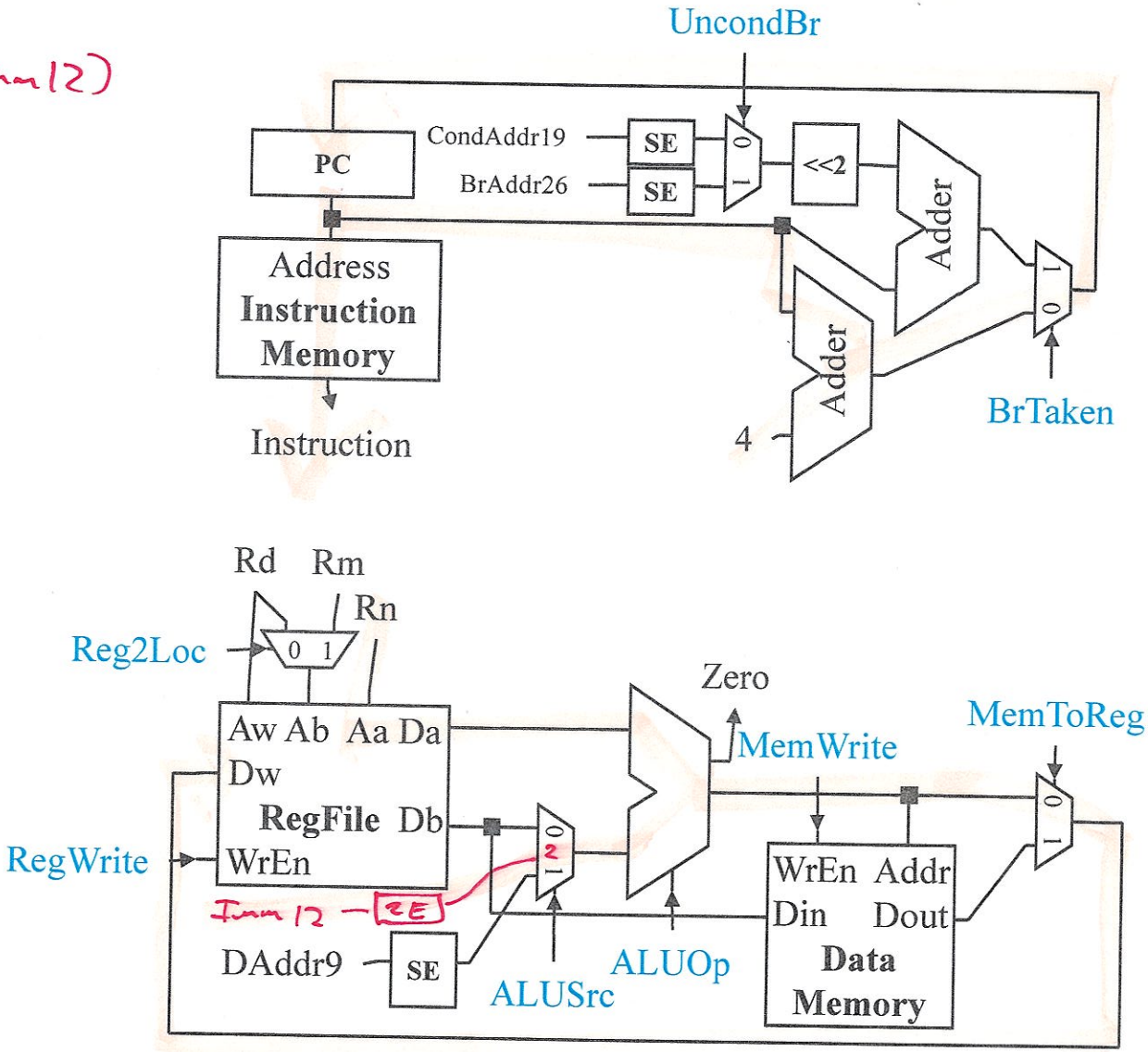
Review Problem 25

- Implement ADDI Rd, Rn, imm12 on our CPU

instr = Mem[PC];
$Reg[Rd] = Reg[Rn] + SE(imm12)
PC = PC + 4;

Signal	Value
Reg2Loc	X
ALUSrc	1
MemToReg	0
RegWrite	1
MemWrite	0
BrTaken	0
UncondBr	X
ALUOp	+



