ECE 550D
Fundamentals of Computer Systems and Engineering

Fall 2023

Datapaths

Xin Li & Dawei Liu

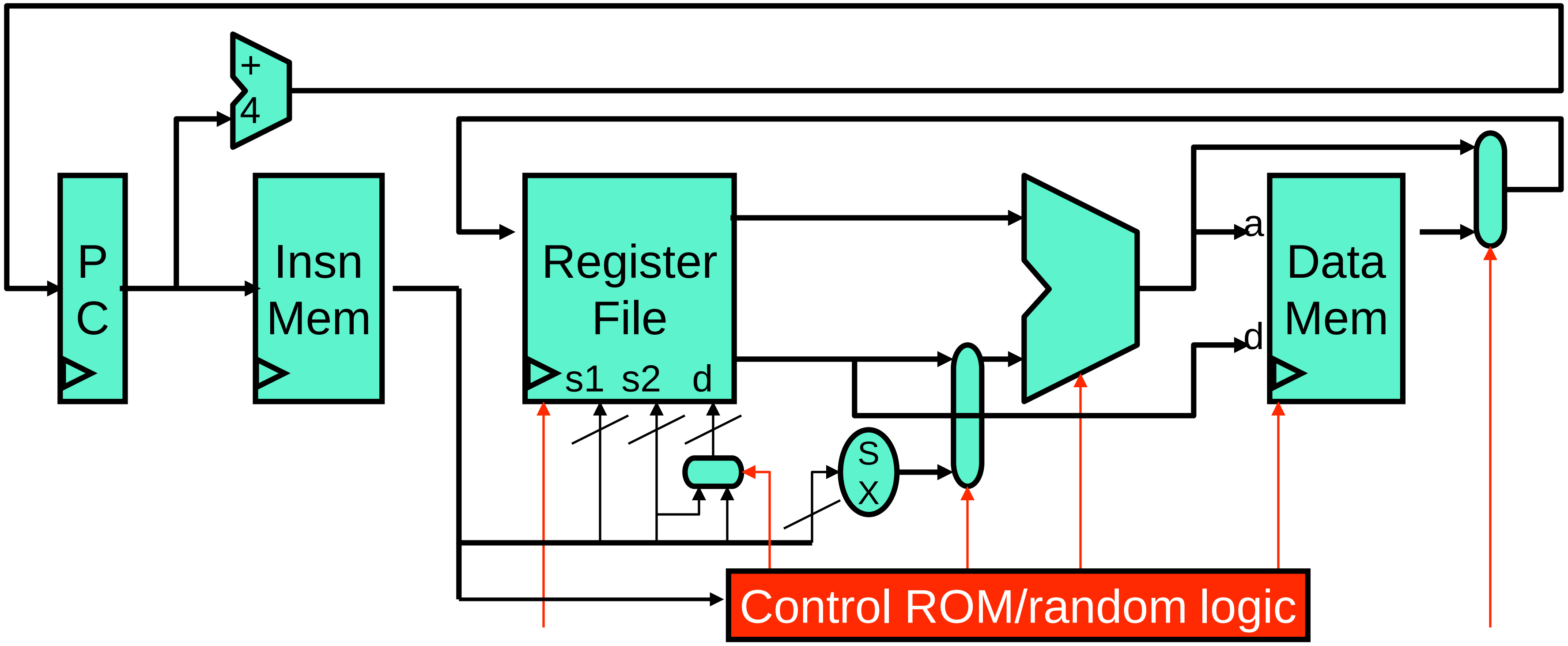
Duke Kunshan University

Slides are derived from work by
Andrew Hilton, Tyler Bletsch and Rabih Younes (Duke)

Last time

- What did we do last time?
- Datapaths:
  - Control signals
  - How control is implemented

Single-Cycle Datapath Performance



- Observations
  - + Low Cycles Per Instruction (**CPI**): 1
  - – Long clock period: to accommodate slowest insn

Interlude: Performance

- Previous slide alludes to something new: **Performance**
  - Don't just want it to work…
  - But want it to go fast!

- Three components to performance:

  Number of instructions
  x Cycles per instruction (CPI)
  x Clock Period (1 / Clock frequency)

$$\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}} = \frac{\text{Seconds}}{\text{Program}}$$

Interlude: Performance

- Three components to performance:

  Number of instructions <- Compiler's Job

  x Cycles per instruction (CPI)

  x Clock Period (1 / Clock frequency)

$$\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}} = \frac{\text{Seconds}}{\text{Program}}$$

- Insns/Program: determined by compiler + ISA
  - Generally assume fixed program when doing **micro-architecture**

Micro-architectural factors

- Micro-architecture:
  - The details of how the ISA is implemented
  - Affects CPI and Clock frequency

- Often will look at fixed program, and consider **MIPS**
  - Million Instructions Per Second
  - MIPS = IPC * Frequency (in MHz)
  - IPC = Instruction Per Cycle (1 / CPI)
  - Gives "Bigger is better" number

The use of "MIPS" to mean "Millions of Instructions Per Second" has nothing to do with the CPU architecture also called "MIPS", which actually stands for "Microprocessor without Interlocked Pipeline Stages".

This fact that a major CPU architecture shares a name with an important metric for performance is incredibly confusing and dumb, and I apologize.

I blame the cocaine-fueled CPU architects of the 1980s.

$$\underset{\textbf{(IPC)}}{\frac{\text{Instructions}}{\text{Cycle}}} \times \underset{\textbf{(Frequency)}}{\frac{\text{Cycles}}{\text{Second}}} = \underset{\textbf{(Throughput)}}{\frac{\text{Instructions}}{\text{Second}}}$$

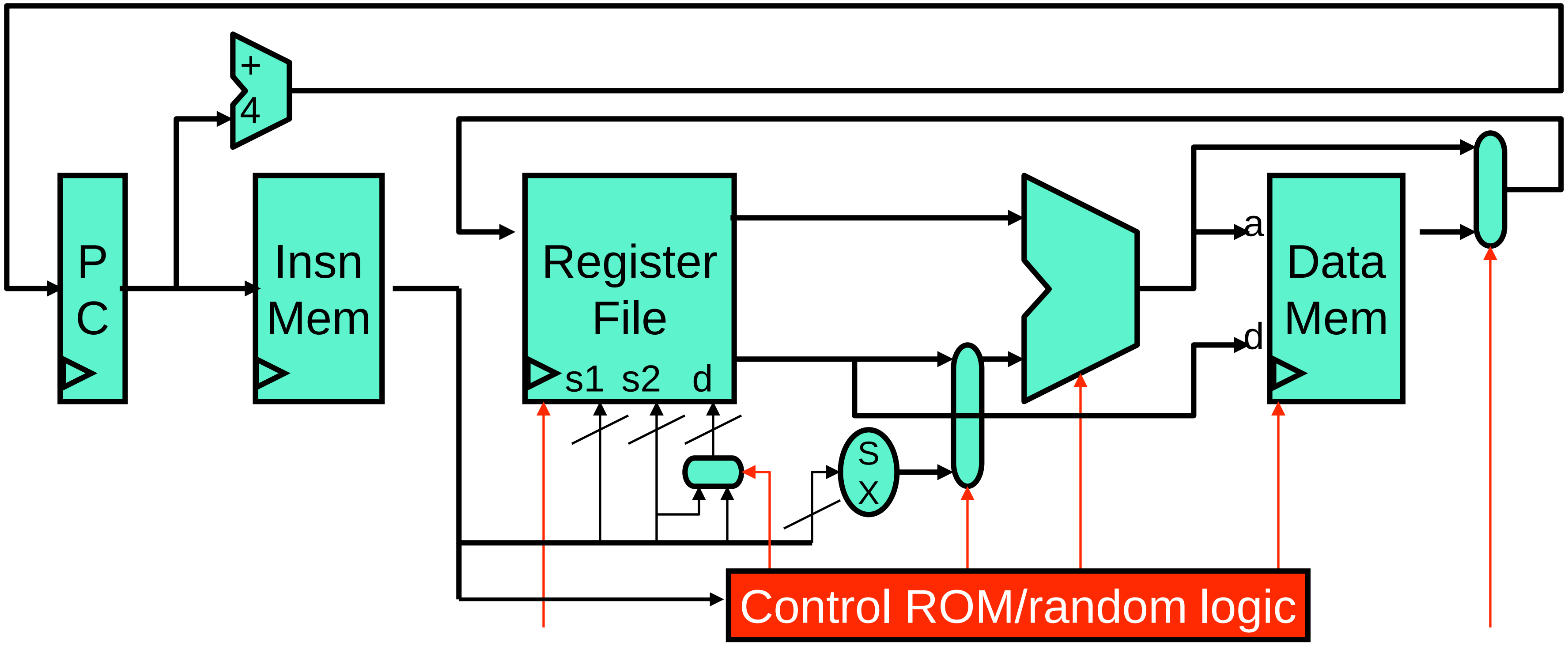
Performance vs ….

