Introduction to Pipelining on the y86

- Topic
  - Are there performance issues with our current y86 implementation issue?
  - What is pipelining?
  - How/why is it useful?
- Learning outcomes
  - Motivate need for pipelining
  - Describe the performance trade offs that pipelining brings
  - Identify the challenges that pipelining is going to introduce
- Reading
  - Section 4.4

y86 Implementation Observations

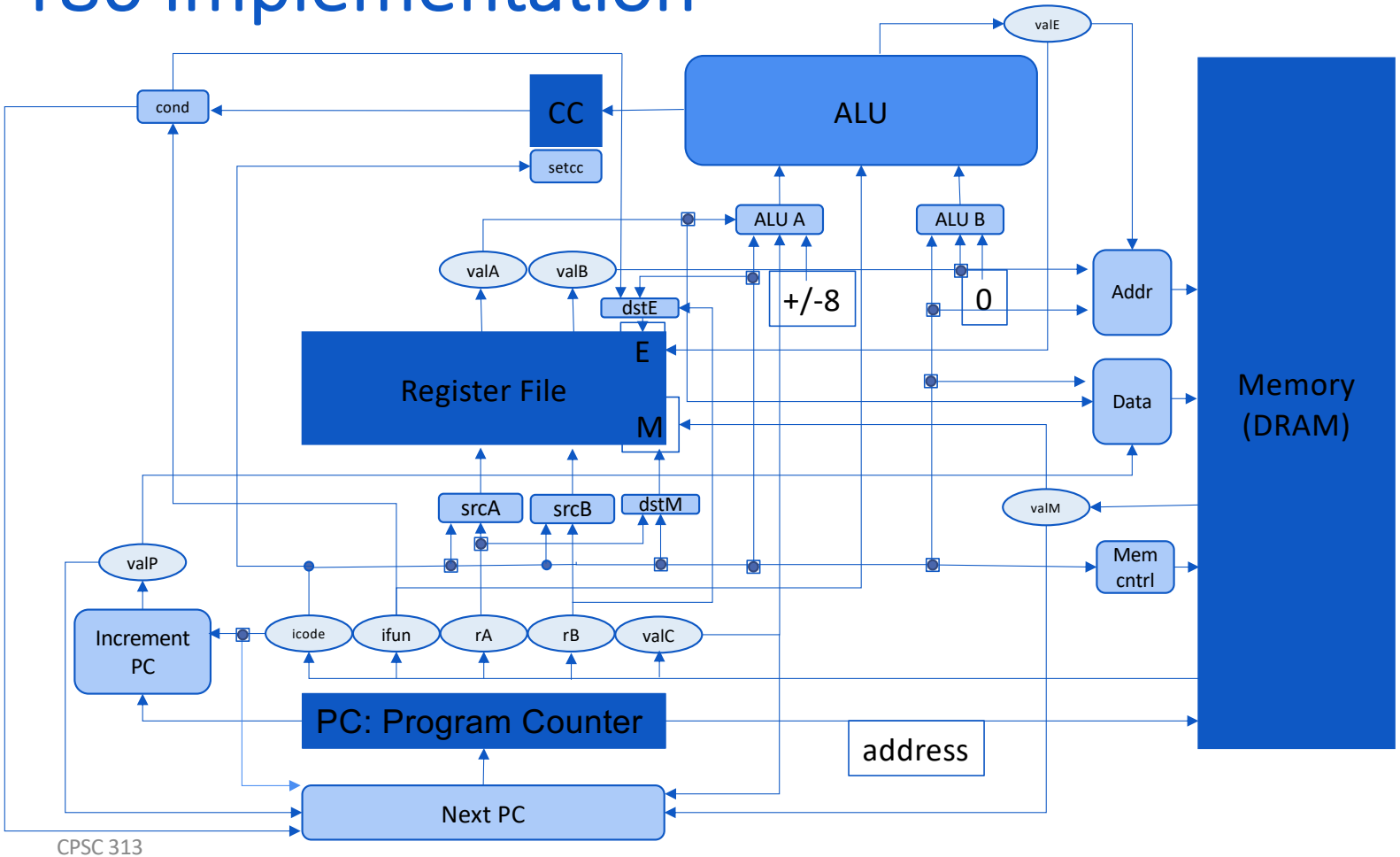
- We only ever read instructions in the FETCH stage
- We only read from the register file in the DECODE stage
- We only use the ALU in the EXECUTE stage
- We only read/write to memory in MEMORY stage
- We only write to the register file in the Writeback stage

