Part-1

Chapter-1

Computer Abstraction & Technology

The computer revolution:

Computer has led to third revolution for civilization. The progress in computer technology largely under-pinned by Moore's Law, which observes that, the number of transistors per chip doubles every two year, leading to exponential improvement in computing power, efficiency and affordability. As a result, novel applications have become feasible reshaping industries and everyday life.

- Key areas of Compluter Revoution:
  - Computers in automobiles
  - Cell phones
  - Human Genome project
  - World Wide Web
  - Search Engines

Computers are Pervasive. (ব্যাপক)

Where do we find computers today?

Computers are not just on desks or laps, it's almost everywhere. Such as - automobiles, smartphones, games, Home automation, education, healthcare, Finance & Retail, Space exploration etc.

How have computers revolutionized life in the 21st century?

Computers have transformed many aspects of life:

- Automobiles – Modern cars have GPS, automatic braking and self driving technology.
- Mobile Phones & Gaming – Smartphones function as mini computers, enabling communication, AI assistant and entertainment.
- Massive distributed Project/Genome Project – Super Computers helped map the human genome, leading to medical advancements.
- World Wide Web – The internet connects billions of people and supports e-commerce, education and communication.
- Search Engines – It provides instant access to information worldwide.

What has fuelled Progress in Computer Technology?

Moore's Law: "The number of transistors per chip doubles every ~~year~~ two years."

Where do we find computers today?

- Desktops, Laptops, Servers (Network based & Cloud computers)
- Embedded processors: hidden as component inside such as cars, toys, phones, irons, toasters, wristwatches.

Classes of Computer:

- Personal Computers – (General purpose, variety of software, cost vs performance)
- Server Computers – (Network based, High capacity and reliability, variety of sizes).
- Supercomputers – (High end scientific and Engineering calculations, Extremely high performance, Rare)
- Embedded computers – (Hidden as component of systems, designed for efficiency

The PostPC Era:

- PMD – Personal Mobile Device
  - → Battery operated
  - → Connects to the internet
  - → Smartphones, tablets, electronic glasses.
- Cloud Computing
  - → Software as a service (SaaS) → web search, social networking
  - → (WSC) Warehouse Scale Computers
  - → Portion of PMD software, Amazon and Google

Cloud Computing: Cloud computing refers to a large collection of servers that provide services over the internet, dynamically varying number of servers as a utility.

A major application of cloud computing is Software as a Service (SaaS), where portion of code runs on PMD and a portion of code runs in the cloud.

Performance of a Computer system: It depends on multiple factors working together –

- Algorithm – Determines no. of operations executed.
- Programming language, compiler, architecture – Determines number of machine instructions executed per operation.
- Processor and memory system – Determines how fast instructions are executed.
- I/O system (including OS) – Determines how fast I/O operations are executed.

Levels of programing code:

① High level language

- Closer to problem domain: Easier to understand and write for humans.
- Productivity and portability: Code can run on different machines with minimal changes (Java, Python).

② Assembly language

- Textual representation of instructions:

