Chapter 06: Instruction Pipelining and Parallel Processing

Lesson 1:
Instruction Pipeline

Objective

- Learn Instruction pipelining and processing in a pipelined processor
- Learn how a designer implements pipeline
- Learn what are cycle time, pipeline latency and throughput

Early Computers

Early Computers

- Executed instructions in a very straightforward fashion
- Five steps sequentially in the processor to execute an instruction

Five steps in Straightforward Execution

(i) Fetch an instruction from memory

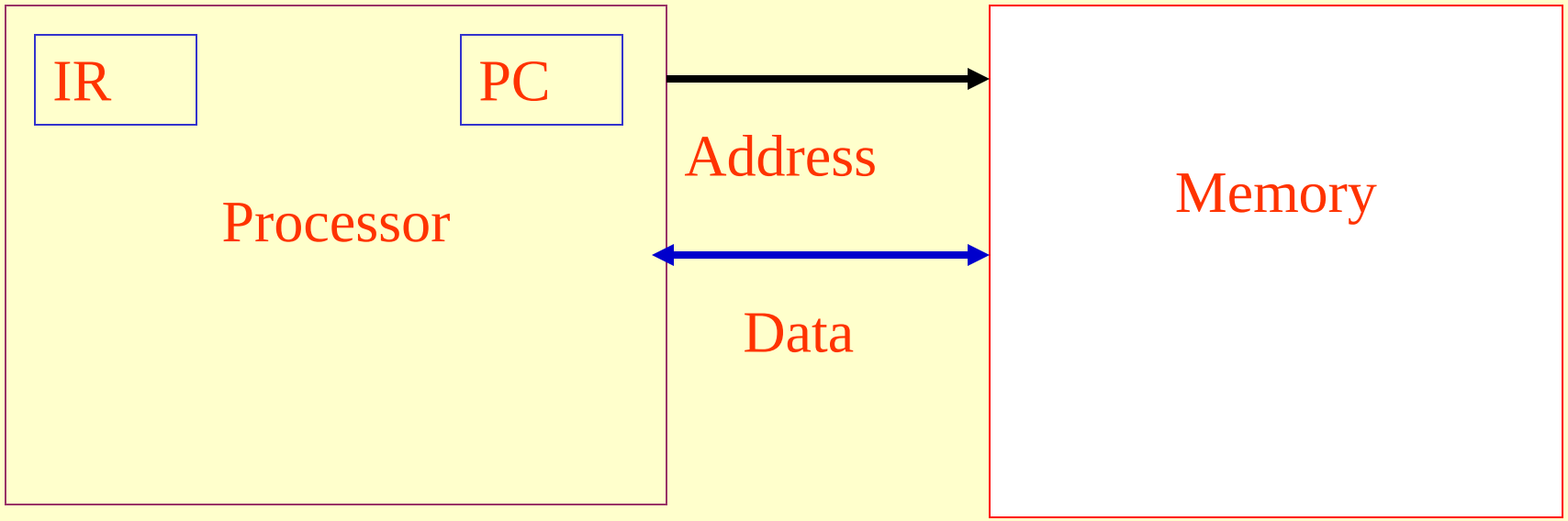
(ii) Decode it to determine what the instruction is

