Performance of computer systems

- Many different factors among which:
  - Technology
    - Raw speed of the circuits (clock, switching time)
    - Process technology (how many transistors on a chip)
  - Organization
    - What type of processor (e.g., RISC vs. CISC)
    - What type of memory hierarchy
    - What types of I/O devices
  - How many processors in the system
  - Software
    - O.S., compilers, database drivers etc.

Moore’s Law

Processor-Memory Performance Gap

What are some possible metrics

- Raw speed (peak performance = clock rate)
- Execution time (or response time): time to execute one (suite of) program from beginning to end.
  - Need benchmarks for integer dominated programs, scientific, graphical interfaces, multimedia tasks, desktop apps, utilities etc.
- Throughput (total amount of work in a given time)
  - Measures utilization of resources (good metric when many users: e.g., large database queries, Web servers)
  - Quite often improving execution time will improve throughput and vice-versa

Execution time Metric

- Execution time: inverse of performance
  \[ Performance = \frac{1}{(\text{Execution time})} \]
- Processor A is faster than Processor B
  \[ \text{Execution time}_A < \text{Execution time}_B \]
  \[ \text{Performance}_A > \text{Performance}_B \]
- Relative performance
  \[ \frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} \]

Measuring execution time

- Wall clock, response time, elapsed time
- Some systems have a “time” function
  - Unix 13.7u 23.6s 18:37 3% 209+1821k 13+240 62p+0u
- Difficult to make comparisons from one system to another because of too many factors
- Remainder of this lecture: CPU execution time
  - Of interest to microprocessors vendors and designers
Definition of CPU execution time

CPU execution time = CPU clock cycles * clock cycle time
• CPU clock cycles is program dependent thus CPU execution time is program dependent
• clock cycle time (nanoseconds, ns) depends on the particular processor
• clock cycle time = 1 / clock cycle rate (rate in MHz)
  – clock cycle time = 1ps, clock cycle rate = 1 MHz
  – clock cycle time = 1ns, clock cycle rate = 1 GHz
• Alternate definition
  CPU execution time = CPU clock cycles / clock cycle rate

CPI -- Cycles per instruction

• Definition: CPI average number of clock cycles per instr.
  CPU clock cycles = Number of instr. * CPI
  CPU exec time = Number of instr. * CPU clock cycles
  – Computer architects try to minimize CPI
    • or maximize its inverse IPC: number of instructions per cycle
  • CPI in isolation is not a measure of performance
    – program dependent, compiler dependent
    – but good for assessing architectural enhancements (experiments with same
      programs and compilers)
  • In an ideal pipelined processor (to be seen soon) CPI = 1
    – but… not ideal so CPI > 1
    – could have CPI < 1 if several instructions execute in parallel (superscalar
      processors)

Classes of instructions

• Some classes of instr. take longer to execute than others
  – e.g., floating-point operations take longer than integer operations
• Assign CPI’s per classes of instr., say CPI_i
  CPU exec_time = Σ(CPI_i * C_i) * clock cycle time
  where C_i is the number of instrs. of class i that have been executed
• Note that minimizing the number of instructions does not necessarily improve execution time
• Improving one part of the architecture can improve the CPI of one class of instructions
  – One often talks about the contribution to the CPI of a class of instructions

How to measure the average CPI

 elapsed time: wall clock

 CPU exec_time = Number of instr. * CPI * clock cycle time

• Count instructions executed in each class
• Needs a simulator
  • interprets every instruction and counts their number
• or a profiler
  • discover the most often used parts of the program and instruments
    – only those
  • or use sampling
• Use of programmable hardware counters
  – modern microprocessors have this feature but it’s limited

Other popular performance measures: MIPS

• MIPS (Millions of instructions per second)
  MIPS = Instruction count / (Exec.time * 10^9)
  MIPS = (Instr. count * clock rate)/Instr. count *(CPU * 10^9)
  MIPS = clock rate *(CPU * 10^9)
• MIPS is a rate: the higher the better
• MIPS in isolation no better than CPI in isolation
  – Program and/or compiler dependent
  – Does not take the instruction set into account
  – can give “wrong” comparative results

Other metric: MFLOPS

• Similar to MIPS in spirit
• Used for scientific programs/machines
  – MFLOPS: million of floating-point ops/second
Benchmarks

• Benchmark: workload representative of what a system will be used for
• Industry benchmarks
  – SPECint and SPECcpu industry benchmarks updated every few years,
    Currently SPEC CPU2000
  – Linpack (Lapack), NASA kernel: scientific benchmarks
  – TPC-A, TPC-B, TPC-C and TPC-D used for databases and data mining
  – Other specialized benchmarks (Olleweb, SPEC JVM98 etc…)
  – Benchmarks for desktop applications, web applications are not as standard
• Beware! Compilers are super optimized for the benchmarks

How to report (benchmark) performance

• If you measure execution times use arithmetic mean
  – e.g., for n benchmarks
    \[ \frac{\sum \text{exec time}_i}{n} \]
• If you measure rates use harmonic mean
  \[ \frac{n}{\sum \frac{1}{\text{rate}_i}} = 1/(\text{arithmetic mean}) \]

Computer design: Make the common case fast

• Amdahl’s law (speedup)
• Speedup = (performance with enhancement)/(performance base case)
  Or equivalently,
  Speedup = (exec.time base case)/(exec.time with enhancement)
• For example, application to parallel processing
  – x fraction of program that is sequential
  – Speedup is at most 1/x
  – That is if 20% of your program is sequential the maximum speedup with
    an infinite number of processors is at most 5