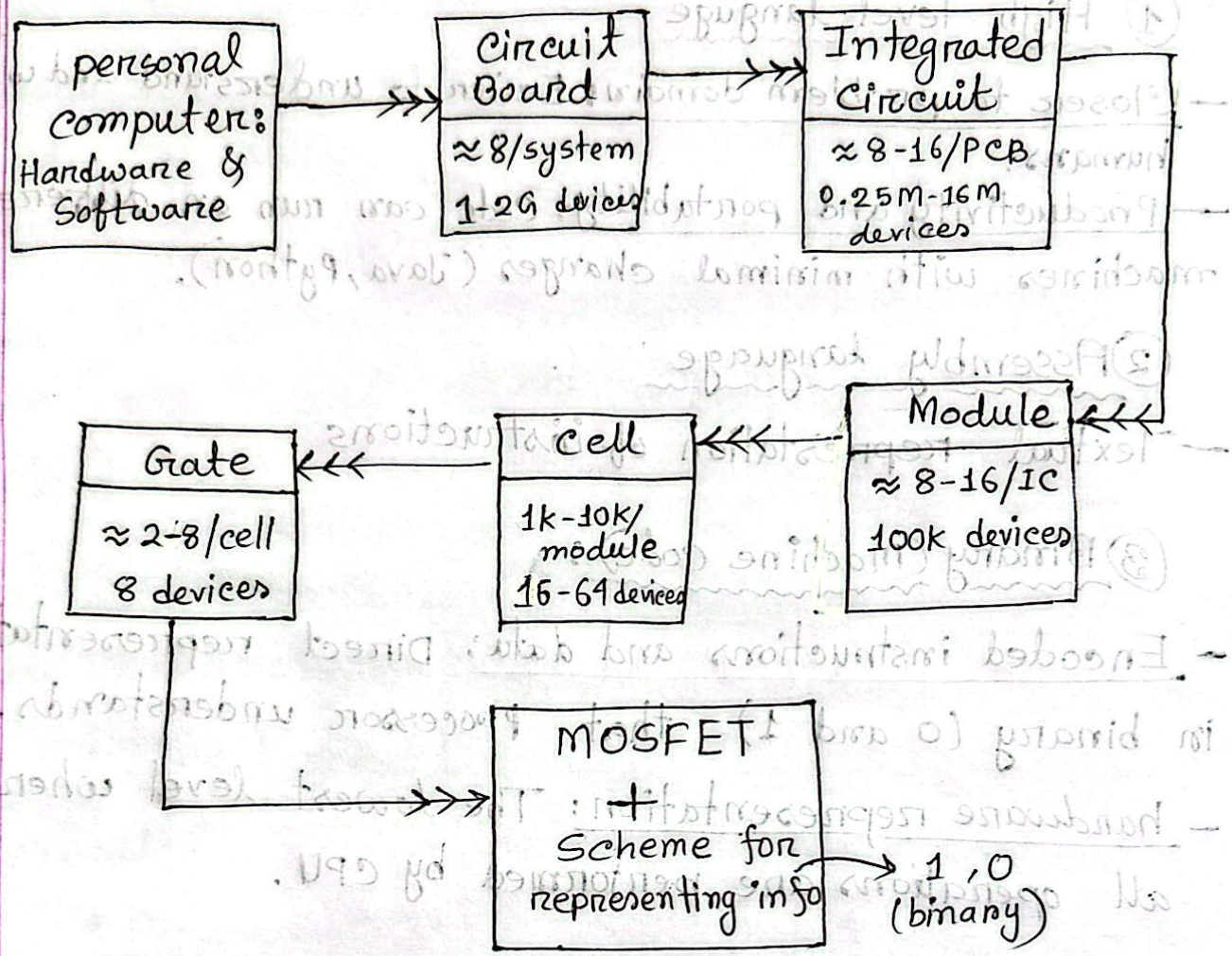
③ Binary (Machine code):

- Encoded instructions and data: Direct representation in binary (0 and 1) that processor understands.
- hardware representation: The lowest level where all operations are performed by CPU.

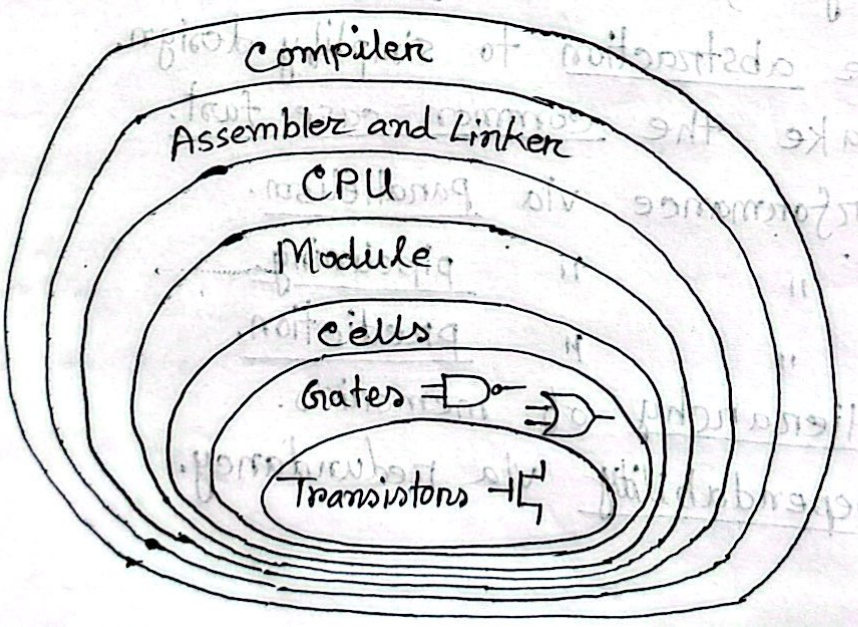
Eight Great Ideas:

① Design for Moore's law.
② Use abstraction to simplify design.
③ Make the common case fast.
④ Performance via parallelism.
⑤ " " pipelining.
⑥ " " prediction.
⑦ Hierarchy of memories.
⑧ Dependability via redundancy.

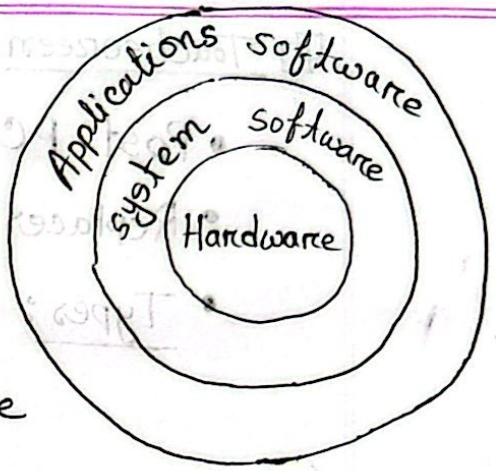
Abstraction : key to building large systems :



A Computer System :



Below your Program:



- Application software — written in high-level language
- System software —

Compiler — translate HLL code to machine code.

Operating System — service code

- Handling Input/Output
- Managing memory and Storage
- Scheduling task and sharing resources.

Hardware — Processor, memory, I/O controllers.

Components of a Computer:

All computers - desktops, server and embedded systems have the same basic components.

- Input/Output devices — Allow users to interact with the computer.
  - Diplay (Monitor
  - Mouse, keyboard
- Storage devices — Store data permanently
  - Hard disk, SSD
  - CD/DVD, USB Flash Drive
- Network Adapters — Enable communication with other devices
  - Wifi
  - Ethernet

Touchscreen:

- Post PC device
- Replaces traditional keyboard and mouse
- Types:
  - (I) Resistive – Works with pressure (finger), less sensitive
  - (II) Capacitive – Used in smartphones/tablets supports multi-touch gesture

Capacitive touchscreen are more common in moder devices due to better responsiveness and durability

Displays:

An output device for presentation of information in visual form.

- Types:
  - (I) CRT – Cathode Ray Tube
    It's the last vaccume tube which is near extinction.
  - (II) LCD – Liquid Crystal Displays

Issues of modern computers:

Major challenge in moder computing is ~~ergy~~ energy or power consumption. Now it has become a limiting factor because of heat ~~and~~ generation, battery drain, needs massive power increasing cost and environmental impact.

Inside the Processor:

Datapath: performs operation on data

Control: sequence datapath, memory

Cache memory: Small fast SRAM memory for immediate access to data:

Example: Intel Core i7 has '4 cores'

- each core is essentially an independent computing engine
- at the top level, memory and I/O is shared by all cores.

Abstraction:

It helps us deal with complexity by hiding lower level details

Level of Abstraction:

① ISA (Instruction set Architecture) — The hardware and software interface.

② ABI (Application binary interface) — ISA plus system software interface.

③ Implementation: details underlying and interface.

Safe place for data:

- Votatile main memory — losses instructions and data when power off
- Non-volatile secondary memory —
  - Magnetic disk
  - Flash memory
  - Optical disk (DVD, CD, ROM)

Networks: Enables communication, resource sharing and non-local access between devices.