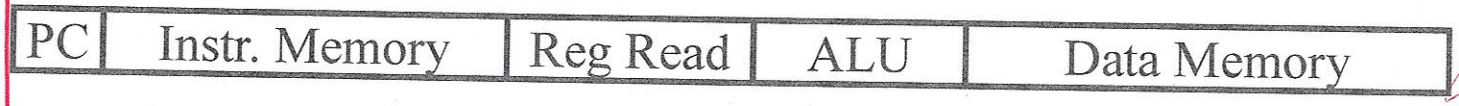
Performance of Single-Cycle Machine

CPI?

ADD, SUB

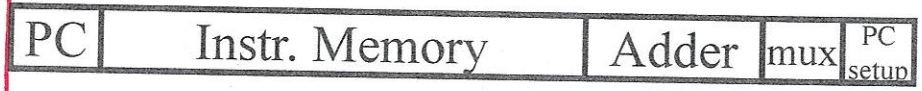
LDUR

STUR



CBZ

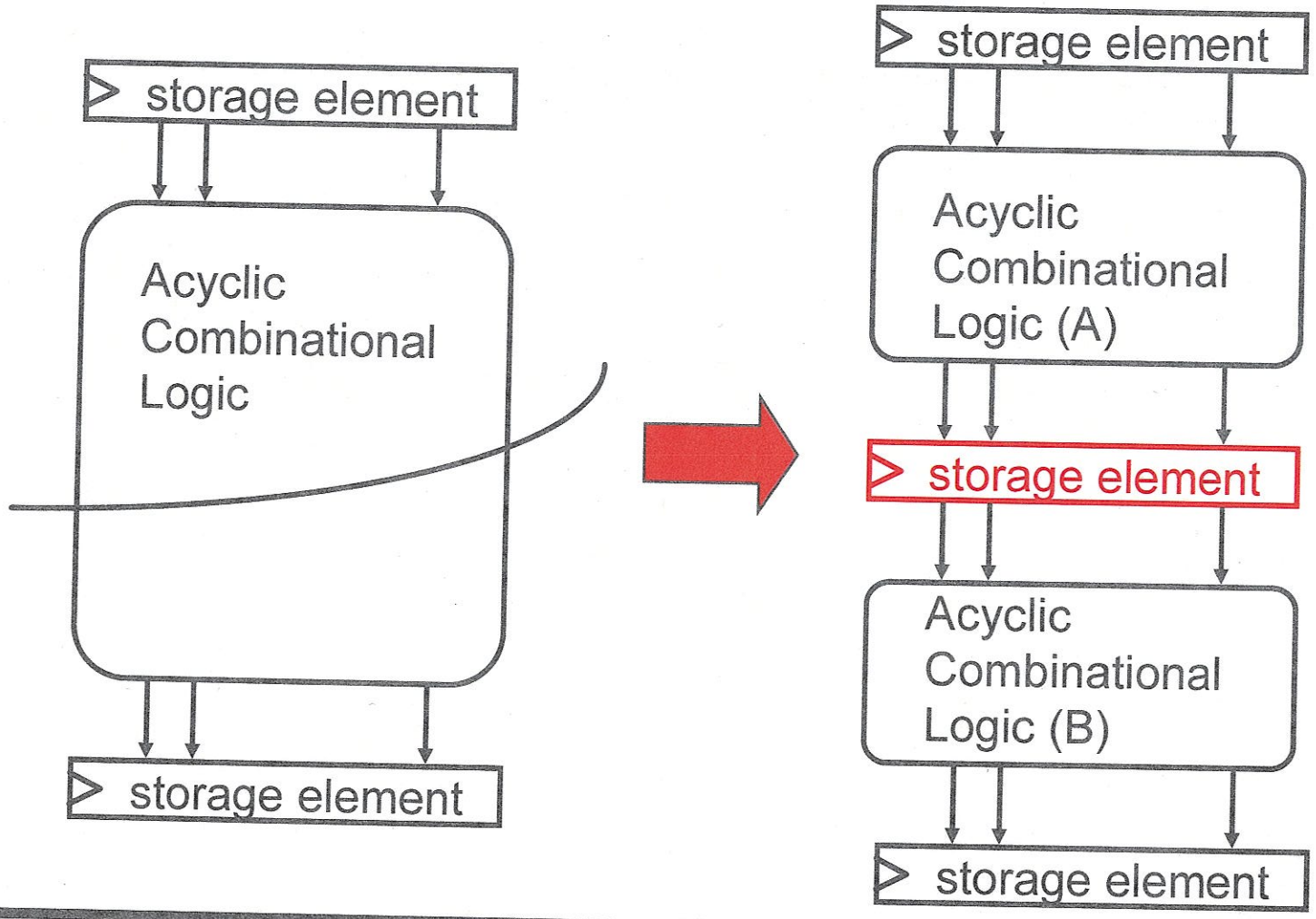