- 1990s: Performance at all cost
  - Actually more "clock frequency" at all cost…
- Now: Care about other things
  - **Energy** (electric bill, battery life)
  - **Power** (cooling, also affects energy)
  - Area (chip cost)
  - Reliability (tolerance of transient faults: e.g., charge particle strikes)
  - …
- Important metric these days "Performance / Watt"
  - Throughput divided by power consumption
  - Why?

Performance Modeling and Analysis

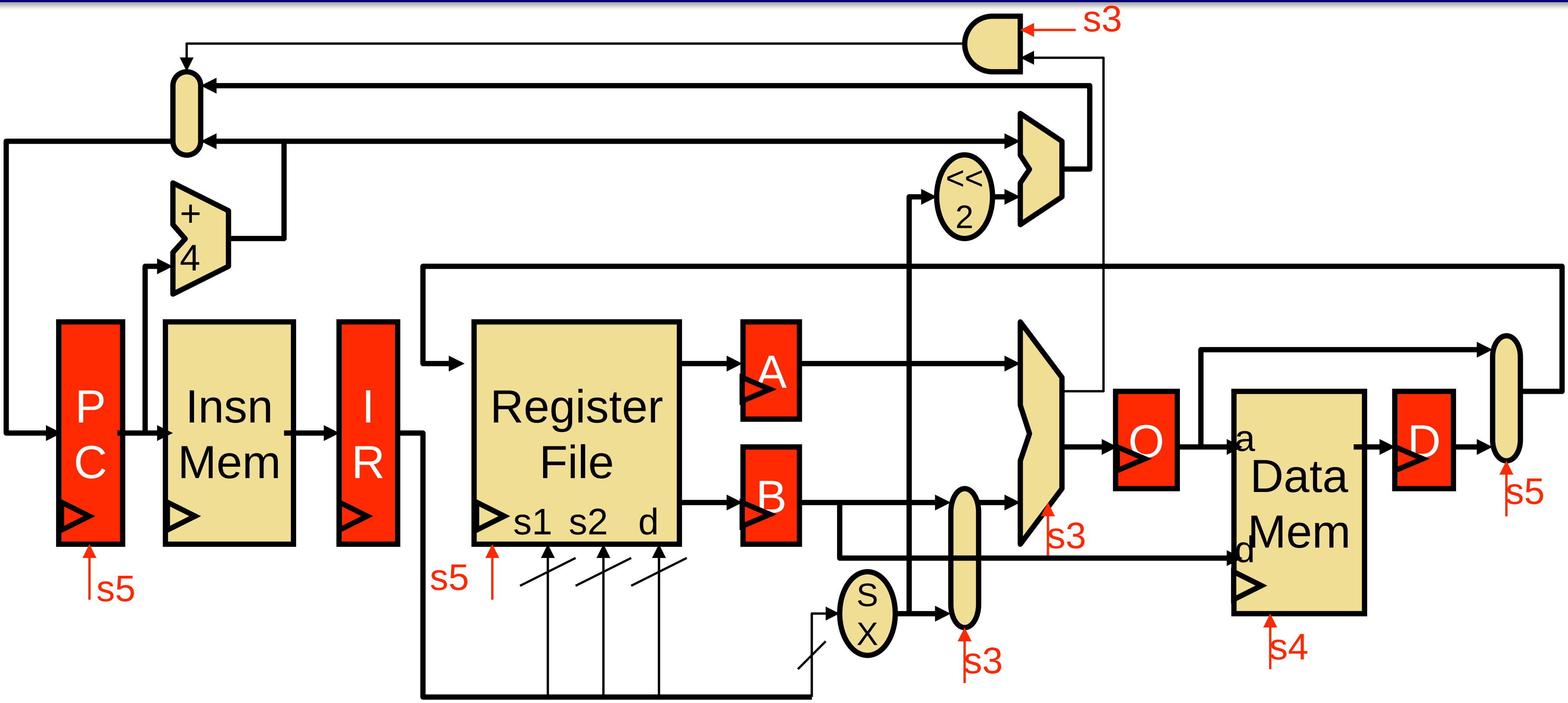
- Speaking of performance
  - Making a processor takes time (years) and money (millions)
  - Want to know it will perform well before you finish
    - If its wrong, doing it all over is painful…
  - Performance can be simulated in software
    - Estimate what IPC will be
    - Guide design

Single-Cycle Datapath Performance



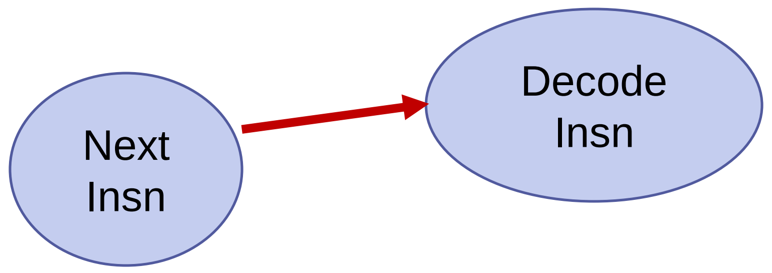
- Observations
  - + Low Cycles Per Instruction (**CPI**): 1
  - – Long clock period: to accommodate slowest insn

Alternative: Multi-Cycle Datapath



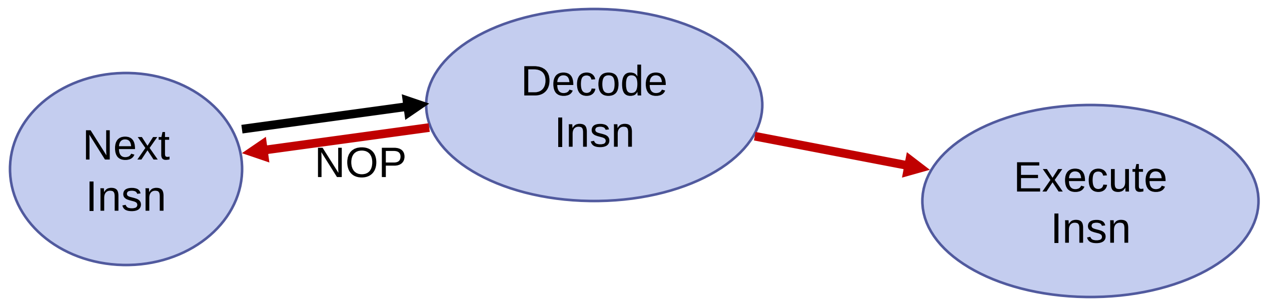
- **Multi-cycle datapath**: attacks high clock period
  - Cut datapath into multiple stages (5 here), isolate using FFs
  - **FSM** control "walks" insns thru stages (by staging control signals)
  - + Insns can bypass stages and exit early