- LAN (Local area network): Ethernet
- WAN (Wide area network): Internet
- Wireless network: WiFi, Bluetooth.

Technology Trends:

Electonics technology continuous to ~~evole~~ evolve making device smarter, smaller and more powerful.

- Increased capacity and performance
- Reduced cost

Semiconductor Technology: Silicon: Semiconductor

Add materials:

- Conductors
- Insulators
- Switch

Implentation Technology:

- Relays
- Vaccum Tubes
- Transistors
- Integrated circuits → Gate-level, Medium Scale Integration, Large " "
- Nanotubes
- Quantum-Effect Devices
- Common links - controllable switch.

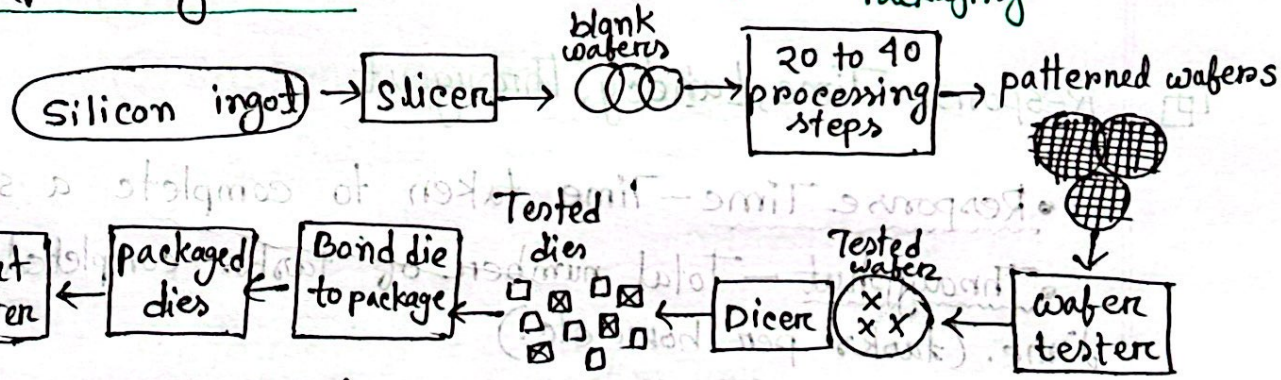
Chips: Silicon wafers

Chip manufacturers build many copies of same circuit onto a single wafer. Only a certain percentage of the ~~so~~ chip will work; those that work will run at different speeds. The yield decreases as the size of chips increases and the feature size decreases

- Wafers are processed by automated fabrication lines. To minimize the chance of contaminants ruining a process step, great care is taken to maintain a clean environment.

Manufacturing ICs:

① Slicing
② Processing
③ Wafer testing
④ Dicing
⑤ Die Testing
⑥ Bonding & Packaging
⑦ Final Testing
⑧ Shipment



↳ Tested packaged dies → Ship to customers

Integrated circuit cost:

- Cost per die = $\frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$
- Dies per wafer ≈ wafer area / Die area
- Proportion of working dies (per wafer) = Yield = $\frac{1}{\left(1 + \left(\frac{\text{Defects per area} \times \text{Die area}}{2}\right)\right)^2}$

Part-2

Performance

What is Performance? Give examples. Why is it important?

Performance refers to how fast a computer completes a task. Example - Short tasks (calculating numbers, displaying PDF, responding to input); Long tasks (record a video; search document on hard drive, Apply a photoshop filter etc.).

Performance is important because it help us make intelligent design choices. It is the key to understand why some hardwares run programs faster, how instruction set impact performance or do we need new machine or OS to boost performance factors.

Response Time, Latency, Throughput

- Response Time – Time taken to complete a single task.
- Throughput – Total number of tasks completed per unit time. (tasks per hour etc.)

Effects of Changes:

- Faster Processor
  - Reduces response time
  - Increases throughput
- Adding more processor:
  - May not reduce response time
  - Improves throughput by handling multiple tasks in parallel.

- Latency – Time taken for an input to produce an output.

