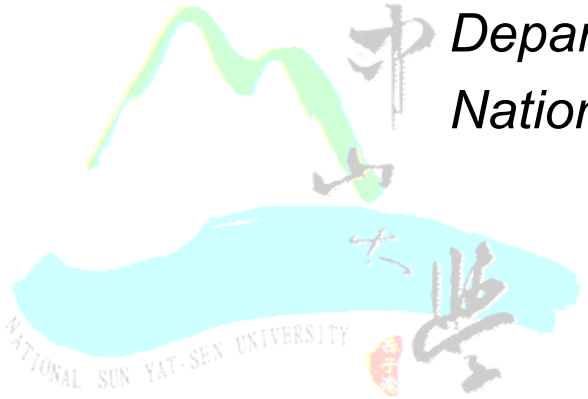
Introduction

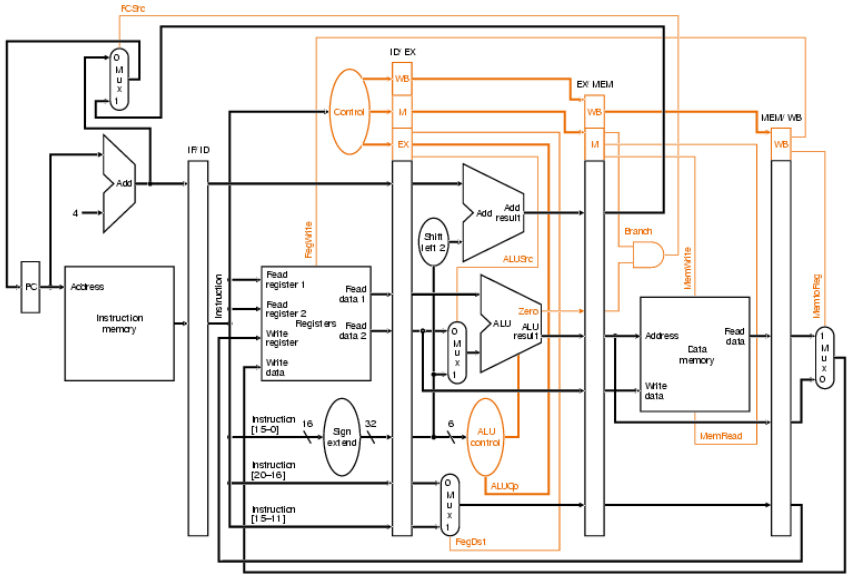
Kun-Chih (Jimmy) Chen 陳坤志

Department of Computer Science and Engineering
National Sun Yat-sen University



What's difference between Computer Organization and Architecture? (1/2)

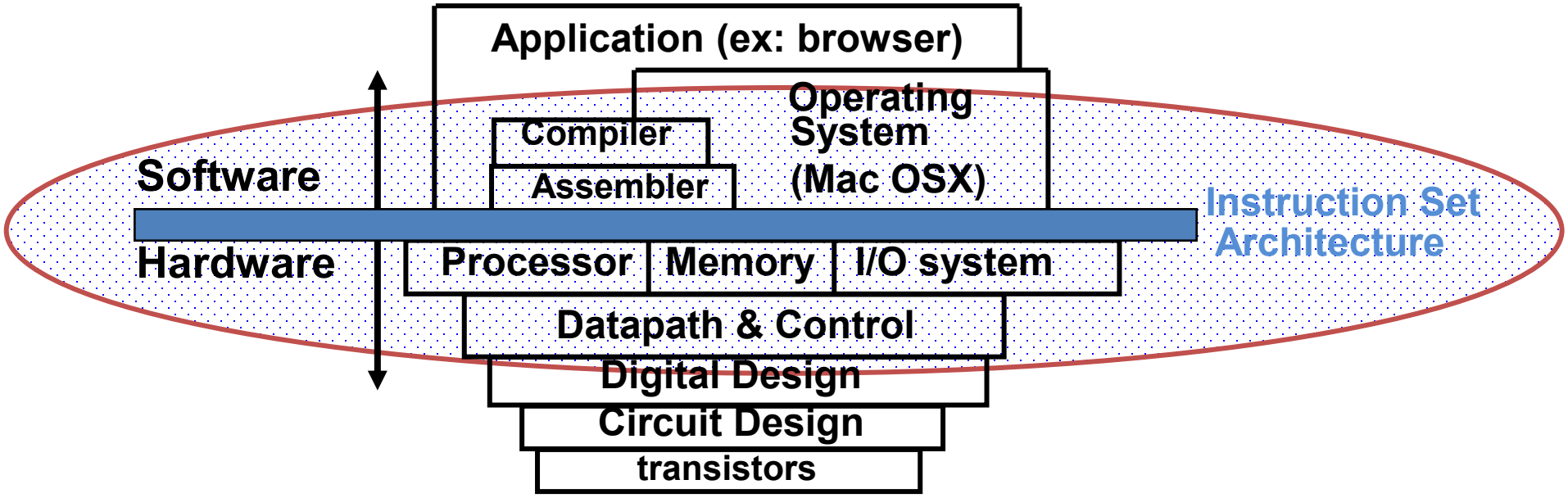
- The "Computer Organization" focuses on the "Instruction Level Architecture" and "Architecture" of "**Signal** processor."



What's difference between Computer Organization and Architecture? (2/2)

- The "Computer Architecture" focus on the "**multiple** processor" and "**parallel** architecture."