Multi-cycle Datapath FSM



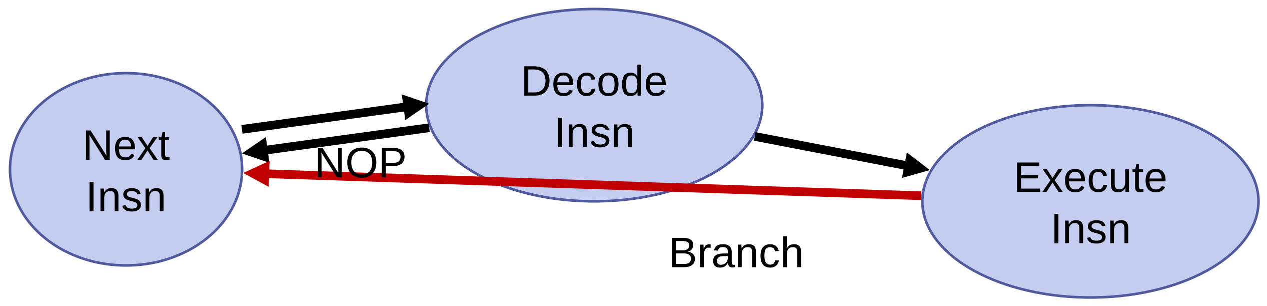
- First state: Get a New Instruction
  - Output signals to fetch (e.g., read enable IMEM)
  - Next State: Always Decode

Multi-cycle Datapath FSM



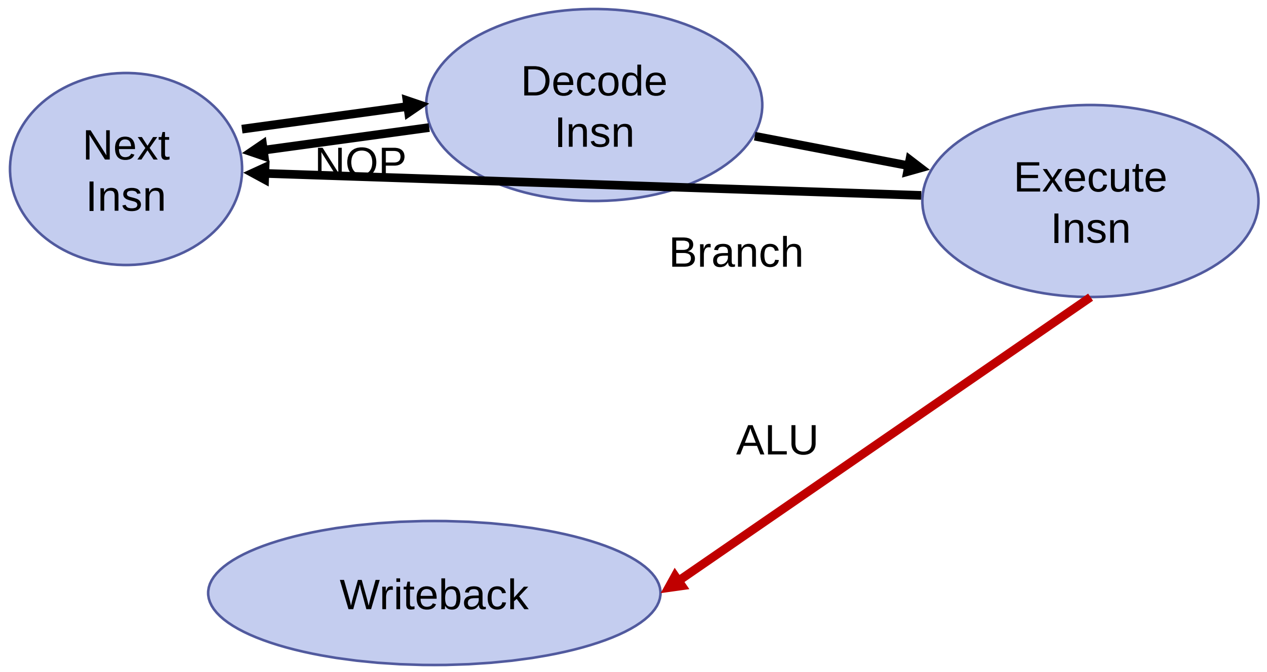
- Second State: Decode
  - Output signals to decode instruction (RdEn RegFile)
  - Go to Next Insn if NOP
  - Otherwise Execute

Multi-cycle Datapath FSM



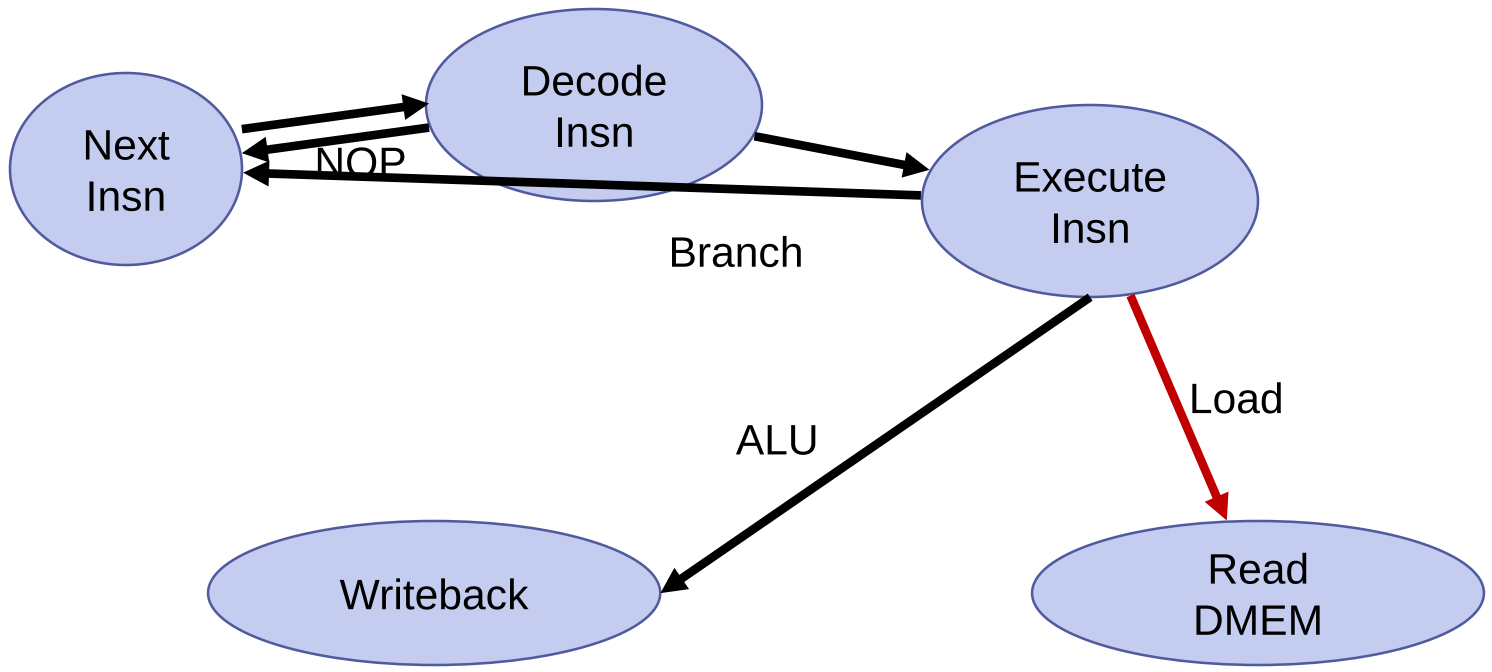
- Execute State
    - Execute Insn (varies by insn type)
    - Next State: Also depends on insn type
        - Branches: Next Insn

