CSE 4305
Computer Organization and Architecture

Performance Issues

Course Teacher: Md. Hamjajul Ashmafee

Lecturer, CSE, IUT

Email: ashmafee@iut-dhaka.edu

Who is the mightiest????

Design for Performance

- A consideration is required **to balance the utilization of computer resources** – develop a **means to assess computer system performance** comparatively
- Processors are so **inexpensive** that used once and then thrown away (like a **testing paper**)
- But todays computers virtually follow **same basic building block as IAS computer** – the **main difference** is the sophisticated techniques of squeezing the maximum performance out of the materials
- Computers in todays can **support** even **complex and powerful applications** like: Image processing, 3D rendering, Speech recognition, Video conferencing, Simulation modeling

Microprocessor Speed

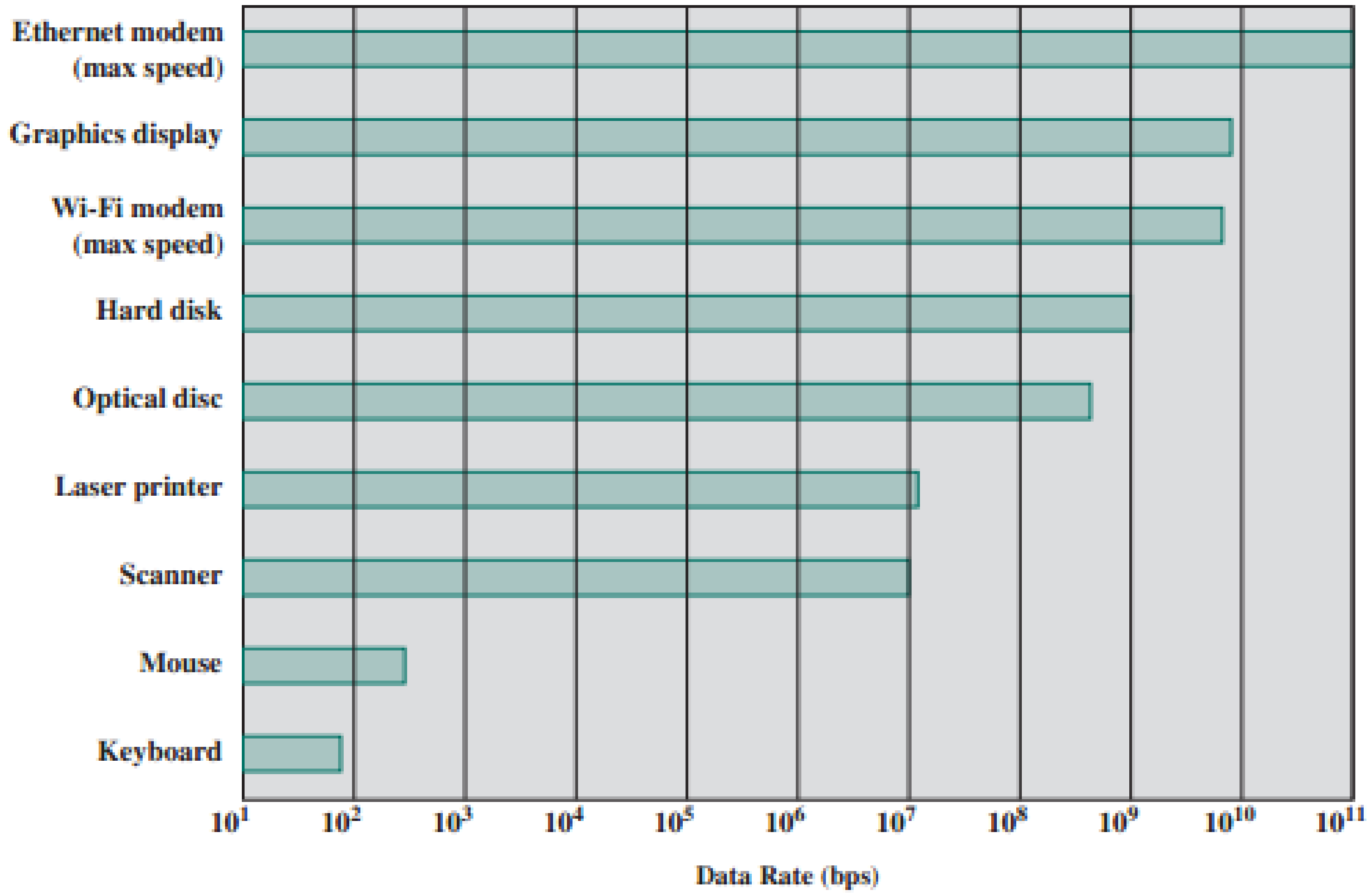
- According to **Moore's Law**:
  - Unleashing **new generation chips** with **4 times more** **transistors** in every three years
  - The **capacity of DRAM** (main memory) **quadrupled** in every three years
  - The **performance of microprocessor** improved **four- or five-fold** every three years
- But we can not achieve the potential unless it is fed in work stream
- **To feed the monster**, designers come with more techniques:
  - **Pipelining:** work simultaneously on multiple instructions
  - **Branch prediction:** prefetching correct instructions and buffer them
  - **Super scaler execution:** multiple parallel pipelines
  - **Data flow analysis**: scheduling instruction execution to prevent delay
  - **Speculative execution**: look ahead execution before their actual appearance keeping processor busy