Machine Structure in “Computer Organization”



Machine Structure in “Computer Architecture”

- Parallel Requests
  Assigned to computer
  e.g., Search “Katz”
- Parallel Threads
  Assigned to core
  e.g., Lookup, Ads
- Parallel Instructions
  >1 instruction @ one time
  e.g., 5 pipelined instructions
- Parallel Data
  >1 data item @ one time
  e.g., Add of 4 pairs of words
- Hardware descriptions
  All gates working in parallel at same time

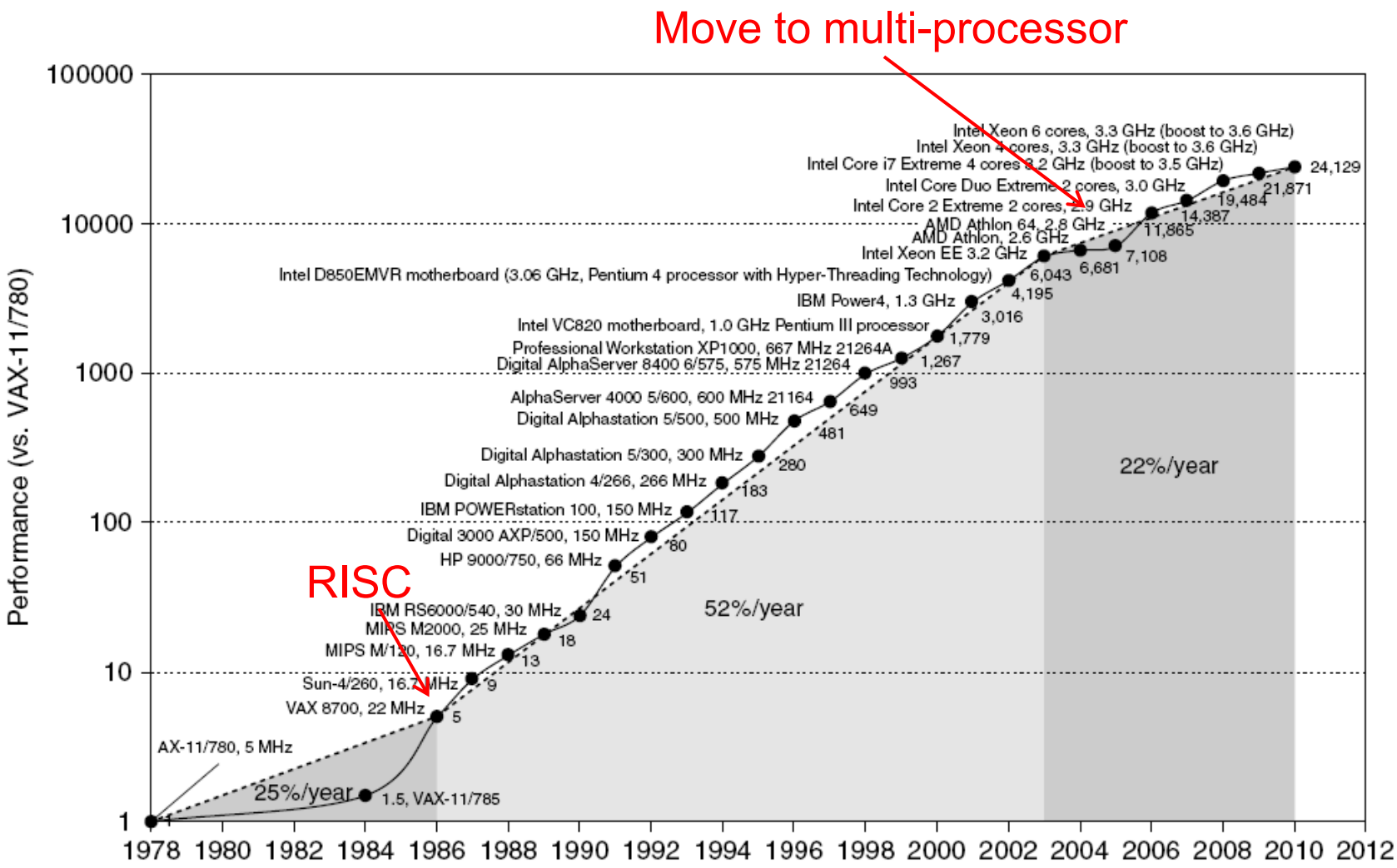
Software | *Hardware*

Harness Parallelism & Achieve High Performance

Warehouse Scale Computer

Smart Phone

Single Processor Performance



Classes of Computers

Feature	Personal mobile device (PMD)	Desktop	Server	Clusters/warehouse-scale computer	Embedded
Price of system	$100–$1000	$300–$2500	$5000–$10,000,000	$100,000–$200,000,000	$10–$100,000
Price of micro-processor	$10–$100	$50–$500	$200–$2000	$50–$250	$0.01–$100
Critical system design issues	Cost, energy, media performance, responsiveness	Price-performance, energy, graphics performance	Throughput, availability, scalability, energy	Price-performance, throughput, energy proportionality	Price, energy, application-specific performance

Defining Computer Architecture

- "Old" view of computer architecture
  - Instruction Set Architecture (ISA) design
    - ISA is playing less of a role today than in 1990
  - decisions regarding
    - registers, memory addressing, addressing modes, instruction operands, available operations, control flow instructions, instruction encoding

- "Real" computer architecture
  - Specific requirements of the target machine
  - Design to maximize performance within constraints: cost, power, and availability
  - Includes ISA, microarchitecture, hardware

Flynn's Taxonomy (1960s)

- Single instruction stream, single data stream (SISD) – Ch3
- Single instruction stream, multiple data streams (SIMD) – Ch4
  - Vector architectures
  - Multimedia extensions
  - Graphics processor units (GPU)
- Multiple instruction streams, single data stream (MISD)
  - No commercial implementation
- Multiple instruction streams, multiple data streams (MIMD)
  - Ch5: Tightly-coupled MIMD – thread-level parallelism
  - Ch6: Loosely-coupled MIMD – request-level parallelism