(iii) Read the instruction's inputs from registers at the register file

(iv) Performed the computations required by the instruction

(v) Wrote the result back into the registers at the register file

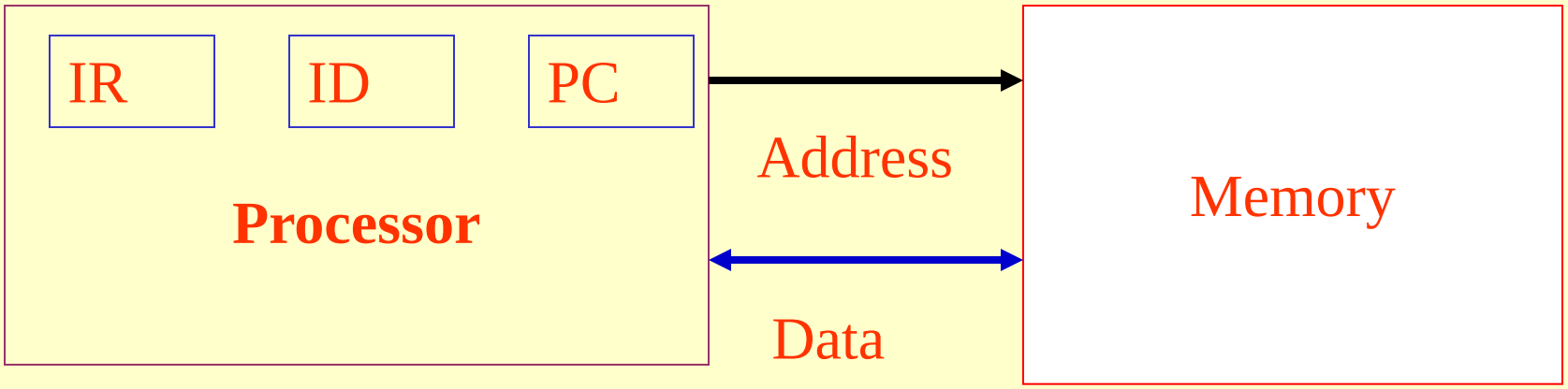
Instruction Fetch



• Fetch an instruction from memory: Steps 1a and 1b) Processor requests instruction from memory using address in PC (or IP) register

Memory using data bus returns the instruction to IR register

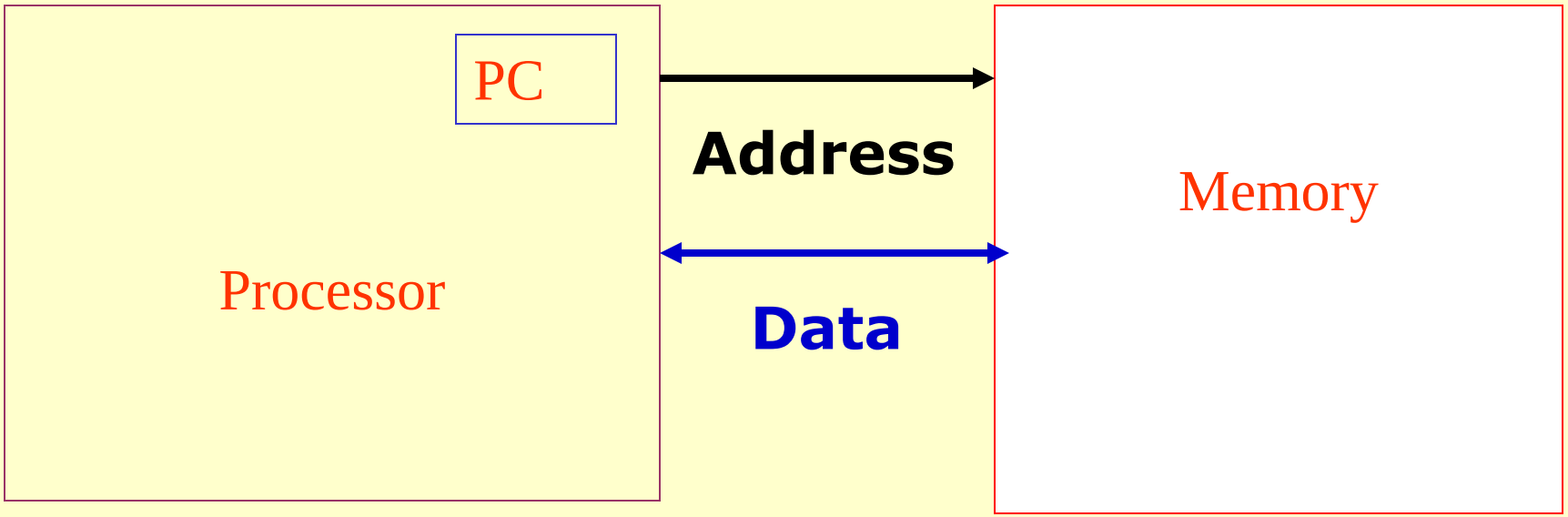
Instruction decoding, read operands and execution



• Decode it to determine what the instruction is in Step 2) Processor decodes instruction and places at ID register

Read Operands and Execute Steps 3and 4) Read the instruction's inputs from registers at the register file and Processor executes instruction at execution unit

Result Write back and PC update for the next



• Write the result back into the registers at the register file and PC updates Steps 5a and 5b) Result of instruction written back for register or memory and PC updates for next instruction

