Computer performance measurement. Amdahl’s Law.

Outline

- Characterizing performance
- Performance and speed
- Basic terminology for measuring
- Performance and Execution Time

Characterizing Performance

- Not always obvious how to characterize performance:
  - motor cars
  - football teams
  - tennis players
  - computers?
- Can lead to serious errors
  - improve the processor to improve speed?

Performance and Speed

- Performance for a program on a particular machine
  - $Performance(X) = \dfrac{1}{Execution(X)}$

$$\frac{Performance(X)}{Performance(Y)} = \frac{Execution(Y)}{Execution(X)} = n$$

  - X is n times faster than Y

Measuring Time

- Execution time is the amount of time it takes the program to execute in seconds.
- Time (computers do several tasks!)
  - elapsed time based on a normal clock;
  - CPU time is time spent executing this program
    - excluding waiting for I/O, or other programs

Execution Time

Elapsed Time
(real time)

CPU time for this program

I/O waiting
& other programs

user
(direct)

system
(time in OS)

Example (UNIX)

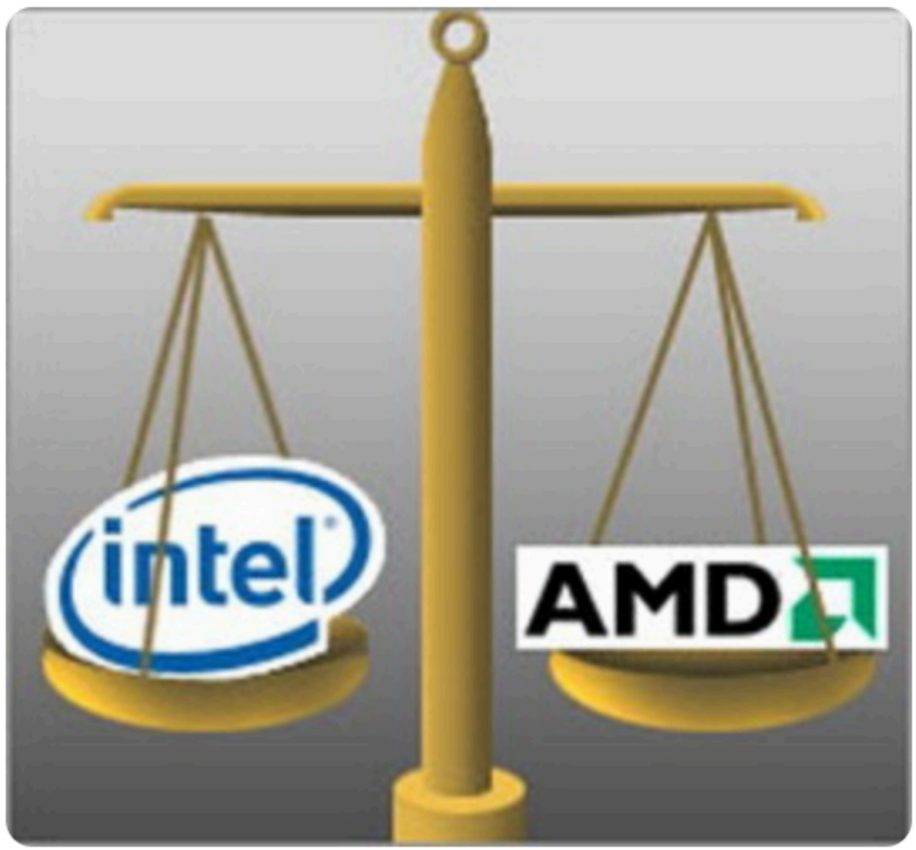
11.099u	8.659s	10:43.67	3.0%
(user)	(system)	(elapsed)	
(seconds)	(seconds)	(min:secs)	

