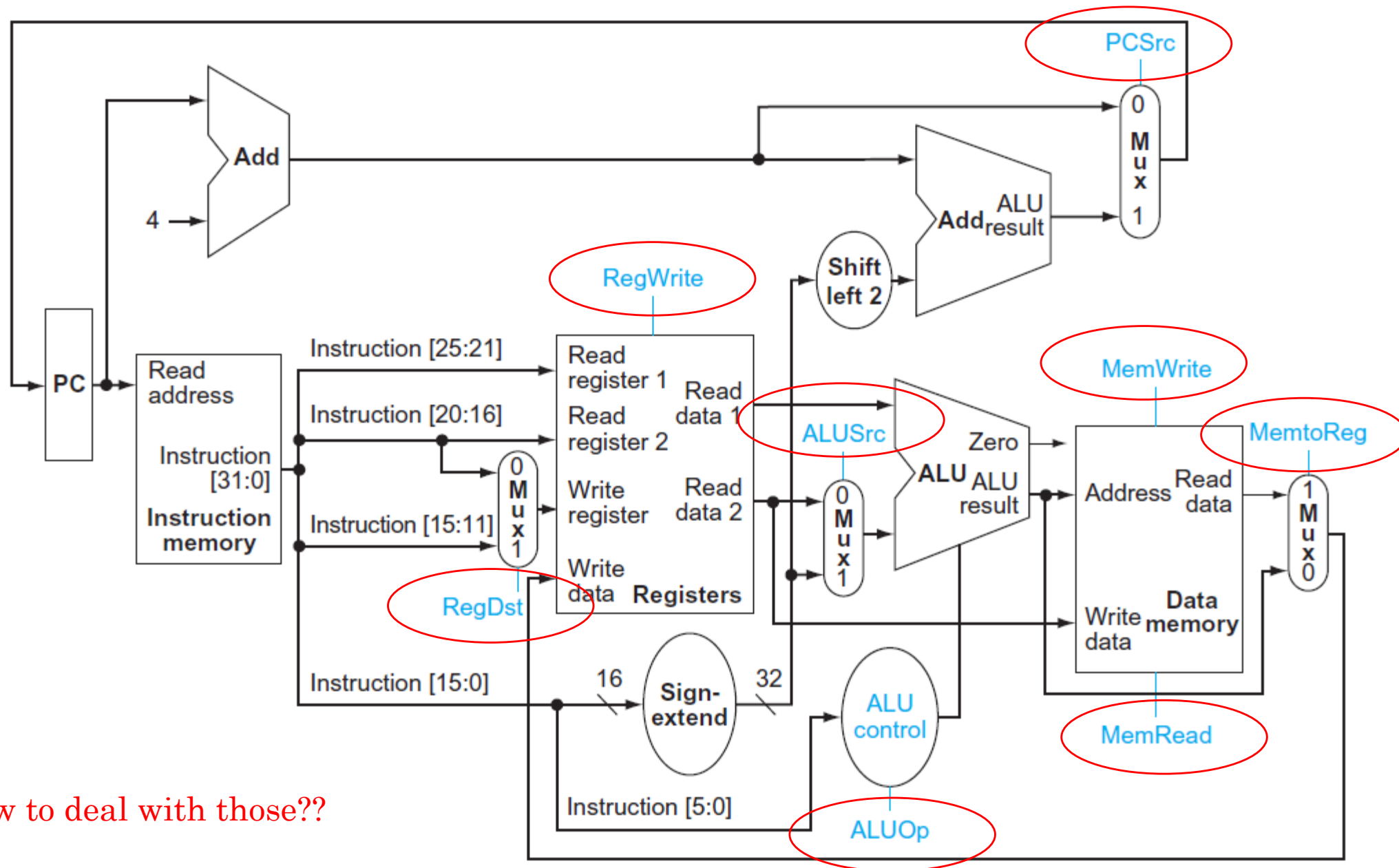
ALU Control Input: ALUOp

Instruction	ALUOp 2-bit control	ALU operation	ALU operation 4-bit Input
LW/SW	00	ADD	0010
BEQ	01	SUB	0110
R-TYPE	10	<i>funct</i>	<i>funct</i>