Multi-cycle Datapath FSM



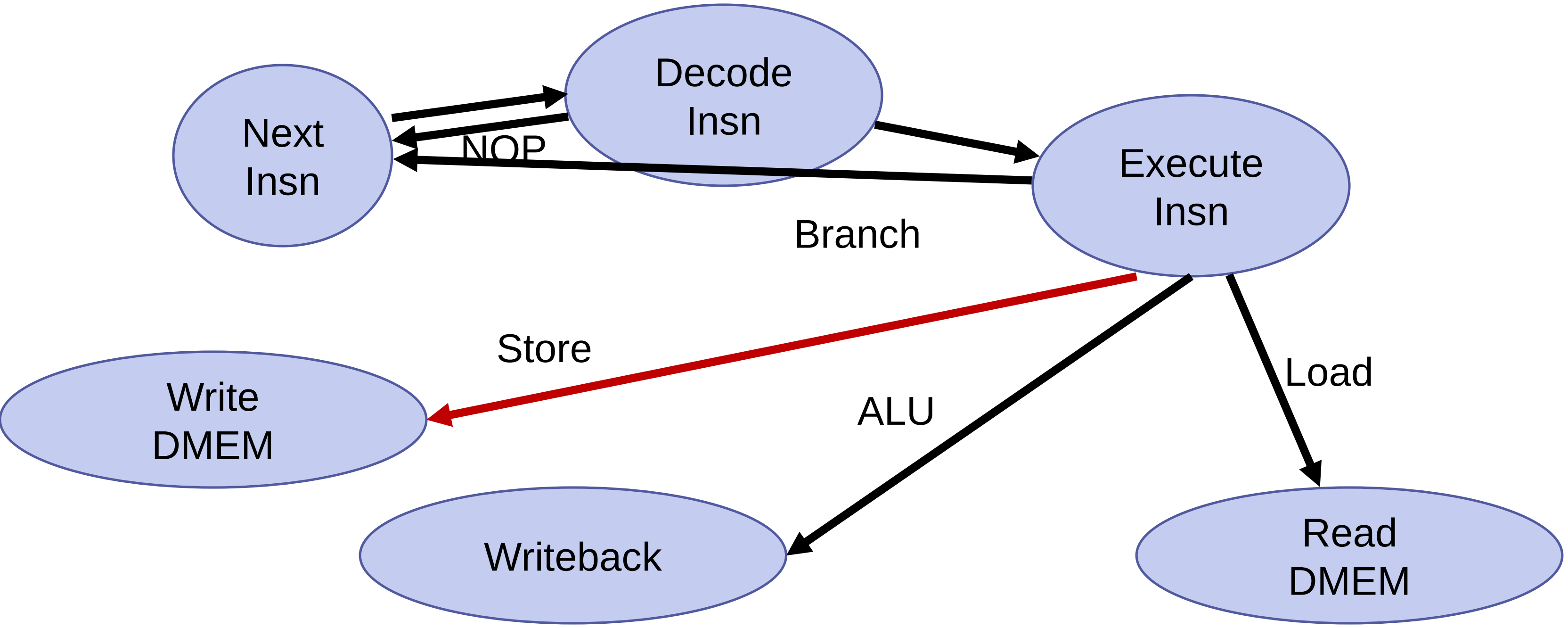
- Execute State
    - Execute Insn (varies by insn type)
    - Next State: Also depends on insn type
        - ALU op: write register

Multi-cycle Datapath FSM



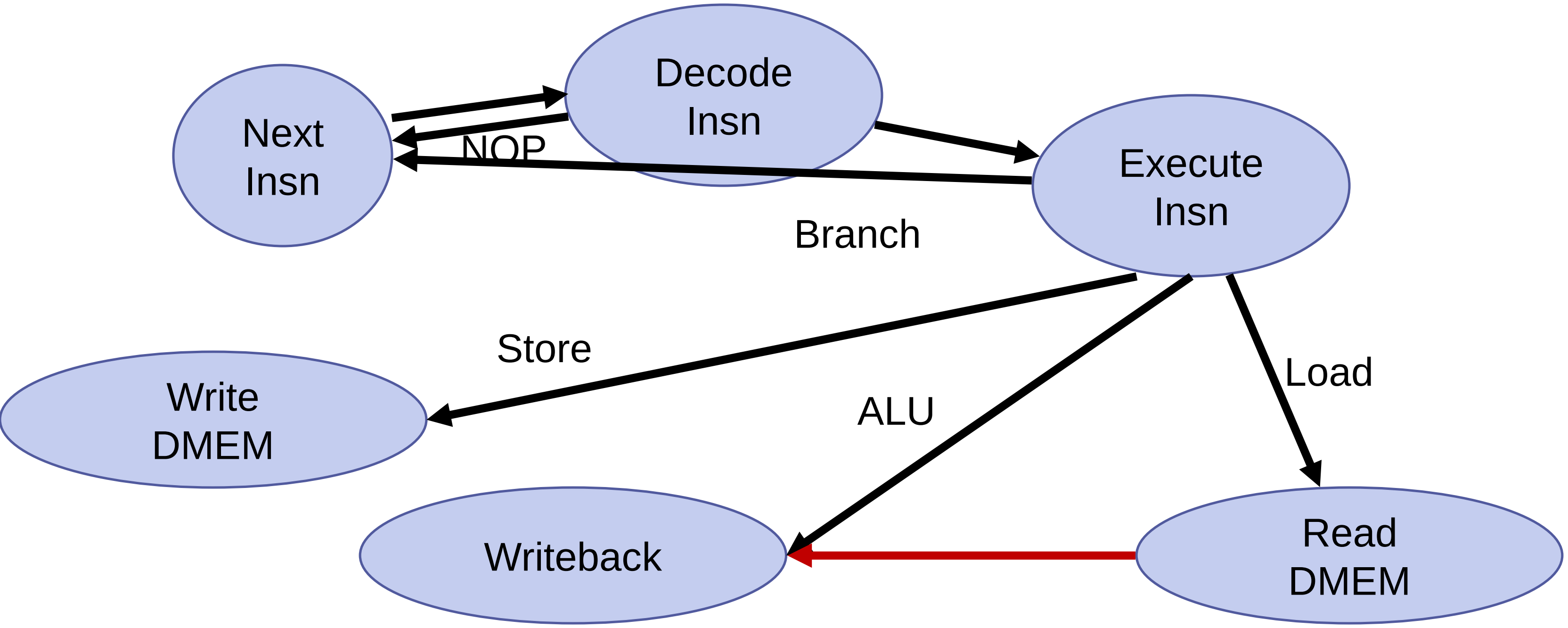
- Execute State
  - Execute Insn (varies by insn type)
  - Next State: Also depends on insn type
    - Load: Read Memory

Multi-cycle Datapath FSM



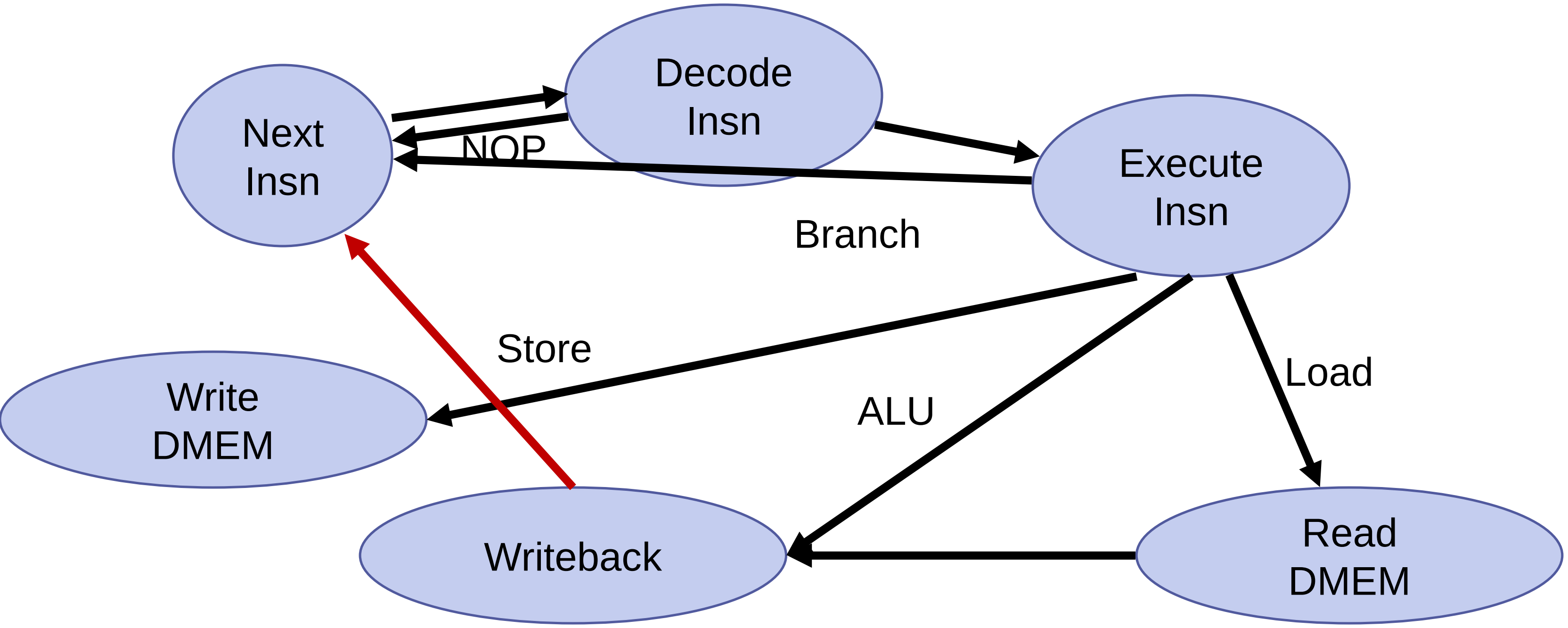
- Execute State
    - Execute Insn (varies by insn type)
    - Next State: Also depends on insn type
        - Store: Write Memory

