Pipelining

Fundamentals of Pipeline Math

- Time for a single stage = `T`
- Time for `S` stages = `ST` (simple enough...)
- Time *between* initiations = `T` (not `ST` as in the multi-cycle approach that we've discussed so far – i.e. where we do 1 instruction right after the next.)
- Idea: Improve **throughput** ... or how often something comes out of the datapath. For example:
  - Assume we have a non-pipelined, multi-cycle datapath where each instruction takes 4 CCs on average. Assume our clock rate is x. How many seconds does it take to finish N instructions?
    - N Instructions $\times \frac{4CCs}{Instruction} \times \frac{xs}{CC}$ = $4N$
  - Now, with a pipelined version, we would have:
    - N Instructions $\times \frac{1CCs}{Instruction} \times \frac{xs}{CC}$ = N

How long will it take or N initiations to finish?

- $N(T) + (S-1)(T)$
  - The `N(T)` part refers to the fact that something finishes every 'T' time units. If there are 'N' items in the pipeline, `N(T)` is one component of the execution time.
  - The `(S-1)(T)` part refers to the fact that we need to fill up the pipeline. I.e. if there are S stages, than (S-1) time units will be spent filling up those stages. No "useful work" will be output during this time. **(see simple trace with stages in row and time in column)**
- Example:
  - 4 loads, 4 stages, 40 minutes/stage (i.e. laundry!)
  - Pipelined: (4)(40) + (4-1)(40) = 160 + 120 = 280
  - Nonpipelined: (4)(4)(40) = 640!

Throughput

- Can divide $N(T) + (S-1)(T)$ by N: $\rightarrow \frac{N(T)+(S-1)(T)}{N} \rightarrow$ As N gets large, equation goes to T. Thus, time per initiation is T.
- As N gets large, the component on the right trends toward 0 and the NT component dominates. Ideally, you see a result produced every (shorted) clock cycle.
- Thus, despite the slight increase in overhead, the pipelined version produces results faster.

Speedup

- Ideally, the speedup one sees is equal to the number of pipe stages...

- Assume/recall that with the multi-cycle approach, we tried to balance the amount of work per cycle. We try to do the same thing here by balancing the amount of work per stage.
- Assume that each cycle takes τ time units. If there are 4 steps, then the total time spent is 4τ.
- The time for 1000 pieces of data/instruction is thus $4\tau \times 1000$.
- Now, what about the time for a pipelined version?
- We can actually use the forumla: $NT + (S-1)T$. Thus, we get: $(1000)\tau + (4-1)\tau = 1003\tau$.
- Therefore, the speedup is essentially equal to the number of stages: $\frac{4000\tau}{1003\tau}$ is essentially 4 (the number of stages).

- (more in the slides)