  Example - delay between pressing 'Enter' and getting a result.

Design Trade-offs – Performance is not only the concern. Cost and Energy consumption is important too.

Key Metrics:
1. Performance / Cost
2. Performance / Power
3. Work / Energy – (for battery powered devices)

Execution Time and CPU Time:

- **Execution time (Elapsed Time / Wall clock Time):**
  - not ideal for comparison
  - Total time ~~start~~ from start to finish, including I/O, disk access, program running.
- **CPU Time** – Only includes time that CPU spends running the program. Divided into:
  1. User time
  2. System time

Our focus

Performance Math:

Formula: For some program running on machine P,

$$\text{Performance}_P = \frac{1}{\text{Execution Time}_P}$$

Relative Performance: "X is n times faster than Y"

$$\Rightarrow \frac{\text{Perf}_X}{\text{Perf}_Y} = n$$

Machine A runs a program in 20s. Machine B runs a program in 25s. By how much ~~time~~ is A faster than B? By How much B is slower than A?

$$\text{Perf}_A = \frac{1}{20} \qquad \therefore \frac{\text{Perf A}}{\text{Perf B}} = \frac{1/20}{1/25} = \frac{25}{20} = 1.25$$

$$\text{Perf}_B = \frac{1}{25} \qquad \therefore \frac{\text{Perf B}}{\text{Perf A}} = \frac{1/25}{1/20} = \frac{20}{25} = 0.8$$

A is 1.25 times as fast as B. B is 0.8 times as fast as A/B is 1.25 times slower than A.

Another way: $\frac{\text{Perf}_A - \text{Perf}_B}{\text{Perf}_B} \times 100\%$

$$= \frac{0.05 - 0.04}{0.04} = 0.25 \times 100\%$$

∴ A is 25% faster than B.

for B, $\frac{0.04 - 0.05}{0.05} \times 100\% = -20\%$ faster

$= 20\%$ slower

∴ B is 20% slower than A.

* ~~So~~, Here, 2 methods are correct but the first one is better than the second way. because second one is a bit confusing. So we will do the math with 1st method.

Execution time taken to run a program.

10s on A, 15s on B ; How much fast is A than B?

$$\frac{\text{Perf A}}{\text{Perf B}} = \frac{1/10}{1/15} = \frac{15}{10} = 1.5$$

So, A is 1.5 times faster than B.

Measuring Execution Time:

- Elapsed Time – Total response time (Processing, I/O, OS overhead, idle time), Determines system performance
- CPU Time – Time spent processing a given job, include user CPU Time and system CPU Time

CPU Clocking:

- Operation of digital hardware governed by a constant rate clock
- Clock period : duration of a clock cycle

$$250ps = 0.25ns = 250 \times 10^{-12}s$$

- Clock frequency rate: cycles per second

$$4GHz = 4000MHz = 4 \times 10^9 Hz$$

CPU Time:

$$\text{CPU Time} = \text{CPU Clock cycle} \times \text{Clock cycle Time}$$

Formula:

$$\text{CPU Time} = \frac{\text{CPU Clock cycle}}{\text{Clock Rate}}$$

Performance improved by –

- Reducing number of cycles
- Increasing clock rate

- Cycle Time = Seconds per cycle
- Clock Rate = Cycles per second (1 Hz = 1 cycle/s) (frequency)

Example: ∴ 200 MHz clock has a cycle time of

$$\frac{1}{200\text{MHz}} = \frac{1}{200 \times 10^6\ \text{Hz}} = 5 \times 10^{-9}\ \text{s} = 5\ \text{ns}$$

