

# CENG 3420

# Computer Organization & Design



## Lecture 11: Performance

Bei Yu

CSE Department, CUHK

[byu@cse.cuhk.edu.hk](mailto:byu@cse.cuhk.edu.hk)

(Textbook: Chapters 1.6 & 1.7)

2024 Spring



## Purchasing perspective

Given a collection of machines, which has the

- best performance ?
- least cost ?
- best cost/performance?

## Design perspective

Faced with design options, which has the

- best performance improvement ?
- least cost ?
- best cost/performance?



Both (purchasing & design) require

- basis for comparison
- metric for evaluation

## Our Goal

Is to understand what factors in the architecture contribute to overall system performance and the relative importance (and cost) of these factors



## Response time (execution time)

- The time between the start and the completion of a task.
- Important to individual users

## Throughput (bandwidth)

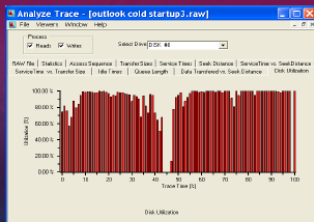
- The total amount of work done in a given time
- Important to data center managers

Will need different performance metrics as well as a different set of applications to benchmark **embedded** and **desktop** computers, which are more focused on response time, versus **servers**, which are more focused on throughput



## It's the Hard Disk, Stupid!

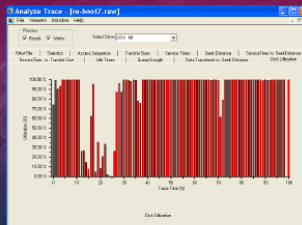
### Re-Boot/Startup on Home PC



Elapsed Time 105.213536, s  
Disk Busy Time 91.368480, s  
Average Data Rate 6.60669, MB/s

**86% BUSY**

### Starting Outlook



Elapsed Time 45.700667, s  
Disk Busy Time 41.056997, s  
Average Data Rate 1.37389, MB/s

**89% BUSY**





- To maximize performance, need to minimize **execution** time

$$\text{performance}_X = \frac{1}{\text{execution\_time}_X}$$

- If X is  $n$  times faster than Y, then

$$\frac{\text{performance}_X}{\text{performance}_Y} = \frac{\text{execution\_time}_Y}{\text{execution\_time}_X} = n$$

- Decreasing **response** time almost always improves throughput.



## EX-1

If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B?



- CPU execution time (CPU time): time the CPU spends working on a task
- Does not include time waiting for I/O or running other programs

$$\begin{aligned}\text{CPU execution time} &= \# \text{ CPU clock cycles} \times \text{clock cycle time} \\ &= \frac{\# \text{ CPU clock cycles}}{\text{clock rate}}\end{aligned}$$

## Can improve performance by reducing

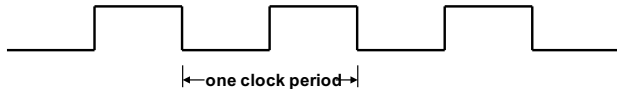
- Length of the clock cycle
- Number of clock cycles required for a program





Clock rate (clock cycles per second in MHz or GHz) is inverse of clock cycle time (clock period)

$$CC = \frac{1}{CR}$$



10 nsec clock cycle => 100 MHz clock rate

5 nsec clock cycle => 200 MHz clock rate

2 nsec clock cycle => 500 MHz clock rate

1 nsec ( $10^{-9}$ ) clock cycle => 1 GHz ( $10^9$ ) clock rate

500 psec clock cycle => 2 GHz clock rate

250 psec clock cycle => 4 GHz clock rate

200 psec clock cycle => 5 GHz clock rate

## EX-2: Improving Performance Example



A program runs on computer A with a 2 GHz clock in 10 seconds. What clock rate must a computer B have to run this program in 6 seconds? Unfortunately, to accomplish this, computer B will require 1.2 times as many clock cycles as computer A to run the program.



- Not all instructions take the same amount of time to execute
- One way to think about execution time is that it equals the number of instructions executed multiplied by the average time per instruction

**CPU clock cycles = # instruction × clock cycle per instruction**

## Clock cycles per instruction (CPI)

- The average number of clock cycles each instruction takes to execute
- A way to compare two different implementations of the same ISA



$$\sum_{i=1}^n CPI_i \times IC_i$$

$IC_i$ : percentage of the number of instructions of class  $i$  executed

$CPI_i$ : (average) number of clock cycles per instruction for that instruction class

$n$ : number of instruction classes

- Computing the overall effective CPI is done by looking at the different types of instructions and their individual cycle counts and averaging
- The overall effective CPI varies by instruction mix
- A measure of the dynamic frequency of instructions across one or many programs



$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{clock cycle}$$

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{clock rate}}$$

## Discussions about the three key factors

- **instruction count**: can be measured by using profilers/ simulators without knowing all of the implementation details
- **CPI**: varies by instruction type and ISA implementation for which we must know the implementation details
- **clock rate**: is usually given



### EX-3: Using the Performance Equation

Computers A and B implement the same ISA. Computer A has a clock cycle time of 250 ps and an effective CPI of 2.0 for some program and computer B has a clock cycle time of 500 ps and an effective CPI of 1.2 for the same program. Which computer is faster and by how much?



