CSE 120
Day 8 Notes

Elijah Hantman

Implementing Logic for Control

Consider the hypothetical

Memwrite is 1 if instruction 0 and 2 and not 5.

Build using combinatorial logic.

A control unit will consume an instruction and output a signal for which logic needs to execute.

Datapath Delay

Assume time for stages is 100ps for register read or write

200ps for other stages

The Critical path is the longest delay path the minimum clock period is determined by critical path.

If the critical path is 800ps, the maximum clock frequency is $\frac{1}{800ps}$

Single Cycle processor

Pros

- Single cycle makes hardware simpl

Cons

- Cycle time is set by worst case
- Hardware is underutilized
  ALU and memory only used for a fraction of a clock cycle
  Not well amortized
- Best possible CPI is 1
- Floating point instructions don't work

Pipelining

Analogy is doing laundry.

Washer takes 30mins

Dryer takes 40mins

Folding takes 20mins

Need to follow order wash $\rightarrow$ dry $\rightarrow$ fold for each laundry load

Each resource can only process one task at a time, however you can have each resource operate on a different task concurrently.

It is possible to complete four loads of laundry in 210 mins.

The potential speedup is proportaional to the number of pipeline stages, and the pipeline rate is limited by the slowest pipeline stage.

In future n is the number of instructions and k is the number of stages.

Pipelining only improves throughput, which means the speed of dependent or single instructions is unchanged.

Pipelining a type of parallelism, it is temporal parallelism, the same hardware resource is shared between different tasks across time, it is not directly subject to Little's law.

Pipelining is instruction level parallelism, ILP.

5 Stage RISC Pipeline

5 stages

- Instruction fetch
- instruction decode and register read
- execute operation or calculate address
- access memory operand
- write result back to register

Overlap instruction in different stages

- all hardware is used all the time
- Clock cycle is fast (common case is faster)

For a pipeline the execution time is

$$f(n,k) = kc + (n-1)c \tag{1}$$

Where c is the clock cycle period. This also assumes maximum pipeline utilization and no dependency chains which would disrupt instruction flow.

$$f(n,k) = k + (n-1)$$

Is the number of cycles required.

cool projects would be architecture simulators.

Speedup

$$\frac{unpipelined}{pipelined}$$

This will vary both on the number of hazards, k, n, and other issues.

Pipelining **only** improves throughput. Viewed from Iron law it can reduce cycle time but increase CPI.

How do you implement stages?

1. Seperate into discrete stages
2. Additional registers are used to hold data between stages.
3. Each pipeline register adds a one cycle delay

We need to delay register write backs until the last stage, so we need to pass the destination register through all the pipeline registers until we are ready to write it.

For control you can pipeline them just like data. Additional complexity but overall a difference of quantity not quality difficulty wise.

Data Hazards

A hazard is anything that prevents the next instruction from executing on the following clock cycle due to interdependence between the execution of consecutive instructions.

- Structural Hazards

  Two or more instructions in the pipeline require the same hardware resource at the same time.

  Comes up often with floating point units as well as architectures where instructions and data share a resource.

  Multiple pipeline paths are one of the reasons instructions might have different timings. In addition, it creates the opportunity for data hazards in the memory access and writeback stages.

  Processors which break the instruction order is known as Out of Order Processors. This only happens with multiple pipelines.