Classes of "Architecture" Parallelism

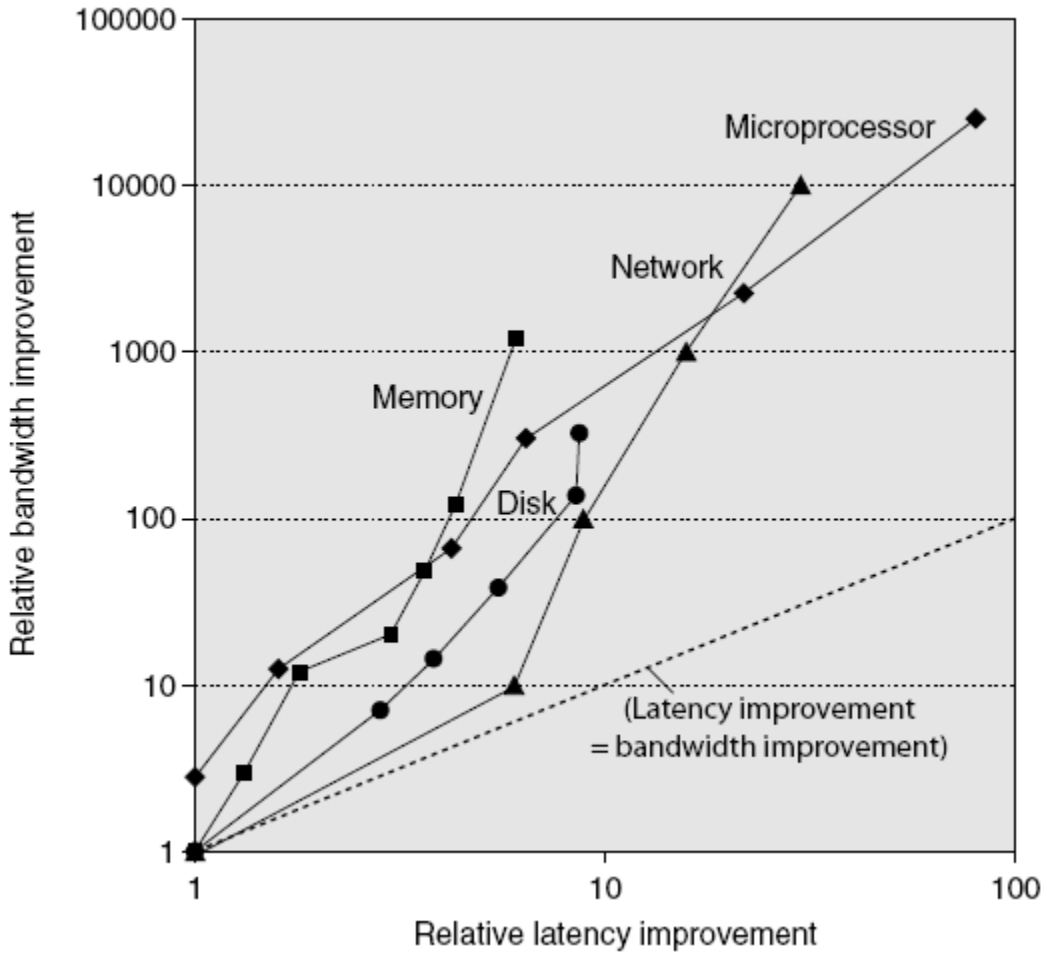
- ***Instruction-Level Parallelism (ILP)***
  - Exploit DLP with compiler or speculative execution
  - Cannot continue to leverage Instruction-Level parallelism (ILP)
    - Single processor performance improvement ended in 2003
- ***Vector Architecture and Graphic Processor Units (GPUs)***
  - Exploit DLP by applying a single instruction to a collection of data in parallel
- ***Thread-Level Parallelism***
  - Exploit DLP or TLP in a tightly coupled hardware model that allows for instruction among parallel threads
- ***Request-Level Parallelism (RLP)***
  - Exploit parallelism among largely decoupled tasks specified by the programmer or the OS

Pipeline Stages of Intel CPU family

Microarchitecture	Pipeline stages
P5 (Pentium)	5
P6 (Pentium 3)	10
P6 (Pentium Pro)	14
NetBurst (Willamette)	20
NetBurst (Northwood)	20
NetBurst (Prescott)	31
NetBurst (Cedar Mill)	31
Core	14
Bonnell	16
Sandy Bridge	14
Silvermont	14 to 17
Haswell	14
Skylake	14
Kabylake	14

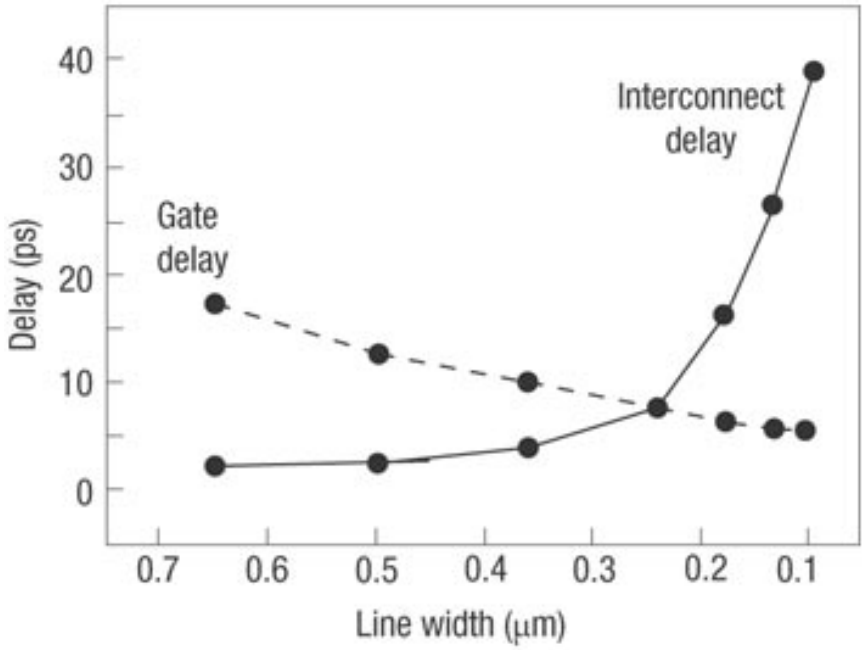
Performance Trends: Bandwidth over Latency

- Bandwidth or throughput
  - Total work done in a given time
  - 10,000-25,000X improvement for processors
  - 300-1200X improvement for memory and disks
- Latency or response time
  - Time between start and completion of an event
  - 30-80X improvement for processors
  - 6-8X improvement for memory and disks