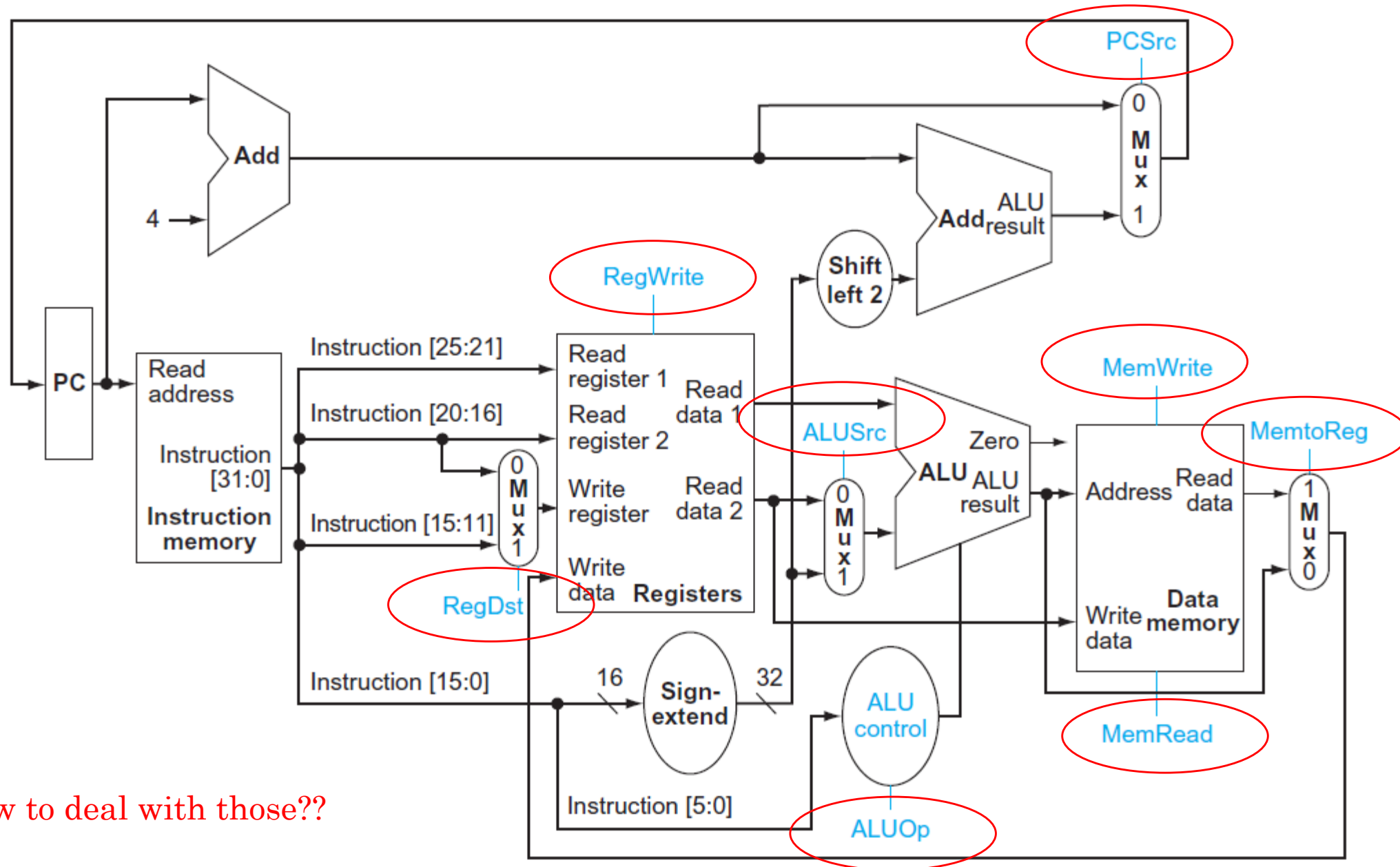
<i>funct</i>	ALU operation	ALU operation 4-bit Input
100000	ADD	0010
100010	SUB	0110
100100	AND	0000
100101	OR	0001
101010	set-on-less-than	0111

...+ ALU control



How to deal with those??

