

Pipelining and Performance

(Ch 6, 2)

Processor Performance

- Exec. time = Instr * CPI * cycle time
- An example:
 - Instr. Class A : 1, B : 2, C : 5 cycles
 - code sequence 1:
 - A = 5; B = 3; C = 1 instructions executed
 - code sequence 2:
 - A = 3; B = 2; C = 2 instructions executed
 - which is faster?

Benchmarks

- How to compare performance of machines?
- What are benchmarks?
- How do you choose benchmarks?
- Small programs Vs. real programs
- Summarizing performance
 - total exec time of all benchmark programs
 - arithmetic mean Vs. geometric mean
 - weighted arithmetic mean
 - take into account relative frequency of programs in the workload mix

Sources of Perf. Improvement

- Increases in clock speed
- Organization leading to lower CPI
- Issuing multiple instructions every clock cycle (superscalar processor design)
- Compiler enhancements
- All of the above and more in later courses...

Passage of an Instruction

- IF: fetch an instruction
- ID: decode
- EX: execute
- what portion of datapath used for each state?
- how can we use all of the hardware in the datapath all the time?

An Example

- assume processor spends
 - 2 cycles to do IF, EX
 - 1 cycle to do ID
- Consider 3 successive load instructions
 - how much time to execute?
 - picture the execution
 - how can you make this faster?

Pipelining

- consider an auto assembly line
- apply to instruction execution
 - breakup datapath into “autonomous sections”
 - IF, DECODE&RR, EXEC, RW stages
 - what is each part of the datapath working on?
- how to keep them autonomous?
- what info is needed as instruction moves from one stage to the next?

Pipeline-Conscious Architecture

- need for symmetric instruction format
- need to make sure equal amount of work in each clock cycle
- things to worry about
 - structural hazards
 - control hazards
 - data hazards
- more in later courses....