B



Reducing Cycle Time

Cut combinational dependency graph and insert register / latch

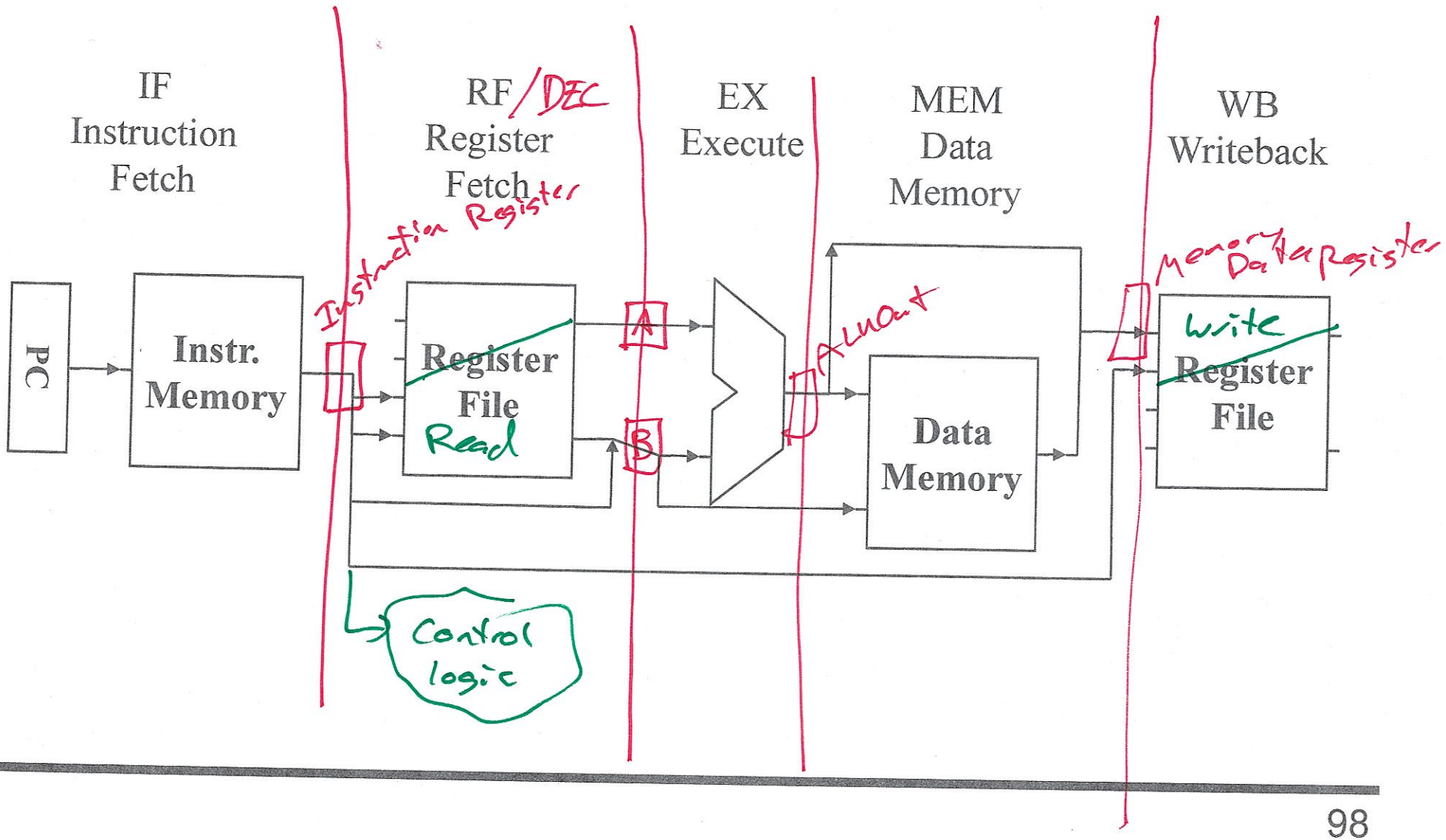
Do same work in two fast cycles, rather than one slow one