Performance Balance: Why???

- Processor power is speeded up very fast, but other components are not at the same pace – maintain balance performance
- **One problem** – interface between memory and processor
- To attack this problem:
  - **Increasing the number of bits** to retrieve from DRAMs – **wide data bus**
  - **Including Cache** to change the DRAM interface **buffering more**
  - **Complex and efficient cache** to **reduce memory access**
  - Include **higher bandwidth** by using **higher speed bus**

Performance Balance…

- **Another Problem:** handling slower I/O devices – **strategies taken:** caching, buffering, higher speed bus, elaborate interconnection structure, dedicated microprocessor



Design factors: keeping in Mind

The **key** is **balance all demands from all components** – focusing the factors:

- The **rate at which performance is changing** in the various technology areas (processor, buses, memory, peripherals) **differs greatly** from one type of element to another.
- **New applications and new peripheral devices** constantly **change the nature** of the demand on the system in terms of typical instruction profile and the data access patterns.

Improvements in Chip

To improve processor speed:

- **Increase the hardware speed** of the processor – more gates, smaller size, increased clock rate (rapid execution)
- <u>Increase</u> the **size and speed of the cache**
- Make **change to the processor organization and architecture** like parallelism in different forms

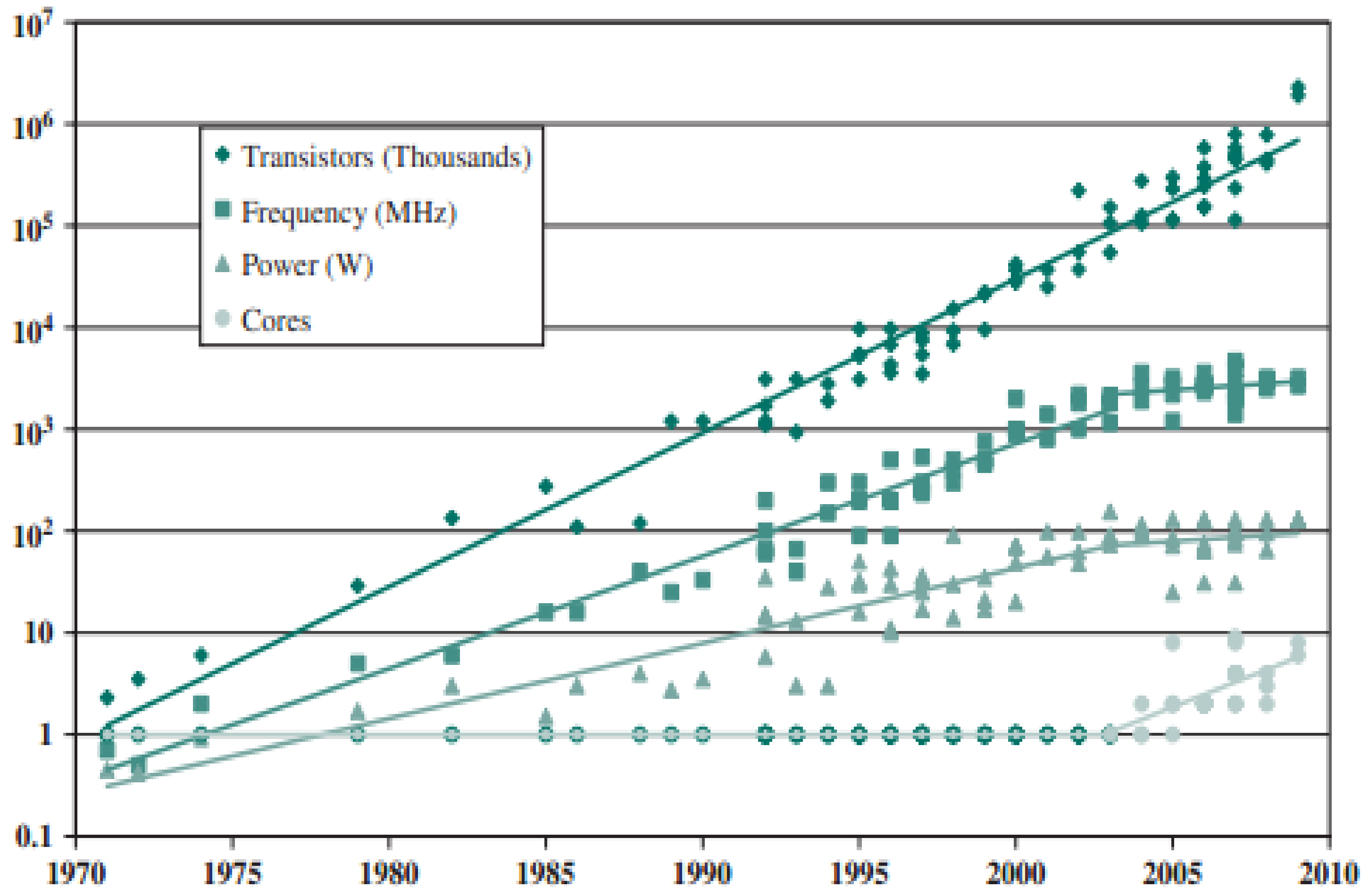