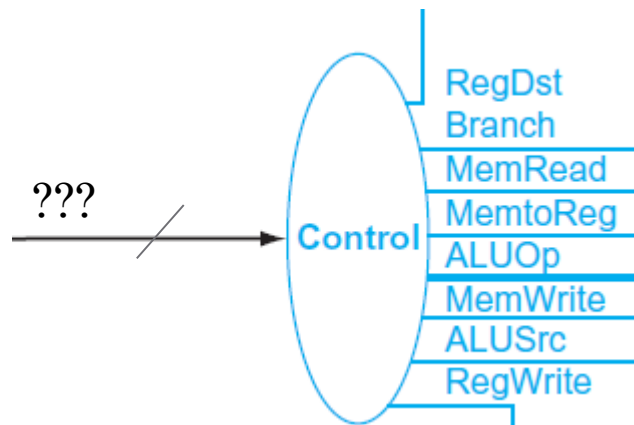
Simple Hardware Implementation



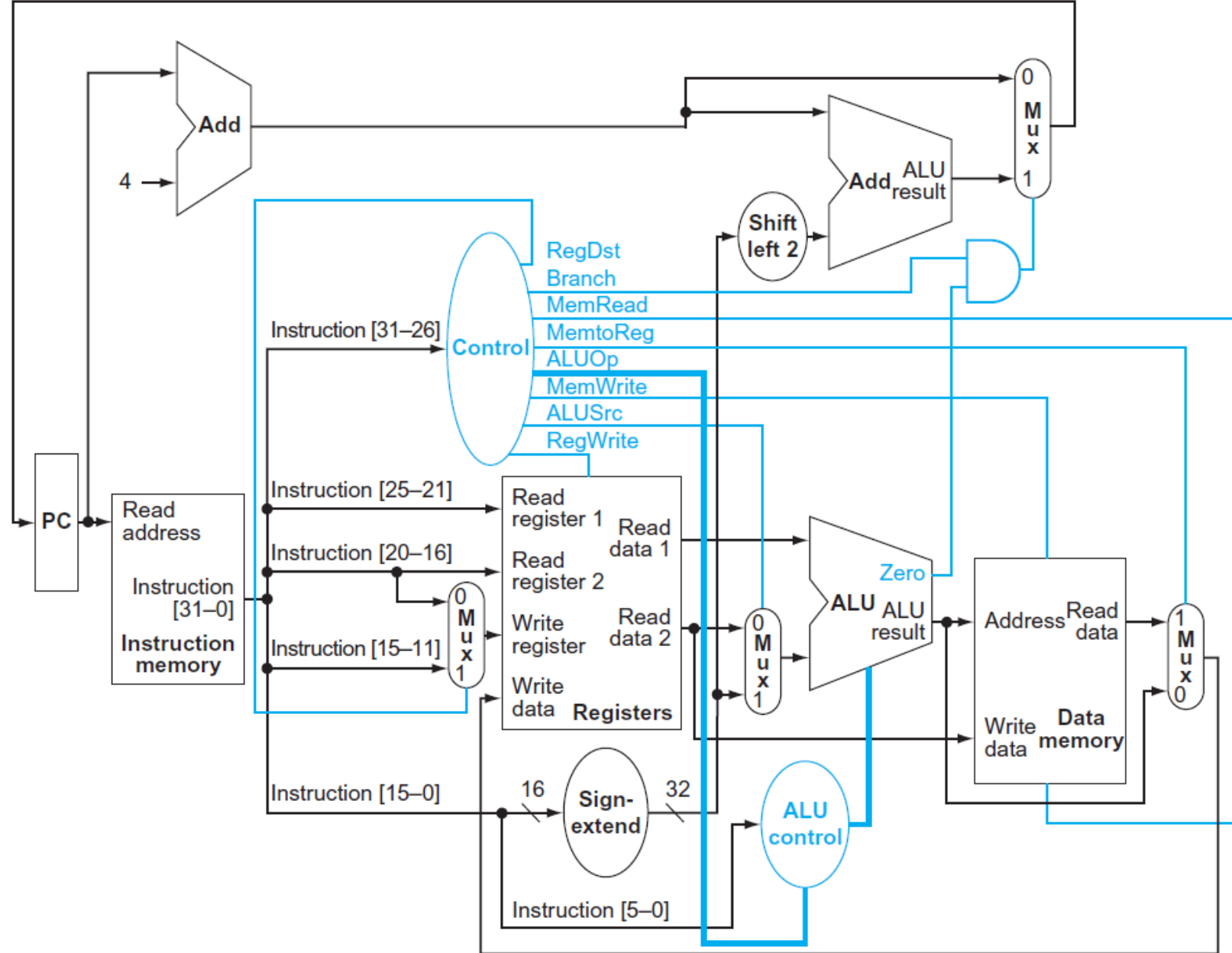
How to deal with those??