Instruction Execution Approach in Early Computers

- Instructions at the processor instruction set that accessed memory slightly different
- But each instruction completely finished before the execution of next one began

Problem

- Different Hardware (circuits) needed to perform each of the five steps (instruction fetch, instruction decode, register read, instruction execution, and register write back)
- Most of the hardware (most part of the processor circuits) idle at any given moment

Remaining Steps Hardware Waiting

- Most part of the processor circuits waiting for the other parts of the processor to complete that part of executing an instruction

Analogy

- In many ways, this is similar to baking several loaves of bread cakes
- Making the batter for one cake loaf (step 1)
- Baking the cake loaf (step 2)
- Cooling of the cake (step 3)
- Repeating the entire three step process for next cake

New Computers

New Computers

- Hardware (circuits) needed to perform each of the five steps (instruction fetch, instruction decode, register read, instruction execution, and register write back) utilized simultaneously for 5 instructions
- Most of the hardware not idle at any given moment

Remaining steps hardware does not wait

- No wait of the most part of the processor circuits for the other parts of the processor to complete that part of executing an instruction

Baking Analogy

- While each of the three steps of baking each loaf cake of bread has to be done in order and takes a set amount of time, one person could bake several cakes loaves of bread much faster

Baking Analogy

- By making the batter for the second loaf cake while first loaf cake is baking
- Making the batter for the third cake loaf while the second loaf cake is baking
- First loaf cake cooling
- Continuing this process with each loaf cake so that there are three loaves of bread cakes in progress at any time

Baking Analogy

- Each loaf of bread would take the same amount of time but the number of loaves for cakes made in a given amount of time would increase (by about three)

Instruction Pipeline

Instruction Pipeline

- A technique for overlapping the execution of several instructions to reduce the execution time of a set of instructions

Exemplary processors using Instruction Pipeline

- 80x86
- Pentium
- ARM

Instruction Execution Time and Execution Time of Set of Instructions

- Similar to the baking analogy, each instruction takes the same amount of time to execute in a pipelined processor as it would in an unpipelined processor (longer, actually, because pipelining adds hardware to the processor), but overlapping instruction execution increases the rate at which instructions can be executed

Computer performance in terms of Latency and throughput

- Latency— the amount of time that a single operation takes to execute
- Throughput — the rate at which operations get executed (generally expressed as operations/second or operations/cycle)

Throughput and Latency in an unpipelined processor

- Throughput = 1/latency, since each operation executes by itself

Computer performance in Pipelined Processor

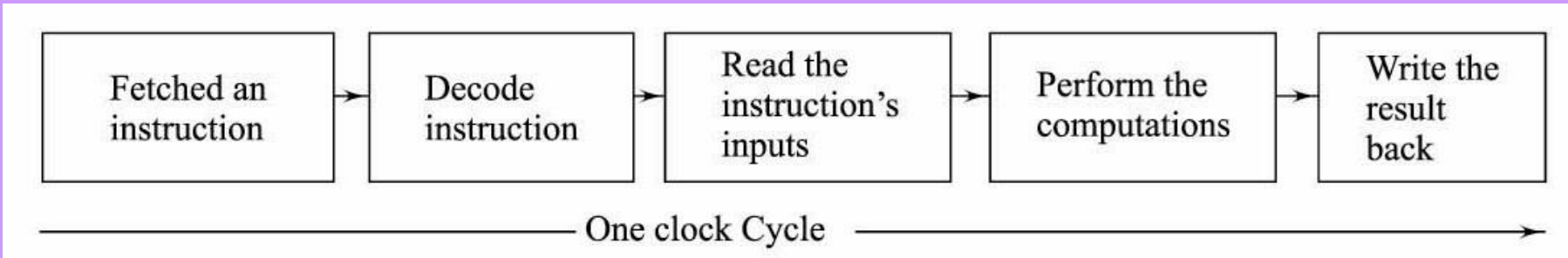
- Throughput > 1/latency, since instruction execution is overlapped
- The latency of a pipelined processor is still important, however, as it determines how often dependent instructions may be executed