Transistors and Wires

- Feature size
  - Minimum size of transistor or wire in x or y dimension
  - 10 microns in 1971 to .032 microns in 2011
  - Transistor performance scales linearly
    - Wire delay does not improve with feature size!
  - Integration density scales quadratically



Power and Energy

- Problem: Get power in, get power out
- Thermal Design Power (TDP)
  - Characterizes sustained power consumption
  - Lower than peak power, higher than average power consumption
  - Used as target for power supply and cooling system
    - Two kinds of thermal management in the modern processors
- Clock rate can be reduced dynamically to limit power consumption
- Energy per task is often a better measurement

Case Study: Profile of Intel i7-8850H

Intel® Core™ i7-8850H Processor

9M Cache, up to 4.30 GHz

Essentials

Export specifications

Product Collection	8th Generation Intel® Core™ i7 Processors
Code Name	Products formerly Coffee Lake
Vertical Segment	Mobile
Processor Number	i7-8850H
Status	Launched
Launch Date	Q2'18
Lithography	14 nm
Recommended Customer Price	$395.00
Use Conditions	PC/Client/Tablet

Performance

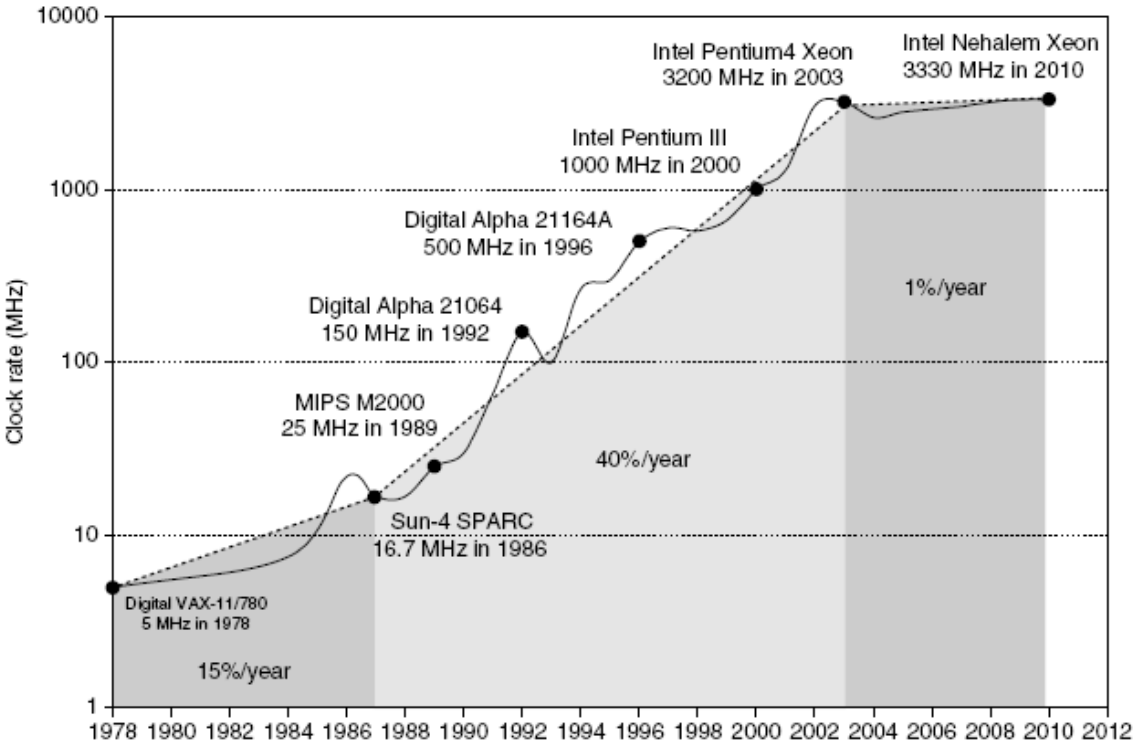
# of Cores	6
# of Threads	12
Processor Base Frequency	2.60 GHz
Max Turbo Frequency	4.30 GHz
Cache	9 MB SmartCache
Bus Speed	8 GT/s DMI
TDP	45 W
Configurable TDP-down	35 W

Dynamic Energy and Power

- Dynamic energy
  - Transistor switch from 0 -> 1 or 1 -> 0
  - ½ x Capacitive load x Voltage^2
- Dynamic power
  - ½ x Capacitive load x Voltage^2 x Frequency switched
- Reducing clock rate reduces power, not energy

