What is meant by:
performance of computer systems

Many different factors to be taken into account:

1. Technology
   - basic circuit speed (clock, MHz)
   - process technology (how many transistors on a chip)
2. Organization
   - what type of processor (RISC vs. CISC)
   - what type of memory hierarchy
   - what type of I/O devices
   - how many processors in the *system*
3. Software
   - how good are compilers, O.S., database drivers etc.
4. and many more

What are the metrics that can be used to measure the performance of computer systems?

1. Raw speed (peak performance but it is never attained)
2. Execution time (also called response time, i.e., time to execute ONE program from beginning to end). Need specific benchmarks for:
   - Integer dominated programs (compiling, formatting etc.)
   - Scientific
   - Graphical interface (previewers, plotters etc.)
   - Multimedia tasks (video, audio etc.)
3. Throughput (total amount of work in a given time)
   - Good metrics for the manager of a computer system. Best utilization of resources
   - Database programs (keep many users happy at the same time)
   - Web servers
4. Quite often improving response time will also improve throughput and vice-versa

Measure of performance - execution time

- Execution time

$$Performance_A = \frac{1}{Executiontime_A}$$

- Processor A is faster than processor B, i.e.,

$$Executiontime_A < Executiontime_B$$

$$Performance_A > Performance_B$$

- Relative performance

$$\frac{Performance_A}{Performance_B} = \frac{Executiontime_B}{Executiontime_A}$$

Measuring execution time

- Wall clock, response time, elapsed time
- Unix time function
  13.7u 23.6s 18:27 3% 2069+1821k 13+24io 62pf+0w
- Difficult to make comparisons form one system to another
- Remainder of this lecture:
  execution time = CPU execution time

Definition of CPU execution time

$CPU\ Execution\ time = CPU\ clock\ cycles \times clock\ cycle\ time$

$CPU\ Execution\ time$ is program dependent

$CPU\ clock\ cycles$ is program dependent

$clock\ cycle\ time$ (nanoseconds, ns.) depends on the particular machine

Since $clock\ cycle\ time = 1/clock\ cycle\ rate$ ($clock\ cycle\ rate$ in MHz) an alternate definition is:

$$CPU\ Execution\ time = \frac{CPU\ clock\ cycles}{clock\ cycle\ rate}$$

Clock cycle rate of 1 MHz $\rightarrow$ Cycle time 1 μs
Clock cycle rate of 1 GHz $\rightarrow$ Cycle time 1 ns

CPI – Cycles Per Instruction

Definition:
CPI Average number of clock cycles per instruction

$$CPU\ clock\ cycles = Number\ of\ Instructions \times CPI$$

or

$$CPU\ Exec.time = Number\ of\ Instructions \times CPI \times clock\ cycle\ time$$

CPI is what computer architects try to minimize but CPI in isolation is not a measure of performance (program dependent, compiler dependent)

Ideally CPI $= 1$ (perfect pipeline, see later chapters) but this might impose that clock is slower (compromise)

CPI $\neq 1$ (i.e., CPI > 1) because of breaks in flow of control (branches), impact of memory hierarchy etc.

Could CPI be less than 1? Yes in superscalar processors where several instructions can be executed concurrently

Classes of instructions

You can give different CPI's for various classes of instructions (e.g., floating-point arithmetic instructions take longer than integer ones)

$$CPU\ Exec.time = \sum_{1}^{n} (CPI_i \times C_i) \times clock\ cycle\ time$$

where C_i is the number of instructions of the i^{th} class that have been executed.

Note that minimizing the number of instructions does not necessarily improve the execution time of the program.

Improving part of the architecture can improve a C_j. One often talks about the contribution to CPI of a class of instructions

How to measure average CPI for a given program

Instruction count: need a simulator or (maybe less precise) a profiler

- Simulator "interprets" every instruction and counts them
- Profiler can either count how many times each basic block has been executed or use some sampling technique
- In modern processors, can use programmable hardware counters.

CPU Exec.time can be measured (elapsed time)
clock cycle time, a given of the processor

Other popular measures of performance – MIPS

MIPS = Millions of Instructions per Second

$$MIPS = \frac{Instruction\ count}{Exec.time \times 10^6}$$

or

$$MIPS = \frac{Instruction\ count \times clock\ rate}{Instruction\ count \times CPI \times 10^6}$$

or

$$MIPS = \frac{clock\ rate}{CPI \times 10^6}$$

Since MIPS is a rate, the higher, the better!

But MIPS in isolation is no better than CPI in isolation. MIPS is:

- Program dependent
- Does not take into account the instruction set
- Can give "wrong" results: Machine A with compiler C1 executes program P faster than with compiler C2 but the MIPS under C1 is lower than the MIPS under C2 (because, e.g., C1 generates fewer instructions)

Other popular measures of performance – MFLOPS

MFLOPS = Millions of floating-point operations per second

$$MFLOPS = \frac{Number\ of\ floating-point\ operations}{Exec.time \times 10^6}$$

Same problems as MIPS

- Program dependent
- Does not take into account the instruction set
- Counts operations, no time to execute instructions

Benchmarks

Benchmarks $\Longrightarrow$ workload representative of what the computer will be used for.

Industry benchmarks to compare machines: SPEC benchmarks (SPECint and SPECfp) updated about every 3 years

Benchmarks for scientific and parallel applications (Perfect Club, NASA)

There exists benchmarks for databases, TPC-A, TPC-B, TPC-C

Benchmarks for desktop (PC) applications? A ffew but not widely recognized.

Caveats:

- Compilers superoptimized for the benchmarks
- Utilities, user interface etc. often not in benchmarks

How to report (benchmark) performance

- Use arithmetic mean for execution times

$$\frac{\sum_i^n exec_i}{n}$$

- Use harmonic mean for rates

$$\frac{n}{\sum_i^n 1/rate_i} = \frac{1}{arithmetic\ mean}$$