But increased clock speed and logic density **causes problems**:

- **Power:** misuse of generated heat
- **RC delay:** more logic, more resistance (R) and capacitance (C), more delay
- **Memory Latency & Throughput:** as <u>processor speeded up</u>, <u>memory access speed and transfer speed delayed</u>

In 1980s, two strategies taken:

1. Cache
2. Parallel and complex instruction execution

Processor Trends



New Approaches in Recent Processors

- **Multicore:** complex logic (general purpose processor)
- **On chip and shared cache:** in different levels
- **MIC:** *many integrated core*– more than 50 cores per chip (general purpose processor)
- **GPU:** *graphics processing unit* – with other multiple general purpose processors – to perform parallel operation graphics data
- **GPGPU**: *general purpose computing on GPU* – support a broad range of application

Laws that provides insight: Ahmdahl's Law

- Speeding up in any aspect of technology does not result in improvement in performance
- Deals with **potential speed up of a program** using multiple processors compared to a single processor

$$\text{Speedup} = \frac{\text{Time to execute program on a single processor}}{\text{Time to execute program on } N \text{ parallel processors}}$$

$$= \frac{T(1-f) + Tf}{T(1-f) + \frac{Tf}{N}} = \frac{1}{(1-f) + \frac{f}{N}}$$

Fraction (1-*f*) = time in sequential
Fraction *f* = time in parallelism (Speed up)
T = total time
N = # of processors in parallel

- When *f* is small, the use of parallel processors has little effect
- As *N* approaches to infinity, speed up upper bounded by (1/(1-*f*))