"slow": Memories, Regfile, ALU
"fast": muxes, Register, SE, ZE

Pipelined Processor Overview

Divide datapath into multiple stages



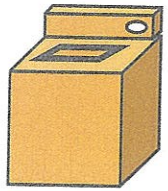
Pipelining

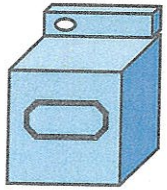
Readings: 4.5-4.8

Example: Doing the laundry

Ann, Brian, Cathy, & Dave 

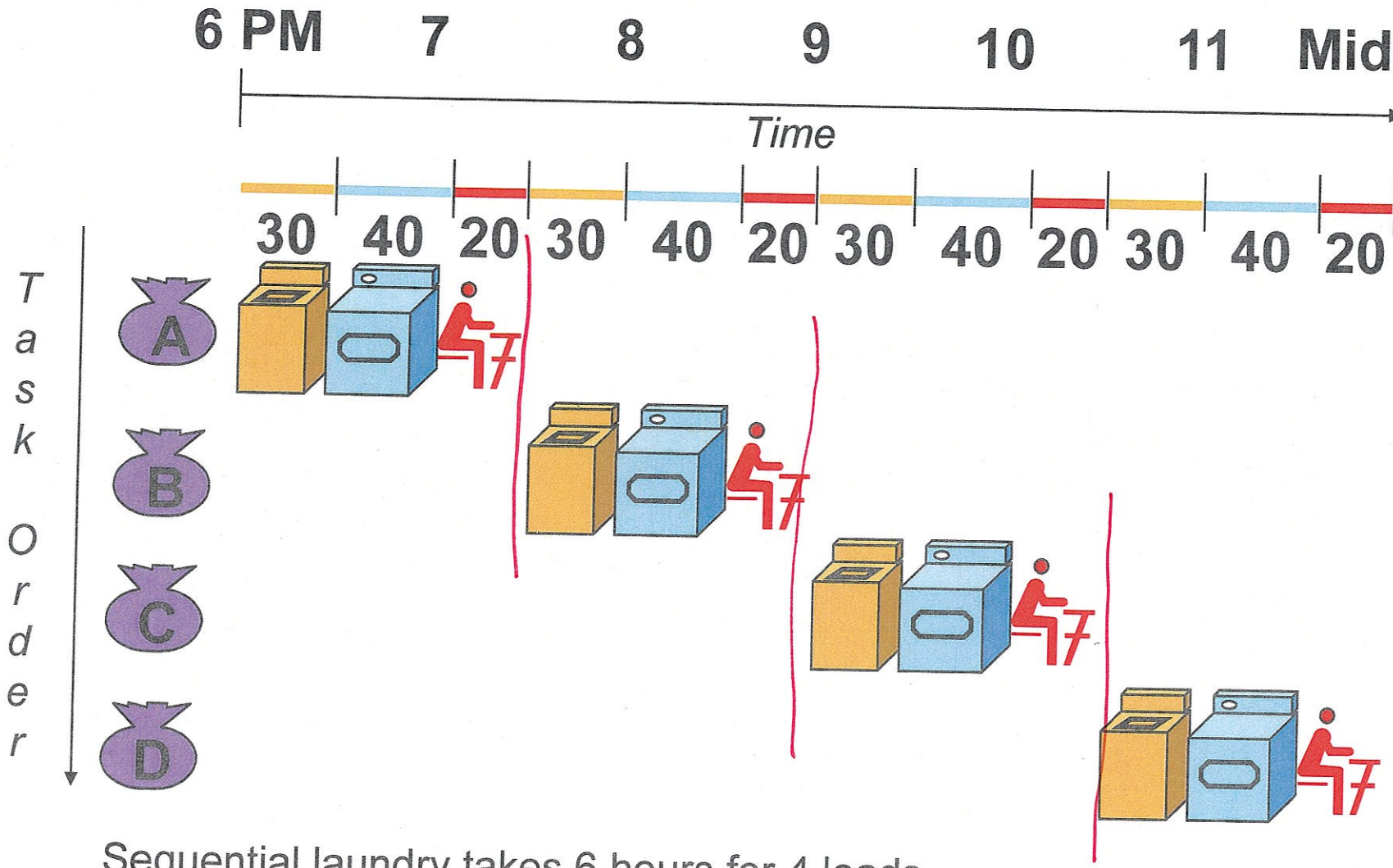
each have one load of clothes to wash, dry, and fold