$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{clock cycle}$$

|                      | Instruction_count | CPI | clock_cycle |
|----------------------|-------------------|-----|-------------|
| Algorithm            |                   |     |             |
| Programming language |                   |     |             |
| Compiler             |                   |     |             |
| ISA                  |                   |     |             |
| Core organization    |                   |     |             |
| Technology           |                   |     |             |



$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{clock cycle}$$

|                      | Instruction_count | CPI | clock_cycle |
|----------------------|-------------------|-----|-------------|
| Algorithm            | X                 | X   |             |
| Programming language | X                 | X   |             |
| Compiler             | X                 | X   |             |
| ISA                  | X                 | X   | X           |
| Core organization    |                   | X   | X           |
| Technology           |                   |     | X           |





## EX-4

| Op     | Freq | CPI <sub>i</sub> | Freq x CPI <sub>i</sub> |
|--------|------|------------------|-------------------------|
| ALU    | 50%  | 1                |                         |
| Load   | 20%  | 5                |                         |
| Store  | 10%  | 3                |                         |
| Branch | 20%  | 2                |                         |
|        |      |                  | $\Sigma =$              |

- 1 How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?
- 2 How does this compare with using branch prediction to shave a cycle off the branch time?
- 3 What if two ALU instructions could be executed at once?



## Benchmarks

A set of programs that form a “workload” specifically chosen to measure performance

- SPEC (System Performance Evaluation Cooperative) creates standard sets of benchmarks starting with SPEC89.
- The latest is SPEC CPU2006 which consists of 12 integer benchmarks (CINT2006) and 17 floating-point benchmarks (CFP2006).
- [www.spec.org](http://www.spec.org)
- There are also benchmark collections for power workloads (SPECpower\_ssj2008), for mail workloads (SPECmail2008), for multimedia workloads (mediabench) ...



| Name           | ICx10 <sup>9</sup> | CPI   | ExTime | RefTime | SPEC ratio |
|----------------|--------------------|-------|--------|---------|------------|
| perl           | 2,1118             | 0.75  | 637    | 9,770   | 15.3       |
| bzip2          | 2,389              | 0.85  | 817    | 9,650   | 11.8       |
| gcc            | 1,050              | 1.72  | 724    | 8,050   | 11.1       |
| mcf            | 336                | 10.00 | 1,345  | 9,120   | 6.8        |
| go             | 1,658              | 1.09  | 721    | 10,490  | 14.6       |
| hmmmer         | 2,783              | 0.80  | 890    | 9,330   | 10.5       |
| sjeng          | 2,176              | 0.96  | 837    | 12,100  | 14.5       |
| libquantum     | 1,623              | 1.61  | 1,047  | 20,720  | 19.8       |
| h264avc        | 3,102              | 0.80  | 993    | 22,130  | 22.3       |
| omnetpp        | 587                | 2.94  | 690    | 6,250   | 9.1        |
| astar          | 1,082              | 1.79  | 773    | 7,020   | 9.1        |
| xalancbmk      | 1,058              | 2.70  | 1,143  | 6,900   | 6.0        |
| Geometric Mean |                    |       |        |         | 11.7       |



## How to summarize performance with a **single** number?

- First the execution times are normalized given the “SPEC ratio” (bigger is faster, i.e., SPEC ratio is the inverse of execution time)
- SPEC ratios are “averaged” using the geometric mean (GM)

$$GM = n \cdot \sqrt[n]{\sum_{i=1}^n \text{SPEC ratio}_i}$$

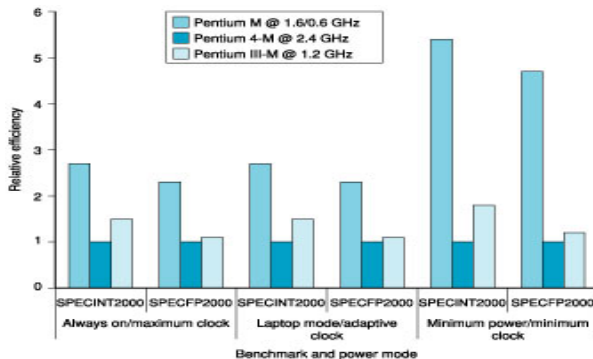
## Guiding principle – reproducibility

List everything another experimenter would need to duplicate the experiment: version of the operating system, compiler settings, input set used, specific computer configuration (clock rate, cache sizes and speed, memory size and speed, etc.)

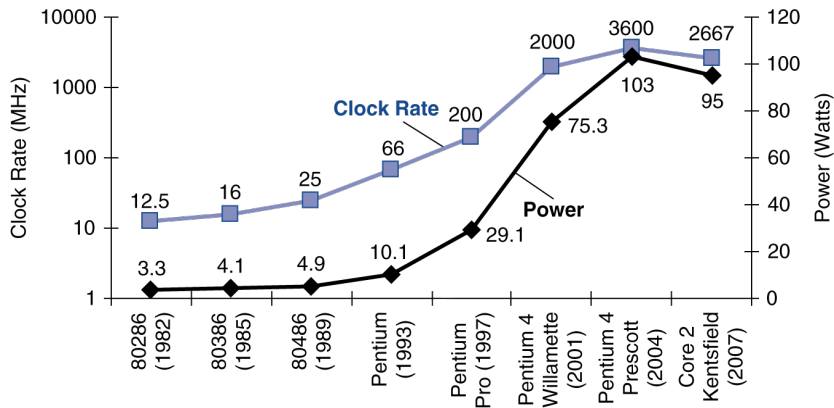


## Power Consumption

- Especially in the embedded market where battery life is important
- For power-limited applications, the most important metric is energy efficiency



# Highest Clock Rate of Intel Processors



What if the exponential increase had kept up? Why not?

- Due to process improvements
- Deeper pipeline
- Circuit design techniques