Instruction Pipeline Implementation

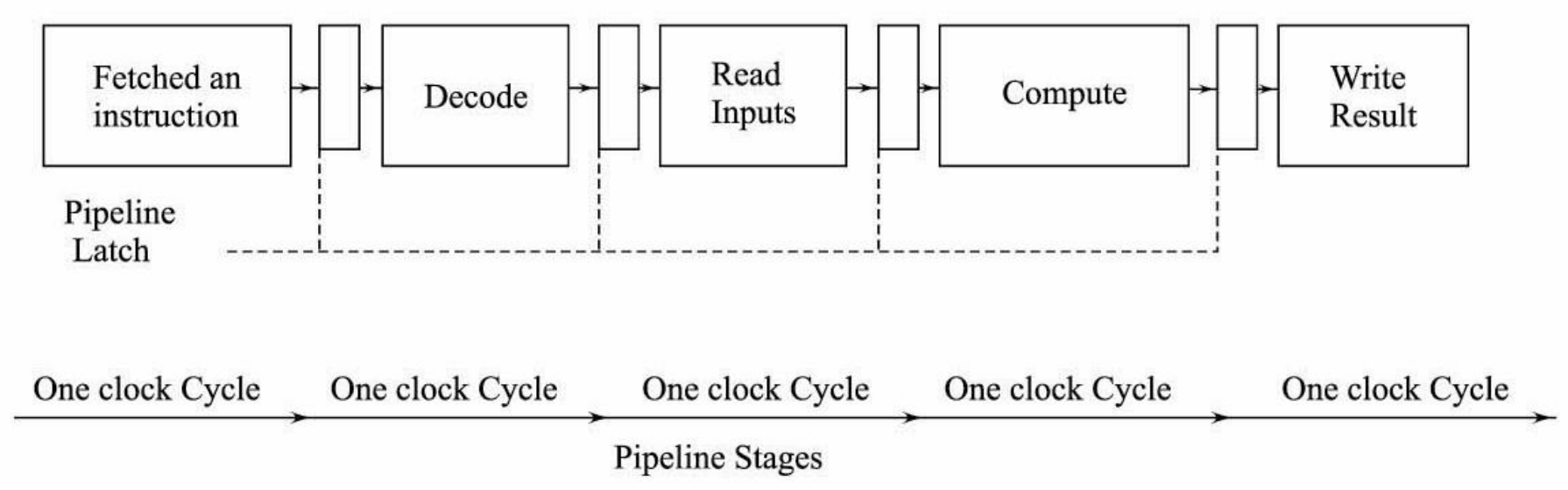
Instruction Pipeline

- To implement pipelining, a designer s divides a processor's datapath into sections (stages), and places *pipeline latches* (also called buffers) between each section (stage)