Washer takes 30 minutes 

Dryer takes 40 minutes 

"Folder" takes 20 minutes 

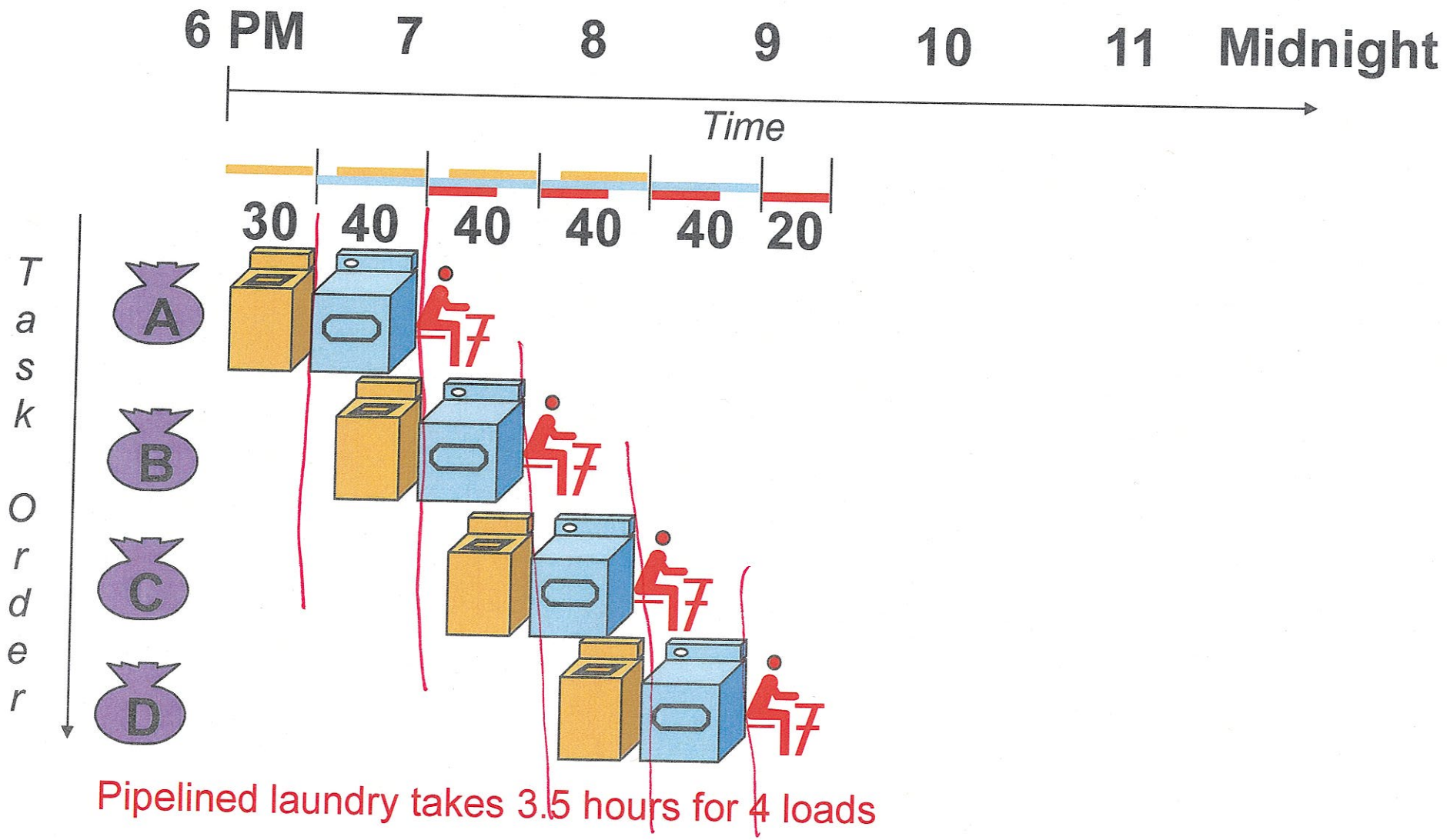
Sequential Laundry