Multi-cycle Datapath FSM



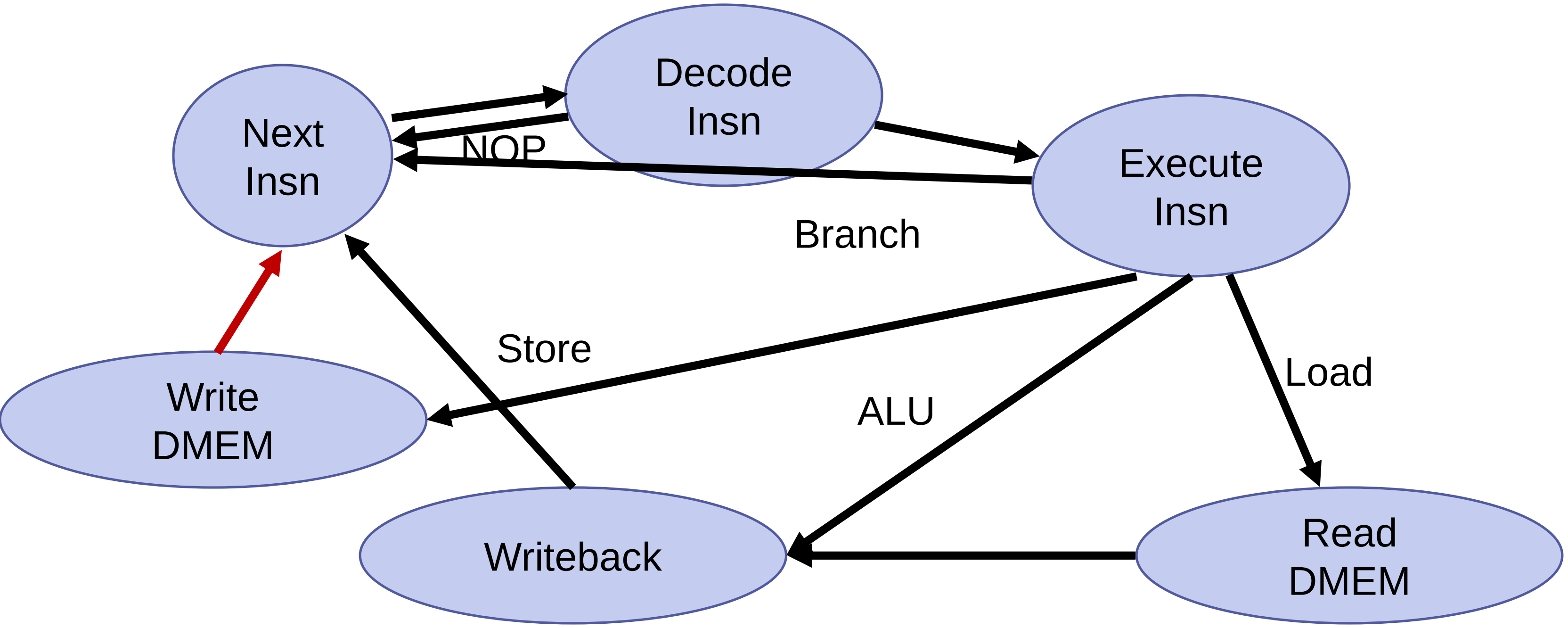
- Read DMEM State
  - Control signals enable DMEM Read
  - Next state is writeback

Multi-cycle Datapath FSM



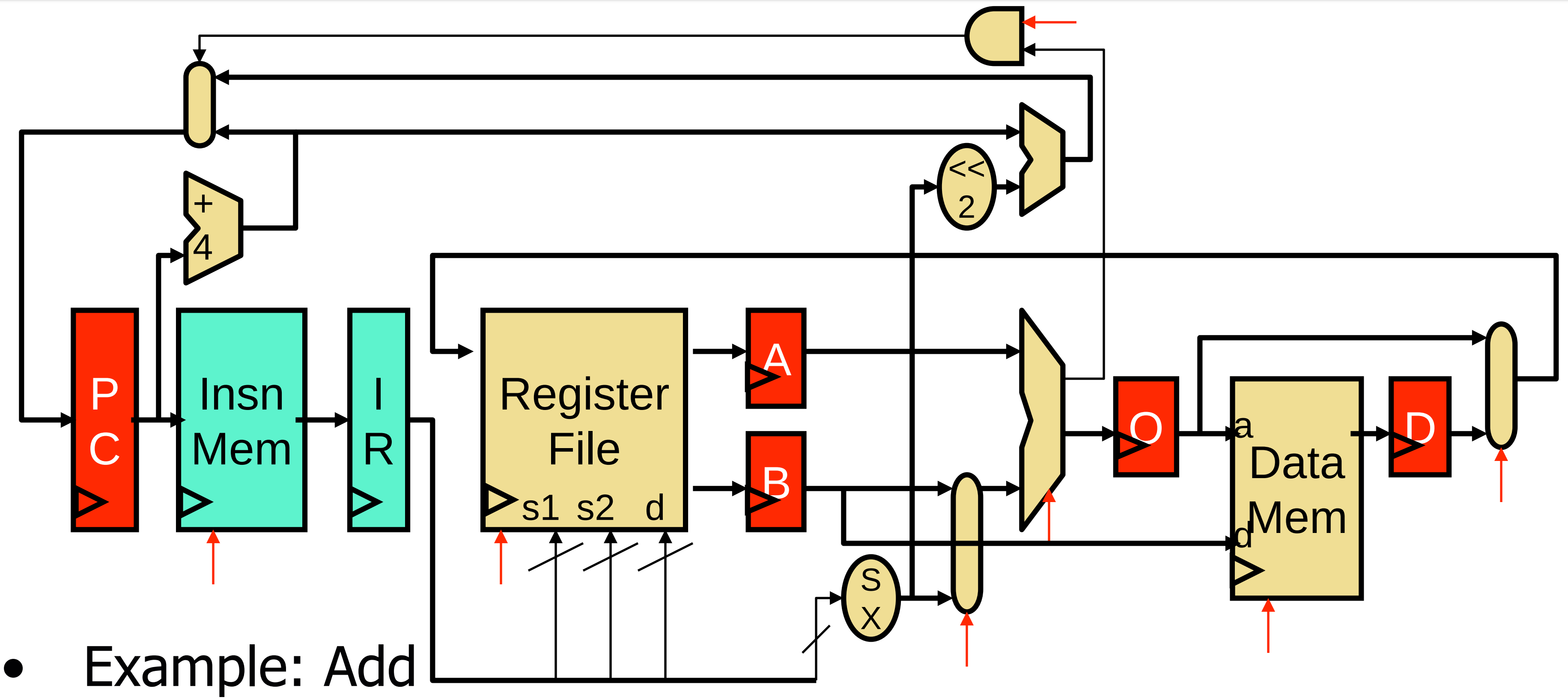
- Writeback state
  - Control signals enable regfile write
  - Next state: Next Insn