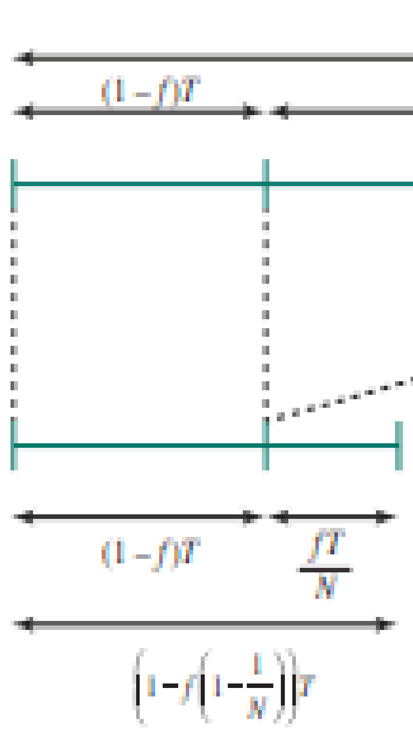
Ahmdahl's Law…

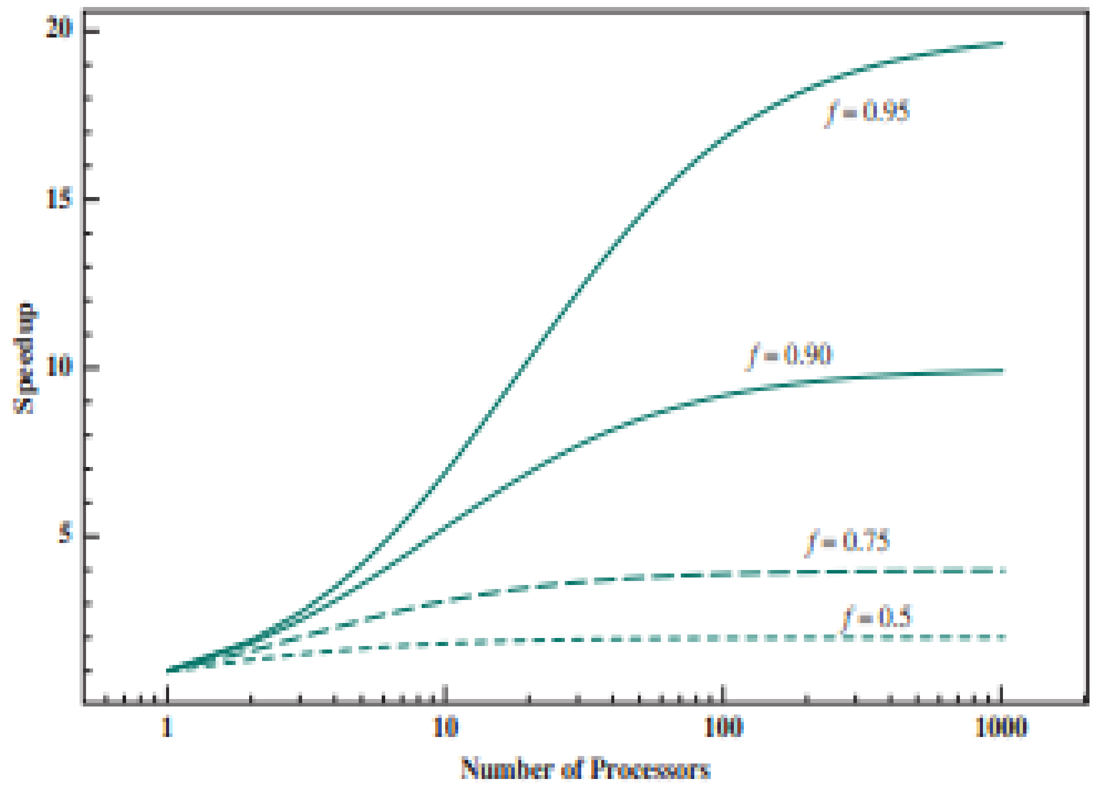
$$\text{Speedup} = \frac{\text{Performance after enhancement}}{\text{Performance before enhancement}} = \frac{\text{Execution time before enhancement}}{\text{Execution time after enhancement}}$$

$$\text{Speedup} = \frac{1}{(1 - f) + \dfrac{f}{SU_f}}$$

f = a fraction of time before enhancement for a feature
SU_f = speed up of that particular feature (not overall)

