

## 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

1/25/99

CSE378 Performance.

1

## 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 data base queries, Web servers)
- Quite often improving execution time will improve throughput and vice-versa

1/25/99

CSE378 Performance.

2

## Execution time Metric

- Execution time: inverse of performance

$$Performance_A = 1 / (Execution\_time_A)$$

- Processor A is faster than Processor B

$$Execution\_time_A < Execution\_time_B$$

$$Performance_A > Performance_B$$

- Relative performance

$$Performance_A / Performance_B = Execution\_time_B / Execution\_time_A$$

## Measuring execution time

- Wall clock, response time, elapsed time
- Some systems have a “time” function
  - Unix 13.7u 23.6s 18:37 3% 2069+1821k 13+24io 62pf+0w
- Difficult to make comparisons from one system to another
- Remainder of this lecture: *CPU execution time*

## 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 = 1 $\mu$ s, 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**

1/25/99

CSE378 Performance.

5

## 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. \* CPI \*clock cycle\_time*
- Computer architects try to minimize CPI
- CPI in isolation is not a measure of performance
  - program dependent, compiler dependent
- 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)

1/25/99

CSE378 Performance.

6

## 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 inst., say  $CPI_i$   
 $CPU\ exec\_time = \Sigma (CPI_i * C_i) * clock\ cycle\_time$   
where  $C_i$  is the number of instr. 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

1/25/99

CSE378 Performance.

7

## How to measure the average CPI

Elapsed time; wall clock

A given of the processor

$$CPU\ exec\_time = \text{Number of instr.} * CPI * clock\ cycle\_time$$

- 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
  - most modern microprocessors have this feature

1/25/99

CSE378 Performance.

8

## Other popular performance measures: MIPS

- MIPS (Millions of instructions per second)  
$$\text{MIPS} = \text{Instruction count} / (\text{Exec.time} * 10^6)$$
$$\text{MIPS} = (\text{Instr. count} * \text{clock rate}) / (\text{Instr. count} * \text{CPI} * 10^6)$$
$$\text{MIPS} = \text{clock rate} / (\text{CPI} * 10^6)$$
- 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 SPECfp industry benchmarks updated every 3 years
  - Perfect Club, NASA kernel: scientific benchmarks
  - TPC-A, TPC-B, TPC-C and TPC-D used for databases and data mining
  - Benchmarks for desktop applications, web applications are not as standard
  - Beware!
    - Compilers are super optimized for the benchmarks

1/25/99

CSE378 Performance.

11

## How to report (benchmark) performance

- If you measure execution **times** use **arithmetic mean**
  - e.g., for  $n$  benchmarks
$$(\sum exec\_time_i) / n$$
- If you measure **rates** use **harmonic mean**
$$n / (\sum 1/rate_i) = 1 / (\text{arithmetic mean})$$

1/25/99

CSE378 Performance.

12