Multi-cycle Datapath FSM



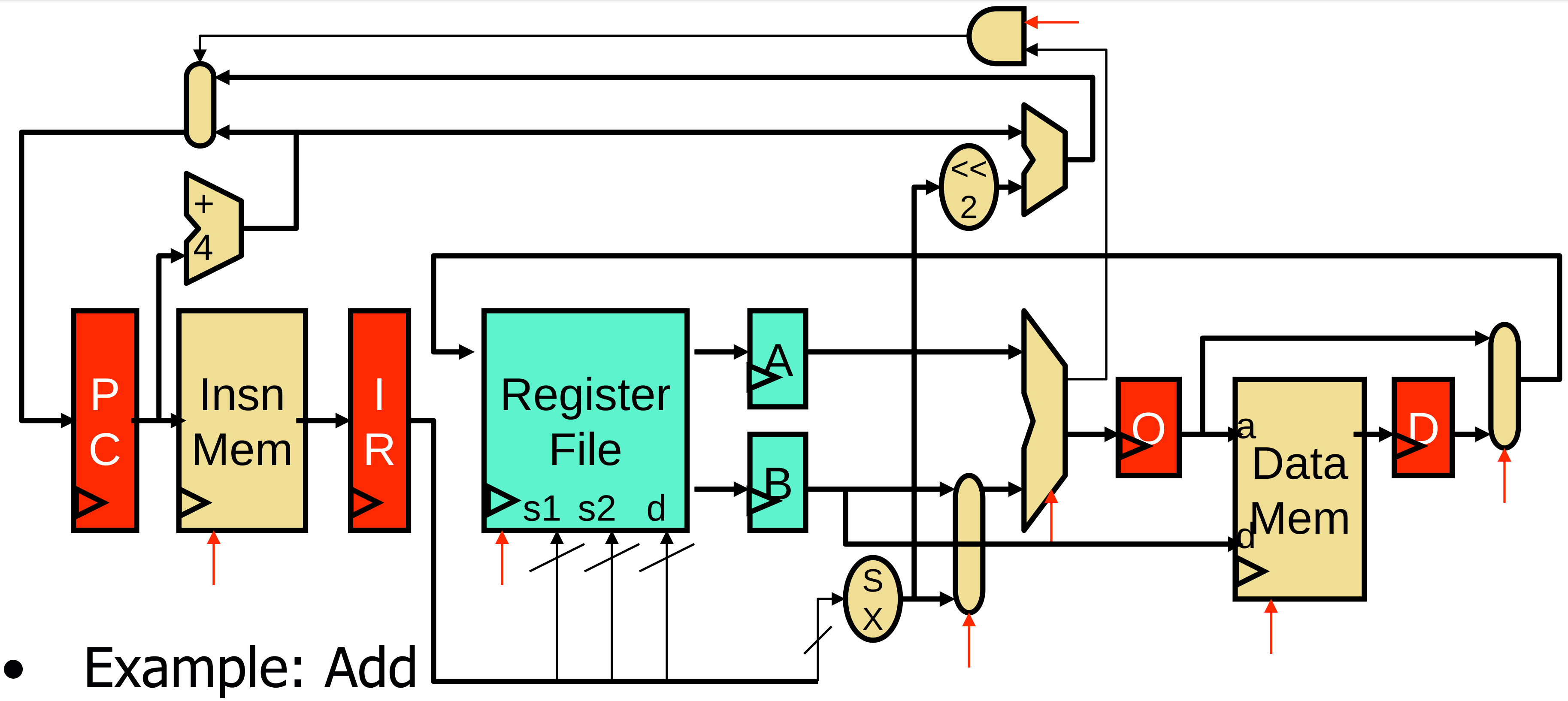
- Write DMEM state
  - Control signals enable memory write
  - Next state: Next Insn