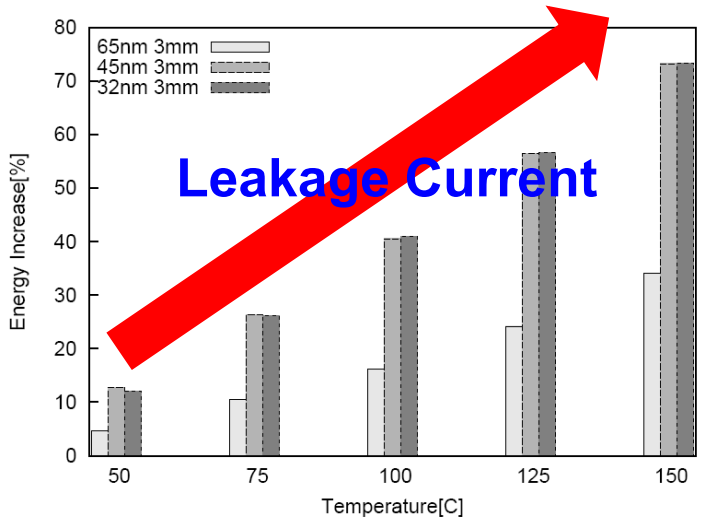
Power

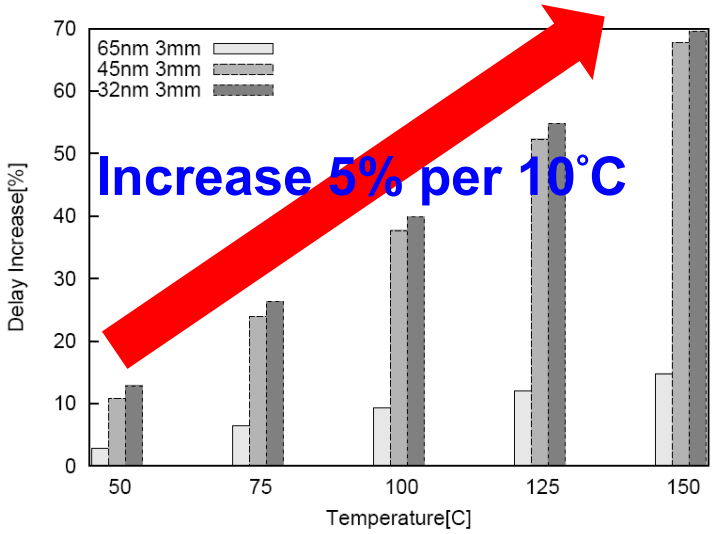
- Intel 80386 consumed ~ 2 W
- 3.3 GHz Intel Core i7 consumes 130 W
- Heat must be dissipated from 1.5 x 1.5 cm chip
- This is the limit of what can be cooled by air



Power vs. Thermal Issue

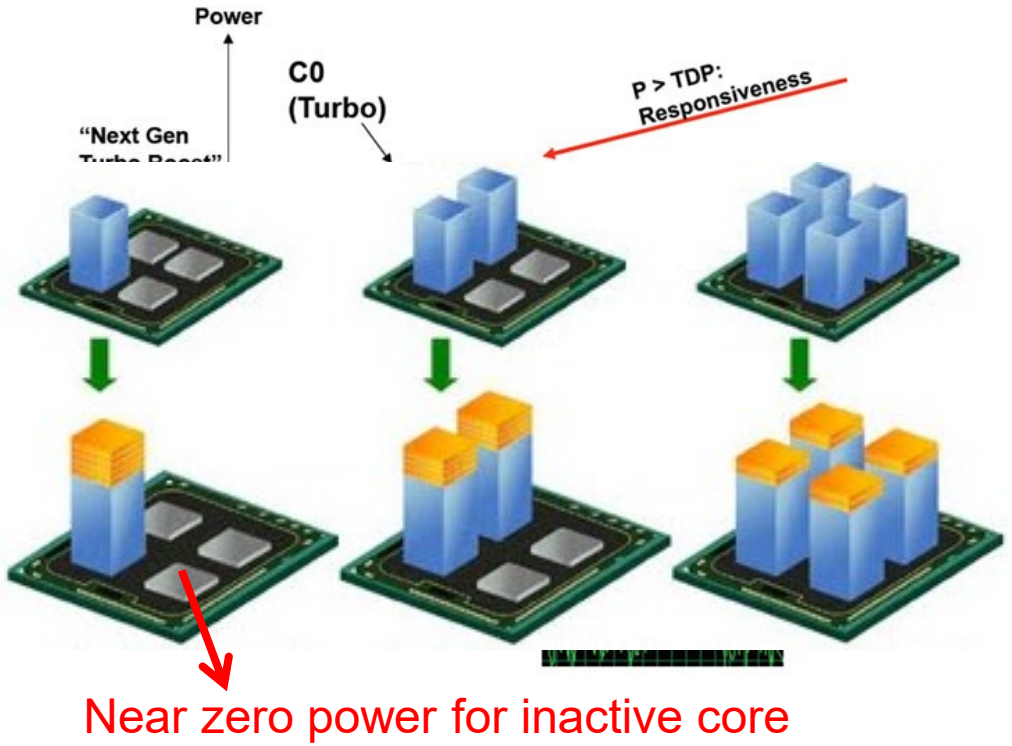
- Distributing the power, removing the heat, and preventing hot spots have become emerging topics.
- Techniques for reducing power
  - Do nothing well
  - Dynamic Voltage-Frequency Scaling
  - Low power state for DRAM, disks
  - Overclocking, turning off cores





Case Study: Intel© Turbo Boost Technology 2.0

- Features
  - Under a Thermal Design Power(TDP), maximize the frequency state depends on the workload and:
    - Number of active cores
    - Estimated current consumption
    - Estimated power consumption
    - Processor temperature
- Commercial Products
  - Intel Core i7-2xxx series
  - Intel Core i5-2xxx series
  - Intel Xeon E3-12xx series



Exercise (Energy Power)

Example:

Some microprocessors today are designed to have adjustable voltage, so a 15% reduction in voltage may result in a 15% reduction in frequency. What would be the impact on dynamic energy and on dynamic power?

Answer:

Since the capacitance is unchanged, the answer for energy is the ratio of the voltages since the capacitance is unchanged:

$$\frac{\text{Energy}_{\text{new}}}{\text{Energy}_{\text{old}}} = \frac{(\text{Voltage} \times 0.85)^2}{\text{Voltage}^2} = 0.85^2 = 0.72$$

Cost of an IC

- Integrated circuit

$$\text{Cost of integrated circuit} = \frac{\text{Cost of die} + \text{Cost of testing die} + \text{Cost of packaging and final test}}{\text{Final test yield}}$$

$$\text{Cost of die} = \frac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{Die yield}}$$

$$\text{Dies per wafer} = \frac{\pi \times (\text{Wafer diameter}/2)^2}{\text{Die area}} - \frac{\pi \times \text{Wafer diameter}}{\sqrt{2 \times \text{Die area}}}$$

- Bose-Einstein formula
  - Assume that defects are randomly distributed over the wafer

$$\text{Die yield} = \text{Wafer yield} \times 1/(1 + \text{Defects per unit area} \times \text{Die area})^N$$

  - Defects per unit area is measure of the random manufacturing defects that occur
    - 0.016-0.057 defects per square cm (2010)
  - N = process-complexity factor, a measure of manufacturing difficulty
    - 11.5-15.5 (40 nm, 2010)

Example

Find the number of dies per 300mm (30cm, 12in) wafer for a die that is 1.5 cm on a side

Answer:

When the die area is 2.25 cm^2

$$\text{Dies per wafer} = \frac{\pi \times (\frac{30}{2})^2}{2.25} - \frac{\pi \times 30}{\sqrt{2 \times 2.25}} = 270$$

Example

Find the die yield for dies that are 1.5cm on a side and 1.0cm on a side, assuming a defect density of 0.031 per cm^2 and N is 13.5.