Control Signal Table

Instruction	ALUOp	RegDst	ALUSrc	RegWrite	MemRead	MemWrite	Branch	MemtoReg
LW	00	0	1	1	1	0	0	1
SW	00	X	1	0	0	1	0	X
BEQ	01	X	0	0	0	0	1	X
R-TYPE	10	1	0	1	0	0	0	0




Full Simple Hardware Implementation



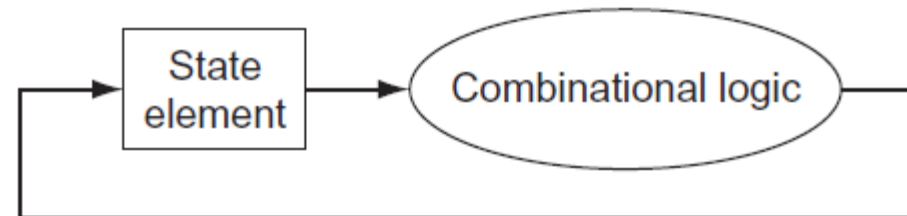
Control function for the simple implementation

OPCODE	Instruction	ALUOp	RegDst	ALUSrc	RegWrite	MemRead	MemWrite	Branch	MemtoReg
100011	LW	00	0	1	1	1	0	0	1
101011	SW	00	X	1	0	0	1	0	X
000100	BEQ	01	X	0	0	0	0	1	X
000000	R-TYPE	10	1	0	1	0	0	0	0

Instruction cycles: Simple implt

- How many cycles to execute an instruction? Why?
- Edge triggered 
- Memory for instructions separate from one for data! *not von-neuman or some cache approach idk*
- And any other element needed more than once must be duplicated

memory has a cache, data cache is separate from instruction cache. from an implementation pov, I'm seeing the cache. so I have the instruction cache as a representative of the instr. memory and the data cache representing the data memory



Performance of the Single-Cycle Implementation

$$\text{Execution time (in Seconds/Program)} = \frac{\text{Instructions}}{\text{Program}} * \frac{\text{Clock cycles}}{\text{Instruction}} * \frac{\text{Seconds}}{\text{Clock cycle}}$$

1

How big should it be?