y86 Implementation Observations

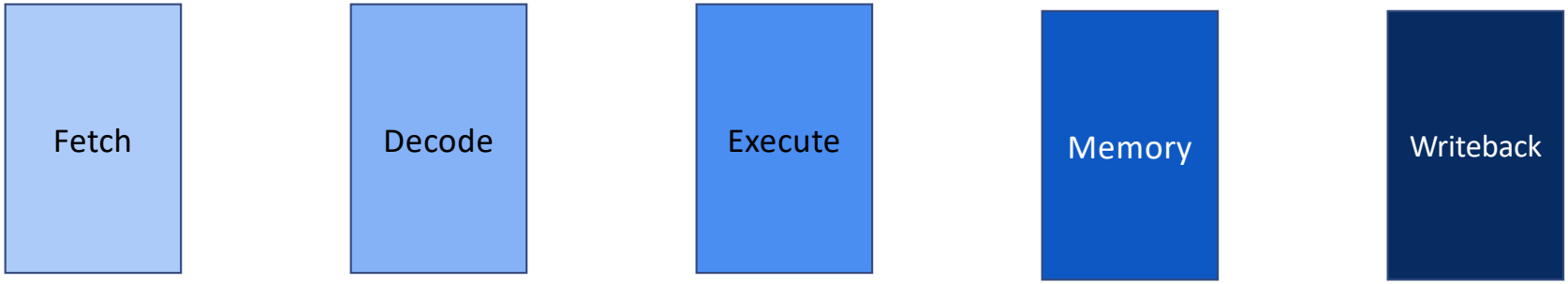
- We only ever read instructions in the FETCH stage
- We only read from the register file in the DECODE stage
- We only use the ALU in the EXECUTE stage
- We only read/write to memory in MEMORY stage
- We only write to the register file in the Writeback stage
- Implications:
  - For any given instruction, we have to wait for the signals to propagate through the entire circuit, but
  - At any given instant, most of the hardware is unused …

