Announcements

- Course web site: http://www.inf.ed.ac.uk/teaching/courses/car
  - Lecture slides
  - Tutorial problems
  - Courseworks

- Piazza discussion forum: http://piazza.com/ed.ac.uk/spring2018/car

- Tutorials start in week 3
Summary: Computer Architecture

What is Computer Architecture?

- ISA
- Microarchitecture
- Hardware implementation

Technology

Logic design

Operating System (OS)

Computer Architecture:
- ISA
- Microarchitecture
- Hardware implementation

Programming Languages

Compilers

Applications

Users

Metrics of interest for computer architects
- Performance
- Cost
- Reliability
- Power
- Area
Performance of Computer Architectures

- **Metrics:**
  - Execution time, response time: overall time for a given computation (e.g. full program execution, one transaction)
  - Latency: time to complete a given task (e.g. memory latency, I/O latency, instruction latency)
  - Bandwidth, throughput: rate of completion of tasks (e.g. memory bandwidth, transactions per second, MIPS, FLOPS)

- **MIPS, FLOPS:** must use with caution

- **Benchmarking:**
  - toy benchmarks, synthetic benchmarks, kernels, real programs,
  - Input sets
  - Standard benchmarking suites: SPEC INT/FP, EEMBC Coremark, CloudSuite
A is \( n \) times faster than \( B \) means

\[
\frac{\text{Execution time of } B}{\text{Execution time of } A} = n
\]

A is \( m\% \) faster than \( B \) means

\[
\frac{\text{Execution time of } B - \text{Execution time of } A}{\text{Execution time of } A} \times 100 = m
\]
Improving Performance: Principles of CA Design

- **Take advantage of parallelism**
  - System level: multiple processors, multiple disks
  - Processor level: operate on multiple instructions at once (e.g., pipelining, superscalar issue)
  - Circuit level: operate on multiple bits at once (e.g., carry-lookahead ALU)

- **Principle of locality**
  - Spatial and Temporal Locality
  - 90% of program executing in 10% of code
  - E.g. Caches

- **Focus on the common case**
  - Amdahl’s law, CPU Performance equation
  - E.g. RISC design principle
Amdahl’s Law

Speedup due to an enhancement $E$:

$$\text{Speedup}(S) = \frac{\text{Execution time before } E}{\text{Execution time after } E}$$

Suppose that enhancement $E$ accelerates a fraction $F$ of the task by a factor $S$, and the remainder of the task is unaffected. What is the Execution time after and Speedup($S$)?
Amdahl’s Law

\[ \text{Execution time}_{after} = \text{ExecTime}_{before} \times \left[ (1 - F) + \frac{F}{S} \right] \]

\[ \text{Speedup}(S) = \frac{\text{ExecTime}_{before}}{\text{ExecTime}_{after}} = \frac{1}{\left[ (1 - F) + \frac{F}{S} \right]} \]
Amdahl’s Law - Example

Floating point instructions contribute 10% to the program execution time and are improved to run 2x faster.

Q: What is the execution time and speedup after improvement?

Answer:

\[ F = 0.1, \quad S = 2 \]

\[ ExTime_{after} = ExTime_{before} \times \left[ (1 - 0.1) + \frac{0.1}{2} \right] = 0.95 \, ExTime_{before} \]

\[ Speedup(S) = \frac{ExTime_{before}}{ExTime_{after}} = \frac{1}{0.95} = 1.053 \]
Factors that affect CPU performance

Instruction count (IC)
  - Compiler, ISA

Cycles per instruction (CPI)
  - ISA, microarchitecture

Clock time ($1/f$)
  - Microarchitecture, technology

What determines these factors?
The CPU Performance Equation

\[
\text{CPU time} = IC \times CPI \times \text{Clock time}
\]

where:  
- CPU time = execution time
- IC = number of instructions executed (instruction count)
- CPI = number of average clock cycles per instruction
- Clock time = duration of processor clock

\[
\text{CPU time} = \left( \sum_{i=1}^{n} IC_i \times CPI_i \right) \times \text{Clock time}
\]

where:  
- IC<sub>i</sub> = IC for instruction (instruction group) \( i \)
- CPI<sub>i</sub> = CPI for instruction (instruction group) \( i \)

\[
\text{CPI} = \frac{\left( \sum_{i=1}^{n} IC_i \times CPI_i \right)}{IC} = \sum_{i=1}^{n} \left( \frac{CPI_i \times IC_i}{IC} \right)
\]
Examples

Example:

- Branch instructions take 2 cycles, all other instructions take 1 cycle
- CPU A uses extra compare instruction per branch
- Clock frequency of CPU A is 1.25 times faster than CPU B
- On CPU A, 20% of instructions are branches (thus other 20% are compare instructions)

Find the relative performance of CPUs A and B

\[
\begin{align*}
CPI_A &= \text{CPI}_{\text{branch}} \times \frac{\text{IC}_{\text{branch}}}{\text{IC}} + \text{CPI}_{\text{others}} \times \frac{\text{IC}_{\text{others}}}{\text{IC}} = 2 \times 0.2 + 1 \times 0.8 = 1.2 \\
CPI_B &= \text{CPI}_{\text{branch}} \times \frac{\text{IC}_{\text{branch}}}{\text{IC}} + \text{CPI}_{\text{others}} \times \frac{\text{IC}_{\text{others}}}{\text{IC}} = 2 \times 0.25 + 1 \times 0.75 = 1.25
\end{align*}
\]

CPU time _B_ = IC _B_ \times CPI _B_ \times Clock time _B_ = 0.8 \times IC _A_ \times 1.25 \times (1.25 \times \text{Clock time}_A) = 1.25 \times IC _A_ \times \text{Clock time}_A

CPU time _A_ = IC _A_ \times CPI _A_ \times Clock time _A_ = IC _A_ \times 1.2 \times \text{Clock time}_A

CPU A is faster!
Improving CPU Performance

• IC:
  • Compiler optimizations (constant folding, constant propagation)
  • ISA (More complex instructions)

• CPI:
  • Microarchitecture (Pipelining, Out-of-order execution, branch prediction)
  • Compiler (Instruction scheduling)
  • ISA (Simpler instructions)

• Clock period:
  • Technology (Smaller transistors)
  • ISA (Simple instructions that can be easily decoded)
  • Microarchitecture (Simple architecture)