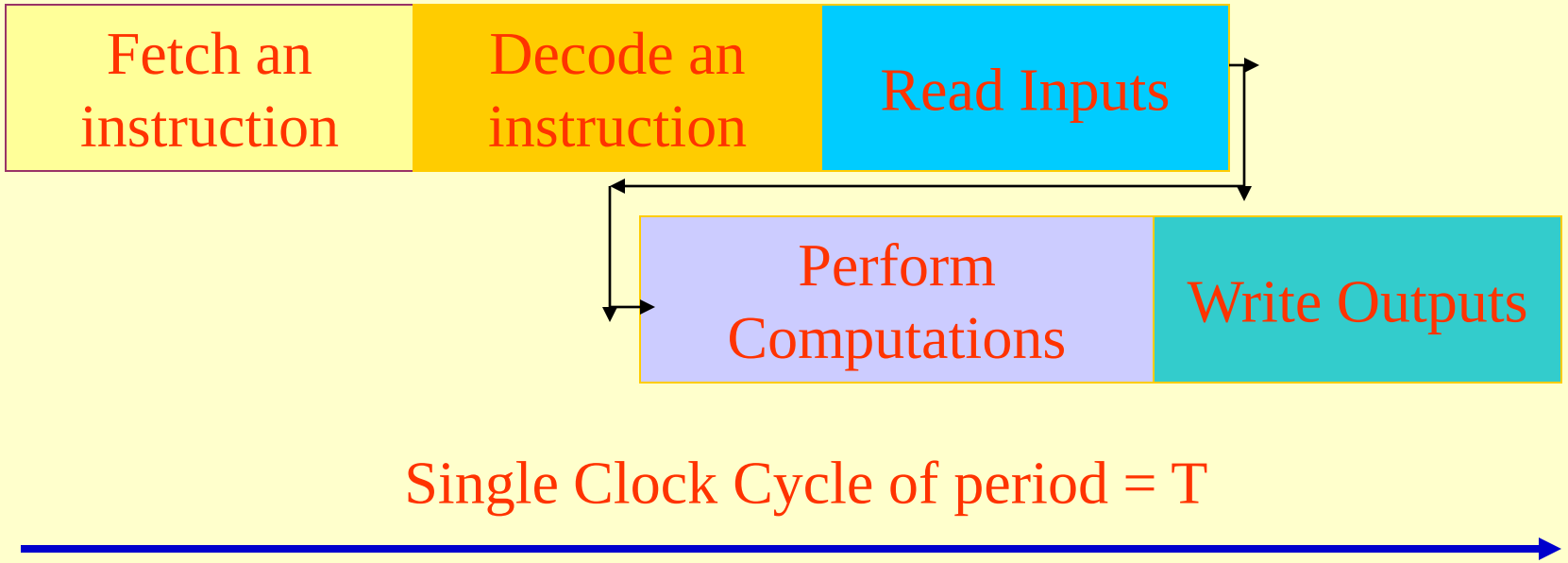
Unpipelined processor



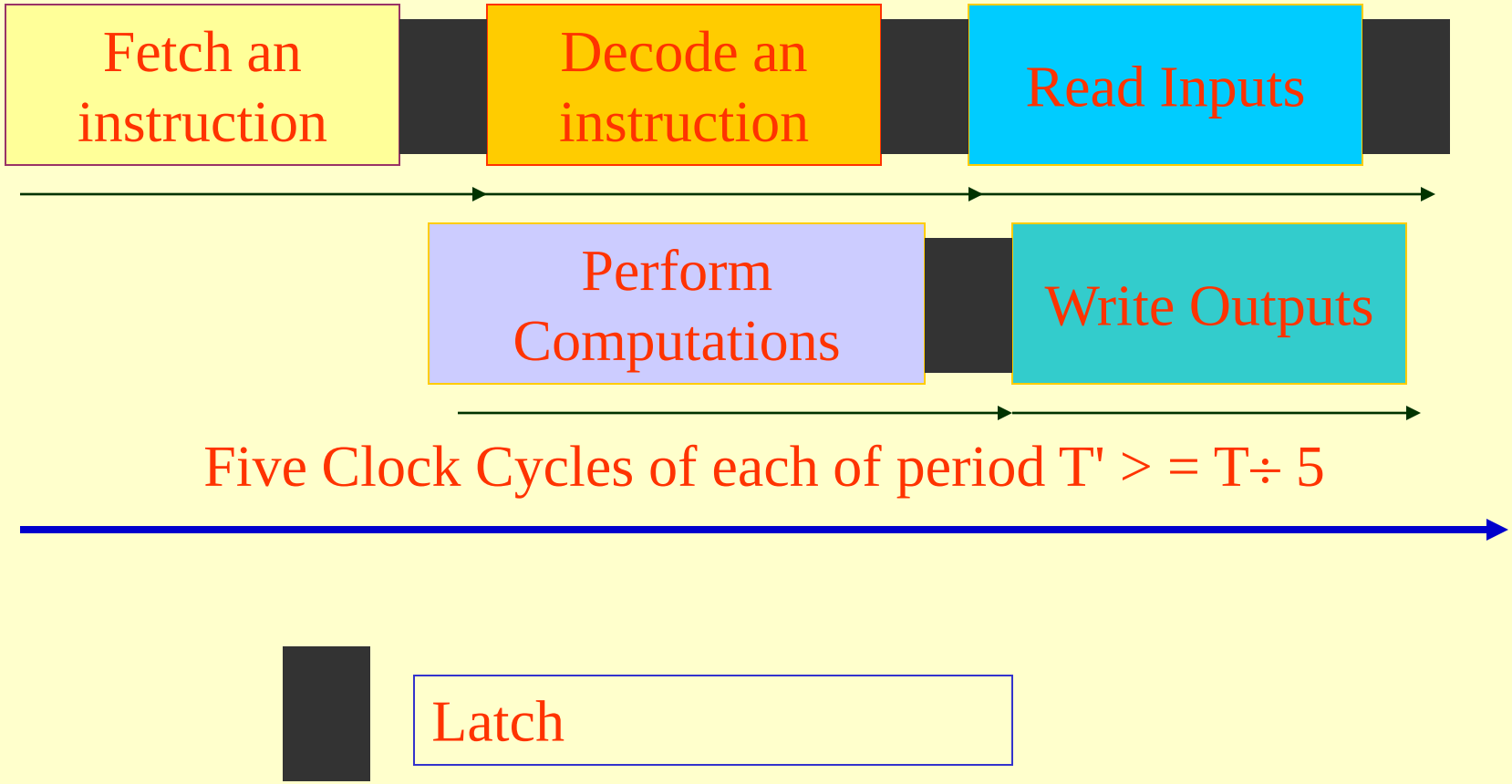
Pipelined five stages processor



Unpipelined Processor



Five Stage Pipelined Processor



Pipeline latches (buffers)

- The latches read their inputs at the start of each cycle
- Copy them to their outputs, which then remain constant throughout the rest of the cycle
- Datapath breaking into several sections (stages)
- Each stage latency = one clock cycle, since an instruction cannot pass through a pipeline latch (buffer) until the start of the next cycle

Amount of Datapath

- Stage of the pipeline─ the amount of the datapath that a signal travels through in one cycle
- Pipeline that takes *n* cycles in an *n*-stage pipeline
- Each cycle period now about one-fifth of unpipelined case period

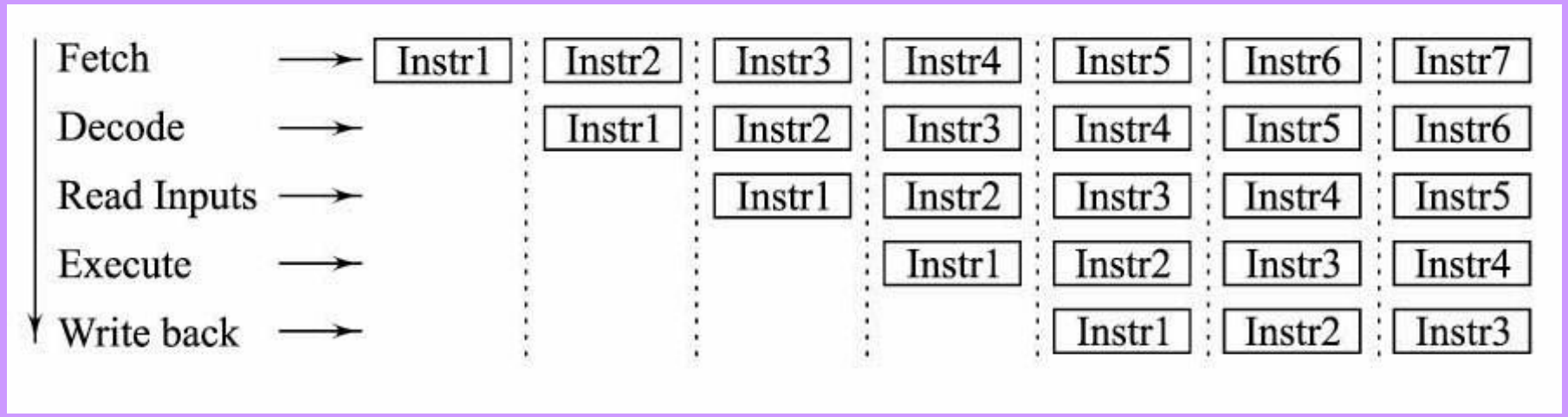
Pipeline five stages

- Stage 1─ the fetch instruction block and its associated pipeline latch (buffer)
- Stage 2─ the decode instruction block and its pipeline latch (buffer)
- Stages 3, 4, and 5─ the subsequent blocks of the pipeline

Architecture Designer's Differences

- Computer architects differ on whether a pipeline latch (buffer) is the last part of stage or the first part of the next stage, so an alternate division of the pipeline into stages would be to count the fetch instruction block as stage 1, the first pipeline latch (buffer) and the decode instruction block as stage 2, and so on

Instructions in the pipeline stages



Summary

We learnt

- Instruction pipelining
- A designer implementation of pipeline
- Processing in pipelined processor
- Cycle time
- Pipeline latency
- Throughput

End of Lesson 1

Instruction Pipeline