Pipelining

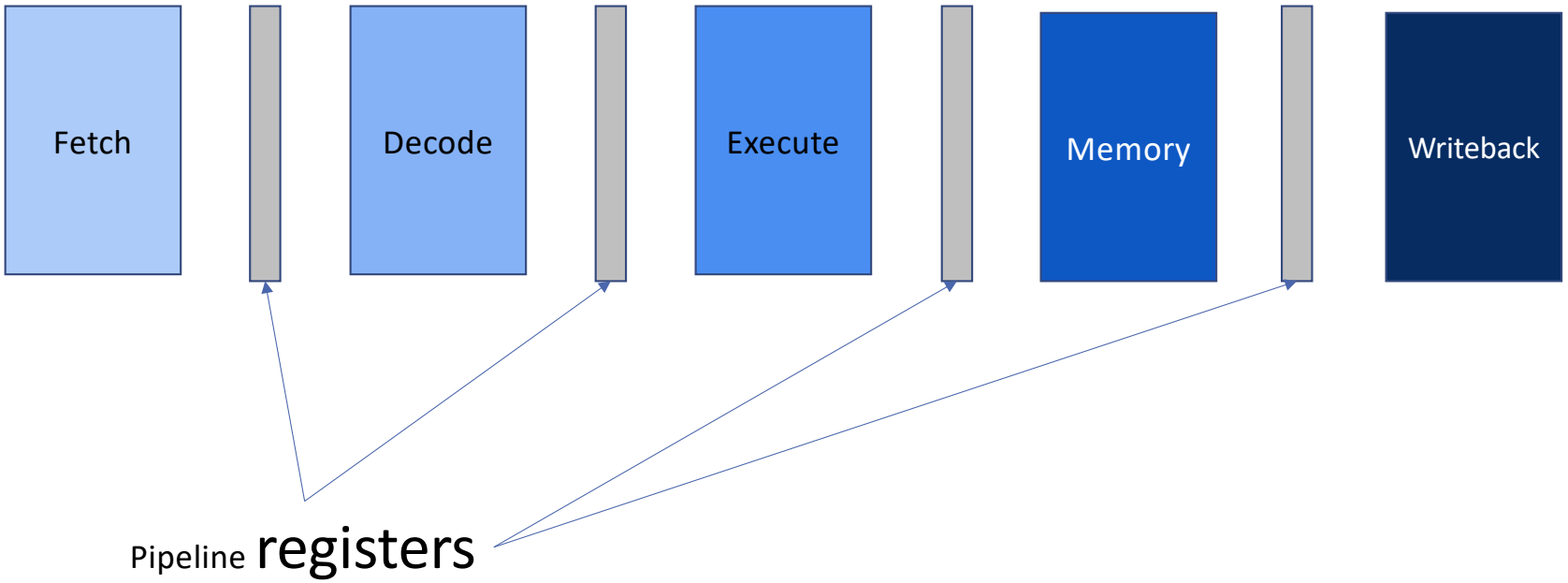
Y86 Implementation



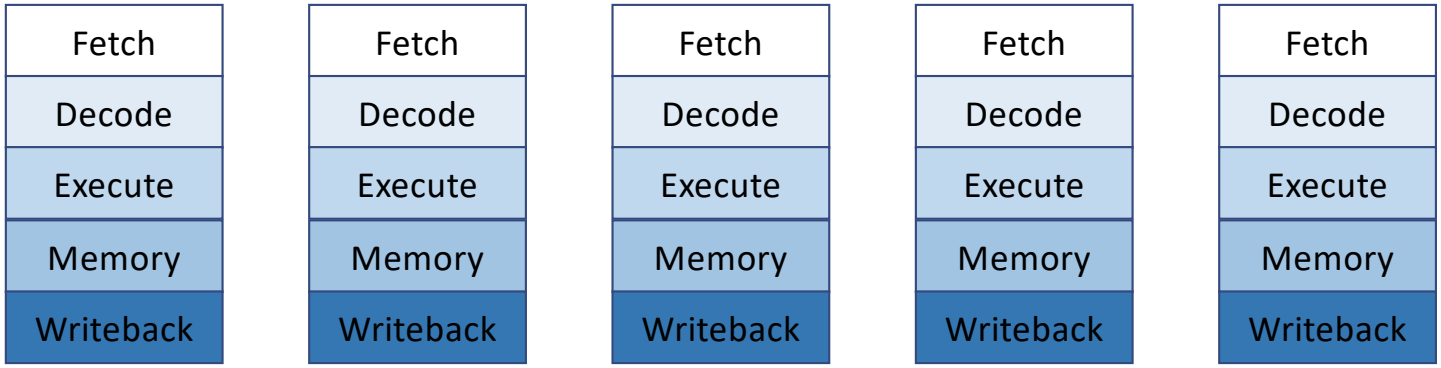
Consider these five stages



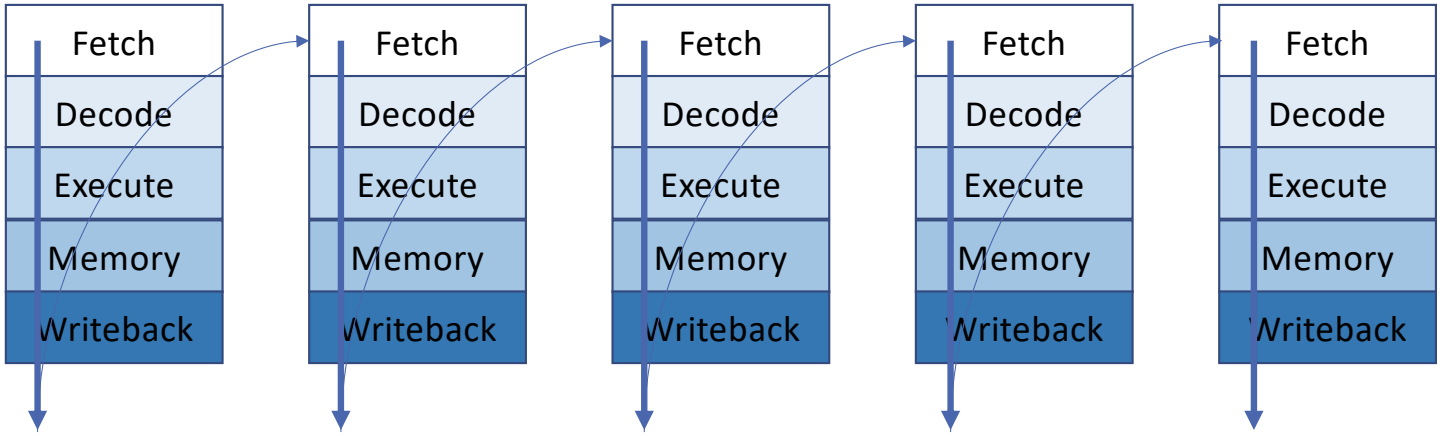
Introducing Pipeline registers



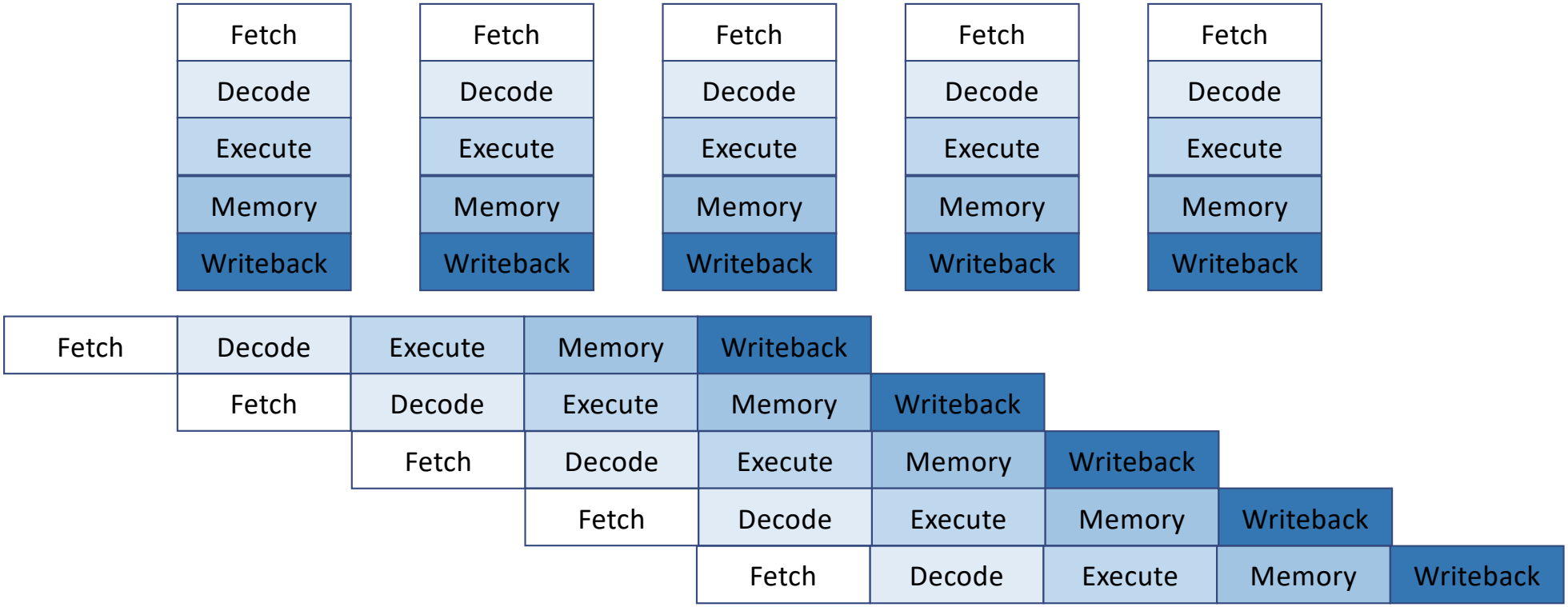
From Sequential to Pipelined