Answer:

We assume the wafer yield is 100%

For the larger die area (i.e., 2.25 cm^2), the yield is

Die yield = $\frac{1}{(1+0.031\times2.25)^{13.5}} = 0.40$

Dependability

- As we head to feature sizes of 32nm and smaller, the both transistor faults and permanent faults will become more commonplace
- The architects must design systems to cope with these challenges and consider…
  - Mean Time to Failure (MTTF)
  - Mean Time to Repair (MTTR)
  - Mean Time between Failure (MTBF) = MTTF + MTTR
  - Availability = MTTF / MTBF
  - Failure rate = 1 / MTTF

Example

Assume a disk subsystem with the following components and MTTF:

- 10 disks, each rated at 1,000,000-hour MTTF
- 1 ATA controller, 500,000-hour MTTF
- 1 power supply, 200,000-hour MTTF
- 1 fan, 200,000-hour MTTF
- 1 ATA cable, 1,000,000-hour MTTF

Using the simplifying assumptions that the lifetimes are exponentially distributed and that failures are independent, compute the MTTF of the system as a whole.

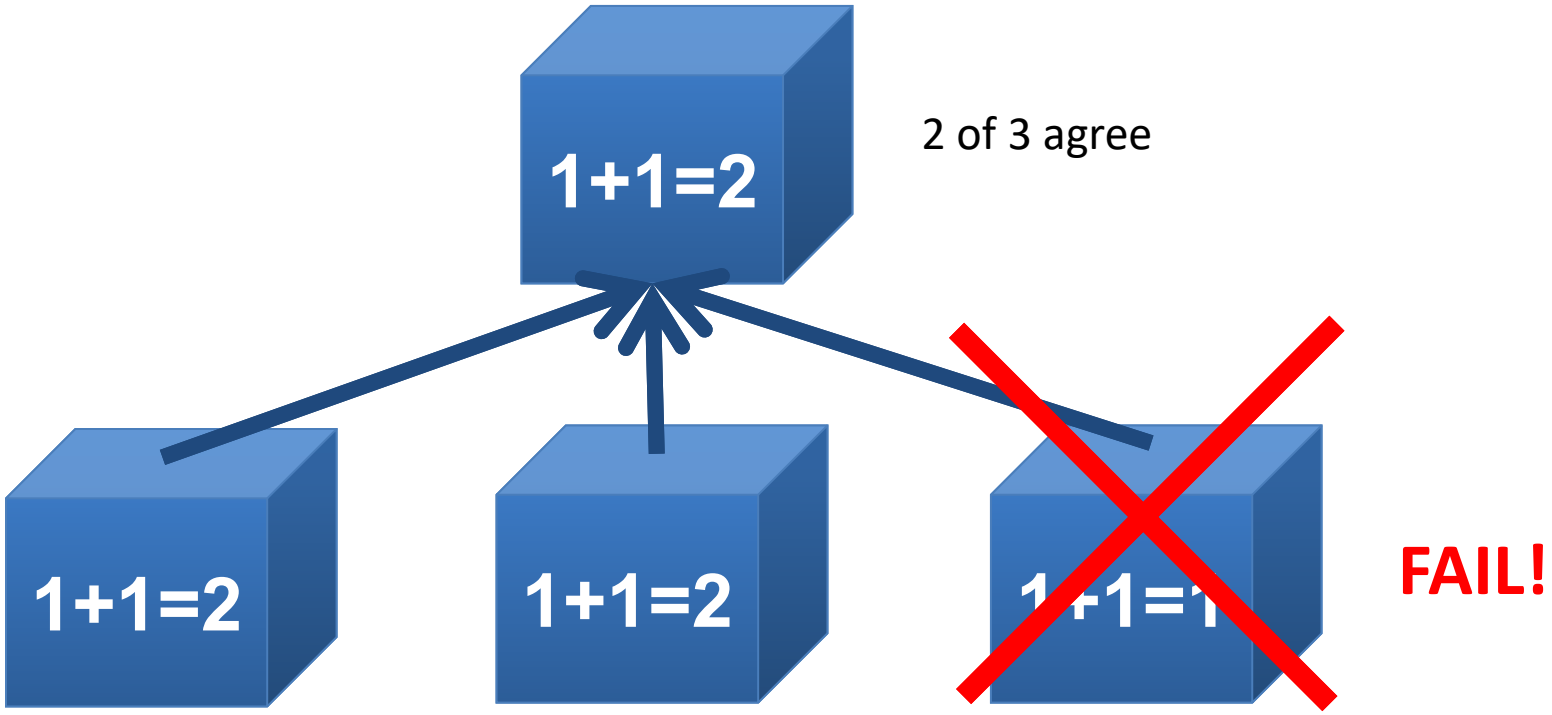
Answer:

$$\text{Failure rate}_{\text{system}} = 10 \times \frac{1}{1000000} + \frac{1}{500000} + \frac{1}{200000} + \frac{1}{200000} + \frac{1}{1000000} = \frac{23000}{1000000000}$$

$$\text{MTTF}_{\text{system}} = \frac{1}{Failure\ rate} = 43500\ hours$$

Dependability via Redundancy (1/2)

- Redundancy so that a failing piece doesn't make the whole system fail



Dependability via Redundancy (2/2)

- Applies to everything from datacenters to storage to memory to instructors
  - Redundant data centers so that can lose 1 datacenter but Internet service stays online
  - Redundant computer was Google's original internal innovation
  - Redundant disks so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID)
  - Redundant memory bits so that can lose 1 bit but no data lose (Error Correcting Code/ECC Memory)

