Computer Architecture (Spring 2020)

Pipelining

Dr. Duo Liu (刘铎)
Office: Main Building 0626
Email: liuduo@cqu.edu.cn

Can We Do Better?

- What limitations do you see with the multi-cycle design?

- Limited concurrency
  - Some hardware resources are idle during different phases of instruction processing cycle
  - "Fetch" logic is idle when an instruction is being "decoded" or "executed"
  - Most of the datapath is idle when a memory access is happening

Can We Use the Idle Hardware to Improve Concurrency?

- Goal: Concurrency → throughput (more "work" completed in one cycle)

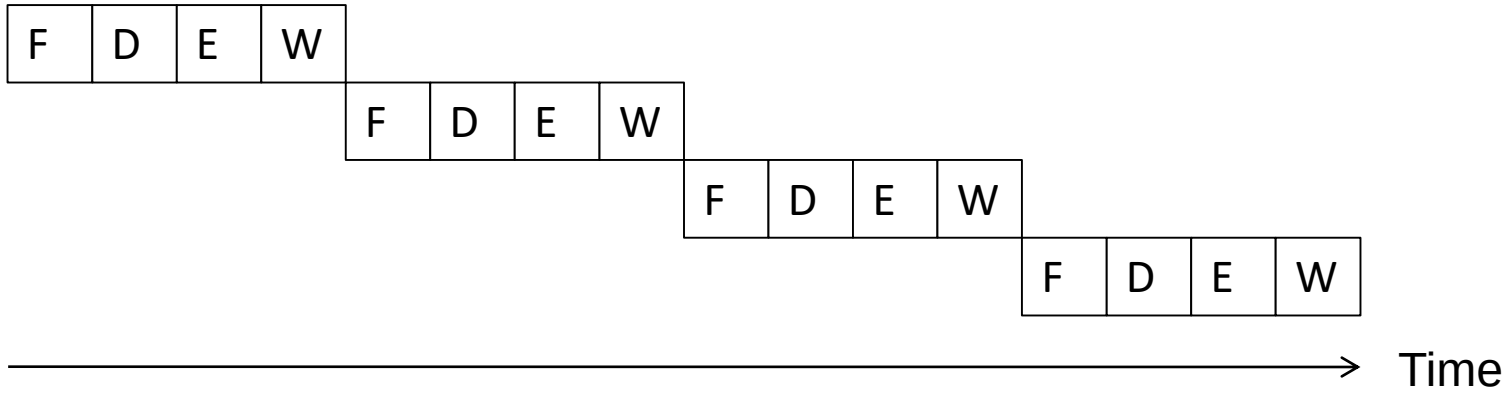
- Idea: When an instruction is using some resources in its processing phase, process other instructions on idle resources not needed by that instruction
  - E.g., when an instruction is being decoded, fetch the next instruction
  - E.g., when an instruction is being executed, decode another instruction
  - E.g., when an instruction is accessing data memory (ld/st), execute the next instruction
  - E.g., when an instruction is writing its result into the register file, access data memory for the next instruction

Pipelining: Basic Idea

- More systematically:
  - Pipeline the execution of multiple instructions
  - Analogy: "Assembly line processing" of instructions
- Idea:
  - Divide the instruction processing cycle into distinct "stages" of processing
  - Ensure there are enough hardware resources to process one instruction in each stage
  - Process a different instruction in each stage
    - Instructions consecutive in program order are processed in consecutive stages
- Benefit: Increases instruction processing throughput (1/CPI)
- Downside: Start thinking about this…

Example: Execution of Four Independent ADDs

- Multi-cycle: 4 cycles per instruction



- Pipelined: 4 cycles per 4 instructions (steady state)