Multi-Cycle Datapath Example: Add



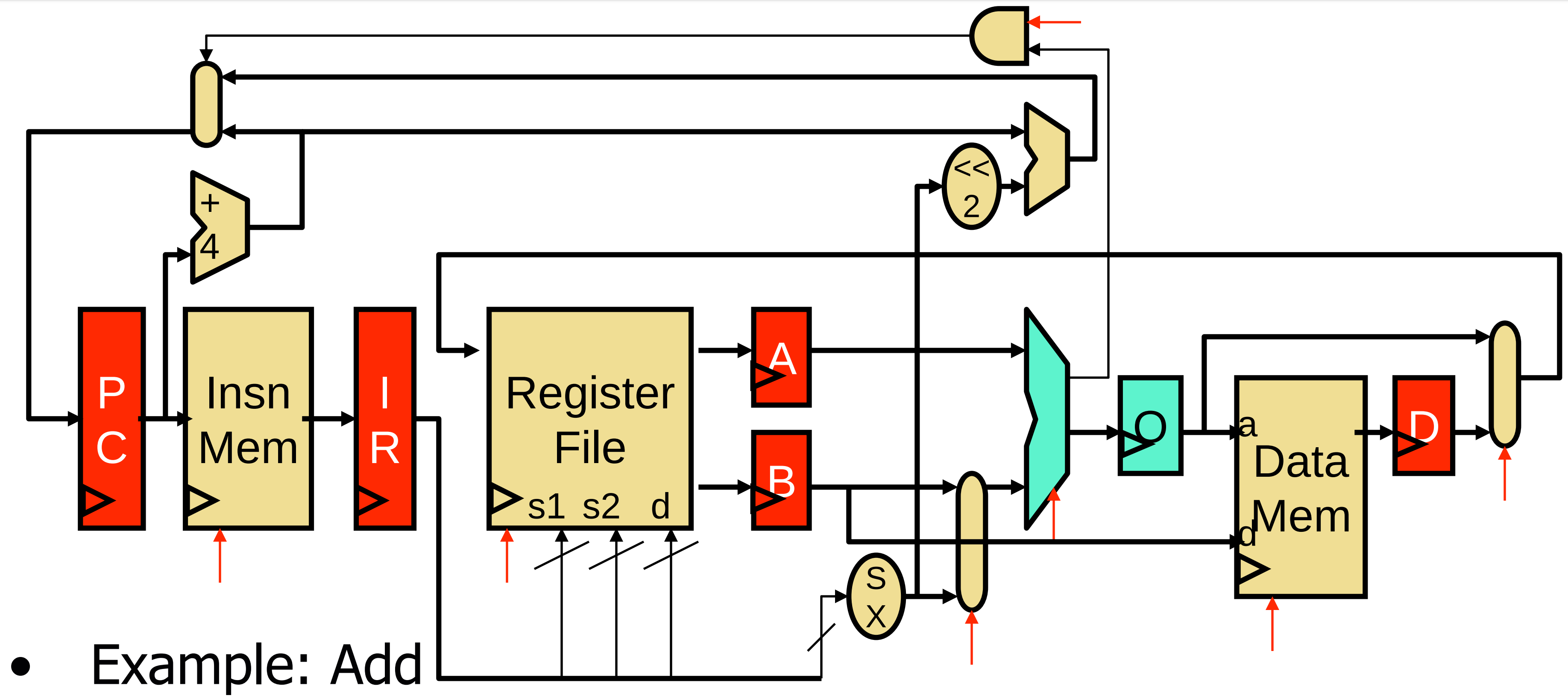
- Example: Add
  - Cycle 1: Read IMEM

Multi-Cycle Datapath Example: Add



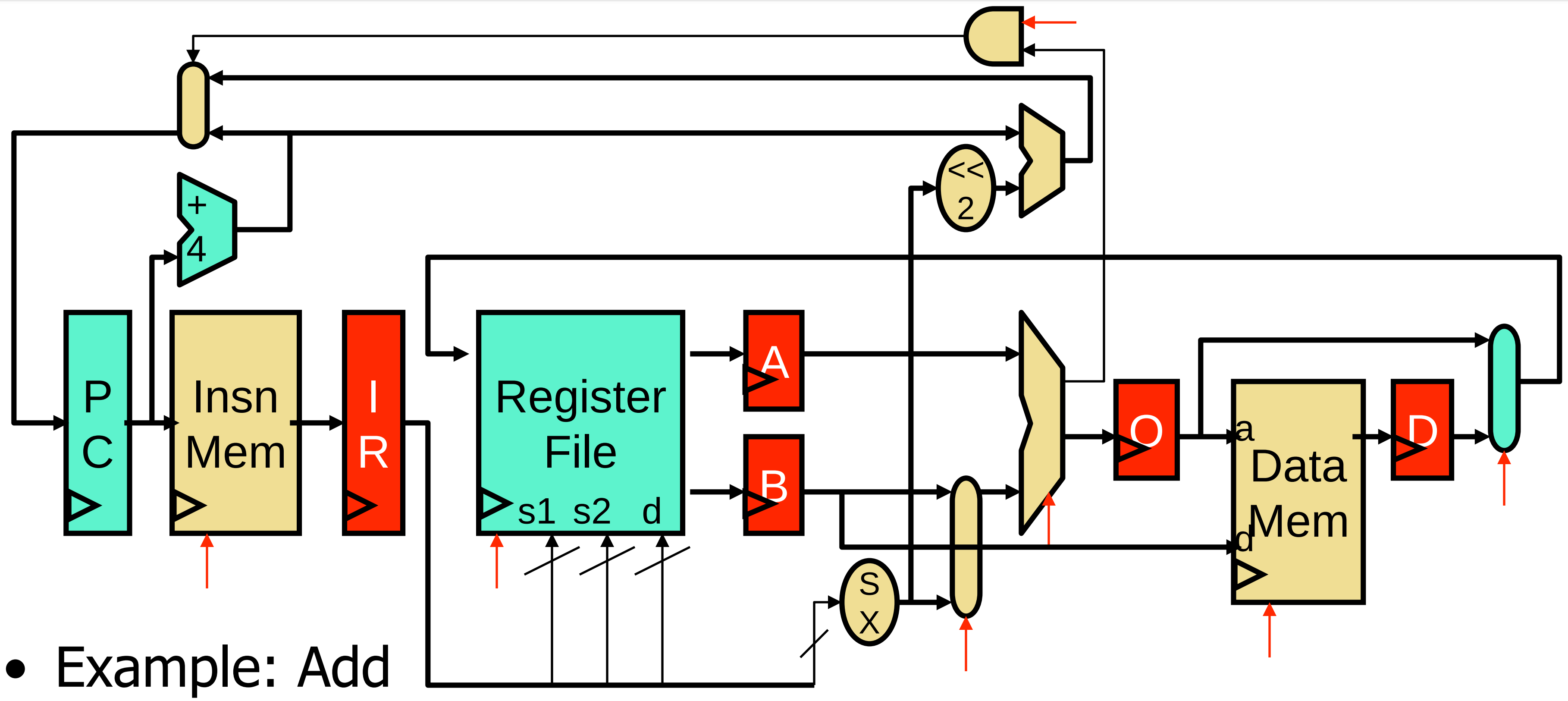
- Example: Add
  - Cycle 1: Read IMEM
  - Cycle 2: Decode + Read RF

Multi-Cycle Datapath Example: Add



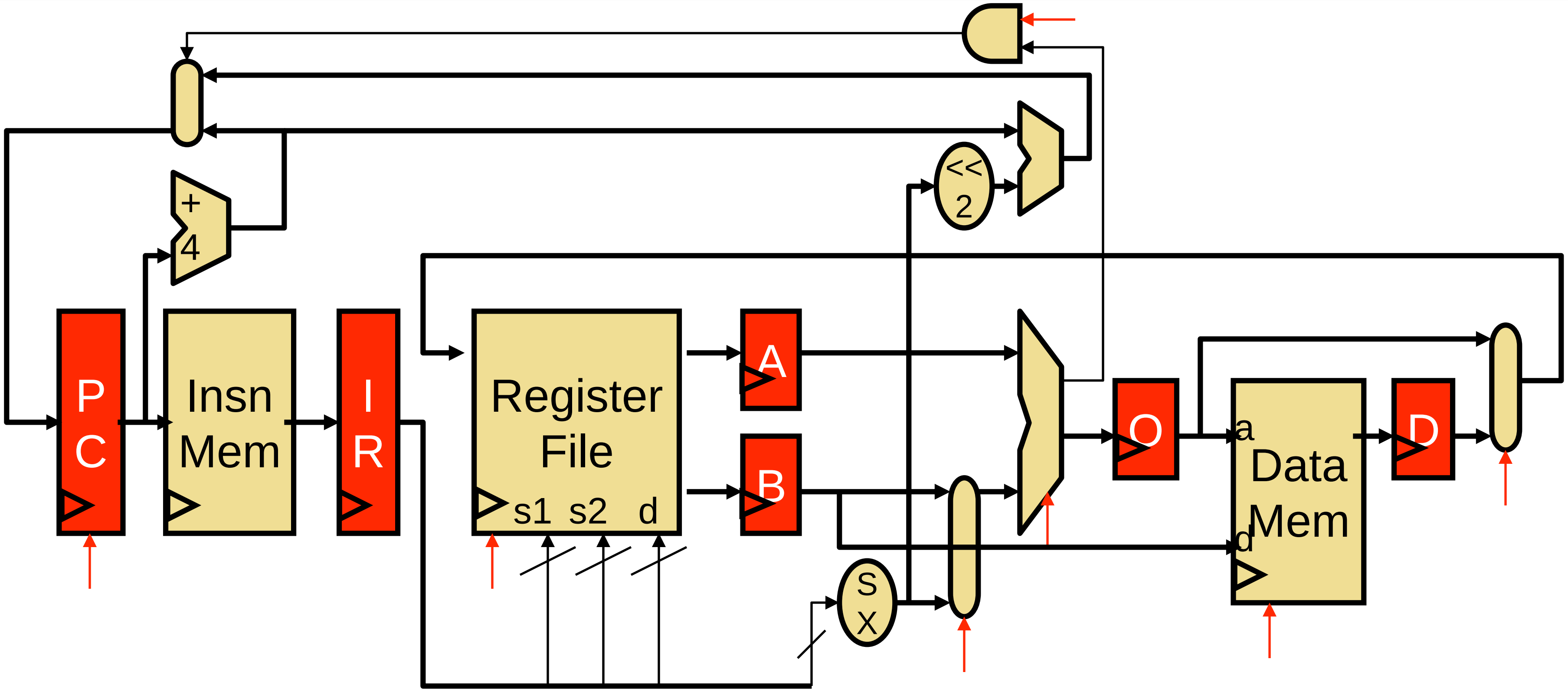
- Example: Add
  - Cycle 1: Read IMEM
  - Cycle 2: Decode + Read RF
  - Cycle 3: ALU

Multi-Cycle Datapath Example: Add



- Example: Add
  - Cycle 1: Read IMEM
  - Cycle 2: Decode + Read RF
  - Cycle 3: ALU
  - Cycle 4: Writeback + Increment PC

Multi-Cycle Datapath Performance



- Opposite performance split of single-cycle datapath
  - + Short clock period
  - – **High CPI**

Multi-cycle Data-path CPI

- CPI depends on instructions
  - Branches / Jumps: 3 cycles
  - ALUs: 4 cycles
  - Stores: 4 cycles
  - Loads: 5 cycles

- Overall CPI is weighted average

- Example:
  - 20% loads, 15% stores, 20% branches, 45% ALU

Multi-cycle Data-path CPI

- CPI depends on instructions
  - Branches / Jumps: 3 cycles
  - ALU: 4 cycles
  - Stores: 4 cycles
  - **Loads: 5 cycles**

- Overall CPI is weighted average

- Example:
  - **20% loads**, 15% stores, 20% branches, 45% ALU

CPI= 0.20 * 5 +

Multi-cycle Data-path CPI

- CPI depends on instructions
  - Branches / Jumps: 3 cycles
  - ALU: 4 cycles
  - **Stores: 4 cycles**
  - Loads: 5 cycles

- Overall CPI is weighted average

- Example:
  - 20% loads, **15% stores**, 20% branches, 45% ALU

CPI= 0.20 * 5 + 0.15 * 4 +

Multi-cycle Data-path CPI

- CPI depends on instructions
  - Branches / Jumps: 3 cycles
  - ALU: 4 cycles
  - Stores: 4 cycles
  - Loads: 5 cycles

