CSE301 – Computer Organization

Tutorial 4

Single-Cycle CPU Implementation

On the next page is a **single-cycle datapath** for a machine different from the one seen in lecture.
It supports the following (complex) instructions:

```
lw_add   rd, (rs), rt      # rd = Memory[R[rs]] + R[rt]
addi_st  (rs), rs, imm     # Memory[R[rs]] = R[rs] + imm
sll_add  rd, rs, rt, imm   # rd = (R[rs] << imm) + R[rt]
```

All instructions use the same format (shown below), but not all fields are used by every instruction.

Field	op	rs	rt	rd	imm
Bits	31-26	25-21	20-16	15-11	10-0

Part (a) – Control Signals

For each of the above instructions, specify how the **control signals** should be set for correct operation. Use **X** for *don't care*. `ALUOp` can be one of: `ADD`, `SUB`, `SLL`, `PASS_A`, or `PASS_B` (*e.g.*, `PASS_A` means pass through the top operand unchanged).

inst	ALUsrc1	ALUsrc2	ALUsrc3	ALUop1	ALUop2	MemRead	MemWrite	RegWrite
lw_add								
addi_st								
sll_add								

Part (b) – Functional Unit Latency

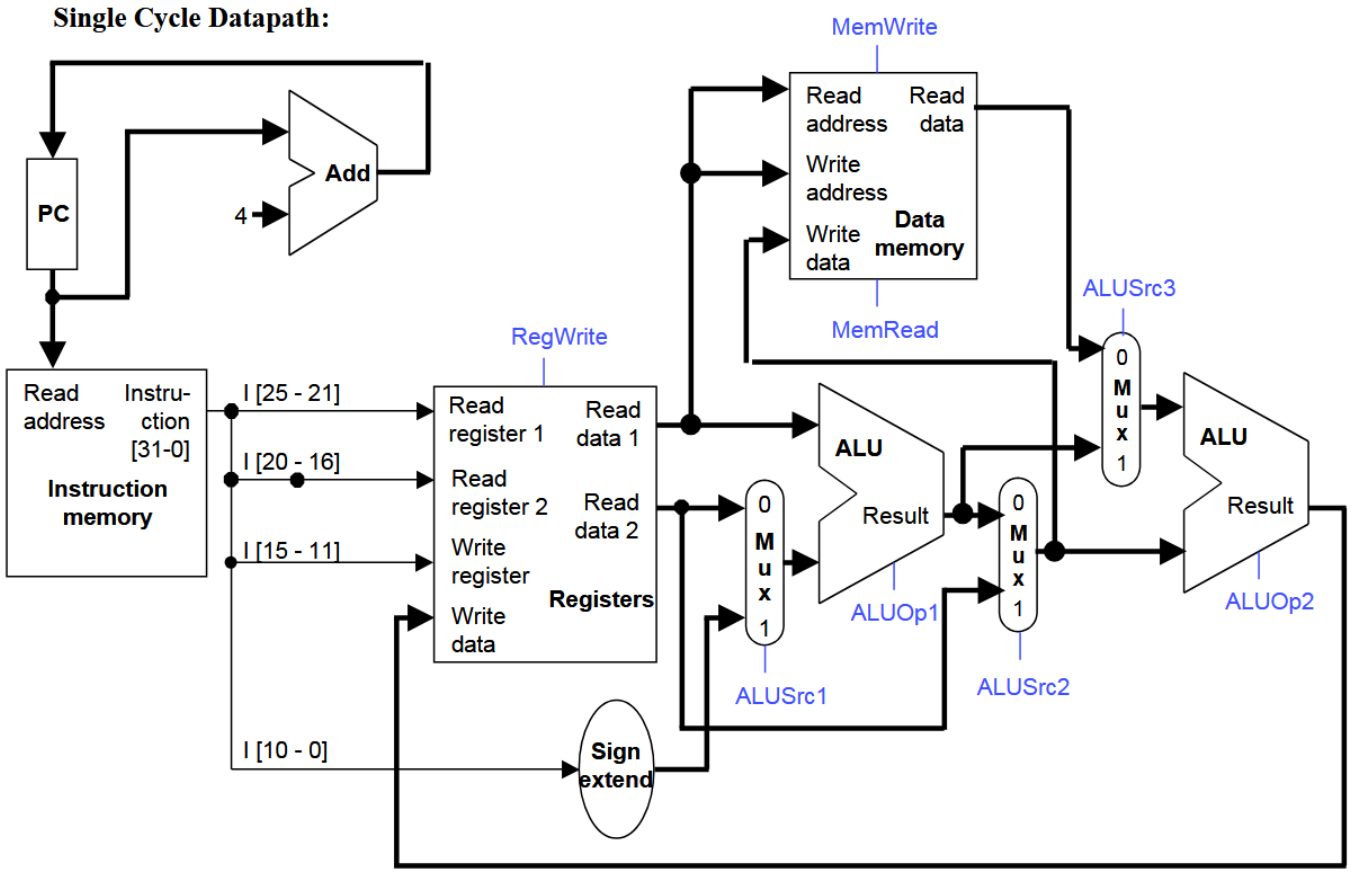
Given the following **functional unit latencies**, compute the **minimum time** to perform each instruction type. Explain your reasoning.

Functional Unit	Latency
Register File	2 ns
ALU	4 ns
Memory	3 ns

inst	Minimum time	Explain
lw_add		
addi_st		
sll_add		

Part (c) – CPI and Cycle Time

What are the **CPI** and **cycle time** for this processor?



Part 2: Questions with Answers

Part (a) – Control Signals

inst	ALUsrc1	ALUsrc2	ALUsrc3	ALUop1	ALUop2	MemRead	MemWrite	RegWrite
lw_add	X	1	0	X	ADD	1	0	1
addi_st	1	0	X	ADD	X	0	1	0
sll_add	1	1	1	SLL	ADD	0/X	0	1

Part (b) – Functional Unit Latency

inst	Minimum time	Explain
lw_add	14 ns	IMEM (3 ns) + RF_read (2 ns) + DMEM (3 ns) + ALU (4 ns) + RF_write (2 ns)
addi_st	12 ns	IMEM (3 ns) + RF_read (2 ns) + ALU (4 ns) + DMEM (3 ns)
sll_add	15 ns	IMEM (3 ns) + RF_read (2 ns) + ALU (4 ns) + ALU (4 ns) + RF_write (2 ns)

Part (c) – CPI and Cycle Time

Since the processor is a single-cycle implementation, the CPI is 1. The cycle time is set by the slowest instruction, which in this case is the sll_add, yielding a clock period of 15ns.