Example: Page - 57

	Computer A	Computer B
CPU Time :	10s	6s
Clock Rate	2GHz	?
Clock Cycle	x	1.2x

$$\therefore \text{CPU Time}_A = \frac{\text{Clock Cycle}_A}{\text{Clock Rate}_A}$$

$$\Rightarrow 10 = \frac{x}{2\text{GHz}}$$

$$\Rightarrow x = 20\text{GHz}$$

$$\Rightarrow x = 20 \times 10^9\ \text{Hz}$$

for B,

$$\therefore \text{CPU Time}_B = \frac{\text{Clock cycle}_B}{\text{Clock Rate}_B}$$

$$\Rightarrow 6s = \frac{1.2 \times 20 \times 10^9}{\text{Clock Rate}}$$

$$\therefore \text{Clock Rate} = \frac{1.2 \times 20 \times 10^9}{6}$$

$$= 4 \times 10^9\ \text{Hz}$$

$$= 4\ \text{GHz}$$

Instruction Count and CPI (Cycles Per Instruction)

Instruction count x CPI = Clock cycle

∴ CPU Time = Clock cycle x Clock cycle Time

= Instruction count x CPI x Clock cycle time

$$= \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock Rate}}$$

Instruction count → Total number of instructions a program can execute during its runtime.
It is determined by program and ISA.

Average cycles per Instruction → Determined by CPU hardware.

Computer Performance Measure –

MIPS = Millions of Instructions per Second.

$$\text{MIPS} = \frac{\text{Clock Rate}}{\text{CPI}}$$

- How to improve performance?
  - Decrease # of required cycles for a program
  - " the clock cycle time
  - " the CPI
  - Increase the Clock Rate

Performance is determined by execution time and these factors.

CPI Example:

	Computer A	Computer B
Clock cycle (time)	250 ps	500 ps → picosecond
CPI	2.0	1.2

$$\text{CPU Time}_A = \text{Instruction count}_A \times \text{CPI}_A \times \text{Cycle Time}_A$$

$$= IC \times 2.0 \times 250\text{ps} = IC \times 500\text{ps}$$

$$\text{CPU Time}_B = IC \times 1.2 \times 500\text{ps}$$

$$= IC \times 600\text{ps}$$

Here we can see, B takes more time than A.

Relative Performance $= \dfrac{\text{Perf A}}{\text{Perf B}} = \dfrac{1/500}{1/600} = \dfrac{600}{500} = 1.2$

A is 1.2 times faster than B.

☐ CPI in more detail:

$$\text{Clock cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction count}_i)$$

Weighted average CPI,

$$\text{CPI} = \frac{\text{Clock cycles}}{\text{Instruction count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \frac{\text{Instruction count}_i}{\text{Instruction count}} \right)$$

Example:

no. of cycle =	class A (1)	class B (2)	class C (3)
Compiler 1	5	1	2
Compiler 2	7	1	1

of instructions:

∴ Compiler 1, 5+1+2 = 8 million

∴ Compiler 2, 7+1+1 = 9 million

So, compiler 1 uses fewer instructions

of Clock cycles:

Compiler 1, (5×1) + (1×2) + (2×3) = 13 million cycles

Compiler 2, (7×1) + (1×2) + (1×3) = 12 million cycles

So, compiler 2 uses fewer cycles.

$$CPI_1 = \frac{\text{Clock cycle}}{\text{Instruction count}} = \frac{13 \text{ million}}{8 \text{ million}} = 1.625$$

$$CPI_2 = \frac{12 \text{ million}}{9 \text{ million}} = 1.33$$

Compiler 2 has lower CPI, which means, it's instructions are more efficient on average.

Performance depends on algorithm, programming language, compiler and ISA (Instruction set architecture)

Benchmarks: A benchmark is a test used to measure the performance of a computer, processor, or software. It is determined by running a real application such as compiler, editors, scientific applications, graphics etc.

- **Small bench marks:**
  - nice architects and designers
  - easy to standardize
  - can be abused
- **SPEC benchmarks:** (System Performance Evalution Cooperative)
  - valuable indicator of performance and compiler technology
  - Companies use it to compare systems fairly.