- Overall CPI is weighted average

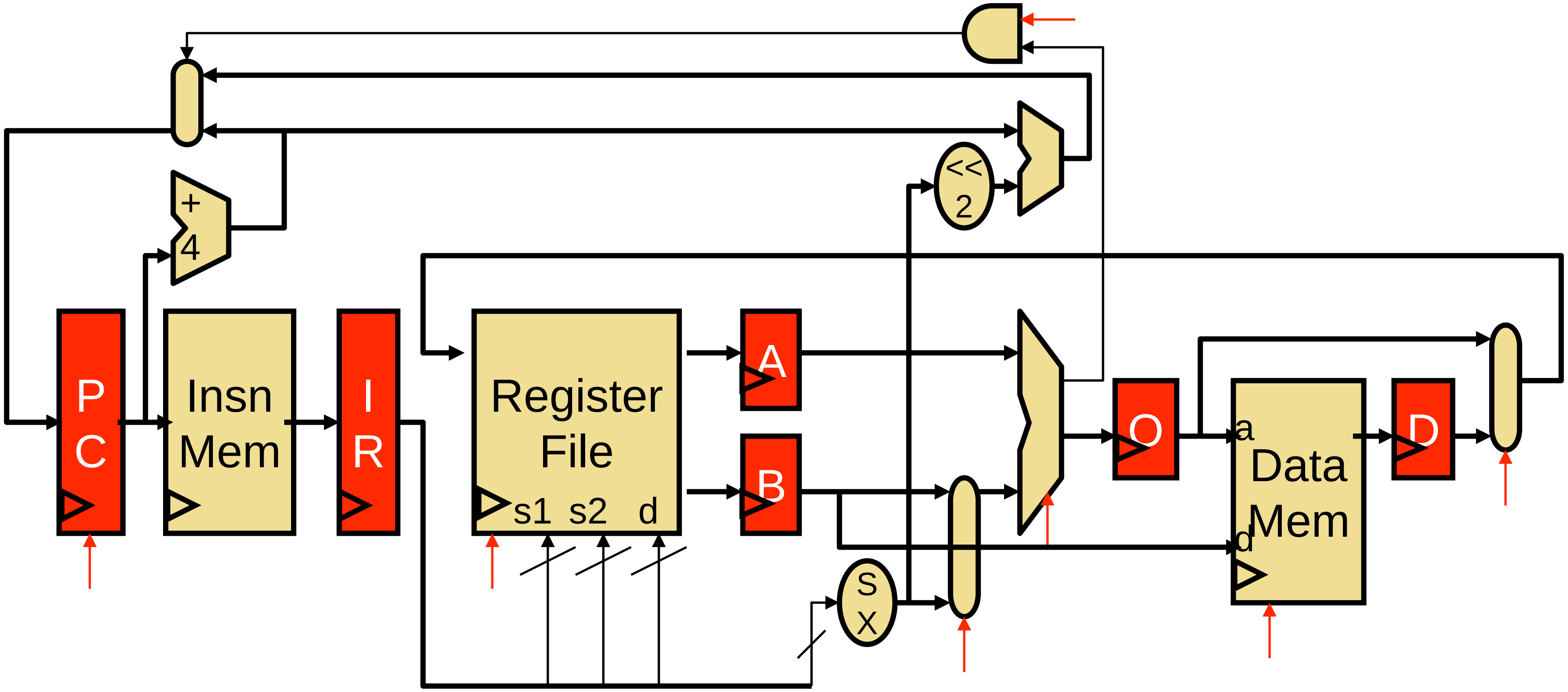
- Example:
  - 20% loads, 15% stores, 20% branches, 45% ALU

CPI= 0.20 * 5 + 0.15 * 4 + 0.20 * 3 + 0.45 * 4 = **4.0**

Multi-cycle Datapath Performance

- Single-cycle
  - Clock period = 50ns, CPI = 1
  - Performace = 50 ns/insn
- Multi-cycle
  - Clock period = 10ns
  - CPI = (0.2*3+0.2*5+0.6*4) = 4
  - Performance = 40 ns/insn
- But wait…

Multi-Cycle Datapath Performance



- Did not just cut up existing logic into 5 pieces
- Also added logic (flip flops)
  - So clock period not 1/5 of single cycle, but slightly longer

Multi-cycle Datapath Performance

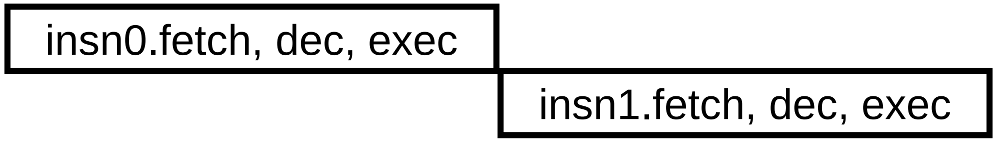
- Single-cycle
  - Clock period = 50ns, CPI = 1
  - Performace = 50 ns/insn

- Multi-cycle
  - Clock period = **12ns**
  - CPI = (0.2*3+0.2*5+0.6*4) = 4
  - Performance = **48 ns/insn**

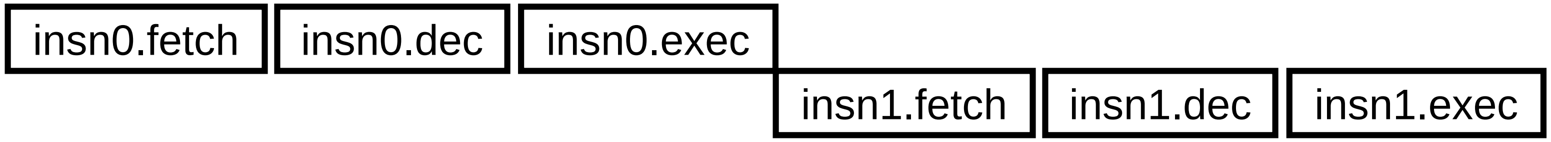
- Better, but not as exciting…
  - Can we do better still?
  - Have our cake (low CPI) and eat it too (high clock frequency)?

Clock Period and CPI

- Single-cycle datapath
  - + Low CPI: 1
  - – Long clock period: to accommodate slowest insn



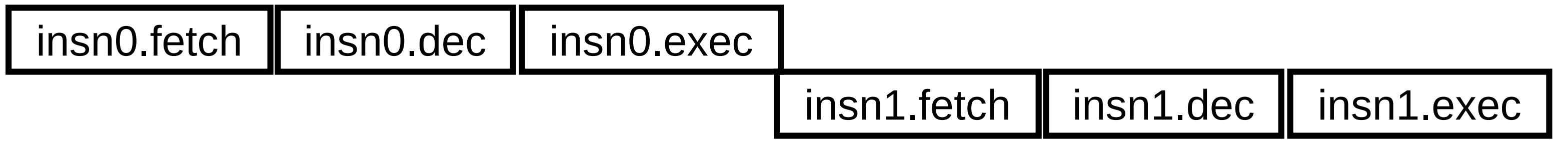
- Multi-cycle datapath
  - + Short clock period
  - – High CPI



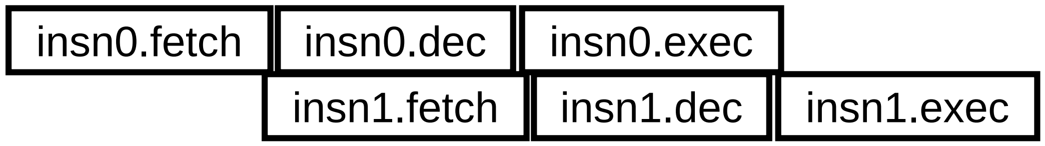
- Can we have both low CPI and short clock period?
  - – No good way to make a single insn go faster
  - + Insn latency doesn't matter anyway … insn throughput matters
  - Key: **exploit inter-insn parallelism**

Pipelining

- **Pipelining**: important performance technique
  - **Improves insn throughput rather than insn latency**
  - **Exploits parallelism at insn-stage level to do so**
  - Begin with multi-cycle design



  - When insn advances from stage 1 to 2, next insn enters stage 1



  - Individual insns take same number of stages
  + **But insns enter and leave at a much faster rate**
  - Physically breaks "atomic" VN loop ... but must maintain illusion
- Revisit soon

Summary

- Datapaths:
  - Multi-cycle
    - Faster clock (yay!)
    - Worse CPI (boooo)
  - Performance:
    - IPC
    - Performance / Watt
    - CPU Performance Equation
  - Pipelining
    - Teaser for later!