Pipelining to increase CPU performance
Multi-cycle Restaurant

Everyone spends 15 seconds at each station that she orders from.
Pipelined Restaurant

Everyone spends 15 seconds at each station that she orders from, but each station can serve a different customer.
Pipelined Datapath

• Just like with multi-cycle implementation there are stages,
• but each stage executes concurrently.
• A new instruction begins execution every clock cycle.
Pipeline concept

• Break up instruction into tasks
• Balance the amount of work (time) between stages
• Allow each segment to complete and start next instruction
Properties of Pipelines

• Latency: the time it takes for a single instruction to execute. Pipelining makes latency slightly worse.
• Throughput: number of instructions executed per unit time. Pipelining improves throughput.
• Five stages in classical pipeline: IF, ID, EX, MEM, WB. Just like our multicycle pipeline.
• Clock is constrained by slowest stage of pipeline.
• Pipelining isn’t free: complexities in design and additional resources (more later).
A MIPS pipeline

The “pink” registers hold data between stages
Pipeline Performance

• If all instructions took N Cycles, and
• each of N pipeline stages did 1/Nth of the work, and
• the clock speed didn’t change,
• the pipeline implementation’s “throughput” would be N times faster than the multi-cycle implementation.
• Because an instruction would be finished every clock cycle.
MIPS ISA and Pipelining

- All instructions are the same length
- Few instruction formats (and similar as possible)
- Memory operands only in load and store and simple addressing mode
- Operands are aligned in memory
Processor Performance

- It takes 1 clock cycle to execute each of 5 stages of executing an instruction (IF, ID, EX, MEM, WB) and a 1 GigaHertz (1 ns period) clock

- 95% cache hit rate no pipelining
  - $5 \text{ c/instruction} \times 0.95 + 45 \text{ c/instruction} \times 0.05 = (4.75 + 2.25) \text{c/instruction}$ on average.
  - $10^9 \text{ c/s} / 7 \text{ c/instruct.} = 143 \text{ MIPS}$

- Traditional Pipeline
  - $1 \text{ c/inst} \times 0.95 + 40 \text{ c/inst} \times 0.05 = (0.95 + 2.0) \text{c/inst} = 2.95$ cycles/inst on average.
  - $10^9 \text{ c/s} / 2.95 \text{ c/instruct.} = 339 \text{ MIPS}$
  - $339/143 = \text{almost 2.4X}$

- With imperfections $1.2 \times 0.95 + 40 \times 0.05 = 318 \text{ MIPS}$
Why Separate Data and Instruction Caches

• So that data and an instruction can be read in one clock cycle allowing the pipeline to do fetch another instruction when doing a load or store.