Performance of the Single-Cycle Implementation

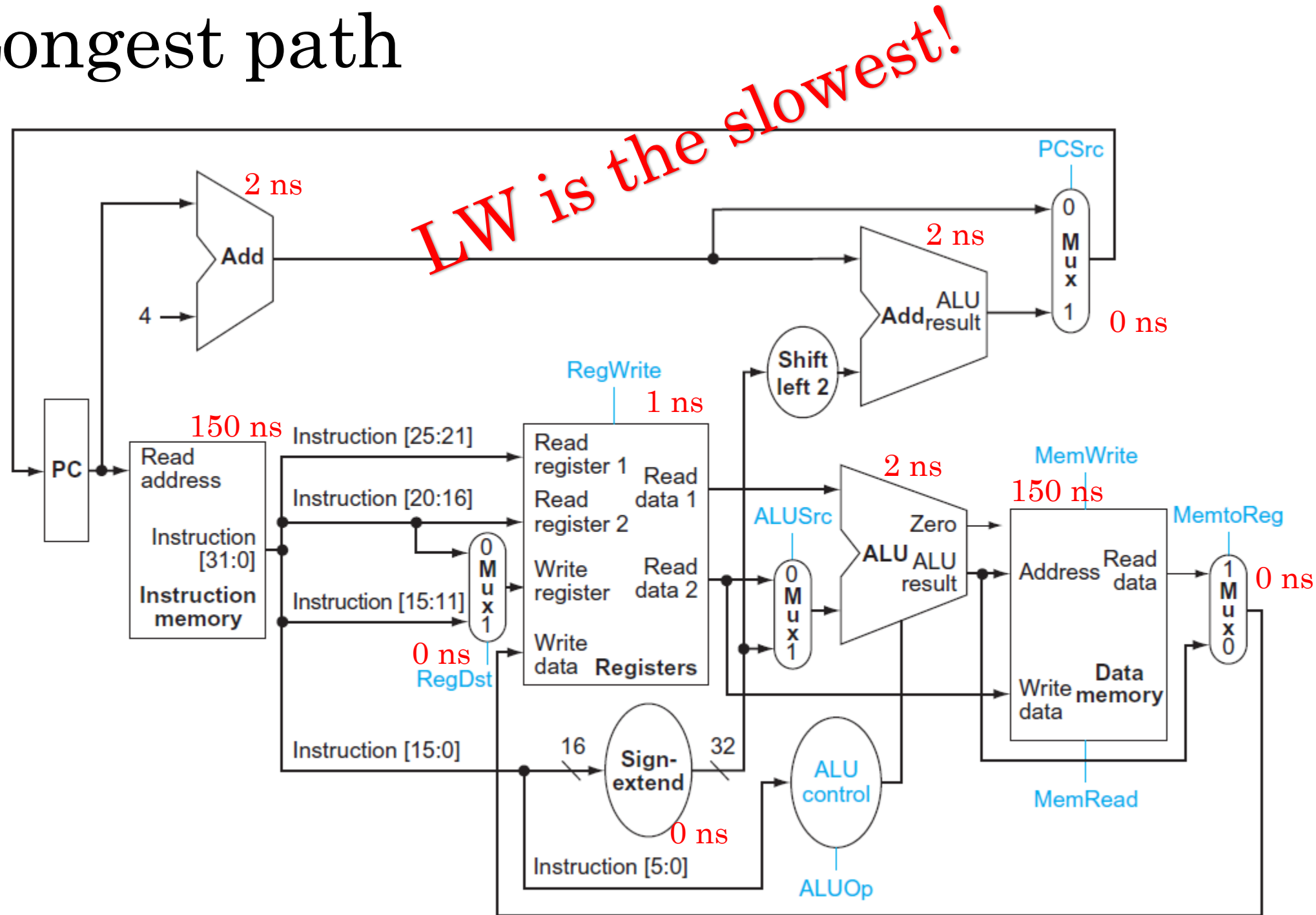
- Single-cycle design will work correctly but inefficiently
- Clock cycle must have the same length for every instruction
- The longest possible path in the processor determines the clock cycle
- Which instruction is the slowest?

$$\text{Execution time (in Seconds/Program)} = \frac{\text{Instructions}}{\text{Program}} * \frac{\text{Clock cycles}}{\text{Instruction}} * \frac{\text{Seconds}}{\text{Clock cycle}}$$

1

How big should it be?

Longest path



Performance of the Single-Cycle Implementation

- Lw is the slowest:

1. reading the instruction memory: 150ns
2. reading the base register: 1ns
3. computing memory address: 2ns
4. reading the data memory: 150ns
5. storing data back in a register: 1ns

So it takes 304 ns

So the clock should be $1/304 \times 10^{-9} = 3 \text{ MHz}$

$$\text{Execution time (in Seconds/Program)} = \frac{\text{Instructions}}{\text{Program}} * \frac{\text{Clock cycles}}{\text{Instruction}} * \frac{\text{Seconds}}{\text{Clock cycle}}$$

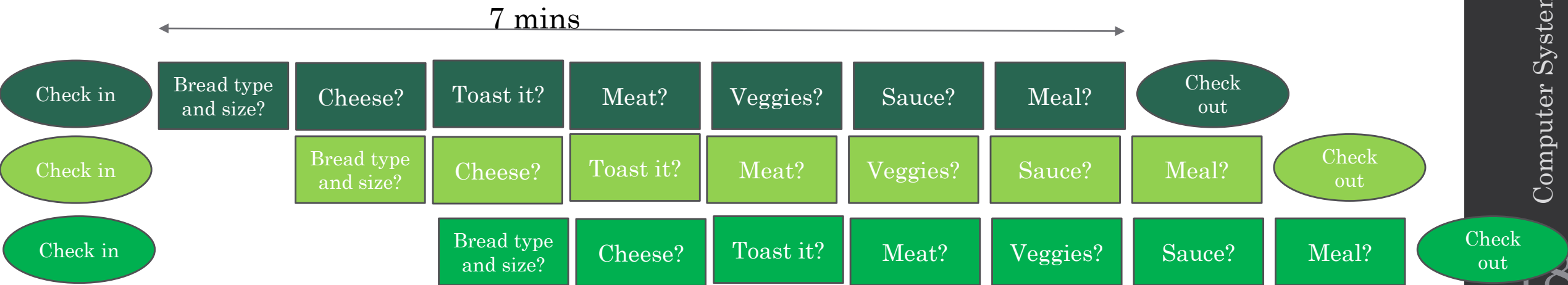
1

How big should it be?