Sequential laundry takes 6 hours for 4 loads

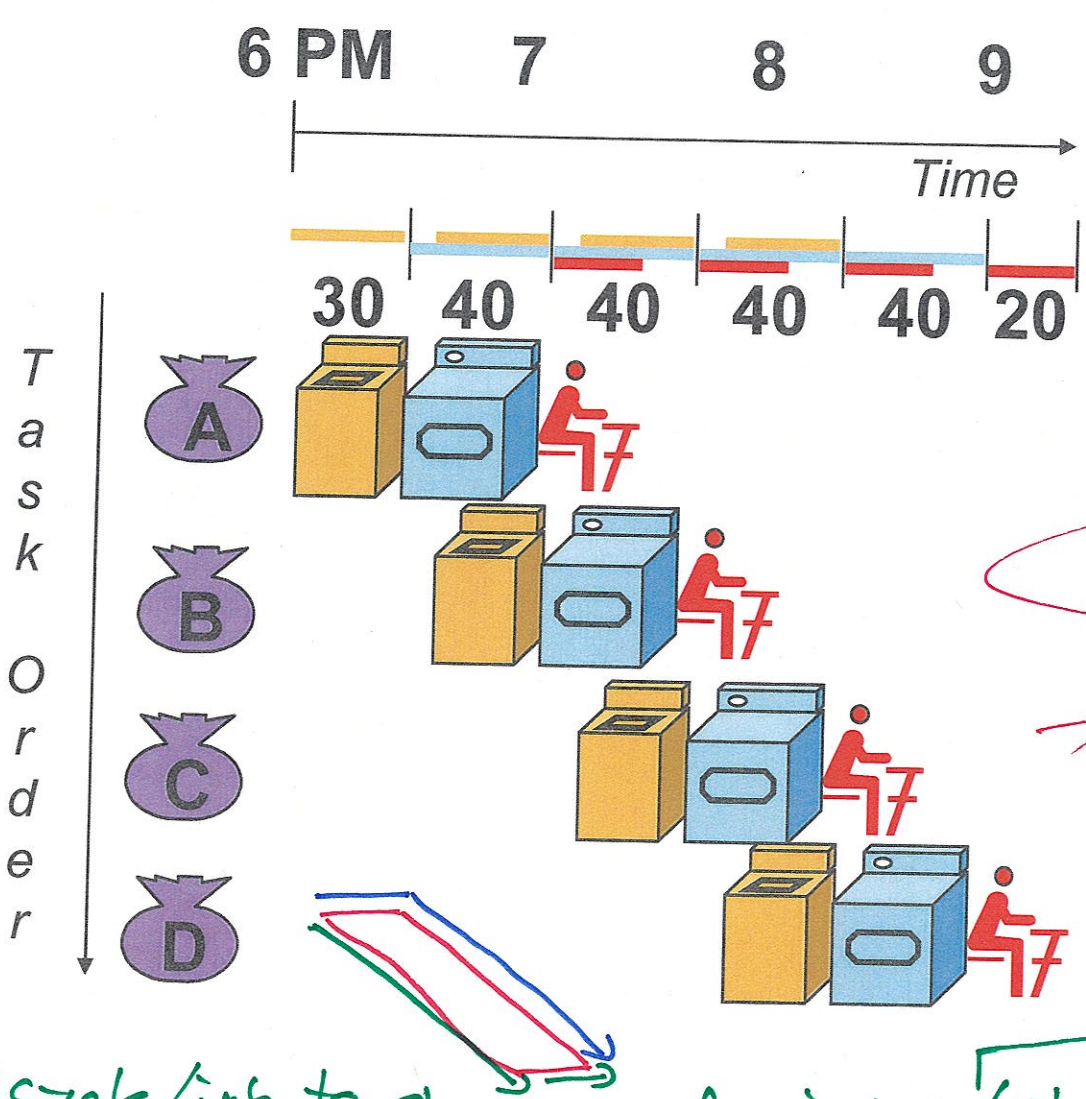
If they learned pipelining, how long would laundry take?