CPU time = 3.0% of elapsed time

Measuring Amounts

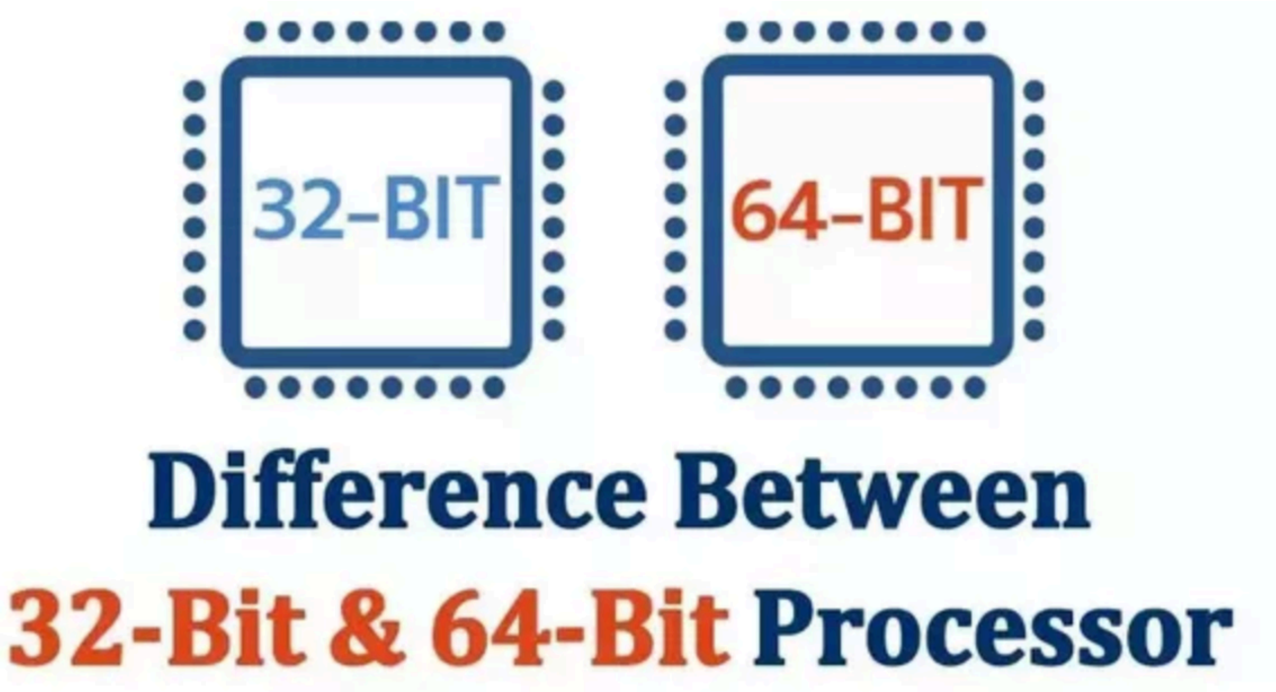
- 1 bit
- 8 bits = 1 byte
- 1024bytes = 1 kilobyte = 1KByte = 1K = 2^{10}
- 1024KBytes = 1 megabyte = 1MB = 2^{20}
- 1024MB = 1 gigabyte = 1GB = 2^{30}
- 1024GB = 1 terabyte = 2^{40}
- and on to infinity

Intel or AMD ?



Measuring Times

- Duration
  - 1 second
  - 1/1000 second = 1 millisec = 1ms = 10^{-3} s
  - 1/1,000,000 s = 1 microsec = 10^{-6} s
  - 1/1,000,000,000s = 1 nanosec = 10^{-9} s
- Frequency
  - 1 Hertz = 1 cycle per second
  - 1 MHz = 1,000,000 cycles per sec
  - 100MHz = 100,000,000 cycles per sec.



The type of processor a computer has not only affects its overall performance, but it can also dictate what type of software it uses.

Differences
32-bit and 64-bit

- number of calculations per second
- on the amount of RAM
  - 32-bit computers maximum of 3-4GB
  - 64-bit computer over 4 GB.
    - The first smartphone with a 64-bit chip (Apple A7) was the iPhone 5s.