Ahmdahl's Law in Graph





Little's Law

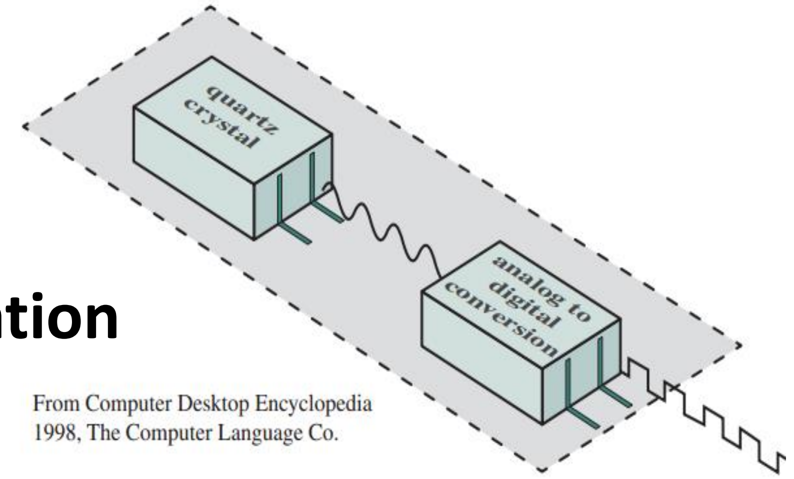
- **Applicable in any system** which are in **steady state** and without any leakage
- Used in **queuing theory terminology** – central element is **server**; provides **service** – *some items are served from that server*
- **L = λW**; where, λ = **average rate of items** per unit time, W = **average stay time** for any item, L = **average units of items** at any time
- Example: A **processor** provides service to processes; a **transmission line** provides a transmission service to packets or frames of data; and an **I/O device** provides a read or write service for I/O requests

Basic Measures on Performance

- **Performance** is key parameter to consider along with cost, size, security, reliability, power consumption **to evaluate processor** and **set requirements**
- Difficult to **make meaningful performance comparisons** among different processors, even in same family
- **Raw speed** is **not** that much important *because* **application performance** doesn't depends on it alone – also on instruction set, programming language, programming skill, compiler efficiency

Measure of Performance: Clock Speed

- **Operations** performed by processor **governed by a system clock** – all operations begin with a clock pulse – processor speed measured in **Hertz (Hz)**
- **Clock signal** – **generated** by a quartz crystal (generates constant **sine wave**)
- This sine wave **converted** to digital voltage pulse stream
- **Clock speed** = clock rate
- **Clock cycle** = a clock tick
- **Cycle time** = time between two pulses
- Must be **appropriate with processor**, not arbitrary
- Signals should be aligned with processor – **synchronization**
- Tells about the **rapidness in instruction execution**



From Computer Desktop Encyclopedia
1998, The Computer Language Co.

Measure of Performance: Instruction Execution Rate

- Processor **driven by a clock** with **constant frequency (*f*)** or **constant cycle time (τ)**; τ = 1/*f*
- Another important parameter: **average cycles per instruction (*CPI*)**
- But the number of clock cycles required **varies** for **different types of instructions** like load, fetch, store, branch and so on

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{I_c}$$

CPI_i = number of cycles required for instruction type i
I_i = number of instruction **executed** of type i for a given program
I_c = **executed** instruction count for a program

- The **processor time (T)** to execute a given program:

$$T = I_c \times CPI \times \tau$$

Instruction Execution Rate…

- But during **an instruction execution**, partial task done **by processor (p)** and other part involved **memory transfer (m)**

$$T = I_c \times [p + (m \times k)] \times \tau$$

p = number of cycles required by processor
m = number of memory references required
k = **ratio** between memory and processor cycle time (**normalized**)

- ***I_c, p, m, k, τ*** are **influenced** by four **system attributes** like:
  - Design of instruction set
  - Compiler technology
  - Processor implementation
  - Cache and memory hierarchy

Instruction Execution Rate…

- Another **common expression** to measure of performance for processor as – **MIPS rate (millions (10^6)** of instructions per second)

$$\text{MIPS rate} = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

f = frequency of clock cycles

- Another common performance measure **dealing with only floating point instructions** for scientific and game applications – MFLOPS rate (millions of floating point operations per second)

$$\text{MFLOPS rate} = \frac{\text{Number of executed floating point operations in a program}}{\text{Execution time} \times 10^6}$$

