Performance Issues :

⇒ Longest delay determines clock period.
↳ We execute one instruction per clock cycle.

⇒ Typically Load instruction takes the longest time to execute

# Sub	=	8 ps
Mul	=	10 ps
lw	=	15 ps
sw	=	13 ps

Clock period = 15 ps

∴ Total time to execute this sequence
= 15+15+15+15 = 60 ps

Vs

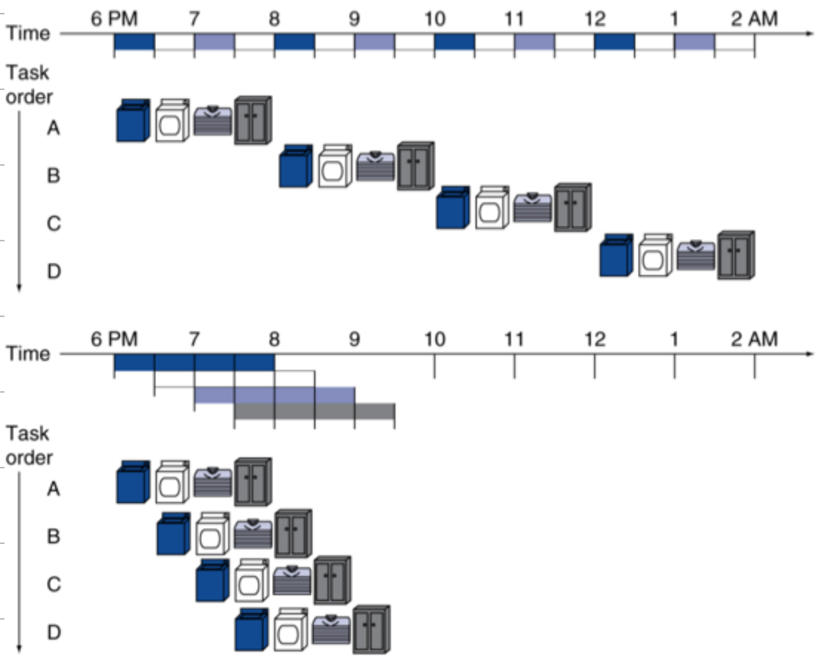
∴ Actual time to execute this sequence
= 8+10+15+13 = 46 ps

$$\underbrace{\text{Time Wastage} = (60-46)\,ps = 14\,ps}$$

to overcome this wastage we move on to $\underbrace{\text{pipelining}}$.
(parallel execution of instruction)

Pipelining Analogy

- Pipelined laundry: overlapping execution
  - Parallelism improves performance



instruction: laundry

1. Wash
2. Dry
3. Fold
4. Store

[at a time, you can only Dry/Wash/Fold/Store 1 cloth]

RISC-V Pipeline

- Five stages, one step per stage
  1. IF: Instruction fetch from memory
  2. ID: Instruction decode & register read
  3. EX: Execute operation or calculate address
  4. MEM: Access memory operand
  5. WB: Write result back to register

Pipeline Performance

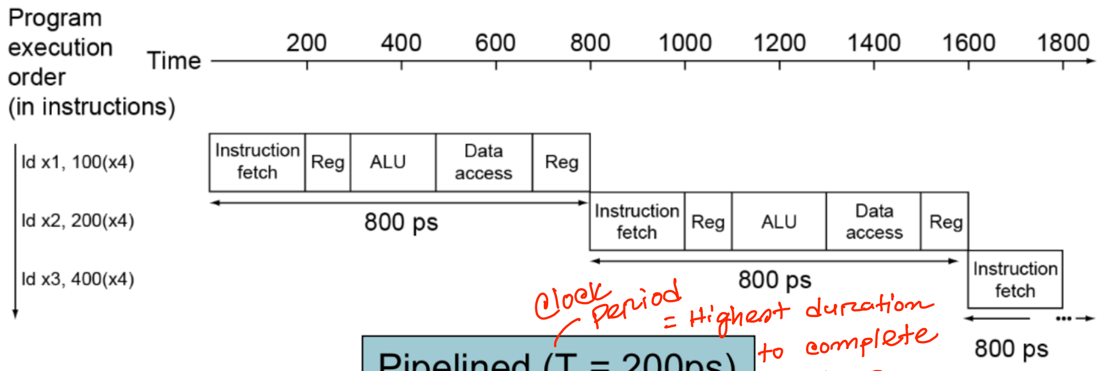
- Assume time for stages is
  - 100ps for register read or write
  - 200ps for other stages
- Compare pipelined datapath with single-cycle datapath

Instr	Instr fetch	Register read	ALU op	Memory access	Register write	Total time
ld	200ps	100 ps	200ps	200ps	100 ps	800ps
sd	200ps	100 ps	200ps	200ps		700ps
R-format	200ps	100 ps	200ps		100 ps	600ps
beq	200ps	100 ps	200ps			500ps

Pipeline Performance

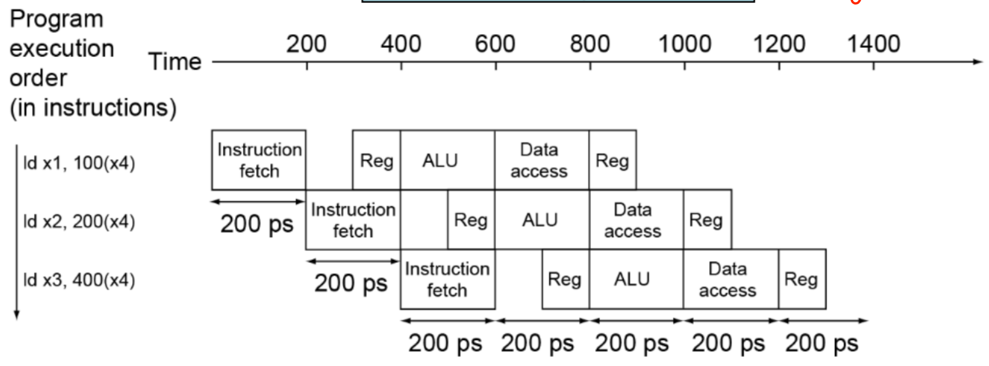
Clock Period = Highest duration to complete an instruction

Single-cycle (T_c = 800ps)



Clock Period = Highest duration to complete a stage

Pipelined (T_c = 200ps)


Program execution order (in instructions)
Time
200 400 600 800 1000 1200 1400
ld x1, 100(x4)
ld x2, 200(x4)
ld x3, 400(x4)
Instruction fetch
Reg
ALU
Data access
Reg
200 ps
200 ps
200 ps 200 ps 200 ps 200 ps 200 ps