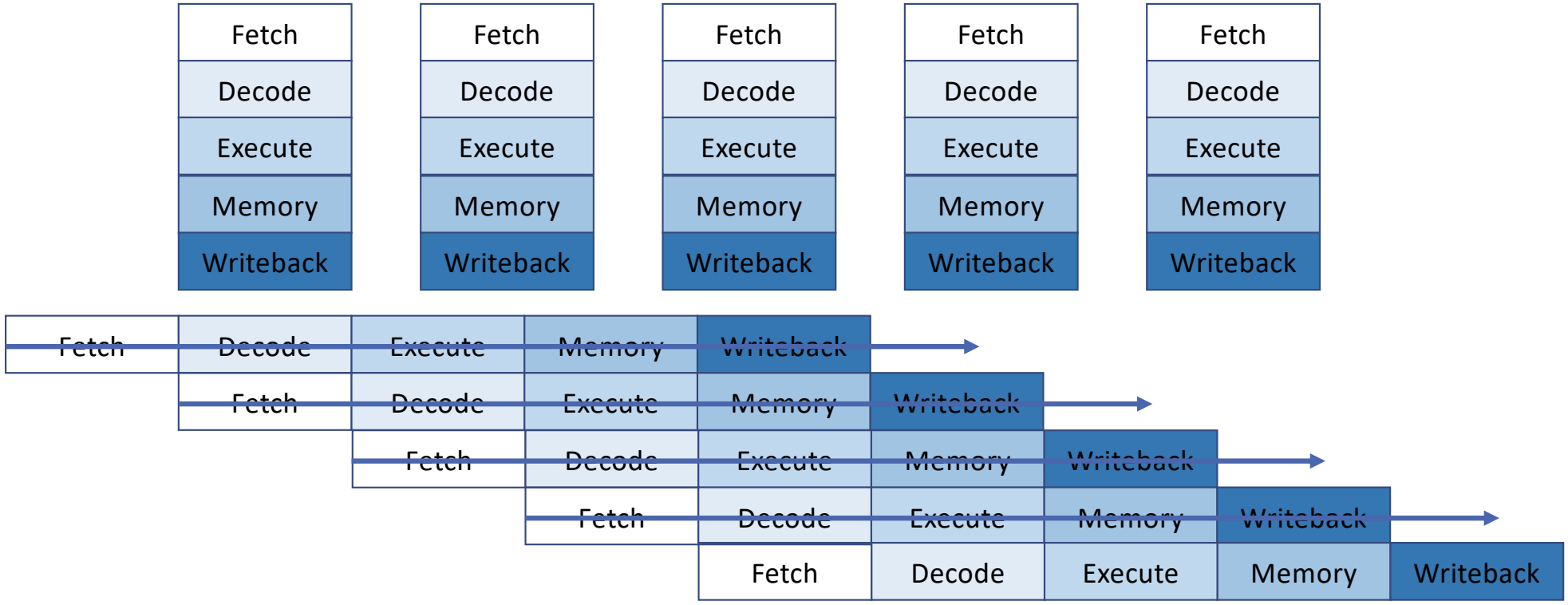
From Sequential to Pipelined



From Sequential to Pipelined



From Sequential to Pipelined



Pipelining: Pros and Cons

- Pros
  - Uses hardware more efficiently (units work in parallel)
  - A collection of instructions completes more quickly
- Cons
  - An individual instruction takes a bit longer (because the pipeline registers add latency).
  - If some phases take longer than others, short phases might have to wait for longer phases.
  - Sometimes you didn't have the right information at the right time.