Computer Clock Times

- Computers run according to a clock that runs at a steady rate
- The time interval is called a clock cycle (eg, 10ns).
- The clock rate is the reciprocal of clock cycle - a frequency, how many cycles per sec (eg, 100MHz).
  – 10 ns = 1/100,000,000 (clock cycle), same as:-
  – 1/10ns = 100,000,000 = 100MHz (clock rate).

Purchasing Decision

- Computer A has a 100MHz processor
- Computer B has a 300MHz processor
- So, B is faster, right?

NOPE!

- Now, let’s get it right.....

Measuring Performance

- The only important question: "HOW FAST WILL MY PROGRAM RUN?"
- CPU execution time for a program
  - = CPU clock cycles * cycle time
  - (= CPU clock cycles / Clock rate)
- In computer design, trade-off between:
  - clock cycle time, and
  - number of cycles required for a program

Cycles Per Instruction

- The execution time of a program clearly must depend on the number of instructions
  - but different instructions take different times
- An expression that includes this is:-
  - **CPU clock cycles = N * CPI**
    - **N** = number of instructions
    - **CPI** = average clock cycles per instruction

Example

- Machine A
  - clock cycle time
    - 10ns/cycle
  - CPI = 2.0 for prog X

- Machine B
  - clock cycle time
    - 30ns/cycle
  - CPI = 1.0 for prog X

Let I = number of instructions in the program.

CPU clock cycles (A) = I * 2.0
CPU time (A) = CPU clock cycles * clock cycle time
= I * 2.0 * 10
= I * 20 ns

CPU clock cycles (B) = I * 1.0
CPU time (B) = CPU clock cycles * clock cycle time
= I * 1.0 * 30
= I * 30 ns

Execution(B) / Execution(A) = 30 / 20 = 1.5

Basic Performance Equation

- CPU Time = I * CPI * T
  - I = number of instructions in program
  - CPI = average cycles per instruction
  - T = clock cycle time
- CPU Time = I * CPI / R
  - R = 1/T the clock rate
    - T or R are usually published as performance measures for a processor
    - I requires special profiling software
    - CPI depends on many factors (including memory).

Amdahl’s law

Amdahl’s Law

- **Amdahl's law**, named after computer architect Gene Amdahl, is used to find the maximum expected improvement to an overall system when only part of the system is improved.
- **Amdahl’s law** can be interpreted more technically, but in simplest terms it means that it is the algorithm that decides the speedup not the number of processors.

Amdahl’s Law

- A program (or algorithm) which can be parallelized can be split up into two parts:
  - A part which cannot be parallelized
  - A part which can be parallelized

Introduction

- If F is the fraction of a program that is sequential, and (1-F) is the fraction of program or algorithm that can be parallelized, then the maximum speed-up that can be achieved by using P processors is:

$$\frac{1}{F + (1 - F)/N}$$

Examples

- If **90%** of a calculation can be parallelized (i.e. **10%** is sequential) then the maximum speed-up which can be achieved on **5** processors is **1/(0.1+(1-0.1)/5)** or roughly **3.6** (i.e. the program can theoretically run **3.6** times faster on five processors than on one)

Examples

- If **90%** of a calculation can be parallelized then the maximum speed-up on **10** processors is **1/(0.1+(1-0.1)/10)** or **5.3** (i.e. investing twice as much hardware speeds the calculation up by about **50%**).
- If **90%** of a calculation can be parallelized then the maximum speed-up on **20** processors is **1/(0.1+(1-0.1)/20)** or **6.9** (i.e. doubling the hardware again speeds up the calculation by only **30%**).

Examples

- If **90%** of a calculation can be parallelized then the maximum speed-up on **1000** processors is **1/(0.1+(1-0.1)/1000)** or **9.9** (i.e. throwing an absurd amount of hardware at the calculation results in a maximum theoretical (i.e. actual results will be worse) speed-up of **9.9** vs a **single processor**).