Pipelined Laundry: Start work ASAP



Pipelined laundry takes 3.5 hours for 4 loads

Pipelining Lessons



- Pipelining doesn't help latency of single task, it helps throughput of entire workload
- Pipeline rate limited by slowest pipeline stage
- Multiple tasks operating simultaneously using different resources
- Potential speedup = Number pipe stages
- Unbalanced lengths of pipe stages reduces speedup
- Time to "fill" pipeline and time to "drain" it reduces speedup
- Stall for Dependences

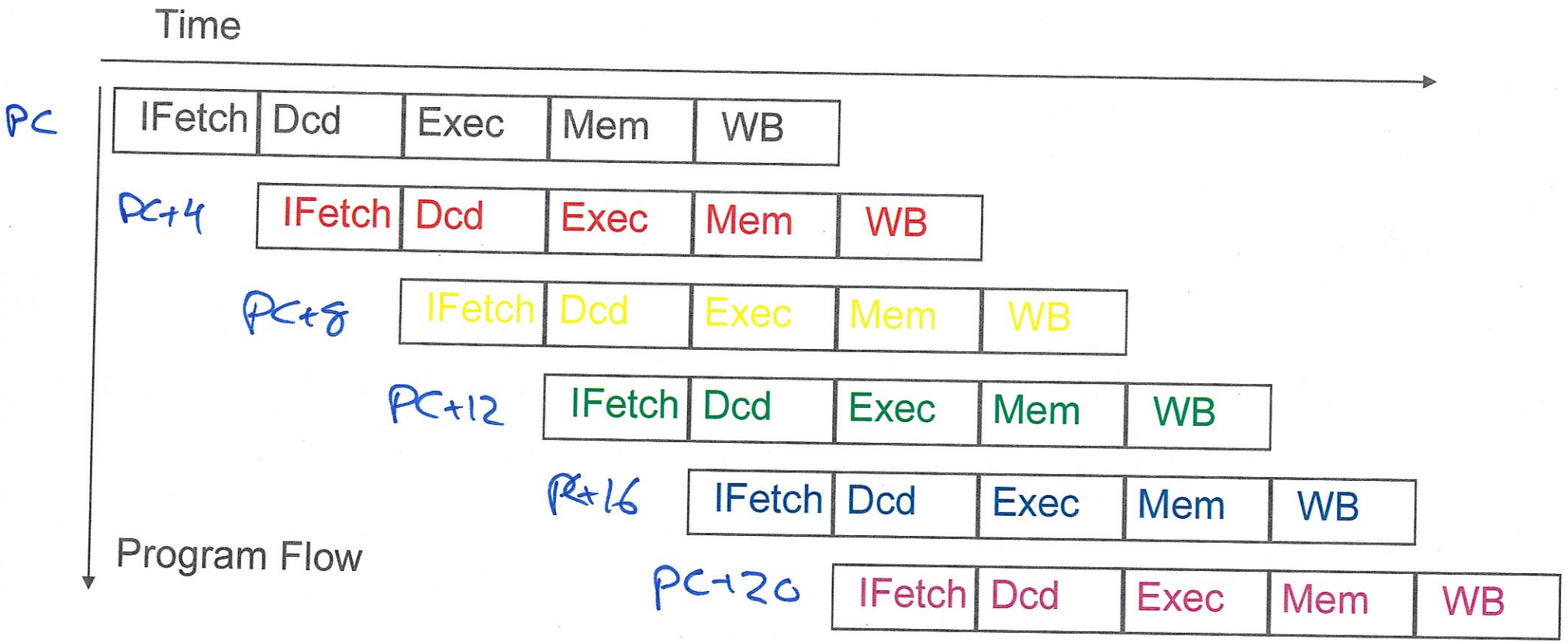
drain

1 cycle/job to start each job + (stages - 1) cycles to finish the last job

(stages - 1) cycles to get first job to the last stage + 1 cycle/job finish

fill time

Pipelined Execution



Now we just have to make it work