Performance Measures

- Typical performance metrics
  - Response time (latency): time between start and completion
  - Throughput (bandwidth) -- rate: work done per unit time

- Speedup of X relative to Y
  - Execution time$_Y$ / Execution time$_X$
  - B is n times faster than A
    - Means exec_time_A/exec_time_B == rate_B/rate_A

- Execution time
  - Wall clock time (or response time, or elapsed time): includes all system overheads
  - CPU time: only computation time

Measuring Performance

- Time is the measure of computer performance
- Elapsed time (Wall-clock time) = program execution + I/O + wait
  - important to user
- Execution time = user time + system time (but OS self measurement may be inaccurate)
- CPU performance = user time on unloaded system
  - important to architect

Real Performance Measurement

- Benchmark suites
  - Attempts at running programs that ate much simpler than a real application have led to performance pitfalls
- Performance is the result of executing a workload on a configuration
- Workload = program + input
- Configuration = CPU + cache + memory + I/O + OS + compiler optimizations
- compiler optimizations can make a huge difference!

Improve Performance by

- Changing the
  - algorithm (ex. Bubble sorting vs. Quick sorting)
  - data structures (ex. Structure vs. pointer)
  - programming language (ex. C vs. Matlab)
  - compiler
  - compiler optimization flags
  - OS parameters (ex. Windows vs. Linux)
- Improving locality of memory or I/O accesses
- Overlapping I/O (ex. Single direction port vs. bi-direction port)
- On multiprocessors, you can improve performance by avoiding cache coherency problems (e.g., false sharing) and synchronization problems

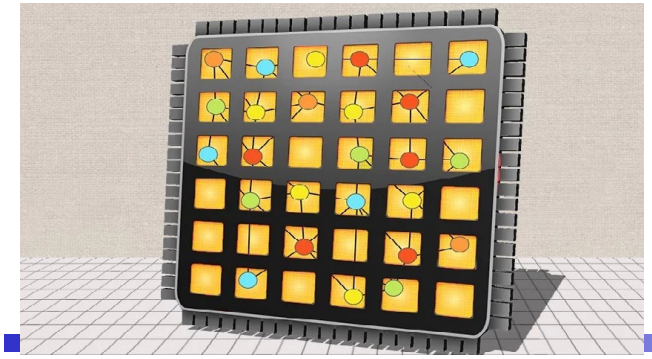
Principle of Computer Design

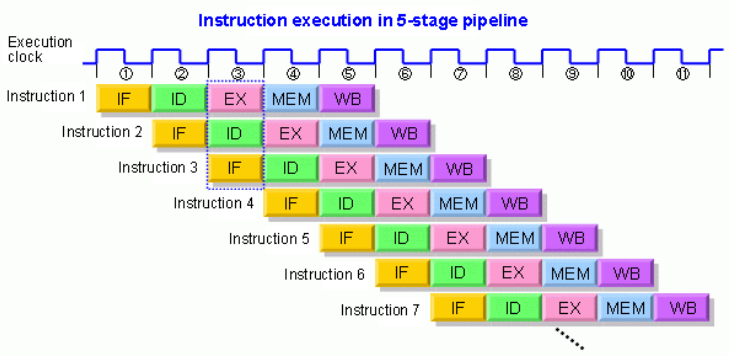
Parallelism at the system level

- The workload of handling requests can then be spread among the processors and disks
  - GPU or multicore CPU
- Spreading of data across many disks for parallel reads and writes enables data-level parallelism
  - Distribution computing

Parallelism among instructions in an individual processor

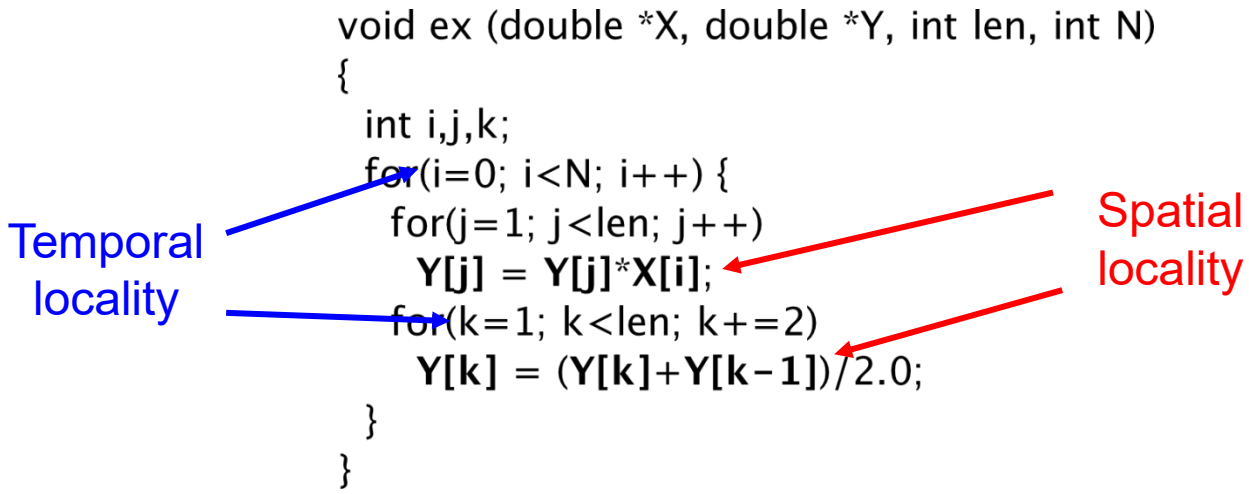
- Pipeline is the best-known example of instruction-level parallelism