Amdahl's Law:

most important law regarding computer performance:

$$t_{improved} = \frac{t_{affected}}{p \text{ speedup}} + t_{unaffected}$$

It helps to understand maximum speedup we can acheive by improving part of a system

Example: (Page-72)

□ for 4 times faster,

program runs in 100 seconds

to make it 4 times faster, $\frac{100}{4} = 25s$

So, it should run in 25s. = $t_{improved}$

Time affected = 80% of the time (is spent on multiplication)

$= 80s$

Time unaffected = $100 - 80 = 20s$

$\therefore t_{improved} = \frac{t_{affected}}{r_{speedup}} + t_{unaffected}$

$\Rightarrow \frac{100}{4} = \frac{80}{r} + 20$

$\therefore r = 16$

$\therefore$ So, multiplication must be 16x faster.

□ for 5 times faster,

$\frac{100}{5} = \frac{80}{r} + 20$

$\Rightarrow 20 = \frac{80}{r} + 20$

$\Rightarrow \frac{80}{r} = 0$

$\therefore r = \infty$

This is impassible to make 5 times faster.

r = Speed up factor = how much faster? program runs.

(Page 73)

Example:

□ $r_{speedup} = 5$

$t_{affected}$ = half of 10s = 5s spent executing

$t_{unaffected} = 5s$

Applying Amdahl's law,

$$t_{improved} = \frac{5}{5} + 5 = 6s$$

So, the new execution time is 6s, making the speedup :-

$$\text{Speed Up} = \frac{10}{6} = 1.67 \qquad \left[\text{SpeedUp} = \frac{\text{Old Execution Time}}{\text{New Execution Time}}\right]$$

□ Original execution time 10s

We want 3x faster program

$$\therefore t_{improved} = \frac{100}{3} = 33.33s$$

let,

$t_{affected} = X$ = time spent on floating operation.

$$\therefore t_{improved} = \frac{t_{affected}}{r_{speedup}} + t_{unaffected}$$

$$\frac{100}{3} = \frac{X}{5} + (100 - X)$$

$$\Rightarrow X = 83.33$$

Power consumption:

Power = energy consumed per unit time

Two contributors of power consumption:

- **Dynamic Power**

- power consumed when doing actual work
- called dynamic because components and wires are switching between '0' and '1'

- **Static or leakage power**

- power consumed even when everything is idle or 'static'
- due to some small amount of leakage current that still flows.

Dynamic Power consumption:

$$\therefore P_{dynamic} = CV_{DD}^2 * \frac{f}{2} = \frac{1}{2} CV_{DD}^2 f$$

C = total capacitance of circuit (capacitive load)

V_{DD} = Supply voltage

f = switching frequency

Static power consumption:

$$\therefore P_{static} \text{ or } P_{leakage} = I_{DD} V_{DD}$$

V_{DD} = supply voltage

I_{DD} = leakage current

Example:

$V_{DD} = 1.2V$

$C = 20nF$

$f = 1GHz$

$I_{DD} = 20mA$

$$P = \frac{1}{2} C V_{DD}^2 f + I_{DD} V_{DD}$$

$$= \frac{1}{2} (20nF)(1.2)^2 (1GHz) + (20mA)(1.2V)$$

$$= 11.4W + 24mW \quad [0.24W]$$

$$= 14.424W$$

~~Power = $\frac{1}{2}$ · CPUold = Capacitance, C = 85%~~

CPUnew, Capacitance $= 0.85 \times C_{old}$

Voltage $= 0.85 \times V_{old}$

Frequency $= 0.85 \times F_{old}$

$$\text{Reducing Power} = \frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^2 \times F_{old}}$$

$$= (0.85)^4 = 0.52$$

So, the new CPU is 52% of the old CPU's Power consumption.

Multicore Processors:

These have multiple processors on a single chip, requiring explicit parallel programming. (not automatic like instruction level parallelism) Compare with Instruction level parallelism, Hardware executes multiple ~~pt~~ instructions at once. It needs programmer to handle tasks like load balancing, optimizing communication and synchronization.

Pitfalls of MIPS: MIPS measures the speed of a processor by calculating how many instruction it can execute per second (in million).

$$MIPS = \frac{\text{Instruction Count}}{\text{Execution time} \times 10^6}$$

Pitfalls: ① It doesn't account for instruction complexity.

② Doesn't reflect real world performance since different CPUs have different instruction set.

③ Ignores memory access time and pipeline effects.