What to do?

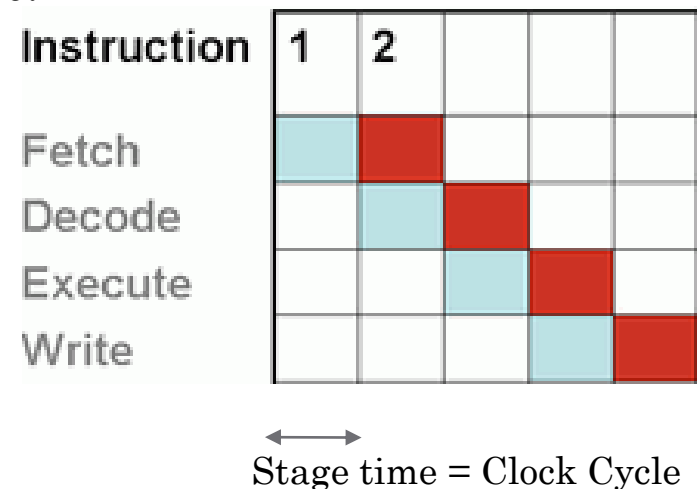
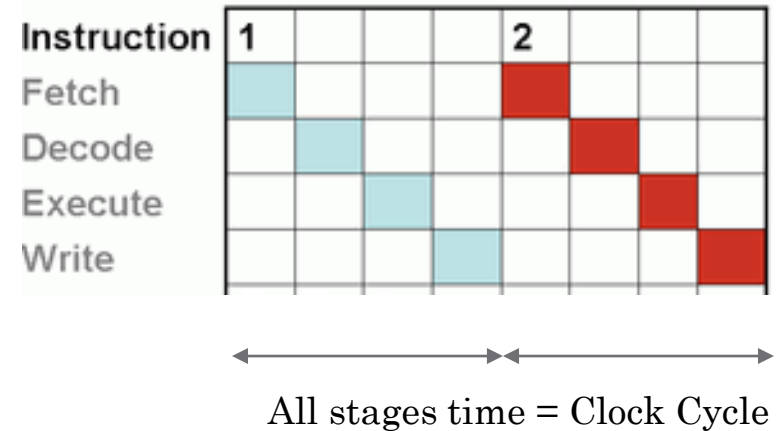
Sandwich making Pipelining



Pipelining

- What are the instruction stages:
 - Fetch
 - Decode (+Read registers)
 - Execute (+ calculate an address)
 - **Read memory**
 - Write result into a register
- In **single cycle**: instructions **must wait** for their turn
- Pipelining: eliminate the wait and **overlap** instructions!
- Single-Cycle versus Pipelined Performance?

Tutorial!



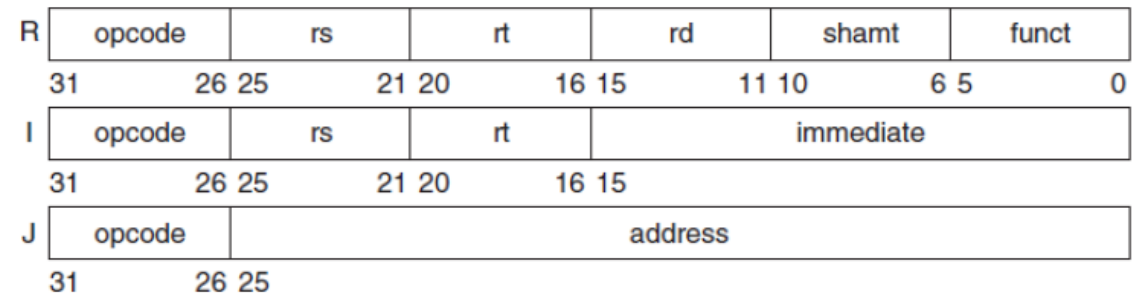
Designing ISA for Pipelining

What makes fetch and decode easier?

1. **Fixed-sized** instructions.
2. Instruction format where source register is in the **same place** in each instruction

What makes execute and memory read easy?

1. Memory operands **only** appear in **loads or stores**
2. Memory **alignment**



Research

- How much technology has influenced time spent in the different components?