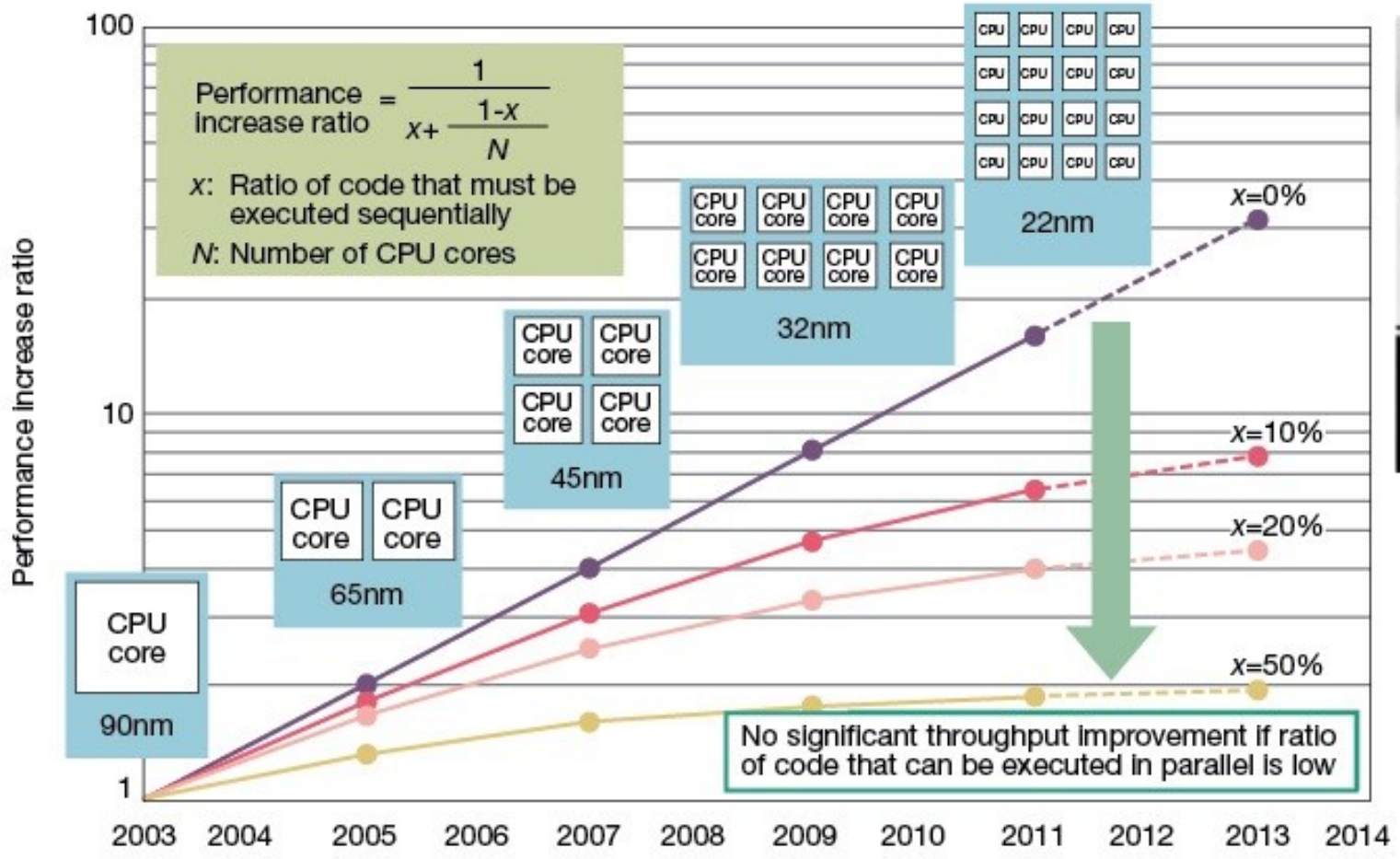


Principle of Locality

- Program tend to reuse data and instructions they have used recently
- Temporal Locality
  - Recently accessed items are likely to be accessed in the near future
- Spatial Locality
  - The items whose addresses are near one another tend to be referenced close together in time



Amdahl's Law: A method to quantify performance speedup



Gene Amdahl
Computer Pioneer

Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor performance. Diagram assumes no overhead in parallel processing. Years shown for design rules based on Intel planned and actual technology. Core count assumed to double for each rule generation.

Amdahl's Law

- Amdahl's law gives us a quick way to find the speedup from some enhancement, which depends on two factors:
  - Fraction of the computation time in the original computer that can be converted to take advantage of the enhancement
    - How much fraction of the computation time can be enhanced?
    - The fraction is always less than 1
  - How much faster the task would run if the enhanced mode were used for the entire program
    - Speedup = (execution time without enhance.) / (execution time with enhance.) = (time without) / (time with) = $T_{w/o} / T_{w/}$

$$\text{Execution time}_{\text{new}} = \text{Execution time}_{\text{old}} \times \left((1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Example

Suppose we are considering an enhancement to a web server. The enhanced CPU is 10 times faster on computation but the same speed on I/O. Suppose also that 60% of the time is waiting on I/O

Answer:

$\text{Fraction}_{\text{enhanced}}=0.4$

$\text{Speedup}_{\text{enhanced}}=10$

$\text{Speedup}_{\text{overall}}=\frac{1}{0.6+\frac{0.4}{10}}=\frac{1}{0.64}\approx 1.56$

CPU Performance Equation (1/2)

- Almost all computers are constructed using a clock running at a constant rate
- Computer designers refer to the time of a clock period by its
  - Duration (e.g. 1ns)
  - Rate (e.g. 1GHz)

- The Processor Performance Equation

$$\text{CPU time} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

$$\text{CPU time} = \text{Instruction count} \times \text{Cycles per instruction} \times \text{Clock cycle time}$$

$$\text{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}}$$

CPU Performance Equation (2/2)

- Processor performance is dependent on
  - **Clock cycle:** hardware technology and organization
  - **CPI:** organization and instruction set architecture
  - **Instruction count:** instruction set architecture and compiler technology

- It is useful in designing the processor to calculate the number of total processor clock cycles

$$\text{CPU clock cycles} = \sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i$$

$$\text{CPU time} = \left(\sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i \right) \times \text{Clock cycle time}$$