Calculating the Mean

- Sometimes to evaluate the system performance, a **single number** is required to **characterize that system** (execution time, consumed memory) - to simplify the comparison
- According to the **benchmarks**, we use **mean value** to compare systems
- There are **three formulas** to calculate the mean:
  - Arithmetic mean
  - Geometric mean
  - Harmonic mean

Arithmetic Mean

- Appropriate measure **if all the values are same meaningful and interesting**
- Example: Execution Time (or any time based variables)
- **Linearly** related

$$AM = \frac{x_1 + \cdots + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Harmonic Mean

- Sometimes total rate from different individual rates is meaningless and also same for mean rate **because they are inversely proportional** (not linear)
- **Time and rate relationship**: t and 1/t, if time is increased, rate will not be increased
- It takes those values into account **which are reciprocals in sequence** holding **constant denominator**
- Example: Capacitance in Series, Resistance in Parallel

$$HM = \frac{n}{\left(\frac{1}{x_1}\right) + \cdots + \left(\frac{1}{x_n}\right)} = \frac{n}{\sum_{i=1}^{n}\left(\frac{1}{x_i}\right)} \qquad x_i > 0$$

- Very much **similar with arithmetic mean** but in **reciprocal form** – can be driven from AM.

Geometric Mean

- AM and HM are **dominated by largest or smallest values** respectively **neglecting** other values.
- **It emphasizes ratio**; in how much **number of times** things are changing
- Used to measure **relative performance**: **comparison**
- **Non-monotonic** in property relative to AM or HM: may or may not be increased if the values are increased. (1:2, 2:4, 3:5)

$$GM = \sqrt[n]{x_1 \times \cdots \times x_n} = \left(\prod_{i=1}^{n} x_i\right)^{1/n} = e^{\left(\frac{1}{n}\sum_{i=1}^{n}\ln(x_i)\right)}$$

Benchmark Principles

- MIPS or MFLOPS are **not adequate** **to compare whole performance of processors** as there are **still some other issues** like difference in instruction sets (**RISC vs CISC**) or different programs (**integer vs floating point**)
- In 1980s, industry and academy were interested to measure performance **using a set of benchmark programs** running them in different machines
- **Characteristics** of BENCHMARK program:
  - Written in **higher level language**
  - **Portable** across different machines
  - **Representative of a particular domain**: systems, numerical, commercial
  - **Easily measurable**
  - **Wide distribution** of tasks

SPEC Benchmarks

- A **standard benchmark suits** are required from industry, academy and research community
- **Benchmark suit**: collection of programs defined in high level language representing a test of a computer in a particular area
- **Best known benchmark suit:** **SPEC** (Standard Performance Evaluation Corporation) - best release is **SPEC CPU2006** based on computation (**5th generation**)
- SPEC CPU2006 are **drawn from real life applications** rather than any artificial or synthetic cases like use of loop
- SPEC CPU2006 consists of **12 integer and 17 floating point benchmarks** written in C, C++, Fortran over **3 million lines of code**

SPEC Benchmarks…

Common terms in SPEC CPU2006 documentation:

- **Benchmark**: A program
- **System under test**
- **Reference Machine**: A base machine used by SPEC to prepare baseline
- **Base metric**: required for all
- **Peak metric**: used to optimize the system performance
- **Speed metric**: used to measure time
- **Rate metric**: used to measure the rate of task accomplishments

SPEC Benchmarks: How to Calculate the Assessments

Calculation is **a three step process** [*focusing the integer programs with 12 programs suit*]:

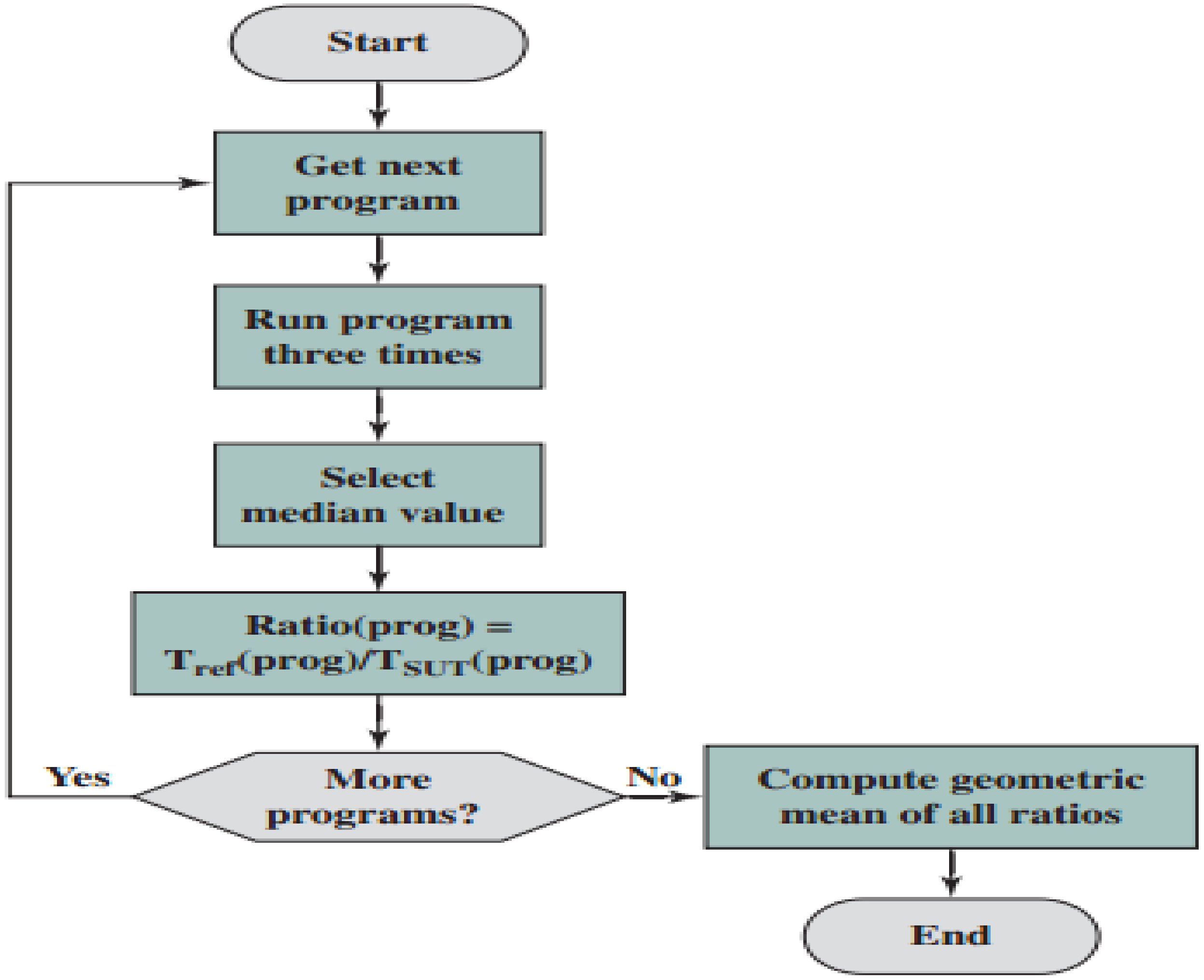
- **Compile the system under test and run each program on the system three times**. For each program, the runtime is measured and the **median value** is selected to avoid the variation in execution time .
- Next, each of the **12 results is normalized** by calculating the **runtime ratio** of the reference run time to the system run time:

$$r_i = \frac{Tref_i}{Tsut_i}$$

- Finally, the **geometric mean of the 12 runtime** ratios is calculated to yield the **overall metric**:

$$r_G = \left(\prod_{i=1}^{12} r_i\right)^{1/12}$$

Calculate the Assessments…



A system with multiple processor

- Each **individual test program's rate** is determined by taking the **median of three runs** where **each run** consists of **N** copies of the program running **simultaneously** on the test system in **N processor** or **N threads**. Here rate metric:

$$rate_i = N \times \frac{Tref_i}{Tsut_i}$$

- The **final rate** score of the total system is determined from **geometric mean of rates** for each program in the test suite

Important !!!

All kinds of math problems covering the preceding formulas…